CEUR Workshop Proceedings http://ceur-ws.org ISSN 1613-0073

Vol-596
urn:nbn:de:0074-596-3

ORES-2010
Ontology Repositories and Editors for the Semantic Web

Proceedings of the 1st Workshop on Ontology Repositories and Editors for the Semantic Web

Hersonissos, Crete, Greece, May 31st, 2010.

Edited by

Mathieu d'Aquin, The Open University, UK
Alexander García Castro, Universität Bremen, Germany
Christoph Lange, Jacobs University Bremen, Germany
Kim Viljanen, Aalto University, Helsinki, Finland

10-Jun-2010: submitted by Christoph Lange
11-Jun-2010: published on CEUR-WS.org

Context-aware access to ontologies on the Web

Patrick Maué[1], Alejandro Llaves Arellano[1], and Thore Fechner[1]

Institute for Geoinformatics (ifgi), University of Muenster, Germany
`patrick.maue|alejandro.llaves|thore.fechner@uni-muenster.de`

Abstract. Domain vocabularies capture the ontology engineer's context-specific perspective on reality. Existing solutions for serving such ontologies often lack intuitive means to avoid conflicts due to logically inconsistent concept descriptions. In addition, no efficient and simple techniques for selecting only relevant terms from extensive vocabularies exist. We present an implementation of a concept repository which shifts the focus from ontologies towards individual concept descriptions. The description's identity is defined by its title and an optional set of subjects. We introduce the notion of profiling concept descriptions to distinguish between context-independent and context-specific (and potentially conflicting) properties. A flexible approach for constructing the concept identifiers supports context-aware access. Furthermore, an extensible set of query actions allows for retrieving certain parts of ontologies, e.g. the neighbourhood of one particular concept or all concepts associated with a certain subject. We illustrate the findings with an implementation of an ontology repository.

1 Introduction

Integrating information across domains relies on a consistent interpretation of the underlying data models. Such semantic interoperability depends on mappings between different domain-specific vocabularies[1]. Ontologies are commonly used to formally represent such vocabularies. Aligning these ontologies to upper-level ontologies , or creating rules mapping between potentially conflicting domain ontologies [2], ensures integration without losing domain-characteristic features. Reasoning engines use the alignments to infer matching conceptualisations. This long-term vision of semantic interoperability across information communities is based, amongst others, on the assumption that:

(a) all domain ontologies are published on the Web. The ontology elements are resources with an identity, and are *accessible* [3] via unique and resolvable identifiers. Such Uniform Resource Locators (URL) are required for relating local application-specific schema to terms in shared vocabularies, and let reasoners retrieve the descriptions from the Web [4].
(b) the relationships between ontology elements are consistent and valid. URLs used by relations referring to external terms have to be accessible and return a valid resource.

(c) elements in local application schema are referenced to shared vocabularies using semantic annotations [5].

Ontologies are traditionally implemented as downloadable files encoded in one particular ontology language. Scope and encoding are defined by the ontology engineer, the ontology's content is usually static. Even though this approach complies to the assumptions (a) and, if performed carefully, also (b), it poses a great problem for (c). Only few examples of accessible and actively re-used ontologies exist. These are usually abstract and thematically narrow proposals such as FOAF [6] for modelling social networks. Published domain-specific ontologies, e.g. the SWEET ontologies for Earth science [7], are either only partially re-used or heavily adapted to local needs. Reasons for this are, amongst others: they are too extensive, which impairs navigation and understanding. They are too limited in scope. They are biased and don't capture the shared consensus of different domain experts. They are not maintained and therefore not representing the current state of knowledge. Or they are inconsistent, linking to remote, but non-existent resources. The implementation of the concept repository (CORE) addresses the first three issues, with the potential to also target the last two. An in-depth discussion of basic principles for re-usable ontologies can be found in [8]. Using mature methodologies for ontology engineering [9] can help to avoid some of the issues which we encountered during the creation of the domain ontologies.

