Design and Development of a Core Grid Ontology *

Wei Xing[1], **Marios D. Dikaiakos**[1],
Rizos Sakellariou[2], **Salvatore Orlando**[3], **Domenico Laforenza**[3]
[1] Department of Computer Science, University of Cyprus
[2] School of Computer Science, University of Manchester
[3] Dipartimento di Informatica, Universita Ca' Foscari di Venezia
Email:{xing,mdd}@ucy.ac.cy, rizos@cs.man.ac.uk,
orlando@dsi.unive.it, domenico.laforenza@isti.cnr.it

Abstract. In this paper, we introduce a core Grid ontology that defines fundamental Grid domain concepts, vocabularies and relationships. This ontology is based on a general model of Grid infrastructures, and described in Web Ontology Language OWL. Such an ontology can play an important role in building Grid-related Knowledge base and in supporting the realization of the Semantic Grid.

1 Introduction

The Semantic Grid is perceived as an extension of current Grids in which information and services are given a well-defined meaning, better enabling computers and people to work in cooperation [5]. Ontologies are among the key building blocks for the semantic Grid. They determine the terms of Grid entities, resources, capabilities and the relationships between them, with which any kind of content can be *meaningful* by the addition of ontological annotations.

The main problem for building an ontology for Grids is that there is currently a multitude of proposed Grid architectures and Grid implementations, and these are comprised of thousands of Grid entities, services, components, and applications. It is thus very difficult, if at all feasible, to develop a complete Grid ontology that will include all aspects of Grids. Furthermore, different Grid sub-domains, such as Grid resource discovery and Grid job scheduling, normally have different views or interests of a Grid entity and its properties. This makes the definition of Grid entities and the relationships between them very hard.

To tackle these issues, we propose a core Grid ontology (CGO) that defines fundamental Grid-specific concepts, vocabularies and relationships, based on a general model for Grids. This ontology can provide a common basis for representing Grid knowledge about Grid resources, Grid middleware, services, applications, and Grid users. A key challenge that needs to be addressed in this context is to make the core Grid ontology extensible and general enough to be used by, or incorporated in, different Grid systems and tools. To address

* Work supported in part by the European Commission under the CoreGRID project.

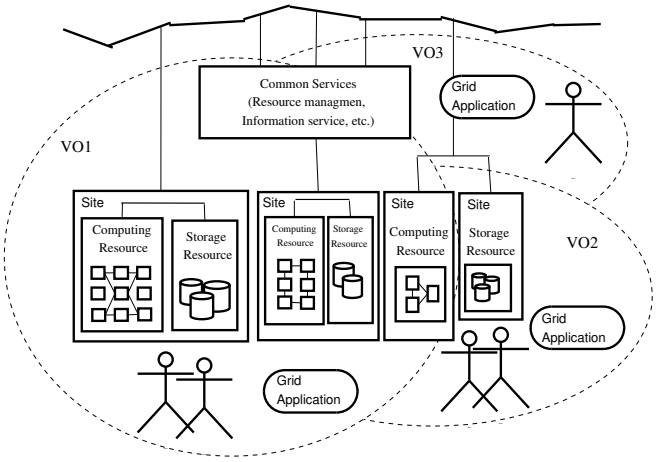

Fig. 1. The Overview of the Proposed Grid Model

this challenge, we design the CGO so that it represents a general model for Grids, which is compatible to major Grid infrastructures [14, 15, 1, 10, 4]. Also by adopting the Resource Description Framework (RDF) graph data model [9], a W3C recommendation, as the data model of choice for representing the CGO concepts and their relationships. The characteristics of the RDF data model make the ontology easier to extend by either adding new classes in it or adding extra properties (slots) into a defined class without any conflict with existing definitions.

The remaining of this paper is organized as follows. In Section 2, we introduce the proposed Grid model. In Section 3, we present an overview of the Core Grid Ontology. We illustrate how to use the Core Grid Ontology with an example in Section 4. Finally, we conclude our paper in Section 5.

