Catalogue of Anti-Patterns for formal Ontology debugging

Oscar Corcho[1], Catherine Roussey[2], Luis Manuel Vilches Blazquez[1]

[1] OEG, Universidad Politécnica de Madrid
ocorcho@fi.upm.es, lmvilches@delicias.dia.fi.upm.es
[2] LIRIS CNRS UMR5205, Université de Lyon, UCBL
catherine.roussey@liris.cnrs.fr

Abstract: Debugging of inconsistent OWL ontologies is normally a tedious and time-consuming task where a combination of ontology engineers and domain expert is often required to understand whether the changes to be performed in order to make the OWL ontology consistent are actually changing the intended meaning of the original knowledge model. This task is aided by existing ontology debugging systems, incorporated in existing reasoners and ontology engineering tools, which ameliorate this problem but in complex cases are still far from providing adequate support to ontology engineers, due to lack of efficiency or lack of precision in determining the main causes for inconsistencies. In this paper we describe a set of anti-patterns commonly found in OWL ontologies, which can be useful in the task of ontology debugging in combination with those debugging tools.

Mots-clés: Ontologies, OWL, Correction d'Erreurs, AntiPattern, Debugger.

1 HydrOntonlogy

The Spanish National Geographic Institute (IGN-E) developed a common reference model by means of a domain ontology, called hydrOntology. IGN-E wants to build this ontology in order to facilitate the semantic harmonization of hydrographic information among data producers at different levels (national, regional and local).

The statistical data (metrics) and its different taxonomic relations appearing below provide an overview of the hydrOntology characteristics.

HydrOntology is saved in the OWL format; it has 150 classes, 34 object properties, 66 data properties and 256 axioms. Some examples of the four taxonomic relations defined in the Frame Ontology [3] and the OKBC Ontology[2], namely, Subclasses, Disjoint-Decomposition, Exhaustive-Decomposition and Partitions, have been implemented in the ontology. Further details are shown in [9]. The ontology documentation is exhaustive, thus, definitions and their definition sources can be found in each concept (class). The ontology has an important amount of labels with alternative names (synonyms) as well as concept and synonym provenances.

A domain expert about geographical information was trained to build an ontology in Description Logics using Protégé tools (Protégé-OWL version 4). He built the ontology following METHONTOLOGY, a widely-used ontology building methodology. A detailed description of this methodology can be found in [4].

HydrOntology has been developed according to the ontology design principles proposed by [5] and [1]. Some of its most important characteristics are that the concept names (classes) are sufficiently explanatory and rightly written. According to some naming conventions, each class is written with a capital letter at the beginning of each word, while object and data properties are written with lower case letters. At the end of the development process 102 concepts were classified as incoherent by the classifier.

When implementing this ontology in OWL several issues arose with respect to its consistency, given its complexity. In the first iteration of implementation, where the domain expert took the conceptualization following Methontology's intermediate representations and encoded it with Protégé 4, all the classes in the ontology were considered inconsistent. Then the process of refinement started, using the OWL ontology debugging facilities of Protégé. Indeed, the debugging systems used did not provide enough information about root unsatisfiable classes or adequate (e.g., understandable by domain experts) justifications of the reasons for their unsatisfiability. Thus, we made an effort to understand inconsistency-leading patterns used by domain experts when implementing OWL ontology. Moreover in several occasions during the debugging process the generation of justifications for inconsistencies took several hours, what made these tools hard to use.

Ontology developer needs more recommendation for debugging than those provided by actual tools. We found out that in several occasions domain experts were just changing axioms from the original ontology in a somehow random manner, even changing the intended meaning of the real definitions instead of correcting errors in their formalisations.

After several iterations, which resulted in a large number of changes to the original implementation, the final consistent ontology could be delivered.

In this paper we propose a detailed list of such anti-patterns, compiling all the relevant cases that we came across when helping ontology developers to debug their ontologies.

