Pitfalls in Ontologies and TIPS to Prevent Them

C. Maria Keet[1(✉)], Mari Carmen Suárez-Figueroa[2], and María Poveda-Villalón[2]

[1] School of Mathematics, Statistics, and Computer Science, UKZN/CSIR-Meraka Centre for Artificial Intelligence Research, University of KwaZulu-Natal, Durban, South Africa
keet@ukzn.ac.za

[2] Ontology Engineering Group, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain
{mcsuarez,mpoveda}@fi.upm.es

Abstract. A growing number of ontologies are already available thanks to development initiatives in many different fields. In such ontology developments, developers must tackle a wide range of difficulties and handicaps, which can result in the appearance of anomalies in the resulting ontologies. Therefore, ontology evaluation plays a key role in ontology development. OOPS! is an on-line tool that automatically detects pitfalls, considered as potential errors or problems—and thus may help ontology developers to improve their ontologies. To gain insight in the existence of pitfalls and to assess whether there are differences among ontologies developed by novices, a random set of already scanned ontologies, and existing well-known ones, data of 406 OWL ontologies were analysed on OOPS!'s 21 pitfalls, of which 24 ontologies were also examined manually on the detected pitfalls. The various analyses performed show only minor differences between the three sets of ontologies, therewith providing a general landscape of pitfalls in ontologies. We also propose guidelines to avoid the inclusion of such common pitfalls in new ontologies, the Typical pItfalls Prevention Scheme (TIPS), so as to increase the baseline quality of OWL ontologies.

1 Introduction

A growing number of ontologies are already available in different domains thanks to ontology development initiatives and projects. However, the development of ontologies is not trivial. Early ontology authoring suggestions were made by [1], and Rector et al. [2] present the most common problems, errors, and misconceptions of understanding OWL DL based on their experiences teaching OWL. OWL 2 DL contains more features and there is a much wider uptake of ontology development by a more diverse group of modellers since. This situation increases the need for training, for converting past mistakes into useful knowledge for ontology authoring, to prevent common flaws and it requires a clear notion of ontology quality both in the negative sense (what are the mistakes?) and in the positive (when is some representation good?). Several steps have been taken with respect

A. Fred et al. (Eds.): IC3K 2013, CCIS 454, pp. 115–131, 2015.
DOI: 10.1007/978-3-662-46549-3_8

to quality in the negative sense, such as to identify antipatterns [3] and to create a catalogue of common pitfalls—understood as potential errors, modelling flaws, and missing good-practices in ontology development—in OWL ontologies [4,5], and in the positive sense by defining good and 'safe' object property expressions [6] and taxonomies [7]. The catalogue of common pitfalls included 29 types of pitfalls at the time of evaluation and 21 of them are detected automatically by the online OntOlogy Pitfall Scanner! (OOPS! http://www.oeg-upm.net/oops). With the automation of scanning pitfalls as well as advances in ontology metrics, this now provides the opportunity to obtain quantitative results, which has been identified as a gap in the understanding of ontology quality before [8]. Here, we are interested in answering two general questions, being:

A. What is the prevalence of each of those pitfalls in existing ontologies?
B. To what extent do the pitfalls say something about quality of an ontology?

The second question can be broken down into several more detailed questions and hypotheses, which one will be able to answer and validate or falsify through a predominantly quantitative analysis of the ontologies:

1. Which anomalies that appear in OWL ontologies are the most common?
2. Are the ontologies developed by experienced developers and/or well-known or mature ontologies 'better' in some modelling quality sense than the ontologies developed by novices? This is refined into the following hypotheses:
   (i) The prevalence and average of pitfalls is significantly higher in ontologies developed by novices compared to ontologies deemed established/mature.
   (ii) The kind of pitfalls observed in novices' ontologies differs significantly from those in well-known or mature ontologies.
   (iii) The statistics on observed pitfalls of a random set of ontologies is closer to those of novices' ontologies than the well-known or mature ones.
   (iv) There exists a positive correlation between the detected pitfalls and the size or number of particular elements of the ontology.
   (v) There exists a positive correlation between the detected pitfalls and the DL fragment of the OWL ontology.

To answer these questions, we used the 362 ontologies scanned by OOPS! over the past year, 23 novices ontologies, and 21 ontologies that are generally considered to be well-known, where the latter two sets were also scanned by OOPS! and evaluated manually. Although all 21 types of pitfalls have been detected, the most common pitfalls concern lack of annotations and domain and range axioms, and issues with inverses, and to some extent creating unconnected ontology elements and using a recursive definition. The results falsify hypotheses (i), (ii), and (v), partially validate (iv)—for novices, the number of pitfalls/ontology does relate to the size and complexity of the ontology—and validate (iii); i.e., there are no striking differences between the three sets of ontologies, therewith providing a general landscape of pitfalls in ontologies. Taking the pitfall results into account, we propose the **T**ypical p**I**tfall **P**revention **S**cheme, TIPS. The TIPS differ from earlier suggestions [1,2], as the suggestions are applicable to OWL 2 instead of

its predecessor languages, they contain an order of importance, and the TIPS embeds emphases with respect to occurrence of the pitfall so that common pitfalls can be prevented first compare to treating any possible pitfall as equally relevant in the overall ontology authoring activity.

In the remainder of this paper, we describe the state of the art in Sect. 2, report on the experimental evaluation of the ontologies in Sect. 3, present the proposed TIPS (Typical pItfalls Prevention Scheme) in Sect. 4 and conclude in Sect. 5.