2 Grid Model Description

The Grid is a platform for coordinated resource sharing and problem solving in dynamic, multi-institutional Virtual Organizations (VO) [7]. In essence, the Grid can be considered as a collection of Virtual Organizations and different kinds of resources. Resources are combined and organized by Grid middleware to provide Grid users with computing power, storage capability, and services, required for problem solving. VOs enable disparate groups of organizations and/or individuals to share resources in a controlled fashion, so that members may collaborate to achieve a shared goal. In our proposed Grid model, we view the Grid as a constellation of Virtual Organizations (VOs), which includes VOs, users, applications, middleware, services, computing and storage resources, networks, policies of use (see Figure1).

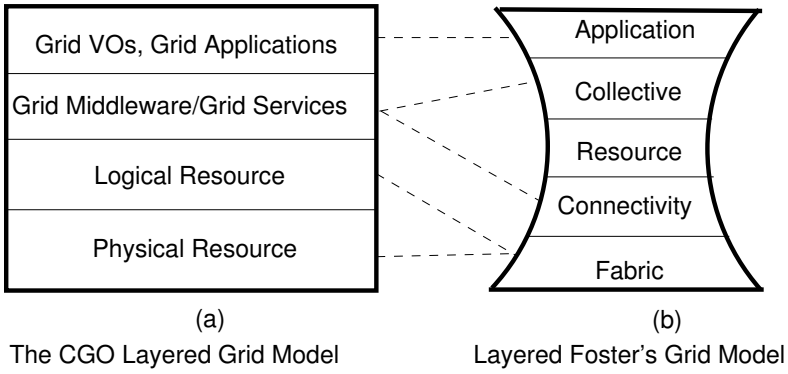

Fig. 2. The Layered Grid Model

One design issue of the core Grid ontology is to capture a "right" model for the Grid, that could be used to further specify Grid concepts, vocabularies, relations, and constrains. This model must remain simple and should have a proper level of abstraction that hides the numerous details involved in Grids. It should also provide a general view of important aspects of Grids [6].

The proposed model is actually a layer-structured model. As shown in Figure 2, the top layer includes multi-VOs, Grid Users, and Applications; Grid middleware and Grid services lie on the second layer; third layer is the VO-based "virtual" resources layer; the "real" physical Grid resources, such as clusters, networks, is at the bottom. Compared with the Foster's layered Grid model in [7], our layered Grid model is designed around a simple four-layer scheme, it combines the features of some adjacent layers of Foster's model and splits other layers apart [7]. In the top layer of our Grid model, we add Grid VOs and users besides applications. From our point of view, the VOs are actually a container that has users and applications in it. Thus, VO should be a fundamental element of Grids. The layer of the Grid middleware and Grid services of our model can be mapped into the three layers of the Foster's model, e.g., Connectivity, Resource, and Collective layer. The two layers, e.g. Logical Resource and Physical Resource, are correspondent with the fabric layer of the Foster's model. The intention of our proposed model is to provide a general, abstract view of Grids, which avoids any specific architectures or functionalities. The results of our division is a more general, extensible, open, VO-oriented model.

One notable aspect of the proposed model is that we distinguish Grid resources as logical or physical. A physical resource (PR) is a "real" resource that is part of a Grid, and the logical resource (LR) is "virtual" resource that a VO controls according to its policies, rules, and availability. Grid middleware and Grid services are responsible for mapping LR into PR. From the view of VOs and Grid Applications, the LR is more "realistic" than the PR, as Grids are VO-oriented. Since the Grid resources normally serve multi-VOs concurrently, the LR are many more in absolute numbers than the PR.

3 A Core Grid Ontology

The core Grid ontology is designed to represent Grid knowledge of Grid systems. Therefore, it should be open and extensible as there are thousands of Grid entities, services, components, and applications of different Grid architectures and Grid implementations. To cope with the openness and extensibility requirements, we adopt the Web Ontology Language OWL to describe the terms and classes in the core Grid ontology [13]. OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web, which is developed as a vocabulary extension of the Resource Description Framework (RDF) [3]. Given the fact that both RDF and OWL are W3C recommendations, the core Grid ontology is thus open, and compatible with other systems. Furthermore, the characteristics of the RDF data model make the ontology easier to extend by either adding new classes in the ontology or adding extra properties (or slots) into a defined class without any conflict with existing definitions. The RDF data model is a directed graph with labeled nodes and arcs; the arcs are directed from one node (i.e. subject) to another node (i.e. object). The object may be linked to other nodes (e.g. other new classes) through a property. One key feature of this data model is that properties in RDF are defined globally, that is, they are not encapsulated as attributes in class definitions. It is thus possible to define new properties that apply to an existing class without changing that class.

