Towards an Approach for Configuring Ontology Validation

Mounira Harzallah[1(✉)], Giuseppe Berio[2], and Pascale Kuntz[1]

[1] LINA, University of Nantes, Rue Christian Pauc, Nantes, France
{mounira.harzallah, pascale.kuntz}@univ-nantes.fr
[2] IRISA, University of Bretagne Sud, Vannes, France
giuseppe.berio@univ-ubs.fr

Abstract. Ontologies are becoming widely recognised as key components in various types of systems and applications. However, ontology validation remains a critical open issue. In previous work, we have proposed a standard typology of problems that need to be removed for validating one ontology. Indeed, there is no standard vocabulary and definitions for those problems. In this paper, we introduce checking dependencies between standard problems; a validation process needs to satisfy checking dependencies for detecting and removing properly problems mentioned above. Then, we report an experience, based on 2 ontologies automatically generated from textual resources, showing how the typology can practically be deployed for configuring a validation process.

Keywords: Ontology validation · Ontology quality evaluation · Ontology quality problems · Quality problem checking · Automatically generated ontology

1 Introduction

Ontologies are becoming widely recognised as key components in various types of systems and applications: e.g. knowledge management systems, social network analysis, business intelligence and personalised applications. Ontologies have been and still are manually designed by human experts. However, the ever-increasing access to textual sources (as technical documents, web pages and so on) has motivated the development of tools for automating as much as possible ontology design and implementation process (being this process often renamed as "ontology learning"), and further enrichment. As a consequence, human involvement is rather minimised when such tools are used. Promising results have been reached [1, 2]. Unfortunately, experimental studies have then put in evidence limits for real-life applications [2, 3] and recent works recommend a better integration of human involvement [4].

Following this recommendation, we see the process of ontology learning as made of two main processes running in parallel and cooperating: (1) a *generation process* and (2) a *validation process*. The generation process focuses on the extraction of relevant items (such as terms and relations) and the identification and naming of relevant knowledge (such as concepts). The validation process is performed anytime

A. Fred et al. (Eds.): IC3K 2014, CCIS 553, pp. 388–404, 2015.
DOI: 10.1007/978-3-319-25840-9_24

when needed during the generation process and beyond. This is because, according to our experiences, validation should be performed as soon as possible focusing on subparts (such as subset of concepts) of the ontology under construction. Furthermore, validation process can be defined as the process guaranteeing the expected quality of the ontology (while the generation process makes the ontology content available). Thus, the validation process is a process specifically (i) looking for any poor (or bad or lower) quality of the ontology under construction through a sort of "*quality evaluation"* (or "*ontology evaluation*") and then (ii) proposing alternatives and applying selected alternatives for increasing quality by finding and removing recognised defects (i.e. modifying/deleting/adding artefacts to the ontology under construction). Since the pioneering works of Gruber in the 90's [5], *quality evaluation* has been discussed in [6], and, often independently from validation, various procedures and features have been proposed [7–11]: e.g. defining a set of quality measures, comparing ontologies to reference ontologies (also called gold standards), performing assessment of formal correctness, quality qualitative evaluation performed by experts, quality evaluation according to the results of a given application using the ontology, using pre-defined anti-patterns corresponding to defects or to potential defects. Roughly speaking, quality evaluation spans over three major criteria: (1) the *dimensions* which are evaluated (e.g. functional dimension, structural dimension or usability dimension) [7, 12], (2) the *evaluation mode* (manual vs automated) [13] and the *user profile* if any (e.g. knowledge engineers, business analysts, practitioners) [9], (3) and the *phase* in which evaluation is conducted (e.g. during the ontology development, before ontology publication and so on) [9, 14].

An analysis of the state of the art reveals that there are *three facets* when referring to "quality evaluation": (i) *scores of ontology quality* (such as, high, poor, bad or numeric scales) evaluated by using several quality measures, (ii) *quality problems* i.e. symptoms of defects or potential defects impacting ontology quality, and (iii) *defects* in the ontology i.e. the specific ontology artefacts (typically, concepts, relationships, axioms) which are causes of any poor quality and/or problem. The three facets are naturally related: for instance, if an ontology is inconsistent, this is a problem in our terminology i.e. a symptom of a defect in the ontology; the defect is the axioms causing inconsistency; quality scoring may be defined as dependent on the number of axioms causing inconsistency (a quality measure). Even though it seems natural that quality measures being related to defects, this is often not the case (as explained in Sect. 2). For instance, a measure like "ontology depth" can be used for scoring quality saying that "ontology quality" is directly proportional to "ontology depth" without referring to any (potential) defect (in this case, sample defects are the "omitted IS-A relationships"). This situation leads to difficulties for using quality scores in practice for validation purposes. On the contrary, problems are introduced and explained as symptoms of some defects and therefore are closer to defects than scores; thus, using problems and their dependencies for removing defects seems more effective than using quality scores. Therefore, in our previous work [15] we have focused on the problem facet. This previous work has specifically targeted one critical aspect of the problem facet i.e. the standardisation of *problems* and *problems definitions*, currently rather variable. Accordingly, we have proposed a *typology of problems* which: makes a synthesis of the state of the art, is extensible, is easy to understand and based on a well-known quality

framework defining quality for conceptual models [16] (ontologies are special cases of conceptual models). However, in that past work, we have not provided details on how, in practice, the proposed typology can be deployed and used in the context of validation process. In this paper, we are going to present one experience (based on 2 ontologies automatically built from textual sources) showing how the proposed typology can concretely be deployed and lessons learned for *configuring a validation process*.