2 State of the Art

When developing ontologies, developers must tackle a wide range of difficulties, which are related to the inclusion of anomalies in the modelling. Thus, ontology evaluation, which checks the technical quality of an ontology against a frame of reference, plays a key role when developing ontologies. To help developers during the ontology modelling, early ontology authoring guidelines to avoid typical errors were provided in [1]. Such guidelines help developers to prevent errors related to the definition of classes, class hierarchies, and properties during frame-based ontology developments, but they are becoming outdated due to increased expressiveness of ontology languages for which guidance is needed, and advances in ontology authoring guidelines have been made over the past 13 years. Rector and colleagues [2] help with the precise meaning of OWL DL and provide some guidelines on how to avoid diverse pitfalls when building OWL DL ontologies. These pitfalls were mainly related to (a) the failure to make information explicit, (b) the mistaken use of universal and existential restrictions, (c) the open world reasoning, and (d) the effects of domain and range constraints. A classification of errors was identified during the evaluation of consistency, completeness, and conciseness of ontology taxonomies [9]. First steps towards a catalogue of common pitfalls started in 2009 [4] leading to a first stable version in [10]. This catalogue is being maintained and is accessible on-line as part of the OOPS! portal. OOPS! [5] is a web-based tool for detecting potential pitfalls, currently providing mechanisms to automatically detect a subset of 21 pitfalls of those included in the catalogue and therewith helping developers during the ontology validation activity. Related to the aforementioned catalogue of pitfalls, is the identification of a set of antipatterns [3]. Theory-based methods to help developers to increase ontology quality include defining good and 'safe' object property expressions [6] and ontologically sound taxonomies [11]. To help developers during the ontology evaluation activity, there are different approaches: (a) comparison of the ontology to a "gold standard", (b) use of the ontology in an application and evaluation of the results, (c) comparison of the ontology with a source of data about the domain to be covered, and (d) evaluation by human experts who assess how the ontology meets the requirements [12]. A summary of generic guidelines and specific techniques for ontology evaluation can be found in [13]. A three-layered approach to ontology evaluation is presented in [14]: (1) O2 (a meta-ontology), (2) oQual (a pattern based on O2 for Ontology

Quality), and qood (for Quality-Oriented Ontology Description). This allows one to measure the quality of an ontology relative to structural, functional, and usability-related dimensions. A compendium of criteria describing good ontologies is reported in [8] (including accuracy, adaptability, clarity, completeness, computational efficiency, conciseness, consistency/coherence and organizational fitness) and it presents a review of domain and task-independent evaluation methods related to vocabulary, syntax, structure, semantics, representation and context aspects.

A separate strand of suggestions for good practices of representations within Semantic Web are for data and SKOS [15,16] that may complement good practices for ontology development once those connections are better established.

To the best of our knowledge, what is missing at present in the ontology and evaluation field is a quantitative analysis of the most common pitfalls developers include in the ontologies. Based on this study, one then may create a relevant set of guidelines to help developers in the task of developing ontologies and refine ontology quality criteria.

3 Experimental Evaluation of Pitfalls in Ontologies

The experimental evaluation is described in the standard order in this section: first, materials & methods regarding data collection and analysis, then the quantitative and qualitative results, and, finally, the discussion of the results.

3.1 Materials and Methods

Data Collection. With the aim of identifying the most common pitfalls typically made when developing ontologies in different contexts and domains, we have collected and analyzed 44 ontologies (Set1 and Set2) and used the data stored in OOPS! for a random set (Set3):

Set1: 23 ontologies in different domains (a.o., furniture, tennis, bakery, cars, soccer, poker, birds, and plants) developed by novices. These ontologies were developed as a practical assignment by Computer Science honours (4th year) students attending the course "Ontologies & Knowledge bases (OKB718)" in 2011 and 2012 at the University of KwaZulu-Natal.

Set2: 21 existing well-known ontologies that may be deemed 'mature' in the sense of being a stable release, well-known, a real OWL ontology (i.e., no toy ontology nor a tutorial ontology, nor an automated thesaurus-to-OWL file), the ontology is used in multiple projects including in ontology-driven information systems, and whose developers have ample experiences in and knowledge of ontologies, and the selected ontologies are in different subject domains; a.o., DOLCE, BioTop, and GoodRelations.

Set3: 362 ontologies analyzed with OOPS! They were selected from the 614 times that ontologies were submitted between 14-11-2011 and 19-10-2012. The full set was filtered as follows: maintain those repeated ontologies for

which OOPS! obtained different results in each evaluation, eliminate those repeated ontologies for which OOPS! obtained the same results in every evaluation, and eliminate those ontologies whose namespace is deferenceable but it does not refer to an ontology.

OOPS! output for the three sets, including calculations, manual analyses of OOPS! detected pitfalls for ontologies in Set1 and Set2, and the names and URIs of the ontologies of Set2 and the names of the ontologies in Set1, are available at http://www.oeg-upm.net/oops/material/KEOD2013/pitfallsAnalysis.xlsx.

All ontologies are evaluated by being scanned through OOPS!, which checks the ontology on most pitfalls that have been collected in the pitfall catalogue that has been presented and discussed in earlier works [4,5] and are taken at face value for this first quantitative evaluation: Creating synonyms as classes (P2); Creating the relationship "is" instead of using `rdfs:subClassOf`, `rdf:type` or `owl:sameAs` (P3); Creating unconnected ontology elements (P4); Defining wrong inverse relationships (P5); Including cycles in the hierarchy (P6); Merging different concepts in the same class (P7); Missing annotations (P8); Missing disjointness (P10); Missing domain or range in properties (P11); Missing equivalent properties (P12); Missing inverse relationships (P13); Swapping intersection and union (P19); Misusing ontology annotations (P20); Using a miscellaneous class (P21); Using different naming criteria in the ontology (P22); Using recursive definition (P24); Defining a relationship inverse to itself (P25); Defining inverse relationships for a symmetric one (P26); Defining wrong equivalent relationships (P27); Defining wrong symmetric relationships (P28); and Defining wrong transitive relationships (P29). Detailed descriptions are available online from the pitfall catalogue at http://www.oeg-upm.net/oops/catalogue.jsp. Note that OOPS! analyses also properly imported OWL ontologies, i.e., when they are available and dereferencable online at the URI specified in the import axiom.