2 Anti-patterns

We have identified a set of patterns that are commonly used by domain experts in their OWL implementations and that normally result in inconsistencies that may be easy or difficult to solve by them. This set of patterns is what we call anti-patterns, and we have categorized them in three groups:

- Logical Anti-Patterns (LAP). They represent errors that DL reasoners detect. These are the ones for which tool support is easier to provide and hence some support already exists.
- Non-Logical (aka Cognitive) Anti-patterns (NLAP). They represent possible modelling errors that are not detected by reasoners (they are not logical but model-

ling errors, which may be due to a misunderstanding of the logical consequences of the used expression).

- Guidelines (G). They represent complex expressions used in an ontology component definition that are correct from a logical point of view, but in which the ontology developer could have used other simpler alternatives for encoding the same knowledge.

In the rest of this section we describe each of the anti-patterns identified in each group, providing their name and acronym, their template logical expressions and a brief explanation of why this anti-pattern can appear. As aforementioned, it is important to note that LAP are identified by existing ontology debugging tools, although the information that is provided back to the user explaining the reason for the inconsistency is not described according to such a pattern, what makes it difficult for ontology developers to find out where the inconsistencies are coming from. With respect to NLAP and G, they are not currently detected by these tools as such, although in some cases their combination may lead into inconsistencies that are detected (although not appropriately explained) by tools. We think that tool support for them could be a major step forward in this task.

Finally, all these anti-patterns should be seen as elementary units that cause ontology inconsistencies. That is, they can be combined into more complex ones.

1 Logical Anti-Patterns

AntiPattern AndIsOr (AIO)

$C1 \subseteq \exists R.C2 \cap C3)$, $Disj(C2,C3)$[1]

This is a common modelling error that appears due to the fact that in common linguistic usage, "and" and "or" do not correspond consistently to logical conjunction and disjunction respectively [10]. For example, I want a cake with milk and chocolate is ambiguous. Does the recipe of cake contain some chocolate plus some milk? (Cake_Recipe $\subseteq$ ($\exists$contain.Chocolate)$\cap$($\exists$contain.Milk). Does the recipe of cake contain chocolate-flavoured milk? (Cake_Recipe$\subseteq$ $\exists$contain.(Chocolate$\cap$Milk)). Does the recipe of cake contain some chocolate or some milk? (Cake_Recipe $\subseteq$ $\exists$contain.(Chocolate$\cup$Milk)). The domain expert makes a confusion between the linguistic "and" and the logical "or". Notice that the position of the logical "and" has an importance in the semantic of an axiom.

This anti-pattern appeared 2 times in HydrOntology debugging process[2].

1. Cano$\subseteq\exists$comunica.(Albufera $\cap$ Mar $\cap$ Marisma)[3]

---

[1] This does not mean that the ontology developer has explicitly expressed that C2 and C3 are disjoint, but that these two concepts are determined as disjoint from each other by a reasoner. We use this notation as a shorthand for $C2 \cap C3 \subseteq \bot$.

[2] All the examples from HydrOntology are in Spanish. Indeed, we cannot translate the examples without changing the meaning of terms, because the conceptualization depends of the language used.

[3] For convenient purpose, we do not add the disjointness relation between classes when the reasoner deduces this relation. Thus notice that all the classes used in our example are found disjoint by the reasoner.

```
2. Ponor⊆∃comunica.(Aguas_Subterráneas ∩
   Aguas_Superficiales)
```

AntiPattern EquivalenceIsDifference (EID)

```
C1≡C2, Disj(C1,C2)
```

This inconsistency comes from the fact that the ontology developer wants to say that C1 is a subclass of C2 (that is, that C1 is a C2, but at the same time it is different from C2 since he has more information). This anti-pattern is only common for ontology developers with no previous training in OWL modelling, since after a short training session they would discover that they really want to express C1⊆C2. This inconsistency can hide also a terminological synonymy relation between classes like in SOE.