The paper is organised as follow. Section 2 provides a short overview on quality and problems in ontologies. Section 3 introduces the proposed typology of ontology problems and dependencies between them. Section 4 describes the performed experience and provides feedback reporting (discussion and lesson learned). Finally, conclusions summarise key results.

2 State of the Art Insights

As reported in the Introduction, ontology quality evaluation concerns three related facets: scoring quality through measurements based on quality measures, quality problems and defects. Figure 1 below provides a simple picture (as a UML class diagram), for representing the three facets (i.e. quality problem, quality measure and defects) and key relationships according to state of the art. Figure 1 can be explained as follow. Whenever precise and scoped definitions of quality problems are available is possible to select (or to develop) techniques for detecting or warning about those problems (a). Quality measures can be used for detecting or warning about defects/problems, especially when combined with reference values: quality measures are therefore techniques for that purpose (b). Quality measures are also techniques to evaluate the ontology quality (c). Ontology problems can be used for a qualitative assessment of ontology quality (d). When a quality problem is detected, it can trigger the usage of techniques for detecting defects causing that problem (e). Finally, even if a problem may not be associated to some techniques for identifying it, this is not suitable.

Sections 2.1 and 2.2 shortly present relevant state of the art insights on quality measures and problems.

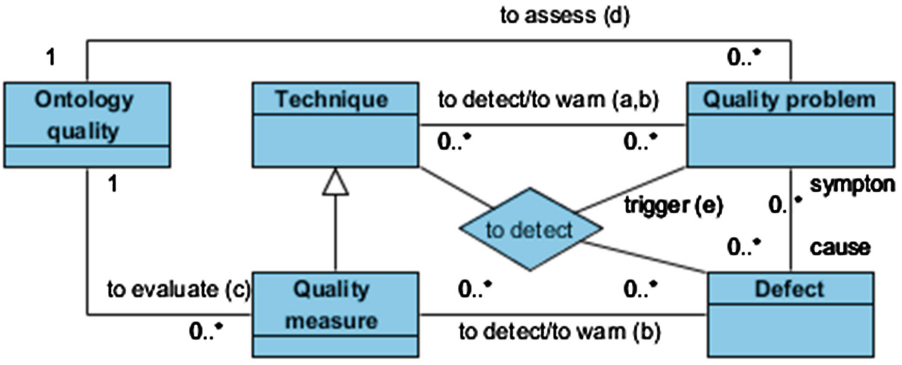

Fig. 1. The three facets of quality evaluation and key relationships.

2.1 Quality Measures

As reported in the Introduction, existing proposals cover various "quality dimensions". In the context of ontologies, dimensions have not been standardised. For instance, they may be referred to as, syntax, semantics, maintenance and ergonomics or, functional, structural and usability.

One of the most complete proposals associating dimensions and measures is probably oQual [12]. In oQual, an ontology is analysed according to three dimensions: (i) structural (syntax and formal semantics of ontologies), (ii) functional (intended meaning of the ontology and its components) and (iii) usability (pragmatics associated with annotations, which contribute to the ontology profiling). A set of measures is associated with each dimension to score the quality. For instance: for structural dimension, depth and breadth of a taxonomy; for functional dimension, precision and recall of the ontology content with respect to its intended meaning; for usability dimension, number of annotations.

More generally, despite the potential interests of measures proposed in literature, some of them remain quite disconnected to defects/problems. For instance, the ratio between number of concepts and number of relations (N°Concepts/N°Relations) is a quality measure for evaluating "cognitive ergonomics", which is in turn closely related to a "easy to use" quality: however, the ontology may not suffer of any problem/defect concerning its artefacts (concepts, relationships, axioms etc.) because representing as it is the targeted domain. Therefore, defining quality measures (possibly organised alongside several dimensions) as entry point does not necessarily make explicit (occurring or potential) defects/problems. On the contrary, defining problems as entry point provides an evidence of occurring defects that, in turn, lead to poor quality.

2.2 Quality Problems

Roughly speaking, the generic notion of "ontological error" covers a wide variety of problems affecting different dimensions. In the relevant literature, it is possible to find precise and less precise definitions for several recognised problems: (1) "taxonomic errors" [8, 10, 17, 18] or "structural errors" [19], (2) "design anomalies" or "deficiencies" [10, 20], (3) "anti-patterns" [11, 19, 21], (4) "pitfalls" or "worst practices [22] and (5) "logical defects" [19]. Additional errors could complete this list: e.g. (6) "syntactic errors" [19].

Hereinafter, we shortly present insights on each of the above mentioned problem cases. Syntactic errors (6) are due to violations of conventions of the language in which the ontology is represented. While interesting in practice for building support tools, they are conceptually less important than others: therefore they will be no longer considered in the remainder.

Taxonomic errors (1) concern the taxonomic structure and are referred to as: inconsistency, incompleteness and redundancy [17]. Three classes of "inconsistency" both logical and semantic have been highlighted: circularity errors (e.g. a concept that is a specialization of itself), partitioning errors (e.g. a concept defined as a specialization of two disjoint concepts), and semantic errors (e.g. a taxonomic relationship in

contradiction with the user knowledge). Incompleteness occurs when for instance, relationships or axioms are missing. Finally, redundancy occurs when for instance, a taxonomical relationship can be deduced from the others by logical inference.