In addition, we collected from the ontologies of Set1 and Set2: DL sublanguage as detected in Protégé 4.1, number of classes, object and data properties, individuals, subclass and equivalence axioms.

Analyses. The data was analysed by computing the following aggregates and statistics. The basic aggregates for the three sets are: (a) percentage of the incidence of a pitfall; (b) comparison of the percentages of incidence of a pitfall among the three sets; (c) average, median, and standard deviation of the pitfalls per ontology and compared among the three sets; and (d) average, median, and standard deviation of the pitfall/ontology.

For Set1 and Set2 ontologies, additional characteristics were calculated, similar to some of the ontology metrics proposed elsewhere [8,14]. Let $|C|$ denote the number of classes, $|OP|$ the number of object properties, $|DP|$ the number of data properties, $|I|$ the number of individuals, $|Sax|$ the number of subclass axioms, and $|Eax|$ the number of equivalences in an ontology. The number of *Ontology Elements* (OE) is computed by Eq. 1, and an approximation of the *Ontology Size* (OS) by Eq. 2.

$$OE = |C| + |OP| + |DP| + |I| \tag{1}$$

$$OS = |C| + |OP| + |DP| + |I| + |Sax| + |Eax| \quad (2)$$

We use two measures for quantifying the 'complexity' of the ontology. First, an *Indirect Modelling Complexity* (IMC) is computed based on the axioms present (Eq. 3), where a lower value indicates a more complex ontology with relatively more axioms declaring properties of the classes compared to a lightweight ontology or bare taxonomy.

$$IMC = |C| : (|Sax| + |Eax|) \quad (3)$$

Second, the OWL features used are analysed twofold: (i) by calculating the overall percentage of use of $\mathcal{S}$, $\mathcal{R}$, $\mathcal{O}$, $\mathcal{I}$, $\mathcal{Q}$ and (D), i.e., a rough measure of the OWL 2 DL features used; (ii) by converting the DL fragment into a numerical value, where $\mathcal{AL}$ is given the lowest value of 0 and $\mathcal{SROIQ}$ the highest value of 10, to be used in correlation calculations (see below). The DL fragment and IMC are compared as well, for they need not be similar (e.g., a bare taxonomy with one object property declared reflexive already 'merits' detection of an $\mathcal{R}$, but actually is still a simple ontology with respect to the subject domain represented, and, vv., an ontology can be comprehensive with respect to the subject domain, but originally developed in OWL DL but not updated since OWL 2).

Basic correlations are computed for the ontology sizes and complexities with respect to the pitfalls, and detailed correlations are computed for certain individual pitfalls: P5, P11, P13, P25, P26, P27, P28, and P29 are pitfalls specific to object properties, hence, the amount of properties in the ontologies may be correlated to the amount of pitfalls detected, and likewise for P3, P6, P7, P10, P21, and P24 for classes, and P8 for classes and ontology elements.

Finally, manual qualitative analyses with ontologies in Set1 and Set2 were conducted on possible false positives and additional pitfalls.

3.2 Results

We first present the calculations and statistics, and subsequently a representative selection of the qualitative evaluation of the ontologies in Set1 and Set2.

Aggregated and Analysed Data. The raw data of the ontologies evaluated with OOPS! are available online at http://www.oeg-upm.net/oops/material/KEOD2013/pitfallsAnalysis.xlsx. The type of mistakes made by novice ontology developers are: P4, P5, P7, P8, P10, P11, P13, P19, P22, P24, P25, P26, P27, and P29. The percentages of occurrence of a pitfall over the total set of 23 ontologies in Set1 is included in Fig. 1, the average amount of pitfalls is shown in Fig. 2, and aggregate data also with minimum, maximum, median and standard deviation is listed in Table 1. The analogous results for Set3 are shown in Figs. 1 and 2, and in Table 1, noting that all OOPS! pitfalls have been detected in Set3 and that the median amount of pitfalls/ontology is similar to that of Set1. The high aggregate values are caused by a few ontologies each with around 5000 or more detected pitfalls; without P8 (missing annotations), there are three ontologies

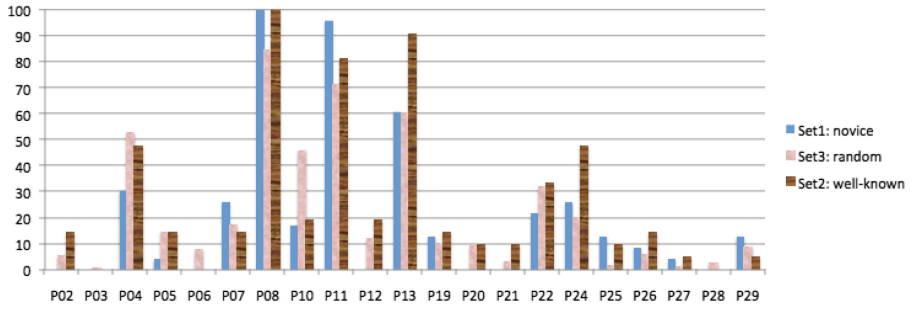

Fig. 1. Percentage of occurrence of a pitfall in the three sets of ontologies.

