Reengineering the Core Grid Ontology*

Paweł Szmeja[2] and Antoine Zimmermann[1]

[1] École Nationale Supérieure des Mines, FAYOL-ENSMSE, LSTI, F-42023 Saint-Étienne, France
[2] Systems Research Institute Polish Academy of Sciences, Warsaw, Poland
antoine.zimmermann@emse.fr, pawel.szmeja@gmail.com

1 Introduction

Ontology engineering is a relatively new and immature science. With new possible applications and often non-normative standard proposals emerging rapidly it is hard to find one universal formalization of ontology engineering rules. This causes many authors to fall into a trap of focusing on the application of an ontology and not giving enough attention to the ontology itself. Such approach results in ontologies that are hardly reusable, overcomplicated and difficult to understand which is the exact opposite of what a good ontology should be. While many problems are very specific we attempt to bring the attention to a set of common ones with the intention of instructing how to fix or avoid them. Here, we explain how we improved an existing ontology, *Core Grid Ontology* (CGO), that we use as an example. It is not our intention to focus on the quality of this ontology, which otherwise fits our needs well. Indeed, *CGO* was reused in the project *Agents in Grid* (AiG), where it was slightly modified then extended [1]. We build on that experience, starting with the reengineering of *CGO* that we will later extend with the improvements of the *AiG* ontology.

2 Problems and Solutions

First, as an ontology is meant to be *shared*, it should be easily available. We updated the ontology IRI to `http://purl/NET/cgo/`, a persistent URL that can be used to get to the ontology file.

Second, in order to be *reusable*, an ontology should contain documentation. A proper documentation being absent, we relied on the associated publication [2] that describes the *CGO*. We noticed that there are discrepancies between the content of this document and the ontology itself. Some classes present in the ontology are not described in [2]. The properties of some classes outlined in the document do not correspond to the content of the ontology. The Web ontology language OWL allows one to document ontologies with OWL annotations in the ontology itself, which ensures that documentation is in line with content. In particular, the property `rdfs:comment` is used to explain the intended use of an ontology term and to keep track of any changes in the file, while `rdfs:label`

* AT2012, 15-16 October 2012, Dubrovnik, Croatia. This work has been supported by COST Action IC0801 Agreement Technologies.

is essential for interfaces to present a human-readable. All natural language metadata should have a language tag, as Web ontologies can be internationalized. For instance, in *CGO*, the class `VO` has a very convenient short identifier, but a label expanding this abbreviation to *"virtual organization"* eases working with the ontology. Adding a description to the properties that do not have any usage, range, domain or annotations defined prevents a user from having to guess the meaning of the entity based only on its name.

Third, an ontology should be *modular*. In [2], the ontology is described as having 3 levels with very general terms in the first level, more specific ones in the second, and application specific ones in the third. However, the ontology file did not reflect this modularization. We simply divided the ontology in two parts. We separated classes and individuals that are application specific. All remaining axioms that can be reused across application are in the "base" ontology that is imported by the other one. This again simplified reuse. Modularity eases maintenance and evolution and can usually be achieved by reflecting the hierarchy of the domain that the ontology describes.

Fourth, sometimes providing concrete usage *examples* is worth a thousand words. The *CGO* had examples that were very helpful, but we also reverse-engineered a few terms by working out an example, and found missing properties by doing so. For instance, the class `StorageSpace` is lacking a property that gives the storage size. The domain and range of properties could also be defined from the example. Some of these observations were already mentioned in [1].

3 Conclusion

In this paper, using the *CGO* as an example we presented common problems that plague ontologies and proposed simple solutions. By applying presented ideas and approaching ontology engineering more formally we can speed up the uptake of ontologies, lower the learning curve and increase the overall quality of ontologies. Quality ontologies are more likely to be reused across various scenarios, facilitating interoperability, which is key in Grid systems in particular, but also in agreement technologies in general.

References

1. M. Paprzycki, M. Drozdowicz, M. Ganzha, K. Wasielewska, I. Lirkov, R. Olejnik, and N. Attaoui. Utilization of Modified CoreGRID Ontology in an Agent-based Grid Resource Management System. In *Proc. of CATA 2010*, pages 240–245, 2010.
2. W. Xing, M. Dikaiakos, and R. Sakellariou. A Core Grid Ontology for the Semantic Grid. In *Proc. of CCGrid 2006*, pages 178–184, 2006.