We understand an ontology as loose collection of related concepts. The notion of *related* is deliberately underspecified: it depends on the client's context which particular representation of an existing vocabulary is considered suitable. We discuss the idea of *profiling concepts* to support conceptualizations conflicting with common sense. This phenomena appears not only in-between different information communities, but also between experts of the same domain. Profiling allows for individual interpretations without losing consistency with the underlying ontology. The presentation of a conceptually simple approach to realize profiling for concept descriptions is the main contribution of this paper.

Profiling supports context-sensitive ontology modularization, and various related work on this subject exists. Most research is focussing on the formal definition of modules in ontologies [10, 11], with focus on describing how to define (and how to separate) modules. In [12], the authors introduce a formal way to link the different modules. D'Aquin et al. [13] discuss different aspects ontology modularization methods have to consider (and can be evaluated against). According to [14], the following three approaches for ontology modularization exist: (1) Query-based methods, (2) Network partitioning, and (3) Extraction by traversal. Their segmentation approach for large-scale ontologies is based on the traversal in the ontology graph. The same is true for the implementation presented here.

The following section 2 lists the reasons explaining why we implemented our own version of an ontology repository, and why existing solutions did not meet our requirements. This section will also introduce a use case which acts as a running example for the remainder. In section 3 we discuss our approach and accordingly the implementation. We introduce the concept of profiling and addressing concepts and how to select relevant collections of concepts in the repository. We conclude the paper with a short evaluation and a summary.

2 Creating and sharing vocabularies on the Web

Authoring sophisticated ontologies in collaboration with domain experts, and making the results accessible on the Web, is only a first step. The active use of these ontologies by other parties (ideally from a different information community) is also needed to enable integration across information communities. Several issues have to be considered to not only complete the first, but also the second phase. In the following section we discuss our experience in knowledge acquisition and ontology engineering, and list the problems encountered which eventually resulted in the implementation of CORE.

2.1 Building Domain Ontologies

Deciding if one particular site may be a suitable location for quarrying mineral resources relies on a variety of criteria. The acquisition, analysis, and presentation of potentially relevant information guiding the decision maker has been the subject of the research project SWING[1]. The relevant information is served by Web services which have been semantically annotated with domain ontologies. The whole process (discovery,pre-processing, and rendering) has been implemented as a workflow. Such Web service compositions were also the focus of the GDI-Grid[2] project. Here, ontologies are used for the semantic validation of the Web service workflows. In SWING we mainly focused on interviews with domain experts like geologists to capture the relevant concepts and their properties [15]. The result of the conceptual phase [9] were extensive concept maps representing the core concepts which had to be implemented in the ontologies. Figure 1 depicts a small excerpt of one concept map. Graphical tools for authoring concept maps ship with two interesting features: Colour has been used to organize the concepts which belong to the same domain. The concept's spatial distribution is particular useful to group those which are in some way related (without the need to explicitly associate them with a domain using colour). Unfortunately, colour and location can not be directly re-used for the implementation of the concept maps as ontologies.

Figure 1 represents the engineer's view of the concept *River*. For computing the river's discharge (the product of the stream velocity and cross-sectional area), environmental models for flood prediction make use of information about the underlying terrain and observations coming from sensors. Computing the cross-sectional area relies on detailed information about the *Depth* of the river. In the remainder of this paper, we are using this particular quality as a running

[1] Project results and videos are available at http://www.swing-project.org/
[2] On-going project, more information available at http://www.gdi-grid.de/