One main challenge in developing a core Grid ontology is to provide formal definitions and axioms that constrain the interpretation of classes and terms. In the following sections, we describe the concepts and, in particular, represent their constraints on the Grid domain according to the knowledge derived from analyzing, evaluating, and experimenting with different Grid architectures, production middleware and large Grid infrastructures, such as, Globus, Unicore, DataGrid, Crossgrid, and EGEE [14, 15, 1, 10, 4]. We first define a set of core Grid ontology classes, which reflect on all the elements of the abstract model. Subclasses can then be added accordingly. After that, we define the relationships and constraints among the ontology classes using RDF properties. These properties are links among the defined classes. Finally, the classes, properties, and instances together form a knowledge base that captures the configuration and state of specific Grid infrastructures. Such a knowledge-base can be used for various inquiries and for decision-support of end-users, application developers, Grid administrators, etc.

3.1 The Core Grid Ontology Classes and the Properties

In the proposed Grid model, we view the Grid as a collection of VOs, any amount of Grid resources, Grid Middleware and Services. Thus we can group the concepts (or Grid Objects) of the Core Grid Ontology into three categories:

1. VO-related: the classes that reflect the Grid entities on the top level of the proposed Grid model, including VO, GridUser, GridApplication, and Policy.

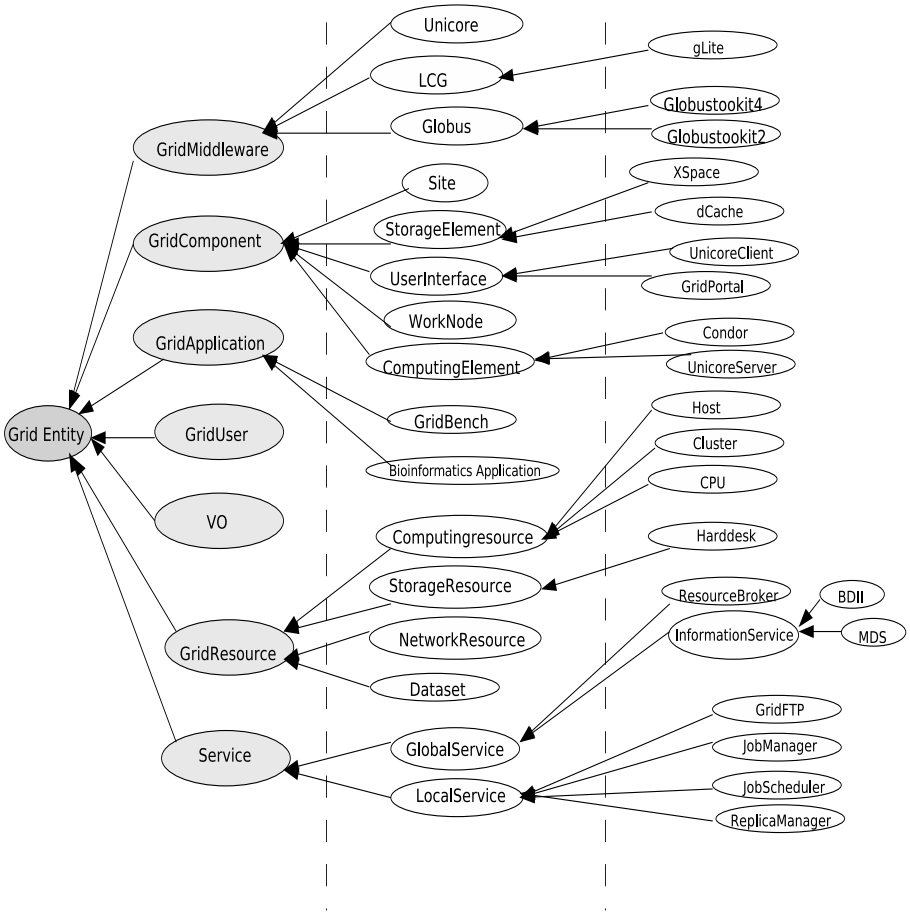

Fig. 3. The Overview of the Core Grid Ontology Classes

2. Grid Resource: Any resources in/on the Grid, including Computing Resource, Storage Resource, Network Resource, and Dataset. The classes of this catalog represent the Grid entities of the logical resource layer and physical resource layer of the Grid model.
3. Grid Middleware & Service: Grid middleware, functions, components, and services, which provide mechanisms and functionalities that provide the Grid Resource to server VOs.

Based on the proposed Grid model, the following types of the core classes are included in the CGO collection: VO, GridResource, GridMiddleware, GridComponent, GridUser, GridApplication, Service. A overview of the CGO classes is given in Figure 3. The meaning of the top level classes is presented as follows:

– VO: A set of individuals and/or institutions defined by sharing policies and rules form what we call a virtual organization (VO).

- GridResource: Any Hardware, Software, computing resource, storage resource, network resource within a Grid.
- GridMiddleware: Software that glue Grid services together following a specific Grid architecture.
- GridComponent: An architecturally significant part of a Grid system with well-defined inbound and outbound interfaces and associated protocols that encapsulate a cohesive documented set of responsibilities.
- GridUser: Users who use the Grid system.
- GridApplication: Applications that can run on Grids.
- Service: Any services that can provide one or more Grid functionalities.