This anti-pattern appeared 5 times in HydrOntology debugging process.

```
1. Afluente ≡ Rio, Disj(Afluente, Rio)
2. Cienaga ≡ Zona_Pantanosa, Disj(Cienaga, Zona_Pantanosa)
3. Cascada ≡ Catarata, Disj(Cascada,Catarata)
4. Raudal ≡ Rapido, Disj(Raudal, Rapido)
5. Aljibe ≡ Cisterna, Disj(Aljibe, Cisterna)
```

AntiPattern OnlynessIsLoneliness (OIL)

```
C1⊆∀R.C2, C1⊆∀R.C3, Disj(C2,C3)
```

The ontology developer has created an universal restriction to say that C1 can only be linked with a R role to C2. Next, a new universal restriction is added saying that C1 can only be linked with R to C3, disjoint with C2. In general, this means that the ontology developer forgot the previous axiom

This anti-pattern appeared 2 times in HydrOntology debugging process.

```
1. Zona_Humeda⊆ Humedal ∩ ∀es_inundada.Aguas_Marinas ∩ ∀
   es_inundada.Aguas_Superficiales ∩ ≥1es_inundada.T
2. Agua_de_transicion⊆ ∀está_proxima.Aguas_Marinas
   ∩∀está_proxima.Desembocadura ∩ =1está_proxima.T
```

AntiPattern OnlynessIsLonelinessWithInheritance (OILWI)

```
C1⊆ C2, C1⊆∀R.C3, C2⊆∀R.C4, Disj(C3,C4).
```

The ontology developer has added a universal restriction for class C1 without remembering that he had already defined another universal restriction with the same property in a parent class. This anti-pattern is a specialization of OIL.

This anti-pattern appeared 2 times in HydrOntology debugging process.

```
1. Ibon⊆ Charca, Ibon⊆ ∀es_originado.(Glaciar ∪
   Masa_de_Hielo), Charca⊆ ∀es_originado.(Arroyo ∪
   Manantial ∪ Rio)
2. Lucio⊆ Charca, Lucio⊆ ∀es_originado.Marisma, Charca⊆
   ∀es_originado.(Arroyo ∪ Manantial ∪ Río ∪ Glaciar ∪
   Masa_de_Hielo)
```

AntiPattern OnlynessIsLonelinessWithPropertyInheritance (OILWPI)

`R1⊆R2, C1⊆∀R1.C2, C1⊆∀R2.C3, Disj(C2,C3)`

The ontology developer misunderstands the subproperty relation between roles, thinking that it is similar to a part-of relation. This anti-pattern is a specialization of OIL because C1⊆∀R1.C2, R1⊆R2 ⊨ C1⊆∀R2.C2

This anti-pattern did not appear in HydrOntology, we derived it from others.

AntiPattern UniversalExistence (UE)

`C1⊆∀R.C2, C1⊆∃R.C3, Disj(C2,C3)`

The ontology developer has added an existential restriction for a concept without remembering the existence of an inconsistency-leading universal restriction for that concept.

This anti-pattern did not appear in HydrOntology, we derived it from others.

AntiPattern UniversalExistenceWithInheritance1 (UEWI_1)

`C1⊆C2, C1⊆∃R.C3, C2⊆∀R.C4, Disj(C3,C4)`

The ontology developer has added an existential/universal restriction in a concept without remembering that there was already an inconsistency-leading universal/existential restriction in a parent class, respectively. This anti-pattern is a specialization of UE.

This anti-pattern appeared 1 time in HydrOntology debugging process.

1. `Gola⊆ Canal_Aguas_Marinas, Gola⊆ ∃comunica.Ría, Canal_Aguas_Marinas⊆ ∀comunica.Aguas_Marinas`

AntiPattern UniversalExistenceWithInheritance2 (UEWI_2)

`C1⊆C2, C1⊆∀R.C3, C2⊆∃R.C4, Disj(C3,C4)`

Same reasons as UEWI_1.

This anti-pattern appeared 1 time in HydrOntology debugging process.