Design anomalies (2) concern ontology understanding and maintainability [10, 20]: lazy concepts (leaf concepts without any instance or not considered in any relation or axiom), chains of inheritance (long chains composed of concepts with a unique child), lonely disjoint concepts (superfluous disjunction axioms between distant concepts), over-specific property range and property clumps (duplication of the same properties for a large concept set which can be retrieved by inheritance).

Anti-patterns (3) are known or recognised templates potentially leading to identified problems [11, 19]. Some classes of anti-patterns are: logical anti-patterns (producing conflicts that can be detected by logical reasoning), cognitive anti-patterns (caused by a misunderstanding of the logical consequences of the axioms), and guidelines (complex expressions true from a logical and a cognitive point of view but for which simpler or more accurate alternatives exist).

Pitfalls (4) cover problems for which ontology design patterns (ODPs) are not available. An ODP cover ad-hoc solutions for the conception of recurrent particular cases [21]. Poveda et al. [23] have established a catalogue of pitfalls grouped on 7 classes, them-self classified under the three ontology dimensions cited above [12]. Four pitfalls classes are associated with the structural dimension: modelling decisions (false uses of OWL primitives), wrong inference, no inference (lacks in the ontology which prevent inferences required to produce desirable knowledge), real-world modelling (common sense knowledge missing). One class is associated with the functional dimension: requirement completeness (e.g. uncovered specifications). And, two classes are associated with the usability dimension: ontology understanding and ontology clarity (e.g. variations of writing-rule and typography for the labels). Poveda et al. [22] have also tried to classify these pitfalls according to the three taxonomic error classes [17]; but pitfalls concerning the ontology context do not fit with this classification.

3 Problem Standardisation Overview

Mentioned in the Introduction and made evident in Sect. 2.2, heterogeneity in quality problems and their definitions is due to distinct experiences, communities and perception of ontologies. Standardisation enables a much better understanding of what problems are and to what extent these problems are critical before using the ontology. We have therefore proposed a two-level rigorous problem typology summarised in Table 1. Level 1 distinguishes logical from social ground problems and level 2 distinguishes errors from unsuitable situations. Errors are problems mostly preventing the usage of an ontology. We add “mostly” because in the case of “inconsistency error” (Table 1), some researches focus on how to make usable inconsistent ontologies [24]. On the contrary, unsuitable situations are problems which do not prevent ontology usage (within specific targeted domain and applications). Therefore, while errors need to be solved, unsuitable situations may be maintained as such.

Social ground problems are related to the interpretation and the targeted usage of the ontologies by social actors (both humans and applications). Logical errors and most

of logical unsuitable cases can be rigorously formalised within a logical framework; for instance, they can be formally defined by considering key notions synthesised by Guarino et al. [25] i.e.: Interpretation (I) (extensional first order structure), Intended Model, Language (L), Ontology (O) and the two usual relations $\models$ and $\vdash$ provided in any logical language. The relation $\models$ is used to express both that one interpretation I is a model of a logical theory L, written as $I \models L$ (i.e. all the formulas in L are true in I: for each formula $\varphi \in L$, $I \models \varphi$), and also for expressing the logical consequence (i.e. that any model of a logical theory L is also a model of a formula: $L \models \varphi$). The relation $\vdash$ is used to express the logical calculus i.e. the set of rules used to prove a theorem (i.e. any formula) φ starting from a theory L: $L \vdash \varphi$). Accordingly, when needed, problems are formalised by using classical description logic syntax that can also be transformed in FOL or other logics.

Problems in Table 1 are not independent and need to be checked out and removed in meaningful order in the context of a validation process. Table 2 below provides a list of *checking dependencies* that constraint orders for checking and removing problems. Checking dependency *A1,.., An → B1,..., Bm* means that before checking out any Bi, all problems Ai (i.e. A1 to An) need normally to be checked out and removed (i.e. left out) by appropriate techniques. For in practice configuring a validation process, dependencies can be used *forward or backward* as follow: *A1,.., An → B1,..., Bm*, is used forward if A1,.., An are checked out and removed before taking into account B1,…, Bm; A1,.., An → B1,…, Bm is used backward if B1,…, Bm are checked out before and this is used for checking out and removing A1,…, Am.

It should be noted that removing a logic ground problem from an ontology results in a new ontology which is or is not logically equivalent to the previous one. Depending on the problem, there may or may not be a way for removing it by guaranteeing logical equivalence. It is clear that L1, L2, L3, L10 can only be removed without guaranteeing logical equivalence; L4, L5, L6, L12 can be removed by guaranteeing logical equivalence; for L7, L8, L11 both ways are possible. It is quite natural to consider by default that removing any social ground problem results in a new non-logically equivalent ontology.

In the context of validation process, checking dependencies:

1. Make meaningful or optimize problem checking (for instance, it does not make sense checking if an ontology is unadapted, if the ontology is inconsistent and inconsistency problem has not removed yet);
2. Suggest which problems can be removed without causing injection of additional problems. For instance, checking out if an ontology is not minimal (L12) and removing the problem, does not make sense without having previously checked out and removed social contradictions (S1) within the ontology; as a concrete case, if redundant IS-A relationships (a case for L12) are found, if one of them is removed and then, another IS-A relationship is removed because suffering of S1, the ontology may become incomplete (injection of L3 or S4). However, when a problem is removed without guaranteeing logical equivalence with the new resulting ontology, all the problems need to be checked out again and all dependencies need to be taken into account once again (as, for instance, when L2 is removed).

Table 1. The typology of quality problems.