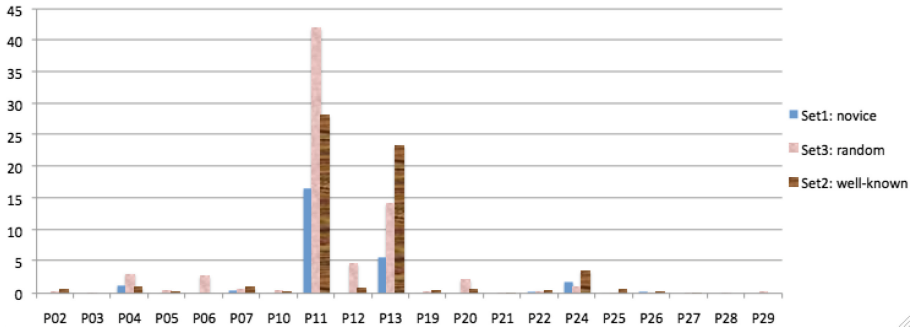

Fig. 2. Average number of pitfall/ontology, by set; for P8, the averages are 62, 297, and 303, respectively.

that have more than 1000 detected pitfalls at the time of scanning the ontology. The results obtained with the 21 well-known ontologies (Set2) can be found in the same table and figures, and include pitfalls P2, P4, P5, P7, P8, P10, P11, P12, P13, P19, P20 (0 upon manual assessment), P21, P22, P24, P25, P26, P27, and P29, noting that the percentages and averages differ little from those of the novices and random ones. The high aggregate values for Set2 is largely due to OBI with a pitfall count of 3771 for P8 (annotations) and DMOP with a pitfall count of 866 for P8; without P8, OBI, DMOP, and the Government Ontology exceeded 100 pitfalls due to P11 (missing domain and range axioms) and P13 (missing inverses—but see also below). P8 is an outlier both in prevalence and in quantity for all three sets of ontologies and only some of the ontologies have very many missing annotations, which skews the average, as can be observed from the large standard deviations.

For Set1 and Set2, we collected data about the content of the ontologies and analysed them against the pitfalls, as described in Sect. 3.1. The usage of the OWL 2 DL features in Set1 are: $\mathcal{S}$ 44 %, $\mathcal{R}$ 26 %, $\mathcal{I}$ 83 %, $\mathcal{O}$ 26 %, $\mathcal{Q}$ 52 %, and D 17 %, whereas for Set2, the percentages are 62 %, 19 %, 81 %, 24 %, 5 %, and 86 %, respectively; the difference is largely due to the difference in timing of

Table 1. Totals for the three sets of ontologies (rounded off), with and without the annotation pitfall (P8).

Ontology	Set1: Novices		Set3: Random		Set2: Well-known		Combined	
Pitfalls	All	All – P8	All	All – P8	All	All – P8	All	All – P8
Total	2046	626	133746	26330	7639	1277	143436	28238
Minimum	23	3	0	0	15	2	0	0
Maximum	366	95	7948	1999	3920	207	7948	1999
Average	89	27	735	145	364	61	353	70
Median	65	19	50	14	137	48	54	16
St. dev.	74	26	1147	244	846	53	1101	231

Table 2. Correlations and p-values for specific pitfalls and ontology size and complexity, with the relatively interesting values in boldface; where $p < 0.0001$, only 0 is written in the cell due to width limitations.

Set	Set1: Novices				Set2: Well-known				Both			
	All		All – P8		All		All – P8		All		All – P8	
pitfalls/onto.	Corr.	p	Corr.	p	Corr.	p	Corr.	p	Corr.	p	Corr.	p
DL fragment	0.33	**0**	0.18	**0.0002**	0.49	0.066	0.52	**0**	0.38	**0.020**	0.38	**0**
IMC	0.06	**0**	−0.14	**0**	−0.21	0.056	−0.36	**0**	−0.14	**0.017**	−0.2	**0**
OE	**0.998**	0.47	0.70	**0.0003**	**0.993**	0.84	0.57	0.068	**0.990**	0.79	0.58	**0.025**
OS	0.58	**0.0072**	0.67	**0**	**0.998**	0.34	0.52	0.10	**0.995**	0.24	0.52	**0.044**

the development of the ontology, with some of the well-known ontologies having been developed before the OWL2 standard, and the use of data properties was discouraged in the lectures for the ontologies in Set1. In order to include the DL fragment in the analyses, we assigned values to the fragments prior to analysis, ranging from a value of 0 for an ontology in $\mathcal{AL}(D)$ to 10 for an ontology in $\mathcal{SROIQ}(D)$, and intermediate values for others (e.g., $\mathcal{ALCHI}(D)$ with a value 3 and $\mathcal{SHIF}$ with value 6—see supplementary data). With the calculated IMC (recall Eq. 3), the correlation between DL fragment and the IMC is −0.18 for the ontologies in Set1 and −0.74 for the ontologies in Set2. This possibly may change a little by tweaking the values assigned to the DL fragments, but not such as to obtain a strong, meaningful correlation between detected DL fragment and the IMC.

Correlations for several measures are included in Table 2. The only substantial correlations found are between all pitfalls per ontology elements and size (in boldface), although with all pitfalls minus P8, there is no obvious correlation anymore. p-values were computed with the 1-tailed unpaired Student t-test, which are also included in Table 2. Using a generous $p < 0.05$ for no difference between the number of pitfalls per ontology and DL fragment, IMC, OE, or OS as measures for ontology size, then the hypotheses have to be rejected mainly for novices (boldface in Table 2). Correlations were also computed for certain pitfalls and relevant ontology elements, as shown in Table 3; e.g., P5 is

Table 3. Correlations by pitfalls and ontology metric, with the most interesting values in boldface, and potentials in italics; "–": no pitfall detected, hence, no correlation.