1. `Charca⊆ Aguas_Quietas_Naturales, Charca⊆ ∀es_originado.(Arroyo∪Manantial∪Río∪Glaciar∪Masa_de_Hielo ∪Marisma), Aguas_Quietas_Naturales⊆ ∃es_originado.(Arroyo ∪Glaciar ∪Manantial ∪Rio), Aguas_Quietas_Naturales⊆ =1es_originado.T`

AntiPattern UniversalExistenceWithPropertyInheritance1 (UEWPI_1)

`R1⊆R2, C1⊆∃R1.C2, C1⊆∀R2.C3, Disj(C2,C3)`

The ontology developer misunderstands the subproperty relation between roles, thinking that it is similar to a part-of relation. This anti-pattern is a specialization of UE because C1 ⊆∃R1.C2, R1⊆R2 ⊨ C1⊆∃R2.C2

This anti-pattern appeared 1 time in HydrOntology debugging process.

1. `se_extrae ⊆ es_alimentada, Fuente_Artificiale⊆ ∃se_extrae.Acuífero∩ =1se_extrae.T, Fuente_Artificiale⊆ ∀es_alimentada.(Tubería)∩ =1es_alimentada.T`

AntiPattern UniversalExistenceWithInverseProperty (UEWIP)

`C2⊆∃R`$^{-1}$`.C1, C1⊆∀R.C3, Disj(C2,C3)`

The ontology developer has added restrictions about C2 and C1 using a role and its inverse. This antipattern is a specialization of UE because: $C2 \subseteq \exists R^{-1}.C1 \models C1.1 \subseteq \exists R.C2, C1.1 \subseteq C1$

This anti-pattern appeared 1 time in HydrOntology debugging process.

1. Aguas_Marinas⊆ ∃alimentada.Aguas_Quietas_Naturales, Aguas_Quietas_Naturales⊆ ∀es_alimentada.Aguas_Corrientes_Naturales

AntiPattern SumOfSomIsNeverEqualToOne (SOSINETO)

C1⊆∃R.C2, C1⊆∃R.C3, C1⊆≤1R.T, Disj(C2,C3)

This anti-pattern can also be written like this

C1⊆∃R.C2, C1⊆∃R.C3, C1⊆=1R.T, Disj(C2,C3)

The ontology developer has added a new existential restriction without remembering that he has already defined another existential and a cardinality restriction for the same concept and role. This pattern is not an elementary one because it contains the NLAP SOS and the G DCC (presented latter), none of these elementary antipattern cause inconsistency; nevertheless it is a good example that a combination of NLAP and G cause inconsistencies.

This complex anti-pattern appeared 1 time in HydrOntology debugging process.

1. Agua_de_transicion⊆ ∃sometida_a_influencia.Aguas_Dulces ∩∃sometida_a_influencia.Aguas_Saladas ∩ ∀sometida_a_influencia.(Aguas_Dulces ∪ Aguas_Saladas)∩ =1sometida_a_influencia.T

2 Non Logical Anti-Patterns

As aforementioned, these anti-patterns are not necessarily errors, but describe common templates that ontology developers use erroneously trying to represent a different piece of knowledge.

AntiPattern SynonymeOfEquivalence (SOE)

C1≡C2

The ontology developer wants to express that two concepts C1 and C2 are identical. This is not useful at all in a single ontology. This is not very useful in a single ontology that does not import others. Indeed, what the ontology developer generally wants to represent is a terminological synonymy relation: the class C1 has two labels: C1 and C2. Usually one of the classes is not used anywhere else in the axioms defined in the ontology.

This anti-pattern appeared 6 times in HydrOntology debugging process.

1. Aguas ≡ Masa_de_Agua,
2. Aguas_Marinas≡Masa_de_Agua_Marina,
3. Aguas_Subterraneas ≡ Masa_de_Agua_Subterraneas
4. Aguas_Superficiales ≡ Masa_de_Agua_Superficial
5. Aguas_Quietas_Artificiales ≡ Masa_de_Agua_Artificial
6. Corriente_Subterranea ≡ Rio_Subterranea

AntiPattern SumOfSom (SOS)

C1⊆∃R.C2, C1⊆∃R.C3, Disj(C2,C3)

The ontology developer has added a new existential restriction without remembering that he has already defined another existential restriction for the same concept and role. Although this could be ok in some cases (e.g., a child has at least one mother and at least one father), in many cases it represents a modelling error.