Logical ground problems	
Errors	L1. Logical inconsistency: no I of s.t. $I \models O$
	L2. Unadapted ontologies: there is a formula φ for some intended models of L, φ is false and $O \models \varphi$
	L3. Incomplete ontologies: there is a formula φ for each intended models of L, φ true and $O \not\models \varphi$
	L4; Incorrect (or unsound) reasoning: when a false formula φ in the intended models $O \not\models \varphi$, can be derived from a suitable reasoning system ($O \vdash \varphi$)
	L5. Incomplete reasoning: when a true formula φ in the intended models $O \models \varphi$, cannot be derived from a reasoning system ($O \not\vdash \varphi$)
Unsuitable cases	L6. Logical equivalence of distinct artefacts: $O \models A_i = A_j$
	L7. Logical indistinguishable artefacts: impossible to prove any of the following statements: ($O \models A_i = A_j$), ($O \models A_i \cap A_j \subseteq \bot$) and ($O \models c \subseteq A_i$ and $c \subseteq A_j$)
	L8. OR artefacts: A_i equivalent to $A_j \cup A_k$, $A_i \neq A_j$, $A_i \neq A_k$, but for which (if applicable) there is neither role R s.t $O \models (A_j \cup A_k) \subseteq \exists R.\top$, nor instance c s.t. $O \models c \subseteq A_j$ and $O \models c \subseteq A_k$
	L9. AND artefacts: A_i equivalent to $A_j \cap A_k$, $A_i \neq A_j$, $A_i \neq A_k$, but for which (if applicable) there is no common (non optional) role/ property for A_j and A_k
	L10. Unsatisfiability: given an artefact A, $O \models A \subseteq \bot$)
	L11. Complex reasoning: unnecessary complex reasoning when a simpler one exists
	L12. Ontology not minimal: unnecessary information
Social ground problems	
Errors	S1. Social contradiction: contradiction between the interpretation and the ontology axioms and consequences
	S2. Perception of design errors: e.g. modelling instances as concepts
	S3. Socially meaningless: impossible interpretation
	S4. Social incompleteness: lack of artefacts
Unsuitable cases	S5. Lack of/poor textual explanations: lack of annotations
	S6. Potentially equivalent artefacts: similar artefacts identified as different
	S7. Socially indistinguishable artefacts: difficult to distinguish different artefacts
	S8. Artefacts with polysemic labels
	S9. Flatness of the ontology: unstructured set of artefacts
	S10. Non-standard formalization of the ontology: unreleased specific logical use
	S11. Lack of adapted and certified version of the ontology in various languages
	S12. Socially useless artefacts

Table 2. Checking dependencies between quality problems.

L1 → L2, L3, L4, L5, L6, L7, L8, L9, L10, L11, L12	If an ontology is inconsistent, inferences do not make sense and any other problem can be trivially detected; this dependency should be only used as forward dependency
L2, L3 → L6, L7, L8, L9, L10, L11, L12 L2, L3 → L4, L5	It does not make sense to assess unsuitable situations on one ontology which is not completely finalised; the same is true for unsound/incomplete reasoning; this dependency should be used as forward dependency; however a backward usage is possible (for instance, L6 can be checked out and this may be used for highlighting unintended models)
S12, S3 → S2, S5, S6, S7, S8	Useless and meaningless artefacts should be removed and ontology updated accordingly before checking out any other problem; however, S10 and S11 can be checked independently; this dependency has be used as forward dependency
S1, S2, S3, S12 → L2, L3, L4, L5, L6, L7, L8, L9, L10, L11, L12	Meaningless and useless artefacts, design errors and social contradictions should be removed before checking out any logic ground problem (except L1); this dependency can be also used backward: for instance, L12 is checked out and among redundant artefacts, S1 is then checked out and removed accordingly
S2 → S1, S4, S9	Modelling errors should be removed before checking out social contradiction, incompleteness and flatness; this dependency has be used as forward dependency

4 Experience

This section presents an experience on the deployment of the typology based on two ontologies automatically generated from different corpora by using Text2Onto [1]. We have used Text2Onto in one of our past research projects [26] and realised a full comparison with similar tools. The comparison results made possible to select Text2Onto as the best choice for realising the work. This was also confirmed by successful work performed in the project, making possible to use extracted ontologies as components for interoperating enterprise systems. However, Text2Onto capability for extracting concepts and taxonomic relationships has been shown to significantly outperform its capability for extracting other types of artefact [27, 28].

4.1 Experience Setting

As said above, we have generated two ontologies by using Text2Onto. Generated raw ontology O1 (resp. O2) contains 441 (resp. 965) concepts and 362 (resp. 408)

taxonomic relationships. The first ontology (O1) has been generated starting from a scientific article in the domain of "ontology learning from texts" containing 4500 words. The second ontology (O2) has been generated starting from a technical glossary composed of 376 definitions covering the most important terms used in the composite material domain. The glossary contains 9500 words. It has been provided by enterprises involved in the project. It should be noted that although showing quite different content features, the size of the two selected textual resources has been deliberately limited to enable further detailed analysis of the experience results.

4.2 Typology Deployment for the Experience

In the experience, the deployment of the typology for the ontology problem detection is performed in two steps. The first step is about the selection of problems that may occur within the ontology to validate. The second step concerns the identification of appropriate techniques or the development of new ones for the detection of each of selected problems.

Problem Selection (First Step). Appropriate techniques are required for the detection of ontology problems. However, there is no need to possess a technique for each typology problem to identify problems within an ontology, especially because ontologies range from very simple (or light) to very complex (or heavy).