Ontology correlation	Set1	Set2	Both
P5 (wrong inverses) – number of object properties	*0.71*	0.52	0.58
P11 (missing domain/range) – number of object properties	0.41	0.34	0.40
P13 (missing inverses) – number of object properties	0.54	*0.78*	*0.77*
P25 (inverse to itself) – number of object properties	0.36	0.30	0.32
P26 (inverse for symmetric) – number of object properties	*0.72*	−0.25	0.25
P27 (wrong equivalence) – number of object properties	*0.71*	0.61	0.59
P28 (wrong symmetric) – number of object properties	–	–	–
P29 (wrong transitive) – number of object properties	−0.20	0.15	−0.03
P3 (adding is-a) – number of classes	–	–	–
P6 (cycles) – number of classes	–	–	–
P7 (merging classes) – number of classes	0.17	0.04	0.06
P10 (missing disjointness) – number of classes	−0.08	−0.13	−0.09
P21 (miscellaneous classes) – number of classes	–	−0.10	−0.06
P24 (recursive definition) – number of classes	0.15	−0.04	0.01
P8 (missing annotation) – number of classes	0.22	**0.9975**	**0.9909**
P8 (missing annotation) – number of ontology elements	0.51	**0.9899**	**0.9848**

about inverse relationships, hence, one might conjecture it is correlated with the amount of object properties in the ontology. This only holds strongly for P8 and the elements in the Set2 ontologies, which explains why there are significant correlations for all pitfalls but not all minus P8 in Table 2. A weakly possibly interesting correlation exists for P5, P26, P27 in the Set1 ontologies, and for P13 in the well-known ontologies.

Comparing pitfalls among Set1, Set2, and Set3 with the 1-tailed unpaired Student t-test, then the null hypothesis—no difference—has to be rejected for novice vs. mature if one ignores pitfall P8 ($p = 0.0096$), i.e., one can observe a difference, but this does not hold anymore for all pitfalls ($p = 0.13$). The results are inconclusive for the other combinations: all pitfalls novice vs. random $p = 0.15$, all mature vs. random $p = 0.98$, all minus P8 novice vs. random $p = 0.37$, and all minus P8 mature vs. random $p = 0.82$.

Qualitative Analysis of the Detected Pitfalls. As the pitfalls in the catalogue (and thus OOPS!) are relatively coarse-grained, we examined the OOPS!-detected pitfalls of the ontologies in Set1 and Set2 on the correctness of detection. That is, although the algorithms in OOPS! are implemented correctly, they may detect more pitfalls than what an ontology developer may see as a problem, and such insights may, on the one hand, help refining a pitfall and, on the other

hand, downgrade a pitfall to being irrelevant practically. Of the analyses carried out (included in the supplementary data file), we highlight four types of pitfalls that illustrate well an aspect of ontology development practices (P4), subject domain peculiarities (P7), language features (P13), and modelling (P24).

P4: Unconnected Ontology Elements. OOPS! correctly identifies 'orphan' classes and properties, but they are debatable in some cases. For instance, an orphan's subclasses are used in a class expression, i.e., the orphan class is used merely as a way of grouping similar things alike a so-called 'abstract class' in UML. The Deprecated and Obsolete orphans are typically present in bio-ontologies, which is deemed a feature in that field. A recurring type of orphan class was to add a class directly subsumed by `owl:Thing` to indicate the subject domain (e.g., a Bakery class for an ontology about bakery things), which might be defensible in a distributed ontology, but not in a single domain ontology. Overall, each of these practices require a more substantive argument whether they deserve to be a false positive or not.

P7: Merging Different Concepts in the Same Class. OOPS! detects a few occurrences that are false positives, besides the many correctly identified ones. For instance, a RumAndRaisinFlavour of ice cream does not constitute merging different classes, but a composite flavour and would not have been a false positive if that flavour had obtained its own name (e.g., RummyRaisin). From a computational perspective, there is no easy way to detect these false positives.

P13: Missing Inverse Relationships. The issues with inverses are contentious and hard to detect, especially since OWL and OWL 2 differ in their fundamental approach. Unlike OWL, OWL 2 has a feature `ObjectInverseOf`, so that for some object property hasOP in an OWL 2 ontology, one does not have to extend the vocabulary with an OPof property and declare it as the inverse of hasOP with `InverseObjectProperties`, but instead one can use the meaning of OPof with the axiom `ObjectInverseOf`(hasOP). In addition, GFO's exists_at and BioTop's abstractlyRelatedTo do not readily have an inverse name, and a modeller likely will not introduce a new property for the sake of having a named inverse property when it is not needed in a class axiom. Overall, P13 is detected more often than warranted from a modeller's viewpoint, and it could be refined to only those cases where the declaration of `InverseObjectProperties` is missing; e.g., both manufacturedBy and hasManufacturer are in the car ontology, but they are not declared inverse though they clearly are, which OOPS! detects already.

P24: Using Recursive Definition. This pitfall is tricky to define and detect. In general, recursive definitions are wrong, such as the pattern $\mathsf{X} \equiv \mathsf{X} \sqcap \mathsf{R.Y}$, which should be detected, and likewise detecting unintended assertions, such as CarrotFilling $\sqsubseteq \exists$ hasFillingsAndToppimg.CarrotFilling (in the bakery (novice's) ontology). However, P24 currently detects whether the class on the left-hand side of the subsumption or equivalence occurs also on the right-hand side, which is not always a problem; e.g., DM-Process $\sqsubseteq \exists$hassubprocess.DM-Process in DMOP is fine. These subtle differences are difficult to detect automatically, and require manual inspection before changing or ignoring the pitfall.

Removal of the false positives reduces the observed minor differences between the three sets of ontologies, i.e., roughly equalize the percentages per pitfall. Put differently, this supports the observation that there is a *general landscape of pitfalls*.