This anti-pattern appeared 4 times in HydrOntology debugging process.

2. Rio⊆ ∃puede_fluir.Corriente_Subterránea , Rio⊆ ∃puede_fluir.Ponor
3. Manantial⊆ ∃origina.Chortal, Manantial⊆ ∃origina.((Aguas_Corrientes_Naturales ∩ not Glaciar)∪ (Aguas_Quietas_Naturales∩ not Bodón∩ not Ibón∩ not Lavajo∩ not Lucio∩ not Masa_de_Hielo))
4. Aguas_Superficiales⊆ ∃es_distribuida.Canal_Aguas_Continentales, Aguas_Superficiales⊆ ∃es_distribuida.Distribución
5. Agua_de_transicion⊆ ∃sometida_a_influencia.Aguas_Dulces ∩∃sometida_a_influencia.Aguas_Saladas ∩ ∀sometida_a_influencia.(Aguas_Dulces ∪ Aguas_Saladas)∩ =1sometida_a_influencia.T

AntiPattern SumOfSomWithInheritage (SOSWI)

C1⊆ C2, C1⊆∃ R.C3, C2⊆∃R.C4, Disj(C3,C4)

The ontology developer has added an existential restriction in a concept without remembering that he had already defined another existential restriction with the same role in a parent class. This Anti-Pattern is a specialization of SOS.

This anti-pattern appeared 3 times in HydrOntology debugging process.

1. Torrente ⊆ Arroyo, Torrente⊆ ∃es_originado.(Glaciar∪ Masa_de_Hielo), Arroyo⊆ ∃es_originado.Nacimiento ∩ =1es_originado.T
2. Arroyo⊆ Aguas_Corrientes_Naturales, Arroyo⊆ ∃es_originado.(Nacimiento∪ Glaciar∪ Masa_de_Hielo)∩ =1es_originado.T, Aguas_Corrientes_Naturales⊆ ∃es_originado.Manantial
3. Rio ⊆Aguas_Corriente_Naturales, Rio⊆ ∃puede_fluir.(Corriente_Subterránea∪ Ponor), Aguas_Corriente_Naturales⊆ ∃puede_fluir.Poza

AntiPattern SumOfSomWithPropertyInheritance (SOSWPI)

R1⊆R2, C1⊆∃R1.C2, C1⊆∃R2.C3, Disj(C2,C3)

The ontology developer misunderstands the subproperty relation between roles, thinking that it is similar to a part-of relation. This Anti-Pattern is a specialization of SOS because C1⊆∃R1.C2, R1⊆R2 ⊨ C1⊆∃R2.C2

This anti-pattern did not appear in HydrOntology, we derived it from others.

AntiPattern SumOfSomWithInverseProperty (SOSWIP)

`C2⊆∃R-1.C1, C1⊆∃R.C3, Disj(C2,C3)`

The ontology developer has created two existential restrictions using a role and its inverse. This anti-pattern specializes SOS because: $C2 \subseteq \exists R^{-1}.C1 \models C1.1 \subseteq C1, C1.1 \subseteq \exists R.C2$.

This anti-pattern did not appear in HydrOntology, we derived it from others.

AntiPattern SomeMeansAtLeastOne (SMALO)

`C1⊆∃R.C2, C1⊆≥1R.T`

The cardinality restriction is superfluous, because if there is an existential restriction that means that the cardinality restriction using the same role is at least equal to 1. The ontology developer had created the axiom $C1 \subseteq \geq 1R.T$ first, to say that C1 should be defined by the R role. Next, he specialized his definition and forgot to remove the first restriction.

This anti-pattern appeared 2 times in HydrOntology debugging process.

1. `Rambla⊆∃es_originado.Torrente, Rambla⊆≥1es_originado.T`
2. `Estero ⊆∃está_proxima.Desembocadura ∩ ≥1está_proxima.T`

3 Guidelines

As aforementioned, guidelines represent complex expressions used in an ontology component definition that are correct from a logical point of view, but in which the ontology developer could have used other simpler alternatives for encoding the same knowledge.