In the experience, ontologies generated with Text2Onto are very simple and basically represented as a list of concepts related by IS-A relationships (i.e. concepts organised as a taxonomy, what is sometimes referred to a *lightweight ontology*).

The logic ground problems L1, L10 and L11 cannot trivially occur. Indeed, they may only occur iff the ontology comprises axioms other than axioms for specifying the taxonomy.

L2 to L5 are not applicable because they can be applied only iff intended models are known in some way, which is not the case within the experience.

The remaining logical ground problems L6, L7, L8, L9 and L12 may occur. Indeed L6, L8 and L9 may occur when two inverse taxonomic relationships exist within the ontology (A IS-A B and B IS-A A). L7 trivially occurs because two concepts that are not equivalents are indistinguishable in this ontology. Finally L12 may occur, for instance, if it exists three taxonomic relationships as A IS-A B, B IS-A C and A IS-A C (possible in the ontologies of the experience).

Concerning the social ground problems, all of them may occur. Specifically, some of them trivially occur. Indeed, S5 (Lack or poor textual explanation) trivially occurs because Text2Onto does not provide any annotation as outcome. Finally, Text2Onto outcome is transformed on OWL but the produced version is not necessarily certified (as rules for making the transformation are proprietary): then, S11 trivially occurs.

The problem selection impacts on general checking dependencies (Table 2). Indeed, whenever a dependency comprises a non-selected problem, this dependency needs to be rewritten by deleting the problem. For instance, in this experience, the dependency: L2, L3 → L6, L7, L8, L9, L10, L11, L12

is rewritten as → L6, L7, L8, L9, L10, L11, L12.

In this case, the dependency is no longer considered because its premises are "void".

Techniques for Selected Problem Detection/Warning (Second Step). For logical ground problems L6, L7, L8 and L12, the OWL ontology version and a reasoner (Pellet) have been used:

- Concerning L6 (Logical equivalence of distinct artefacts), the reasoner has been able to identify equivalences between concepts(e.g. area = domain = issue = end = section = object, path = shape);
- Because of the special form of the ontology comprising only concepts and IS-A relationships, detecting L7 has been made possible by counting the pairs of non logically equivalent concepts (checked with L6).
- Concerning L8 and L9 problems, the reasoner has not been able to find any concept equivalent to union or intersection of other concepts. So that, L8 and L9 do not occur
- Concerning L12 (Ontology non minimal), the reasoner has been able to detect some IS-A relationships (original) as inferred other ones.

Apart S5 and S11 problems mentioned above, the other social ground problems have required to develop our own techniques. However, because most of these problems can only be detected if stakeholders/users are directly involved (such as end-users, experts and so on), employed techniques do not guarantee unbiased results.

Through *formal inspection*, S1 (Social contradiction) has been detected by specifically inspecting IS-A relationships and pointing the ones contradicting our own IS-A relationships. S2 (Perception of design errors) has been detected by focusing on the ambiguity/vagueness of the dichotomy concept vs. instance.

S3 (meaningless artefacts) has been raised for concepts labelled with artificial labels (e.g. a label such as "tx12").

S4 (social incompleteness) has been detected as follow: whenever a concept is connected only to the root (so that it has no other relationship with other concepts because ontology is lightweight), the ontology is considered to be incomplete because probably lacking of additional IS-A relationships; this technique only warns about the problem.

S6 (Potentially equivalent artefacts) has been detected as a problem occurring when labels for concepts are synonyms according to our domain knowledge (e.g. area = field, human = person, sheet = plane) or according to known domain references.

S7 (socially indistinguishable artefacts) has been detected whenever it was impossible for a pair of concepts to both provide factual raison to made them equivalent and factual raison to made them distinct.

S8 (polysemy in artefact labels) has been detected by looking to the existence of several definitions, within the given domain, for the single concept label (e.g. labels such as cycle, repair).

S9 has been simply detected by calculating the average depth of the ontology as the average of taxonomy leaf depth, and comparing it to an expected typical depth (found in a manually built ontology based on the same documents).

Table 3. Identified quality problems in O1 and O2.

Types of problems	Detected problems	
	Ontology O1 (441 concepts and 362 is-a relationships)	Ontology O2 (965 concepts and 408 relationships)
L1	Trivially non occurring	Trivially non occurring
L2	NA	NA
L3	NA	NA
L4	NA	NA
L5	NA	NA
L6	276 (= 24*23/2, because we found 24 equivalent concepts) pairs of equivalent concepts (detected on the OWL version)	57 pairs of equivalent concepts (detected on the OWL version)
L7	Trivially occurring; all pairs of concepts that are not equivalent are indistinguishable ((441*440/2)-276 indistinguishable pairs)	Trivially occurring; all pairs of concepts that are not equivalent are indistinguishable ((965 *964/2)-57 indistinguishable pairs)
L8	No "OR artefact"	No "OR artefact"
L9	No "AND artefact"	" No "AND artefact"
L10	Trivially non occurring	Trivially non occurring
L11	The ontology does not contain any situation that can make inferences more complicated	The ontology does not contain any situation that can make inferences more complicated
L12	32 redundant taxonomic relations	49 redundant taxonomic relation
S1	130 taxonomic relations contradict the evaluator's knowledge	60 taxonomic relations contradict the evaluator's knowledge
S2	2 instances were identified as concepts according to evaluator's knowledge	5 instances were identified as concepts according to evaluator's knowledge
S3	13 concepts with meaningless labels according to evaluator's knowledge	21 concepts with meaningless labels according to evaluator's knowledge
S4	168 concepts only connected to root	360 concepts only connected to root
S5	Trivially occurring (not counted)	Trivially occurring (not counted)
S6	9 pairs of concepts with synonymous labels	3 pairs of concepts with synonymous labels
S7	No couple of socially indistinguishable artefacts	No couple of socially indistinguishable artefacts
S8	7 concepts with polysemic labels	9 concepts with polysemic labels
S9	Flat ontology, Average depth of leaves = 2.02, Expected depth = at least 5	Flat ontology, Average depth of leaves = 1.99, Expected depth = at least 7
S10	No: a OWL version is available	No: a OWL version is available
S11	The ontology is not certified	The ontology is not certified
S12	121 useless concepts according to evaluator's knowledge	31 useless concepts according to evaluator's knowledge

