

Ontology Access in Grids with WS-DAIOnt and the RDF(S) Realization

Miguel Esteban Gutiérrez, Asunción Gómez-Pérez, Oscar Muñoz García, Boris Villazón Terrazas

Ontology Engineering Group, Universidad Politécnica de Madrid
{mesteban, asun}@fi.upm.es, omunoz@delicias.dia.fi.upm.es, bvillazon@fi.upm.es

Abstract

This paper presents WS-DAIOnt, a framework for defining ontology access service interfaces in terms of the WS-DAI specification, extending it with the patterns, properties and behaviours needed for providing ontology access. We also present WS-DAIOnt-RDF(S): a realization of WS-DAIOnt for accessing RDF(S) ontologies.

1. Introduction

The increase of use of semantic technologies has reached almost every computer science related field, including the Grid computing field. The next generation Grid should virtualise the notion of distribution in computation, storage, and communication over unlimited resources with well defined computational semantics. A Grid node may provide new services, functions or even new concepts that are unknown to clients. The *semantics* of such services are defined by means of Ontologies [1], which are formal and explicit specification of a shared conceptualizations.

Ontologies could be used in the Grid for several purposes. Some of them could be: describing policies and sharing information, services and computing resources in Virtual Organizations, describing formal and non formal properties of Grid resources and services, accessing data catalogs in a conceptual and multidimensional way, etc.

Right now, a few Grid applications use ontologies but the access to these ontologies is not integrated in the Grid infrastructure. So, if the ontology server that provides such ontologies is down (or simply does not provide them for any other reason), the ontology-based Grid application won't be able to use it, so it won't work at last.

In the Grid computing area, the current OGSA architecture [2] doesn't consider ontology usage, and there are neither protocols nor standards available in the grid community for tackling with this issue. Therefore the provisioning of proven value mechanisms for accessing and managing ontologies in Grid environments is the priority if the Grid wants to make profit of the semantic technologies already available in other areas such as the Semantic Web [6].

The Semantic Grid community [3] does not start from the scratch. The Semantic Web community has developed languages and tools for building and using ontologies. The W3C has proposed three languages as recommendations to be used for implementing ontologies in the Semantic Web (RDF¹, RDFS² and OWL³) and several ontology development tools (i.e., Protégé⁴, WebODE⁵, KAON⁶) support the creation of ontologies in such languages. There also exist tools for querying ontologies and performing inferences with them. Normally, these querying and inference tools are strongly related to the language in which the ontology is implemented. Such languages differ in their expressiveness (the kind of knowledge that can be represented) and in their inference mechanisms (the kind of reasoning they carry out). For a detailed description and comparison of languages and tools we recommend [1]. However, the diversity of

¹ <http://www.w3.org/RDF/>

² <http://www.w3.org/TR/rdf-schema/>

³ <http://www.w3.org/2001/sw/WebOnt/>

⁴ <http://protege.stanford.edu/>

⁵ <http://webode.dia.fi.upm.es/WebODEWeb/>

⁶ <http://kaon.semanticweb.org/>

ontology languages and tools provokes the translation problem, which appears when an ontology developer decides to reuse an ontology (or part of an ontology) with a tool or language that is different from those where the ontology is available. At the same time, several APIs exist for accessing ontologies implemented in a given language and the ontology user (or the application that uses the ontology) should know how to retrieve the ontology content. As example, we can say that RDF(S) ontologies can be stored in Sesame⁷, 3store⁸, Joseki⁹, Jena¹⁰ or Kowari¹¹ and each one has its own primitives for accessing them. At this moment, the Semantic Web community does not have a standard mechanism or protocol for accessing ontologies implemented in a given ontology language in a storage-and-retrieval system independent fashion, what leads to severe interoperability problems.

In order to be able to apply semantic technologies in the Grid, we must first face and solve these interoperability issues. Thus providing the appropriate means for accessing and using ontologies in the Grid is fundamental if semantic technologies are to be used, and the transition from monolithic, centralized ontology services to a virtual organization of Grid compliant and Grid aware ontology services that can coordinate and cooperate with each other is crucial to progress towards the Semantic Grid.

2. WS-DAIOnt: a framework for specifying ontology access services

One of the main goals of the OntoGrid¹² project is to explicitly share and deploy knowledge to be used for the development of innovative Grid infrastructure and for Grid applications. To address this challenge the OntoGrid project is developing a Semantic Grid reference architecture (named Semantic OGSA¹³, a.k.a. *S-OGSA*) and the technological infrastructure for the rapid prototyping and development of knowledge-intensive distributed open services for the Semantic Grid. A key module on this Semantic OGSA reference architecture is the component that provides access to Ontologies, being our main goal in OntoGrid to architect, build on, adapt and extend existing ontology services to be Grid compliant. This high level, very general objective is refined as follows:

- To access heterogeneous and distributed ontology sources as homogeneous logical resources.
- To provide seamless access to ontologies implemented according to different knowledge representation formalisms in different languages.
- To provide uniform mechanism for providing homogeneous service interfaces for accessing to ontologies in a Grid environment.
- To generate a modular design to ease extensibility according to other ontology source formats.