Guideline DisjointnessOfComplement (DOC)

`C1≡not C2`

The ontology developer wants to say that C1 and C2 can not share instances. Even if the axiom is correct for a logical point of view, it is more appropriate to state that C1 and C2 are disjoint.

This anti-pattern appeared 3 times in HydrOntology debugging process.

1. `Aguas_Marinas ≡ not Aguas_Dulces`
2. `Albufera ≡ not Aguas_Dulces`
3. `Laguna_Salada ≡ not Aguas_Dulces`

Guideline Domain&CardinalityConstraints (DCC)

`C1⊆∃R.C2, C1⊆(≥2R.T) (for example)`

Ontology developers with little background in formal logic find difficult to understand that universal restriction does not imply existential one [10]. This antipattern is a counterpart of that fact. Developers may forget that existential restrictions contain a cardinality constraint: $C1 \subseteq \exists R.C2 \models C1 \subseteq (\geq 1R.C2)$. Thus, when they combine existential and cardinality restrictions, they may be actually thinking about universal restrictions with those cardinality constraints.

This anti-pattern appeared several times in HydrOntology debugging process, we only provide some examples.

1. `Aguas_Quietas_Naturales⊆ ∃es_originado.(Arroyo ∪Glaciar ∪Manantial ∪Rio), Aguas_Quietas_Naturales⊆ =1es_originado.T`
2. `Fuente_Artificiale ⊆∃se_extrae. Acuífero ∩=1se_extrae.T`
3. `Agua_de_transicion⊆ ∃sometida_a_influencia.Aguas_Dulces ∩∃sometida_a_influencia.Aguas_Saladas ∩ ∀sometida_a_influencia.(Aguas_Dulces ∪ Aguas_Saladas)∩ =1sometida_a_influencia.T`
4. `Arroyo⊆ ∃es_originado.Nacimiento ∩ =1es_originado.T`

Guideline GroupAxioms (GA)

`C1⊆∀R.C2, C1⊆(≥2R.T) (for example)`

In order to facilitate the comprehension of complex class definition, we recommend grouping all the restrictions of a concept that use the same role R in a single restriction. The previous restriction becomes C1⊆ (∀R.C2)∩(≥2R.T)

Because the development of an ontology is an iterative process, most part of the class definition using the same R role are split in several expressions. Have a look to previous examples.

Guideline MinIsZero (MIZ)

`C1⊆(≥0R.T)`

The ontology developer wants to say that C1 can be the domain of the R role. This restriction has no impact on the logical model being defined and can be removed.

This anti-pattern appeared 1 time in HydrOntology debugging process.

1. `Laguna_Salada ⊆0≥ es_alimentada.T`

3 Related works

As far as we know there exist only two works about anti-pattern in formal ontology development. In [8], the authors present four Logical Anti Pattern but all of them focus on the domain and range of Role. In our case all the domain and range of role are been remove before consistency checking. Due to the fact that we are in the development process of the ontology, class hierarchy is not valid enough to save the domain and range of role. Our proposition differs from [10] even if we use also the protégé tools in our experiment. In [10], the authors describe common difficulties for newcomers to Description Logics in understanding the logical meaning of expressions. Their use case examples are very small. In our case the ontology is bigger thus the ontology developer builds his ontology in several times. Moreover our ontology developer is not a DL expert but he has already learnt DL primitives.

Automated OWL ontology debugging features have been described, connected to reasoners and ontology engineering tools, in several recent works ([6, 7]). These features are very useful to debug ontologies, and allow identifying the main root unsatisfiable classes with different approaches, so that the debugging process can be guided by them and can be optimized. However, these features are very focused on the logical consequences that can be extracted from the logical theory of an OWL ontology, and are not so focused on the ontology engineering side, hence the explanations are still sometimes difficult to understand for ontology developers.