Finally, useless artefacts (S12) have been considered as such if it is impossible to provide simple and clear raison for including artefacts in the ontology (for instance, 'train', 'cannot' were trivially out of the ontology domain scope).

Table 3 above summarizes the problems detected, by using deployed techniques, in the two generated ontologies. Next section provides a discussion on experience feedback, mostly based on Table 2.

4.3 Discussion

During the experience, we have remarked the interest, when applicable, of keeping in mind "numbers of occurrences" of a given problem (for instance, S1 can be considered occurring several times as many as ontology artefacts suffer of the problem). Indeed, occurrences are a simple way to highlight differences in the two ontologies, then to identify causes of problems (i.e. defects) and potential correlations between problems. However, not all the problems can be counted: for instance, flatness problem (S9) cannot be counted.

The six most occurring problems are the same for O1 and O2, three are social and three are logical problems: S1, S4, S12, L6, L7 and L12. These problems have been checked involving both concepts and relationships. Occurrences of these problems are quite different in O1 and O2: S1 (O1: 130, O2: 45), S12 (O1: 121, O2: 31), L6(O1: 300, O2: 65), These differences may be quite surprising because numbers of concepts and relations in O1 are lower than in O2. We have therefore tried to provide alternatives non-exclusive explanations. Two explanations have been provided.

One alternative explanation concerns the nature of the content of the incoming textual resource. A technical glossary (starting point for O2) naturally providing definitions of terms, is more self-contained and more focused than a scientific paper (starting point for O1). Indeed, few concept labels in O2 can be considered very generic thus loosely related to the domain while this is not the case for O1. This seems to be confirmed by the fact that S12 (useless artefacts) occurs very often in O1 if compared to O2.

A second non-exclusive alternative explanation is traced back to the usage of Wordnet made by Text2Onto. Indeed, generic and rather useless concepts belonging to O1 enable Text2Onto to also introduce IS-A relationships belonging to Wordnet; these IS-A relationships are due to the several meanings associated by Wordnet to terms (for instance, for term "type" in case of O1, Text2Onto extracted: "type" IS-A "case"; "type" IS-A "group" and "type" IS-A "kind"; each IS_A relation concerns one quite specific and distinct meaning of the term "type"). This is confirmed by much higher occurrences of S1 in O1 than in O2 (remember that S1 has been detected by focusing on IS-A relationships only, see Sect. 4.2).

Occurrences can also be fruitful for establishing *potential correlations* between problems. A problem is potentially correlated to another one if the presence of one problem is potentially due to the presence of another one. However, correlations cannot substitute checking dependencies mentioned in Sect. 3: indeed, checking dependencies are by definition independent from the ontology and the technique used to detect problems while correlations are based on occurrences which are consequences of those

techniques and the ontology. As a consequence, dependencies are stronger than correlations. Correlations should confirm established checking dependencies (Table 2) and provide insights on using those dependencies in forward or backward direction; however, correlations can also be additional to those dependencies.

We have found the following three interesting correlations.

Correlation 1: S12 seems correlated with S1 (raised from results reported Table 3: O1(S12: 121, S1: 130) and O2 (S12: 31, S1:45)). The correlation, as also explained above, can be justified because in the experience useless concepts are often source of incorrect taxonomic relationships; the correlation confirms the (forward) checking dependency between S12 and S1.

Correlation 2: S9 (ontology flatness) shows similar values for the two ontologies (Table 3 reports O1(S9: 2:02) and O2(S9: 1:99)). S9 seems to be correlated to S4: frequency of S4 is mostly the same for the two ontologies, and S9 also occurs o. The correlation can be justified because S4 is checked by counting the concepts only related to the root. If S4 occurs often, in any case, the average depth tends to depend on number of concepts only related to the root. This correlation is additional to the ones in Table 2.

Correlation 3: S12 seems correlated with L6 (raised from results reported in Table 3: O1(S12: 121, L6: 276) and O2 (S12: 31, L6: 57)) because useless artefacts have generated additional logical equivalences; the correlation confirms the (forward) checking dependency between S12 and L6).

4.4 Lessons Learned: Configuring a Validation Process

Discussion above points out that explanations for quality problems can be traced back to the content features of the incoming textual resources (e.g. technical content vs scientific content) and the usage of external resources. It is therefore suggested that to reduce the complexity of the validation process contents of the incoming textual resources should be evaluated and possibly improved before learning the ontology.