The core classes (see in Figure 3) are fundamental concepts, elements, and aspects of a Grid system. They provide a "framework" for representing any entity of a Grid system. In other words, all important Grid entities and resources in/on Grids can be "located" within this framework. For example, to describe a DataGrid-based Grid component ComputingElement (CE), which is responsible for providing computing power and comprised of one or more similar machines managed by a single scheduler/job queue, following aspects need to be determined [1]:

- Which VOs does it support?
- What kinds of Service run on it?
- Which GridMiddleware has been installed?
- Which kinds of applications can it run?
- How many Grid resources can it provide?

Therefore, we can represent the Data Grid CE using the defined core ontology classes in the RDF triple model as follows:

{ CE *isA* *ComputingElement*};
{ CE *supportVO* *VO*};
{ CE *runningService* *Service*};
{ CE *hasInstalled* *GridMiddleware*};
{ CE *totalCPU* XMLSchema#int };
{ CE *freeCPU* XMLSchema#int }.

In order to represent the relationships and constrains among the ontology classes, we define the properties that provide the semantic meaning for the Core Grid Ontology entities. As we mentioned earlier, one key design goal of the CGO is to be open and extensible. Hence users can extend the CGO by adding their classes and properties on a "read-to-have" basis. In other words, we intend to provide a "framework" for representing a Grid system instead of having a whole, complete set of classes and properties for a Grid system. In this paper, we only present the core properties that reflect the relationships among the core ontology classes concerning with the class ComputingResource to illustrate how the relationships among the classes are defined (or linked). Currently, the whole defined properties can be founded in [12]. Table 1 shows the properties related with Computing Resource in the core Grid ontology.

Properties	*Description*
supportVO	*belong to VO*
runningService	*Grid service running on*
hasName	*Name of any resource*
coService	*Services related to the running services*
accessControlBaseType	*Policy for accessing Grid resources*
maxRunningJobs	*maxim number of job in one queue*
operatingSystem	*type of OS on host*
runningEnvironment	*middleware or services environment*
storageDevice	*interface of the storage element*
fileSystem	*the type of the file system on the storage element*
totalCPU	*the number of the total CPUs*
freeCPU	*the number of the available CPUs*

Table 1. The Properties related with CE in the Core Grid Ontology

3.2 Representing a Grid Entity using the Core Grid Ontology

We adopted OWL to describe the identified Grid entities, not only defining the concepts of them but also the relationships and constrains among them. For example, we defined the ComputingElement (CE) as:

- *CE is a Grid entity has several kinds of computing resources, such as CPU, Memory. The resource is organized by grid middleware and use Grid services to provide computing power to VO(s).*
- *A CE is comprised of one or more similar machines managed by a single scheduler/job queue.*