With the goal of avoiding the proliferation of different access mechanisms for ontologies implemented in languages of the Semantic Web, the OntoGrid project is specifying and designing an ontology access mechanism for the Grid, whose formal name is **WS-DAIOnt**. It provides a WS-DAI [4] based framework for defining ontology access service interfaces using the standard grid data access vocabulary¹⁴, and extending the data access mechanisms with the patterns, properties and behaviours needed for providing ontology access. The WS-DAIOnt specification and the accompanying realizations (**WS-DAIOnt-RDF(S)**, ...) define the data access services that are needed for dealing with ontologies in Grid environments. It has the following features:

- Is fully compliant with S-OGSA, therefore is also compliant with OGSA.
- Is based on up-to-date Grid standards (GGF): WS-DAI specification, which defines a basic framework for data service interfaces and properties definition¹⁵.
- Is based on Web Service standards (OASIS, W3C): WS-RF, WS-Addressing.

WS-DAIOnt is built over four pillars:

⁷ <http://www.openrdf.org/>

⁸ <http://www.aktors.org/technologies/3store/>

⁹ <http://www.joseki.org/>

¹⁰ <http://jena.sourceforge.net/>

¹¹ <http://kowari.org/>

¹² <http://www.ontogrid.net/>

¹³ Deliverable D1.2 at the OntoGrid website.

¹⁴ Defined in the WS-DAI specification, see [4].

¹⁵ Note that the WS-DAI specification is still under development. As it evolves over time, the WS-DAIOnt specification will also evolve as needed in order to be fully compliant with this base specification.

- *Unified basic terminology.* WS-DAIOnt defines a neutral vocabulary for naming the ontology elements that are to be used when dealing with ontologies in Grid environments. This common and standard vocabulary avoids the usage of multiple different vocabularies that would make difficult the understanding of the provided data elements and functionalities.
- *Ontology elements usage patterns.* WS-DAIOnt defines how the messages, methods, interfaces and services must be specified in order to provide functionalities in a standard way.
- *Ontology elements possible relationships.* WS-DAIOnt defines how to specify the way each ontology element is related to each other.
- *Ontology access services behaviors.* WS-DAIOnt defines which is the expected behavior of the predefined common components and functionalities, so that every concrete implementation must adhere to these behaviors.

By extending WS-DAI, and therefore the OGSA architecture, with WS-DAIOnt and the accompanying realizations, we provide to the current Grid architecture with a standard way for supplying ontology access and management capabilities, thus enabling the future integration of semantic technologies in the Grid architecture. A preliminary detailed version of the WS-DAIOnt specification is available as part of the deliverable D3.1 of the OntoGrid project [8], which is available in the project's website (<http://www.ontogrid.net>)¹⁶.

3. WS-DAIOnt-RDF(S): accessing RDF(S) in the Grid

To test the correctness of our approach, we decided to start the implementation of WS-DAIOnt with ontologies implemented in RDF(S). The WS-DAIOnt-RDF(S) realization provides a framework for defining ontology access service interfaces using the WS-DAI vocabulary, and defining the set of messages, properties and behaviors needed for providing ontology access to ontologies implemented in RDF(S).

The infrastructure chosen for developing WS-DAIOnt-RDF(S) is the Globus Toolkit 4¹⁷ for providing the Grid infrastructure, specifically the Java WS-Core. Regarding the ontology repository, we selected Sesame¹⁸, using SeRQL¹⁹ for accessing the ontologies stored in it.

In the WS-DAIOnt-RDF(S) framework architecture we distinguish three service layers (see Figure 1): the upper service layer allows selecting the repository that is to be used; the intermediate service layer allows interacting with the selected repository directly; finally, the lower service layer includes the services that allow interacting directly with the RDF(S) knowledge components.

Until now the following services have been fully developed:

`RDFSRepositorySelectorService`. This service allows selecting the specific repository that is to be used in case that multiple repositories are available. Each repository is identified by a unique identifier.

`RDFSRepositoryService`. This service provides access to all the knowledge components (properties, statements, classes, etc.) in the repository. Some of the operations provided are:

- Get all classes of the repository (the method name is `getAllClasses`).
- Get a class by URI (the method name is `getClass`).