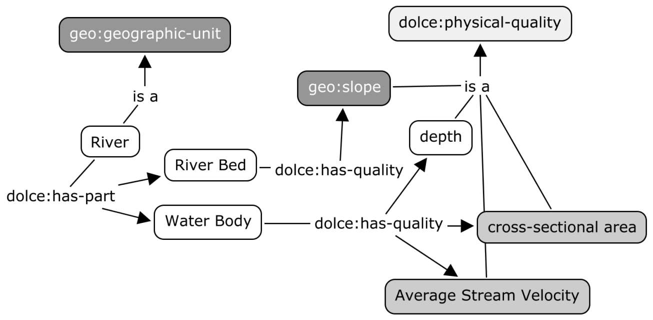

Fig. 1. An excerpt of a concept map

example to explain the idea of domain-independent conceptualizations. The captain steering a vessel through the river has one particular view on the river's depth. He is only concerned about the minimum depth of the official water way. The biologist may be more interested in maximum freezing depth, which allows for modelling the winter conditions for the fish population. In the next section we discuss the problems encountered when we realized that we have to integrate such different perspectives into the domain ontologies.

2.2 Conflicting Conceptualizations

The results of the knowledge acquisition where captured using either tools for building concept maps or by writing protocols of the discussions with the domain experts. For implementation, these intermediate, sometimes inconsistent, and rather informal models had to be transformed into formal ontologies. The problems described in this section have been the motivation for the implementation of the concept repository.

Until now, a concept has been merely described is by a name and relations to other concepts. Ontologies are meant to serve as formal specifications which explicitly describe the concept. These concept descriptions comprise a name, properties (including relations to other concepts), and additionally axiomatic statements which further constrain the properties. If an ontology is lacking ambiguities, e.g. due to homonyms, naming is not necessarily an issue. Hence, re-using the concept's name as part of its identifier is a common approach suitable for simple ontology building tasks. Since we refer to RDF-encoded ontologies shared on the Web, an identifier is implemented as Internationalized Resource Identifier (IRI). An IRI comprises a namespace and a local name. Concepts are often defined through other concepts: a concept description for river may be identified using the term "River", the river's depth as "RiverDepth", and one particular conceptualization even as "MinimumRiverDepth". This approach does not scale well for extensive ontologies, and eventually results in arbitrarily chosen local names which do not reflect the actual name of the concept. This becomes even more apparent if multiple names in different languages are to be supported for one concept. By decoupling the identifier's local name from the concept's name, ontologies can support different descriptions with similar names, as well as descriptions with different names. Within CORE, the concept descriptions have automatically generated local names and one common namespace. We make use of Dublin Core [16] metadata properties - in this case `dc:title` for the concept name - to model the identity of a concept description.

The problem of finding appropriate identifiers becomes more apparent if two concept descriptions within the same namespace (which means, the same ontology) describe the same concept. Figure 2 comprises two examples for a description of the concept *River* using the Manchester Syntax [17] of the Web Ontology Language (OWL).

Example 1 (Captain's Perspective).

```
Class: River
  Annotations:
    dc:title "River"

ObjectProperty: has-depth
  Annotations:
    dc:title "has depth"
  Domain:
    River
  Range:
    minimum-depth

Class: minimum-depth
  Annotations:
    dc:title "minimum depth"
```

Example 2 (Biologist's Perspective).

```
Class: River
  Annotations:
    dc:title "River"

ObjectProperty: has-depth
  Annotations:
    dc:title "has depth"
  Domain:
    River
  Range:
    maximum-freezing-depth

Class: maximum-freezing-depth
  Annotations:
  dc:title "maximum freezing depth"
```

Fig. 2. Two different conceptualisation of river depth.

Depending on the context, either of these two descriptions can be considered to be valid. Both share an identical extension since they refer to the same real world concept. The captain's understanding of *River* may differ from the biologist's, but both use the same term to refer to it. Hence, even though concept descriptions have conflicting properties, their names are identical. Modularization – splitting the ontology up into modules with different namespaces – may provide a solution for conflicting concept descriptions. During the implementation we regularly dealt with concepts whose context were not clearly defined. Such border-line cases can not be clearly associated with one particular domain. The need for modularization forces the ontology engineer to also explicitly assign context to concepts which are either context-free or belong to multiple domains. The *maximum-freezing-depth* of a river may be important in the scope of a Biology domain ontology, but is obviously also related to Hydrology. The concept *Depth* itself is domain-independent. Adding such concepts to one specific domain ontology strengthens the association to the domain, but also impairs re-usability in other contexts. During implementation we decided to interpret modularization differently: ontologies are no longer collections of concepts manually compiled by the ontology engineer. Ontology membership is simply a property itself. Every concept description may be defined to be part of multiple ontologies, and membership can change dynamically. Similarly to the concept's name, we use of the `dc:subject` property to express a concept's membership in a certain domain.