According to the definition, we describe the constrains of the class ComputingElement: a) a CE must support at least one VO; b) a CE machine must run a GridManager service; c) a CE machine must run a JobManager service and JobScheduler service. The three restrictions can be described in Description Logics as follows [2]:

$$\begin{aligned} \text{CE} \sqsubseteq\ & \exists\ \text{supportVO}\ \ \text{VO}. \\ & \exists\ \text{runningSevice}\ (\text{GridManager}), \\ & \exists\ \text{runningSevice}\ (\text{JobManager} \sqcup \text{JobScheduler}). \end{aligned}$$

The above formula represents the necessary conditions of a Computing Element:

(1) isA Computing Resource.
(2) at least one of the values of the runningService property is of type GridManager.
(3) at least one of the values of the runningService property are the union of
  - JobManager
  - JobScheduler
(4) at least one of the values of supportVO property is of type VO.

We thus describe the CE from the following facts: i) There is at least one VO supported; ii) There are a number of CPUs this CE dedicates to its VO; iii) There are some specified services running on it; iv) The GridMiddleware it installed; v) It has some worker nodes. The Table 2 shows the CE description in OWL using the Core Grid Ontology.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
.....

<owl:Class rdf:ID="ComputingElement">
     <rdfs:subClassOf>
       <owl:Restriction>
         <owl:someValuesFrom>
           <owl:Class>
             <owl:unionOf rdf:parseType="Collection">
               <owl:Class rdf:about="#Jobmanager"/>
               <owl:Class rdf:about="#JobScheduler"/>
             </owl:unionOf>
           </owl:Class>
         </owl:someValuesFrom>
         <owl:onProperty rdf:resource="#runningSevice"/>
       </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf rdf:resource="#EDG"/>
     <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#runningSevice"/>
         <owl:someValuesFrom rdf:resource="#Gridmanager"/>
       </owl:Restriction>
     </rdfs:subClassOf>
  </owl:Class>
......
```

Table 2. CE class Definition in Core Grid Ontology

4 Example: Generating the Ontology Instances

After introducing the Core Grid Ontology, we represent a ComputingElement (CE) of the CY01-LCG2 Grid node of the EGEE infrastructure to show how the ontology can help generate the knowledge-based information about a CE [4].

In the CY01-LCG2 Grid node, we have a CE named ce101.grid.ucy.ac.cy. We describe the ce101 instance based on the definition of the ComputingElement class. In CGO, it is defined that there must be three services running on a CE. Furthermore, since the *Gridmiddleware* of the CY01-LCG2 Grid node is *LCG*, we then can "*deduce*" that the jobmanager service is *openPBS* and job scheduler service is *MAUI*. So we can represent the CE as:

1. ce101.grid.ucy.ac.cy is a LCG-based CE;
2. It supports BioMed VO;
3. It has LCG-2.6.0 installed;
4. It runs globus-gatekeeper as Grid manager;
5. It runs openPBS as JobManager, and MAUI as Jobscheduler;
6. It has 20 WorkerNodes, and provides 40 CPUs.

By this way, we can represent an instance of a CE (i.e., ce101.grid.ucy.ac.cy) in RDF/OWL. It is in fact the knowledge about ce101.grid.ucy.ac.cy, which can be queried by Grid users, Grid services, or Grid applications. Figure 4 shows the ontology definition and the OWL description of the example, presented as an OWL graph.

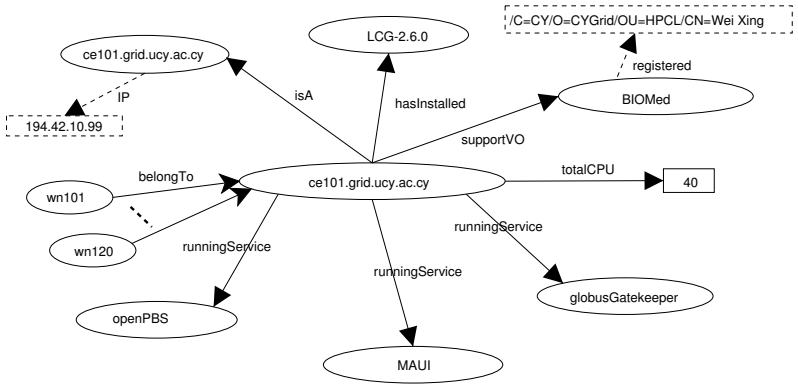

Fig. 4. The Core Grid Ontology Instance

5 Conclusions and Future Work