Correlations (Sect. 4.3) and dependencies (as in Table 2 but rewritten as explained in Sect. 4.2) between problems can be merged, when not in contradiction. Once merged, an order for configuring (and then running) a validation process can be identified (Fig. 2 above). It should be noted that, according to what has been said in Sect. 3, whenever a problem is removed, if the new resulting ontology is not logically equivalent to the previous ontology, all dependencies need to be reconsidered: as a consequence, the validation process moves to the initial step (status "Checking S3, S12" in Fig. 2 above). This order configuration is reasonable for any other *lightweight ontology* built and validated in the same context of the reported experience i.e. (i) with the same tool (e.g. Text2Onto) with the same tool parameters, (ii) the same typology deployment (same selected problems, same associated techniques, see Sect. 4.2), and (iii) whenever a problem is removed, the resulting ontology remains lightweight (otherwise additional problems need to be selected when typology is deployed).

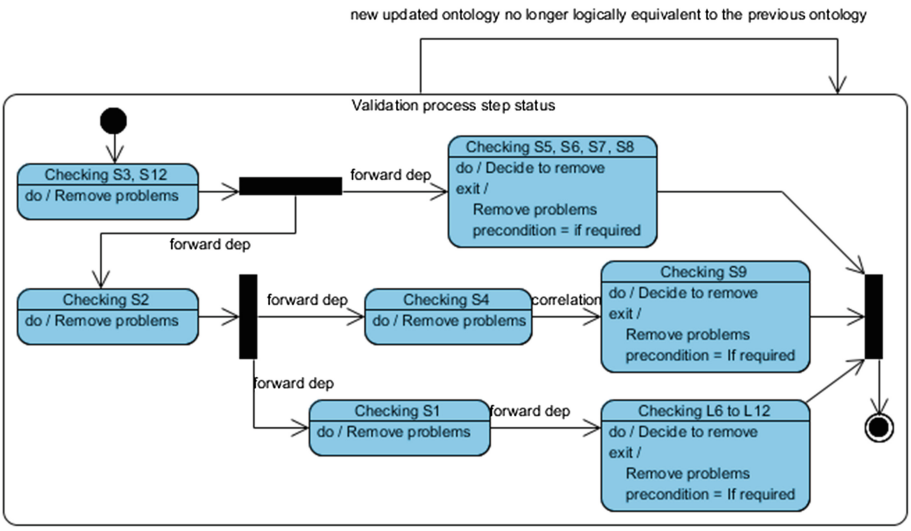

Fig. 2. Identified ordered steps (UML state machine).

5 Conclusions

Through the paper, we have reported a typology of problems impacting the quality of an ontology and introduced checking dependencies for properly detecting and removing problems within a validation process. These dependencies are generally valid and can be reused in any case. Through an experience, we have presented how in practice the typology can be deployed (i.e. problems to be considered and techniques for detecting them). Based on experience results, we have then analysed how a validation process can be configured (i.e. in which order problems need to be checked out and removed). The resulting configuration merges dependencies (independent from the experience) with correlations (dependent from the experience); however, the configuration seems reasonable when other ontologies built with the same tool (Text2Onto) are validated according to the typology deployment made for the experience.

Of course, the experience itself does not cover various important aspects reported below:

- Because ontologies used in the experience are lightweight, typology deployment has only concerned a subset of problems; important problems, especially logic ground errors, are not covered by the deployment; however, specific techniques have been developed for trying to detect most of the logical ground problems; these techniques focus on algorithms for explaining reasoning and supporting users for expressing expected facts; however, some works (through SPARQL queries [10], anti-patterns [21, 29], heuristics [30], tools [23, 30], have undertaken more empirical ways for looking to problems (therefore, often more focusing warning about problems than on detecting occurring problems);

- Deployed techniques for social ground problems are quite simple; several works have investigated techniques that can be associated to social ground problems (for instance [10, 11, 23, 31]). However, some techniques for social ground problems are not clearly confined because problems themselves while well-defined cover a quite large spectre of situations; other social ground problems are even recognised as open issues (such as S10, [32]).

These aspects can be turned into research perspectives for the work targeting ontologies comprising general axioms, dependencies and correlations mixing logic ground and social ground problems, and in the latter case, the preferred usage (forward-backward) of dependencies.

References

1. Cimiano, P., Völker, J.: Text2onto. In: Montoyo, A., Muńoz, R., Métais, E. (eds.) NLDB 2005. LNCS, vol. 3513, pp. 227–238. Springer, Heidelberg (2005)
2. Cimiano, P., Maedche, A., Staab, S., Volker, J.: Ontology learning. In: Studer, R., Staab, S. (eds.) Handbook on ontologies. International Handbooks on Information Systems, pp. 245–267. Springer, Heidelberg (2009)
3. Hirst, G.: Ontology and the lexicon. In: Studer, R., Staab, S. (eds.) Handbook on Ontologies. International Handbooks on Information Systems, 2nd edn, pp. 269–292. Springer, Heidelberg (2009)
4. Simperl, E., Tempich, C.: Exploring the economical aspects of ontology engineering. In: Studer, R., Staab, S. (eds.) Handbook on Ontologies. International Handbooks on Information Systems, 2nd edn, pp. 445–462. Springer, Heidelberg (2009)
5. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquisition **5**(2), 199–220 (1993)
6. Gomez-Perez, A.: Some ideas and examples to evaluate ontologies. In: Proceedings of the 11th Conference on Artificial Intelligence for Applications, pp. 299–305 (1995)
7. Duque-Ramos, A., Fernandez-Breis, J.T., Aussenac-Gilles, N., Stevens, R.: Oquare: asquare based approach for evaluating the quality of ontologies. J. Res. Prac. Inf. Technol. **43**, 159–173 (2011)
8. Gomez-Perez, A.: Ontology evaluation. In: Staab, S., Studer, R. (eds.) Handbook on ontologies. International Handbooks on Information Systems, 1st edn, pp. 251–274. Springer, Heidelberg (2006)
9. Hartmann, J., Spyns, P., Giboin, A., Maynard, D., Cuel, R., Suarez-Figueroa, M.C., Sure, Y.: Methods for ontology evaluation. Technical report. Knowledge Web Deliverable D1.2.3 (2004)
10. Baumeister, J., Seipel, D.: Smelly owls-design anomalies in ontologies. In: Proceedings of 18th International Florida Artificial Intelligence Research Society Conference, pp. 215–220 (2005)
11. Roussey, C., Corcho, O., Blazquez, L.M.V.: A catalogue of owl ontology antipatterns. In: Proceedings of 5th International Conference on Knowledge Capture, pp. 205–206 (2009)
12. Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J.: Modelling ontology evaluation and validation. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 140–154. Springer, Heidelberg (2006)