In SWING, the individual modules represented only a small excerpt of the needed vocabulary, and the aggregated graph was much too extensive for the visualization. Sophisticated query techniques for RDF-based vocabularies exist, but relying on such complex solutions impedes re-usability for generic clients. It would then be the client's responsibility to (a) study the ontology to learn how to formulate the query and (b) execute queries where in fact only one URL should be required for selecting the relevant collection of concepts. It should be possible to construct URLs which not only uniquely identify concept descriptions, but also allow for selecting collections of concept descriptions which are in some sense related and therefore important to the client. descriptions are only valid within a one particular domain, since there exists another conflicting description. In our case, the concept River may be modelled to have a quality "'depth"', which is commonly understood as the average depth measured by a gauge.

2.3 A first implementation

Existing solutions like the Tones Ontology Repository[3], Oyster[4], or Pronto[5] are focused on the ontologies as a subjects of interest, not the individual concept descriptions. A first implementation of CORE was released in late 2008 for the SWING project [18]. Only some of the features discussed in this paper have been realized in this version. In fact, most requirements were identified during its implementation and use.`dc:title` labels the concepts, and the concatenated language tag, e.g. "@en", marks different languages. The namespace defines the scope of the needed ontology. For example, the URL "http://.../core/GDI-Grid/" has been used to request all concepts associated with the "GDI-Grid" domain. The URL "http://.../core/Acoustics/GDI-Grid/" retrieves an intersection of two domains: the result is a list of concepts which have been defined valid for both domains.

The focus on using namespaces for defining scope had one major drawback. Managing the import of namespaces for local ontologies became a tedious task, since nearly every concept description was defined in a different scope. Additionally, the separation between listing all the concepts in one context (only the namespace is used as the URL) and concept description was not accepted by the

[3] See: http://owl.cs.manchester.ac.uk/repository/
[4] See: http://oyster.ontoware.org/
[5] See: http://metadata.net/sfprojects/pronto.htm

users. Hence, a new implementation of the concept repository[6] was initiated, which is currently in active development.

3 Solutions

In the following section, we introduce some suggestions to overcome the encountered problems. These includes the notion of profiling concepts to model the domain-specific perspective without breaking the relation to the original concepts, a solution for selecting subsets of concepts which may be relevant according to certain criteria, and the idea of regular consistency checks for the relationships between concepts.

3.1 Profiling concepts

The example of Figure 2 listed two valid, but conceptually inconsistent, descriptions for the concept *River*. Following Guarino, we consider a concept description (and its associated ontology) to be a "logical theory which gives an explicit, partial account of a conceptualization" [19]. The ontology engineer's subjective view on reality can only result in a partial description. Different perspectives on one concept may result in diverging and sometimes conflicting descriptions. The object's identity criteria are defined through its characteristic properties [20].

Profiling concepts supports different viewpoints on concepts within one ontology. It allows for refining and extending conceptual structures, without losing the applicability of the underlying model [21]. One profile[7] concretises another concept description. The concept itself is always domain-independent; the same is true for characteristic properties. The profile extends (and therefore concretises) a domain-independent description by either refining existing or adding new properties. Profiling is not inheritance. Both, source and profiled concept description, refer to the same concept. Both have an identical extension. All instances of *River* are also instance of *River (Biology)*. But only some instances of *River* can be considered to also be a *Creek* (which is modelled as sub-class of *River*) A profile specifies one particular viewpoint on a concept, but it does not affect its extension. Accordingly, both share the same name for identification.