New and More Detailed Pitfalls. The novices' ontologies had been analysed manually on modelling mistakes before OOPS! and before consulting the catalogue. In addition to detecting the kind of pitfalls already in the catalogue, new ones were detected, which typically occurred in more than one ontology. We refer to them here as new *candidate pitfalls* (Cs) which are currently being added to the catalogue:

C1. *Including some form of negation in ontology element names.* For example, DrugAbusePrevention (discussed in [17]), and NotAdults or ImpossibleHand (in the poker ontology). This pitfall refers to an anomaly in the element naming.
C2. *Distinguishing between file name and URI.* This is related to naming issues where the `.owl` file has a meaningful name, but the ontology URI has a different name (also observed in [18]).
C3. *Confusing part-of relation with subclass-of relation.* This pitfall is a special and very common case of pitfall P23 (using incorrectly ontology elements) (see [19]). As part of this pitfall, there is also the case in which the most appropriate part-whole relation in general is not selected (see also [20]).
C4. *Misusing* min 1 *and* some. This pitfall affects especially ontology feature usage due to the OWL restrictions (note: Protégé 4.x already includes a feature to change all such instances).
C5. *Embedding possibility/modality in the ontology element's name.* This pitfall refers to encapsulating a modality ("can", "may", "should") in an element's name (e.g., canCook).

3.3 Discussion

Whilst giving valuable insight in the prevalence of pitfalls in existing ontologies, the results obtained falsify hypotheses (i) (except for novice vs. mature when discounting P8), (ii), and (v), partially validate (iv) (for all pitfalls and mature ontologies), and validate (iii), which is not exactly as one may have expected, and it raises several possible interpretations.

First, the set of pitfalls currently implemented in OOPS! is limited and with more and more refined checks, substantial differences may be found. Perhaps this is the case, but it does not negate the fact that it is not the case for the 21 already examined and therefore not likely once extended. In addition, recently, the notion of good and safe object property expressions has been introduced [6], where manual evaluation with a random set of ontologies—including some of the ones in Set2—revealed advanced modelling issues concerning basic and complex object property expressions. This further supports the notion that, for the time being, there is a general landscape compared to saliant differences among levels of maturity.

Second, the well-known ontologies are possibly not really mature and exemplary after all (the converse—that the novices' ontologies in Set1 are 'as good as the well-known ones'—certainly does not hold), for they are quite close to the ones in Set3; i.e., that some ontology is widely known does not imply it is 'good'—or, at least: has fewer pitfalls than an—ontology being developed by a novice ontologist. This makes it more difficult to use them in ontology engineering courses, where one would like to point students to 'good' or 'exemplary' ontologies: if well-known ontologies have those pitfalls, they are more likely to be propagated by the students "because ontology x does it that way". This attitude was observed among the novices with respect to P11, because the popular Protégé OWL Pizza tutorial (http://www.co-ode.org) advises against declaring domain and range of properties (page 37), which may explain why P11 was detected often.

Third, it may be reasonable to argue that 'maturity' cannot be characterised by absence of pitfalls at all, but instead is defined by something else. Such a 'something else' may include its usefulness for its purpose—or at least meeting the requirements—or, more abstract, the precision and coverage as introduced by Guarino (see also Fig. 2 in [11]). Concerning the latter, this means both a high precision and maximum coverage of the subject domain one aims to represent in the ontology. It is known one can improve on one's low precision—i.e., the ontology admits more models than it should—by using a more expressive language and adding more class expressions, and recently the approach of letting users choose among possible models is used to add more disjointness axioms [21] and in ontology-driven conceptual modelling [22]. For domain ontologies, another option that influences to notion of being well-known and mature is its linking to a foundational ontology and that therewith less modelling issues occur [18], but this has to do with the knowledge that is represented, not with, e.g., language feature misunderstandings. We leave a more detailed investigation in this direction for future works.

4 Ontology Authoring Guidelines: TIPS

Given the pervasiveness of the pitfalls regardless the level of maturity and usage of an ontology, there is a need to increase the average quality of ontologies at least to the extent that pitfalls are avoided upfront or fixed once detected. While one can take the same approach here as with using an automated reasoner—model first, then check the deductions—and design one's ontology first and then test with OOPS!, from an academic viewpoint, prevention is better, because it means the modeller has a better grasp of ontology development.

The easiest way is to devise a checklist by simply turning around the pitfalls. The pitfall catalogue is a *list*, however, and it is possible categorise the pitfall into different groups; e.g., grouping pitfalls by a structural, functional, and usability profiling dimension or by 'consistency' (not necessarily resulting in logical inconsistencies), completeness, and conciseness [5,10]. Importantly for useful guidelines, is to recognise that first structural and modelling issues—like

P15 and P29—have to be addressed before making an ontology 'neat' with a consistently applied naming pattern or evaluating the somewhat vague notion of meeting the requirements. In addition, in a similar fashion to a "conceptual schema design procedure" [23] or some other ordering of activities in formalization of the subject domain (e.g., DiDOn [24]), one can structure the guidelines accordingly; here, we choose the order of addressing pitfalls as: classes, taxonomy, properties, constraints, documentation. Taking these notions into account, we propose the **T**ypical p**I**tfall **P**revention **S**cheme, TIPS. While there are indeed earlier suggestions [1,2], these TIPS are applicable to the latest OWL 2, contain an order of importance, and embeds emphases with respect to occurrence of the pitfall so that common pitfalls can be prevented first. The descriptions of the tips are written in the imperative indicating what a developer should be checking.

T1: Class Naming and Identification (includes P1, P2, P7, C2, and C5): When identifying and naming classes in ontologies, avoid synonymy and polysemy: distinguish the concept itself from the different names such a concept can have (the synonyms) and create just one class for the concept and provide, if needed, different names for such a class using `rdfs:label` annotations. Regarding polysemy, where the same name has different meanings, try to disambiguate the term, use extension mechanisms and/or axioms. Other important cases regarding class naming and identification are (a) creating a class whose name refers to two or more different concepts by including "and" or "or" in the name (e.g., StyleAndPeriod or ProductOrService) and (b) using modality ("can", "may", "should") in the ontology element's names. In situation (a) consider dividing the class into different subclasses, and in case (b) consider a more appropriate name avoiding the use of modality or change to a logic language that can express it. Take care about providing proper names for both the ontology file and the URI.