In this paper, we present our initial work on building a core Grid ontology. We first introduce an abstract model of Grid. After that, we design and develop a core Grid ontology that expresses the basic concepts and relationships of Grid entities and Grid resources according to the proposed Grid model. The flexibility and extensibility of the ontology allows it to be used, among other things, for Grid information integration, information searching, resource discovery and resource allocation management. Additionally, the fact that it is Grid architecture and implementation independent, renders it quite useful for hybrid large-scale Grids.

In the future, we plan to extend our work in the following directions:

1 Developing a toll that can automatically generate the instances of the CGO using Jena [8]. Currently we generate the instances of the CGO manually using Protege [11]. It is obviously not suit for the Grid system, which includes thousands of instances. Therefore, we need a tool that can automatically generate and update the instances dynamically.
2 Support knowledge-based queries. The ontologies/knowledge will be stored in a distributed manner. We need a suitable OWL query language and distributed query mechanism to query those distributed knowledge efficiently.

3 Knowledge-based resource discovery. Traditionally, the resource discovery is performed by key-word based one-by-one match mechanism. With the Core Grid Ontology, one can search the required resources with a knowledge-based search. Namely, good selection strategies, which use various kinds of knowledge, are used to guide the resource discovery. It will not only improve the performance of the resource discovery, also improve the quality of the results.

6 Acknowledgments

We thank George Tsouloupas,Nick Drummond, Matthew Horridge, Robert Stevens and Hai Wang for helpful discussions.

References

1. http://eu-datagrid.web.cern.ch/eu-datagrid/.
2. *The Description Logic Handbook*. Cambridge University Press, 2003.
3. D. Brickley and R.V. Guha (editors). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft, October 2003. http://www.w3.org/TR/rdf-schema/.
4. S. Campana and A. Sciaba M. Litmaath. LCG-2 Middleware Overview. LCG Technical Document. https://edms.cern.ch/file/498079//LCG-mw.pdf.
5. N.R.Shadbolt De Roure, D.Jennings, editor. *The Semantic Grid: Past, Present and Future*. IEEE, March 2005.
6. M. Dikaiakos and A. Artemiou. Navigating the grid information space: Design and implementation of the ovid browser. Technical Report Technical Report TR-2004-07, Department of Computer Science, University of Cyprus, December 2004.
7. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Supercomputer Applications*, 2001.
8. http://jena.sourceforge.net/ontology/. *Jena 2 Ontology API*.
9. F. Manola and E. Miller (editors). RDF Primer. W3C Working Draft, October 2003. http://www.w3.org/TR/rdf-primer/.
10. J. Marco, W. Xing R. Marco, and M. Dikaiakos et al. First prototype of the crossgrid testbed. volume vol. 2970 of *Lecture Notes in Computer Science series*, pages 67–77. Springer, Santiago de Compostela, Spain, February 2003.
11. N. F. Noy, S. Decker M. Sintek, R. W. Fergerson M. Crubezy, and M. A. Musen. Creating semantic web contents with protege-2000. *IEEE Intelligent Systems*, 16:60–71, 2001.
12. http://grid.ucy.ac.cy/grisen/coreonto.owl.
13. P.F. Patel-Schneider, P. Hayes, and I. Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax*. World Wide Web Consortium, February 2004.
14. S. Tuecke, I. Foster K. Czajkowski, C. Kesselman J. Frey, S. Graham, T. Sandholm T. Maguire, and D. Snelling P. Vanderbilt. Open Grid Service Infrastructure (OGSI) version 1.0. *Open Grid Service Infrastructure WG, Global Grid Forum*, 2002.
15. Ph. Wieder and D. Mallmann. UniGrids - Uniform Interface to Grid Services. In *7th HLRS Metacomputing and Grid Workshop*. Stuttgart, Germany, April 2004.