Figure 3 illustrates how *River* has been refined to reflect the biologist's perspective. It also shows how concept descriptions are stored within the repository. During import, the identifiers are automatically generated (as hexadecimal codes) from the title and, if existing, the subject. A concept is profiled by specifying that a property is only valid within a certain context, i.e., a `dc:subject` annotation is added. An existing property is refined by additionally re-using the source property's `dc:title`-annotation and changing the property's range. In the

[6] This time as part of an open source project. More information is available at http://purl.org/net/sapience/docs/. All source code is publicly accessible via the subversion repository.

[7] The idea of "profiling" is commonly used in the standards communities to explain if one standard is concretising another.

```
Class: 26c623af
  Annotations:
    dc:title "River"
    rdfs:seeAlso River_Biology
  SubClassOf:
    geo:geographic-object

DatatypeProperty: 1a50ca0c
  Annotations:
    dc:title "has depth"
  Domain:
    26c623af
  Range:
    double
```

```
Class: 618c2089
  Annotations:
    dc:title "River"
    dc:subject "Biology"

ObjectProperty: addac6d
  Annotations:
    dc:title "has depth"
    dc:subject "Biology"
  Domain:
    618c2089
  Range:
    d1ce83e7

Class: d1ce83e7
  Annotations:
    dc:title "maximum freezing depth"
    dc:subject "Biology"
```

Fig. 3. The concept *River* (left) and the profiled concept *River (Biology)* (right)

figure, the domain-independent concept description includes the property "has Depth" with a literal as its range. The range of this property has been changed and refers to the "maximum freezing depth" for the profiled concept. The ontology engineer is responsible for creating the profiled concept *River (Biology)*, adapting the properties, and adding a `rdfs:seeAlso` annotation to link the original concept description to the new profiling description. We already mentioned that semantic heterogeneities may not only exist between different information communities, but already within one community, or even within one organization. Profiles can again be source descriptions for other profiles. The transitive nature of profiling enables individual conceptualizations at all levels, with the option to trace the profiles back to the original source. Users are then able to navigate to the profiled concept descriptions if needed. In the following section we explain how to retrieve the concept descriptions either for *River* (without the refined properties) or *River (Biology)* (the value of the `rdfs:seeAlso` annotation in the figure).

3.2 Accessing concepts

Internally, all concept descriptions have automatically generated local names which are used for identification. The concept's identity, on the other hand, is defined by its title and subject. Title and subject can be defined in various ways in the URL. The expressions `River_Biology`, `/subject/Biology/River`, `River?subject=Biology` and `/describe?title=River&subject=Biology` all identify the same concept description. Only one context can be specified in the URL. The first three examples are internally transformed into the fourth. In the end, the query task (see section 3.3) *describe* is triggered with the parameters *title* and *subject*. The result of this query is the concept description listed in figure 4. The resulting RDF is formatted according to the requested URL. As suggested in [22], the concept description's identifier is always the request URL. The style of the other resource identifiers in the concept description, e.g. for the properties, is equivalent to the style of the request URL. If a concept description with the given parameters does not exist, the result is either a redirection (HTTP Response Code 303) to the potentially correct concept description (e.g. if a non-existent domain-dependent concept is requested, a redirect to the domain-independent description is returned) or an exception (HTTP Response Code 404).

```
Class: River_Biology
  Annotations:
    dc:title "River"
    dc:subject "Biology"

ObjectProperty: has-depth_Biology
  Annotations:
    dc:title "has depth"
  Domain:
    River_Biology
  Range:
    maximum-freezing-depth
```

Fig. 4. Result for the URL "http://.../rdf/River_Biology"