T2: Class Hierarchy (includes P3, P6, P17, and P21): The class taxonomy is based on is-a relations, where a class A is a subclass of class B, if and only if every instance of A is also instance of B, and the is-a in the hierarchy is transitive. So, do not introduce such a relation as an object property, but use primitives provided by the ontology language: e.g., `subclassOf` for representing the subclass of relationship, and `instanceOf` for representing membership of an individual in a class. Another issue when creating class taxonomies is to avoid cycles in the hierarchy, i.e., to avoid defining a class as a specialization or generalization of itself either directly or indirectly, because if a cycle is included between two classes in a taxonomy, the implication is like defining such classes and the ones involved in the cycle as equivalent. Also avoid the temptation of creating a class named Unknown, Other or Miscellaneous in a class hierarchy just because the set of sibling classes defined is incomplete. Finally, consider the leaf elements of the hierarchy, and ask yourself whether they are still classes (entities that can have instances) or individuals (entities that cannot be instantiated anymore), if the latter, then convert them into instances.

T3: Domain and Range (includes P11 and P18): When creating an object or data property, answer the question "What is the most general class in the ontology for which this property holds?" and set the answer as domain of the property. If the answer is `owl:Thing` consider using several subclasses joined by "or" operators (i.e., `owl:unionOf`). Repeat the process to set the range for object properties. For a data property's range, answer the question "What would be the format of data (strings of characters, positive numbers, dates, floats, etc.) used to fill in this information?".

T4: Equivalent Relations (includes P12 and P27): Is any object or data property declared equivalent to another one? To have 'safe' equivalent properties that will not generate unexpected deductions, assess the following: the domains of both properties should be the same class (e.g., Country) and also the ranges should be the same class (e.g., LanguageCode) or datatype. Also check that both properties refer to the same meaning (e.g., hasLanguageCode and has_language_code) between classes or to the same attributes in case of datatype properties. Finally, check if both relations are really needed: if they are defined in the same namespace they could be either redundant or refer to different real-world relations and require disambiguation instead.

T5: Inverse Relations (includes P5, P13, P25, and P26): If there is an object property declared inverse to another one, then, to avoid unexpected deductions, be certain to check that the domain class of one is the same class as the range of the other one, and vv. (e.g., Country and LanguageCode for hasLanguageCode and isCodeOf) and try to create sentences from domain to range using the property name for both (potential) inverse properties to double-check it is possible. Note that this TIPS applies for object property pairs; if only a single object property is involved, consider T6.

T6: Object Property Characteristics (includes P28 and P29): Go through the object properties and check their characteristics, such as symmetry, functional, and transitivity. To have 'safe' object property characteristics declared that will not have unexpected deductions, examine: for transitive properties, the domain and range of the object property should be the same class (e.g., both Process) so that a chain can be formed; for symmetric relations, verbalise the assertion both from domain to range and from range to domain to double-check it is possible and has no hidden second property; reflexivity: if the relation holds for all objects in your ontology, declare it reflexive, if only for a particular relation, then use the `Self` construct.

T7: Intended Formalization (includes P14, P15, P16, P19, C1, and C4): A property's domain (resp., range) may consist of more than one class, which is usually a union of the classes (an `or`), not the intersection of them. Considering the property's participation in axioms, the `AllValuesFrom`/only/$\forall$ can be used to 'close' the relation, i.e., that no object can relate with that relation to the class other than the one specified. If you want to say there is at least one such relation (more common), then use `SomeValuesFrom`/some/$\exists$ instead. To state there is *no*

such relation in which the class on the left-hand side participates, put the negation before the quantifier ($\neg\forall$ or $\neg\exists$), whereas stating that there is a relation but just not with some particular class, then the negation goes in front of the class on the right-hand side; e.g., a vegetarian pizza does not have meat as ingredient ($\neg\exists$hasIngredient.Meat), not that it can have all kinds of ingredients—cucumber, marsh mellow, etc.—as long as it is not meat ($\exists$hasIngredient.$\neg$Meat). To avoid the latter (the unintended pizza ingredients), one ought not to introduce a class with negation, like NotMeat, but use negation properly in the axiom. Finally, when convinced *all* relevant properties for a class are represented, consider making it a defined class, if not already done so.

T8: Modelling Aspects (includes P4, P23, and C3): Even though it is too difficult to check whether every ontology element is defined in the most appropriate way, we attempt to provide some general rules for checking basic modelling issues. First, add only elements to the ontology that will be used somewhere in the ontology. Second, in case ontology population is part of the project, one could check whether all the classes will contain instances and, following that, whether the (potential) individual will have property assertions to other individuals, and that those properties indeed do exist. Also, check that the necessary data properties exist for those individuals that would need to be linked to values (e.g., dates, booleans, floats, strings, etc.). Finally, remember that the primitive `subclassOf` is used to define taxonomies and membership for individuals (see T2), and they are different from part-whole relations that do need a separate object property (e.g., Province is part of (or: located in) Country, not a subclass of Country).

T9: Domain Coverage and Requirements (includes P9 and P10): Check if functional ontology requirements—referring to the particular knowledge to be represented by the ontology and the particular terminology to be included in the ontology—are covered by the ontology. One important check in this regard is to inspect whether disjointness has been declared explicitly in the ontology (two ontology elements are disjoint if they cannot share instances or are already different individuals). It is crucial to remember that in OWL, elements are not disjoint unless disjointness statements are stated. Disjointness knowledge is important both as a means to describe the world as it is and to obtain the expected results from ontology inferences.