`RDFSClassService`. This service provides access to a given RDFS class. Some of the operations provided are:

- Get the siblings classes of the class (the method name is `getSiblings`).
- Get all the subclasses of the given class, direct or indirect classes, (the method name is `getSubClasses`).
- Get all the super classes of the given class, direct or indirect classes, (the method name is `getSuperClasses`).

¹⁶ The version available is now obsolete due to the quick evolution of the WS-DAI specification, and also by

¹⁷ <http://www.globus.org/toolkit/>

¹⁸ <http://www.openrdf.org/>

¹⁹ <http://www.openrdf.org/doc/sesame/users/ch06.html>

The latter service, *RDFClassService*, needs to interact with the Sesame repository in order to execute its methods (*getSiblings*, *getSubClasses*, etc.). As we are trying to provide storage-independent access to RDF(S) we have to decouple the repository from the service implementation. Generally speaking, those services which have to interact with the repository must do it in a loosely coupled way. This has been achieved by an extra data access layer, namely *RDFSCConnector*, which adapts the access to the repository to the services requirements.

Figure 1 shows the relationships of these services in the proposed architecture.

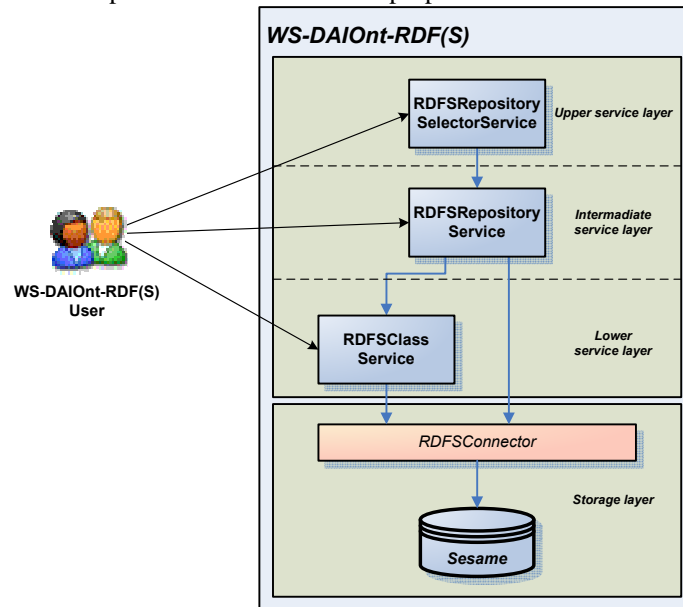


Figure 1. WS-DAIOnt-RDF(S) architecture

We have briefly presented WSDAIOnt goals and the first implementation of the WSDAIOnt-RDF(S). In our future work we will focused on the development of WS-DAIOnt-OWL implementation.

Acknowledgments

This work is supported by the OntoGrid project (FP6-511513) and by a U.P.M. PhD student grant.

References

- [1] A. Gómez-Pérez, M. Fernández-López, O. Corcho; *Ontological Engineering*. Springer Verlag, 2003
- [2] I. Foster (Editor), D. Berry, A. Djaoui, A. Grimshaw, B. Horn, H. Kishimoto (Editor), F. Maciel, A. Savva (Editor), F. Siebenlist, R. Subramaniam, J. Treadwell, J. Von Reich; *The Open Grid Services*
- [3] D. De Roure, N.R. Jennings, N.R. Shadbolt; *The Semantic Grid:Past, Present and Future*. In: *Proceedings of the IEEE* Volume 93, Issue 3, March 2005, pp 669-681.
- [4] M. Antonioletti, M. Atkinson, S. Malaika, S. Laws, N. W. Paton D. Pearson and G. Riccardi; *Web Services Data Access and Integration (WS-DAI)*. DAIS-WG Informational Draft, 13th Global Grid.
- [5] K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe; *The WS-Resource Framework, Version 1.0; The Globus Alliance*, May 3rd 2004.
- [6] T. Berners-Lee, J. Hendler, O. Lassila; *The Semantic Web*. In: *Scientific American* Volume 284, Number 5, May 2001, pp. 34-43.
- [7] OASIS Web Services Security 1.0 (WS-Security 2004) OASIS Web Services Security Technical Committee, 6th April 2004.
- [8] M. Esteban Gutiérrez (ed), S. Bechhofer, O. Corcho, M. Fernández-López, A. Gómez-Pérez, Z. Kaoudi, I. Kotsiopoulos, M. Koubarakis, M.C. Suárez-Figueroa, V. Tamma; *Specification and Design of Ontology Grid Compliant and Grid Aware Services*. D3.1 OntoGrid Project, 27th April 2005.