3.3 Accessing ontologies

An ontology is a collection of related concepts descriptions. Depending on the user's need, the type of the relevant relation may differ. In most cases, though, she might be interested in all concepts associated with one domain, e.g. Biology or Hydrology. Which specific collection, and therefore ontology, is returned by CORE depends on the ontology identifier. As when accessing a concept description, the access to an ontology is defined by the URL parameter. For example, the query action *describe* returns a concept description matching the given query parameters *title* and *subject*. We distinguish between query and update actions. The first triggers a SPARQL [23] query to the internal RDF repository (based on Sesame [24]), and optionally transforms the result. The latter is used to upload ontologies into the repository. We have developed the query tasks *all* and *neighbors* for CORE. The neighbourhood of one particular concept comprises all other concepts which are directly related to the query concept via its properties. If the optional depth-parameter is specified, the properties of the related concepts are considered as well. The all-action returns all concept descriptions which have the given subject-parameter defined as their domain. As for the describe-action, all actions support the encoding either in the URL's path (RESTful approach), or in the query fragment. Figure 5 shows three equivalent URLs to return all concepts within the domain Biology.

```
(1) http://.../rdf/Biology/
(2) http://.../rdf/subject/Biology/
(3) http://.../rdf/all?subject=Biology
```

Fig. 5. Constructing the Ontology identifier.

The idea of query actions is not constrained to the two introduced actions. The action *is-similar* could return similar (but not explicitly related) concepts, the action *has-property* may query for all concepts with the given property. Even though we've implemented CORE as a repository for ontologies, it might also be deployed for other RDF-based vocabularies. Using it, for example, as a gazetteer would require query actions supporting spatial queries like *contains*, which runs not only a SPARQL query, but also performs spatial filtering.

4 Evaluation

The first version of the concept repository has been evaluated in two research projects. The new features discussed here were implemented and tested (using module tests), and will be released in the next version. It is deployed as software as a service using the Google infrastructure, which addresses issues such as scalability, performance, and sustainability [25]. Scalability for RDF repositories is primarily concerned about performance of handling very large numbers of triples. Since reducing the amount of retrieved ontology elements has been the objective of CORE, scalability regarding the extent of the ontologies was not investigated. Other open issues, i.e. the scalability of the profiling approach, have to be tested in a long-term evaluation, which is planned in the just started research project ENVISION (http://www.envision-project.eu).

5 Conclusion

Language independence has always been one of the key requirements. The ontologies in SWING and GDI-Grid were implemented using the Web Service Modeling Language WSML[8]. Support for the more popular OWL Web Ontology Language has been identified as a requirement as well. CORE is not restricted to one particular ontology language, but requires an RDF-encoding. CORE is a sophisticated solution to access resources in an RDF repository.

[8] More information available at: http://www.wsmo.org/wsml/

In this paper, we presented an implementation of an ontology repository. We discussed why ontologies published on the Web are rarely re-used in semantically enriched applications, and listed the problems we encountered during knowledge acquisition and ontology engineering. Our proposal to facilitate the use of existing shared vocabularies included the following recommendations: profiling of concepts supports adaptation of existing domain concepts to local needs, without losing the alignment to the underlying domain ontology. A RESTful approach to access the shared vocabularies simplified local integration. Domain-specific information can simply be encoded in the URL used to identify a concept. Continuously running consistency checks test the relationships within the ontologies to ensure valid connections. If we are able to facilitate re-usability of existing shared vocabularies, the envisioned semantic integration of data across information communities may become reality. We believe the presented implementation of the concept repository CORE can contribute to this vision.

6 Acknowledgments

The presented research has been funded by the BMBF project *GDI-Grid* (BMBF 01IG07012) and the European projects *SWING* (FP6-026514) and *ENVISION*(FP7-249120).