T10: Documentation and Understandability (includes P8, P20, and P22): Because of multi-authored ontologies, understanding of the ontology, and eyeing long-term and broad use and reuse, it is good practice to include human readable descriptions, such as comments and labels, in the ontology. Consider providing names as labels (with `rdfs:label`) and definitions as comments (with `rdfs:comment`) for all ontology elements. Remember not to confuse the name for an ontology element with its definition.

5 Conclusions

We performed a quantitative analysis of the pitfalls developers included in ontologies by analyzing different sets of data obtained after using OOPS!. All implemented types of pitfalls have been detected in the ontologies scanned with OOPS!, but the most common ones are lack of annotations, absence of domain and range axioms, and issues with inverses, and to a lesser extent creating unconnected ontology elements and using a recursive definition. Five new pitfalls have been identified upon closer inspection of the novices' ontologies. Analysis showed that there is no clear evidence of noteworthy differences between ontologies developed by novices, well-known ones, and the random set of ontologies, except for novice vs. mature when disregarding pitfall P8, and for novices, the pitfalls per ontology is related to the size of the ontology complexity of the ontology. Thus, the analysis provides a data-driven general landscape of pitfalls in current ontologies. Taking advantage of the results of our study, we proposed the Typical pItfalls Prevention Scheme (TIPS) in order to facilitate avoiding the inclusion of such common pitfalls in ontologies and thus to benefit the ontology quality.

We are extending the pitfall catalogue, and are working on a better characterization of 'maturity' in ontologies and how such a characterization is related to the set of most common pitfalls.

Acknowledgements. This work has been partially supported by the Spanish projects BabelData (TIN2010-17550) and BuscaMedia (CENIT 2009-1026).

References

1. Noy, N., McGuinness, D.: Ontology Development 101: A guide to creating your first ontology. Number KSL-01-05, and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001
2. Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL pizzas: practical experience of teaching OWL-DL: common errors & common patterns. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS, vol. 3257, pp. 63–81. Springer, Heidelberg (2004)
3. Roussey, C., Corcho, O., Vilches-Blázquez, L.: A catalogue of OWL ontology antipatterns. In: Proceedings of K-CAP 2009, pp. 205–206 (2009)
4. Poveda, M., Suárez-Figueroa, M.C., Gómez-Pérez, A.: Common pitfalls in ontology development. In: Meseguer, P., Mandow, L., Gasca, R.M. (eds.) CAEPIA 2009. LNCS, vol. 5988, pp. 91–100. Springer, Heidelberg (2010)
5. 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)
6. Keet, C.M.: Detecting and revising flaws in OWL object property expressions. 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. 252–266. Springer, Heidelberg (2012)

7. Guarino, N., Welty, C.: An overview of ontoclean. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 201–220. Springer, Heidelberg (2009)
8. Vrandečić, D.: Ontology evaluation. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, 2nd edn, pp. 293–313. Springer, Heidelberg (2009)
9. Gómez-Pérez, A.: Ontology evaluation. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. International Handbooks on Information Systems, pp. 251–274. Springer, Heidelberg (2004)
10. Poveda-Villalón, M., Suárez-Figueroa, M.C., Gómez-Pérez, A.: A double classification of common pitfalls in ontologies. In: Proceedings of Workshop on Ontology Quality (OntoQual 2010). CEUR-WS (2010) C-located with EKAW 2010
11. Guarino, N., Oberle, D., Staab, S.: What is an ontology? In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 1–17. Springer, Heidelberg (2009)
12. Brank, J., Grobelnik, M., Mladenic, D.: A survey of ontology evaluation techniques. In: Proceedings of SiKDD 2005, Ljubljana, Slovenia (2005)
13. Sabou, M., Fernandez, M.: Ontology (network) evaluation. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (eds.) Ontology Engineering in a Networked World, pp. 193–212. Springer, Heidelberg (2012)
14. 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)
15. Poveda-Villalón, M., Vatant, B., Suárez-Figueroa, M.C., Gomez-Perez, A.: Detecting good practices and pitfalls when publishing vocabularies on the web, Sydney, Australia, 21 October 2013
16. Suominen, O., Mader, C.: Assessing and improving the quality of SKOS vocabularies. J. Data Semant. **2**(2), 1–27 (2013)
17. Schulz, S., Stenzhorn, H., Boekers, M., Smith, B.: Strengths and limitations of formal ontologies in the biomedical domain. Electron. J. Commun. Info. Innov. Health (Special Issue on Ontologies, Semantic Web and Health) **3**(1), 31–45 (2009)
18. Keet, C.M.: The use of foundational ontologies in ontology development: an empirical assessment. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 321–335. Springer, Heidelberg (2011)
19. Aguado de Cea, G., Gómez-Pérez, A., Montiel-Ponsoda, E., Suárez-Figueroa, M.C.: Natural language-based approach for helping in the reuse of ontology design patterns. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 32–47. Springer, Heidelberg (2008)
20. Keet, C.M., Fernández-Reyes, F.C., Morales-González, A.: Representing mereotopological relations in OWL ontologies with OntoPartS. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 240–254. Springer, Heidelberg (2012)
21. Curé, O., Prié, Y., Champin, P.-A.: A knowledge-based approach to augment applications with interaction traces. 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. 317–326. Springer, Heidelberg (2012)
22. Braga, B.F.B., Almeida, J.P.A., Guizzardi, G., Benevides, A.B.: Transforming ontoUML into alloy: towards conceptual model validation using a lightweight formal methods. Innov. Syst. Softw. Eng. **6**(1–2), 55–63 (2010)
23. Halpin, T.: Information Modeling and Relational Databases. Morgan Kaufmann Publishers, San Francisco (2001)
24. Keet, C.M.: Transforming semi-structured life science diagrams into meaningful domain ontologies with DiDOn. J. Biomed. Inform. **45**, 482–494 (2012)