13. Vrandecic, D.: Ontology evaluation. In: Studer, R., Staab, S. (eds.) Handbook on ontologies. International Handbooks on Information Systems, 2nd edn, pp. 293–314. Springer, Heidelberg (2009)
14. Tartir, S., Arpinar, I.B., Sheth, A.P.: Ontological evaluation and validation. In: Poli, R., Healy, M., Kameas, A. (eds.) Theory and Applications of Ontology: Computer Applications, pp. 115–130. Springer, Netherlands (2010)
15. Gherasim, T., Harzallah, M., Berio, G., Kuntz, P.: Methods and tools for automatic construction of ontologies from textual resources: a framework for comparison and its application. In: Guillet, F., Pinaud, B., Venturini, G., Zighed, D.A. (eds.) Advances in Knowledge Discovery and Management. SCI, vol. 471, pp. 177–201. Springer, Heidelberg (2013)
16. Krogstie, J.: Specialisations of SEQUAL. In: Model-Based Development and Evolution of Information Systems, pp. 281–326. Springer, London (2012)
17. Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O.: Ontological Engineering: With Examples from the Areas of Knowledge Management, e-commerce and the Semantic Web. Advanced Information and Knowledge Processing. Springer, New York (2001)
18. Fahad, M., Qadir, M.A.: A framework for ontology evaluation. In: Proceedings of the 16th International Conference on Conceptual Structures (ICCS 2008), vol. 354, pp. 149–158 (2008)
19. Buhmann, L., Danielczyk, S., Lehmann, J.: D3.4.1 report on relevant automatically detectable modelling errors and problems. Technical report LOD2-Creating Knowledge out of Interlinked Data (2011)
20. Baumeister, J., Seipel, D.: Anomalies in ontologies with rules. Web Seman. Sci. Serv. Agents World Wide Web **8**(1), 55–68 (2010)
21. Corcho, O., Roussey, C., Blazquez, L.M.V.: Catalogue of anti-patterns for formal ontology debugging. In: Atelier construction d'ontologies: vers un guide des bonnes pratiques, afia, pp. 2–12 (2009)
22. Poveda, M., Suarez-Figueroa, M.C., Gomez-Perez, A.: A double classification of commonpitfalls in ontologies. In: Proceedings of the workshop on ontology quality at ekaw, pp. 1–12 (2010)
23. Poveda-Villalón, M., Suárez-Figueroa, M.C., Gómez-Pérez, A.: Validating ontologies with OOPS! In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 267–281. Springer, Heidelberg (2012)
24. Bertossi, L., Hunter, A., Schaub, T.: Inconsistency Tolerance. Springer, Heidelberg (2005)
25. Guarino, N., Oberle, D., Staab, S.: What is an ontology? In: Studer, R., Staab, S. (eds.) Handbook on ontologies. International Handbooks on Information Systems, 2 edn, pp. 1–17. Springer, Heidelberg (2009)
26. Harzallah, M.: Développement des ontologies pour l'interopérabilité des systèmes hétérogènes, applications aux cas industriels du projet ISTA3. Livrable final de la tâche 2.4 du projet ISTA3. Interop-vlab.eu/workspaces/ISTA%203 (2012)
27. Volker, J., Sure, Y.: Data-driven change discovery-evaluation. Technical report. Deliverable D3.3.2for SEKT Project, Institute AIFB, University of Karlsruhe (2006)
28. Gherasim, T., Harzallah, M., Berio, G., Kuntz, P.: Problems impacting the quality of automatically built ontologies. In: Proceedings of the 8th Workshop on Knowledge Engineering and Software Engineering, held in Conjunction with ECAI 2012, pp. 25–32 (2012)
29. Roussey, C., Scharffe, F., Corcho, O., Zamazal, O.: Une méthode de débogage d'ontologies owl basées sur la détection d'anti-patrons. In: Actes de la 21e conférence en ingénierie des connaissances, pp. 43–54 (2010)

30. Pammer, V.: Automatic support for ontology evaluation-review of entailed statements and assertional effects for owl ontologies. Ph.D. thesis, Graz University of Technology (2010)
31. Burton-Jones, A., Storey, V., Sugumaran, V.: A semiotic metrics suite for assessing the quality of ontologies. Data Knowl. Eng. **55**(1), 84–102 (2005)
32. Kalfoglou, Y.: Cases on semantic interoperability for information systems integration: Practices and applications. IGI Global, Hershey (2010)