References

1. Kuhn, W.: Geospatial Semantics: Why, of What, and How? Journal on Data Semantics III **3534** (2005) 1–24
2. Maué, P., Ortmann, J.: Getting across information communities. Earth Science Informatics **2** (2009) 217–233
3. Hayes, P.J., Halpin, H.: In Defense of Ambiguity. International Journal on Semantic Web & Information Systems **4** (2008) 1–18
4. Berrueta, D., Phipps, J., Miles, A., Baker, T., Swick, R.: Best Practice Recipes for Publishing RDF Vocabularies (2008)
5. Handschuh, S., Staab, S.: Annotation for the Semantic Web (Frontiers in Artificial Intelligence and Applications). IOS Press (2003)
6. Graves, M., Constabaris, A., Brickley, D.: FOAF: Connecting People on the Semantic Web. Cataloging & classification quarterly **43** (2007) 191–202
7. Raskin, R., Pan, M.: Knowledge representation in the semantic web for Earth and environmental terminology (SWEET). Computers & Geosciences **31** (2005) 1119–1125
8. Smith, B.: Against Idiosyncrasy in Ontology Development. In: Proceeding of the 2006 conference on Formal Ontology in Information Systems, Amsterdam, The Netherlands, The Netherlands, IOS Press (2006) 15–26
9. Gomez-Perez, A., Corcho, O., Fernandez-Lopez, M.: Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition (Advanced Information and Knowledge Processing). Springer (2004)
10. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: theory and practice. J. Artif. Int. Res. **31** (2008) 273–318

11. Grau, B.C., Parsia, B., Sirin, E., Kalyanpur, A.: Modularity and web ontologies. In: In Proc. KR-2006. (2006) 198–209
12. Bao, J., Caragea, D., Honavar, V.: On the semantics of linking and importing in modular ontologies. In: The Semantic Web - ISWC 2006. (2006) 72–86
13. d'Aquin, M., Schlicht, A., Stuckenschmidt, H., Sabou, M.: Ontology modularization for knowledge selection: Experiments and evaluations. In Wagner, R., Revell, N., Pernul, G., eds.: Database and Expert Systems Applications. Volume 4653 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg (2007) 874–883
14. Seidenberg, J., Rector, A.: Web ontology segmentation: analysis, classification and use. In: WWW '06: Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, ACM (2006) 13–22
15. Schade, S., Maué, P., Langlois, J., Klien, E.: Knowledge acquisition with geologists - a field report. In: ESSI1 Semantic Interoperability, Knowledge and Ontologies, EGU General Assembly 2008. (2008)
16. Weibel, S.L., Koch, T.: The Dublin Core Metadata Initiative: Mission, Current Activities, and Future Directions. D-Lib Magazine **6** (2000)
17. Horridge, M., Drummond, N., Goodwin, J., Rector, A.L., Stevens, R., Wang, H.: The Manchester OWL Syntax. In Grau, B.C., Hitzler, P., Shankey, C., Wallace, E., Grau, B.C., Hitzler, P., Shankey, C., Wallace, E., eds.: OWLED. Volume 216 of CEUR Workshop Proceedings., CEUR-WS.org (2006)
18. Schade, S.: D3.3 Ontology Repository with ontologies. Technical report, University of Münster (2008)
19. Guarino, N.: Formal Ontology and Information Systems. In Guarino, N., ed.: Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998., Amsterdam, IOS Press (1998) 3–15
20. Guarino, N., Welty, C.: Identity, Unity, and Individuality: Towards a Formal Toolkit for Ontological Analysis. In: Proceedings of the 14th European Conference on Artificial Intelligence (ECAI), IOS Press (2000) 219–223
21. Koutsomitropoulos, D.A., Paloukis, G.E., Papatheodorou, T.S.: Semantic application profiles: A means to enhance knowledge discovery in domain metadata models. Metadata and Semanticss (2009) 23–33
22. Sauermann, L., Cyganiak, R., Völkel, M.: Cool URIs for the Semantic Web (2007)
23. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. Technical report, W3C (2008)
24. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: The Semantic Web ISWC 2002. Lecture Notes in Computer Science. Springer (2002) 54–68
25. Erdogmus, H.: Cloud computing: Does nirvana hide behind the nebula? IEEE Software **26** (2009) 4–6