4 Conclusion and future works

In this paper, we have described a detailed list of anti-patterns commonly used by domain experts when implementing ontologies in OWL. This list is aimed at complementing the work that is done by automated ontology debugging tools when detecting inconsistencies in this type of ontologies, so that we can provide better explanations of the reasons why a specific class or set of classes of the ontology are inconsistent, and hence improve the efficiency of the ontology debugging process.

All these anti-patterns should be seen as elementary units that cause ontology inconsistencies. That is, they can be combined into more complex ones. However, providing a solution for the individual ones will be a good advance to the current state of the art, and our future work will be also devoted to finding the most common combinations and providing recommendations for them.

We have applied this list of anti-patterns to the development of an ontology in the hydrology domain (HydrOntology [9]), resulting in an improvement in the efficiency of the debugging process that we have not actually measured. However, our intuition suggests that the process has been much faster than what it would have been without the use of such anti-patterns, that is, with the use of debugging tools alone.

Our next steps towards providing effective tools to help domain experts in their ontology building tasks are making formal experiments with a set of inconsistent ontologies, built by domain experts that we have been collecting in the past year. The aim of these experiments would be to compare the time needed to complete the debugging process with and without the use of our anti-patterns, and the quality of the final models generated after debugging, in case that there are differences. Finally, another piece of work that we are planning to do in the future is to organize this list of anti-patterns into a set of debugging guidelines for the creation of a better-specified method for ontology debugging that can be more effective.

Acknowledgements

This work is a result of collaboration between the OEG and LIRIS lab. It has been done under the context of the project GeoBuddies, funded by the Spanish Ministry of Science and Technology and it was also partially funded by the COST Action C21 sponsored by the European Commission under the grant number STSM-C21-04241.

References

1. ARPÍREZ JC, GÓMEZ-PÉREZ A, LOZANO A, PINTO HS (1998) (ONTO)2Agent: An ontology-based WWW broker to select ontologies. In: Gómez-Pérez A, Benjamins RV (eds) *ECAI'98 Workshop on Applications of Ontologies and Problem-Solving Methods*. Brighton, United Kingdom, pp 16–24
2. CHAUDHRI VK, FARQUHAR A, FIKES R, KARP PD, RICE JP (1998) Open Knowledge Base Connectivity 2.0.3. *Technical Report KSL-98-06*, Knowledge Systems Laboratory, Stanford, CA, http://www.ai.sri.com/ okbc/okbc-2-0-3.pdf
3. FARQUHAR A, FIKES R, RICE J (1997) The Ontolingua Server: A Tool for Collaborative Ontology Construction. *International Journal of Human Computer Studies*, 46 (6): 707–727
4. GÓMEZ-PÉREZ A, FERNÁNDEZ-LÓPEZ M, CORCHO O (2003) Ontological Engineering. Springer-Verlag, London (United Kingdom)
5. GRUBER TR (1995) Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, vol 43 n.5-6.
6. MATTHEW HORRIDGE, BIJAN PARSIA AND ULRIKE SATTLER for Laconic and Precise Justifications in OWL. In proceedings of ISWC2008.
7. ADITYA KALYANPUR, BIJAN PARSIA, EVREN SIRIN, BERNARDO CUENCA-GRAU (2006) Repairing Unsatisfiable Concepts in OWL Ontologies . In proceedings of ESWC2006.
8. LABORATORY FOR APPLIED ONTOLOGY Collection of antipattern in http://wiki.loa-cnr.it/index.php/LoaWiki:MixedDomains
9. VILCHES-BLÁZQUEZ LM, BERNABÉ-POVEDA MA, SUÁREZ-FIGUEROA MC, GÓMEZ-PÉREZ A, RODRÍGUEZ-PASCUAL AF (2007) *Towntology & hydrOntology: Relationship between Urban and Hydrographic Features in the Geographic Information Domain*. In: Ontologies for Urban Development. Studies in Computational Intelligence, vol. 61, Springer, pp 73–84
10. HAI WANG, ALAN RECTOR, NICK DRUMMOND, MATTHEW HORRIDGE, (2004). OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In proceedings of EKAW 2004