# Design and Development of a core Grid Ontology *

**Wei Xing**[1], **Marios D. Dikaiakos**[1], **Rizos Sakellariou**[2],
[1] Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus
[2] School of Computer Science, University of Manchester, M13 9QS Manchester, UK
Email: xing@ucy.ac.cy, mdd@ucy.ac.cy, rizos@cs.man.ac.uk

**Abstract**

In this paper, we present a core Grid ontology that defines the fundamental Grid domain concepts, vocabularies and relationships based on an abstract Grid model. This ontology can provide a basis of Grid knowledge for Grid middle-ware, services, applications, as well as Grid users.

## 1 Introduction

The Semantic Grid is perceived as an extension of current Grids in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation [4]. Ontologies are one of 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, which includes 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 that defines fundamental Grid-specific concepts, vocabularies and relationships, based on an abstract Grid model. This ontology can provide a basis for representing Grid knowledge in a common way for Grid middleware, services, applications, as well as Grid users. One of our key challenges is to make this core Grid ontology easily extensible and general enough to be used by any Grid architecture or any Grid middleware.

## 2 Grid Model Description

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 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 [5].

Grid is emerging as a platform for coordinated resource sharing and problem solving in dynamic, multi-institutional Virtual Organizations (VO) [6]. In reality, Grid is a collection of Virtual Organizations and different kinds of resources. The resources are combined and organized by Grid middleware to provide computing power, storage capability, and services

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

for Grid users (or Grid applications) to solve their problems. The 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, resources, users, applications, middleware, services, computing element, storage element, network, policy, etc. (see Fig1).

The proposed model is actually a layer-structured model. The top layer includes multi-VOs, Grid Users, and Applications; Grid middleware, Grid services, and VO-based "virtual" resource lie on the middle layer; the "real" physical Grid resource, such as cluster, network, is at the bottom. We distinguish the Grid resource as logical resource (LR) and physical resource (PR) in this model. A PR is a "real" resource that is part of a Grid, and the LR is "virtual" resource that a VO posses according to the policies, rules, and the 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 a absolute numbers than the PR.

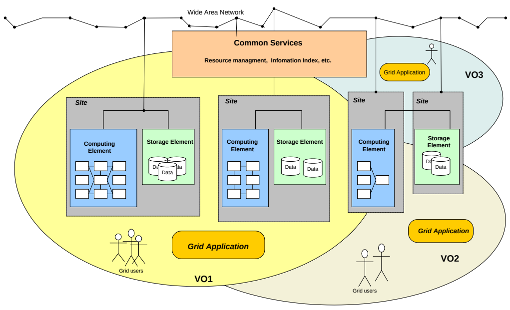


Figure 1: The Overview of the Proposed Grid Model

## 3 A Core Grid Ontology

In this section, we introduce the design of a Core Grid Ontology (CGO). The key role of the CGO is to provide a higher-level framework in which all concepts of Grids can be given a consistent and semantically coherent representation. Thus it is designed as an upper-level ontology, which captures and models the basic concepts and knowledge of Grids. We start with some basic distinctive Grid concepts, such as *VO, GridMiddleware, GridService, GridResource,* etc. Further on, the CGO goes into more details to an extent that Grid entities of general importance are included, like *Policy, ComputingComponent, StorageComponent, ResourceMgt, InfoService, SecurityInfra, ComputingResource, NetworkResource,* etc. The characteristic attributes and relations for the featured entities are defined. Having this Core Grid Ontology as a basis, one could add architecture-specific extensions to it easily, in order to represent an architecture-specific Grid, for instance, *GlobusToolkit4, GridPortal, XSpace, GRAM, MDS, BDII,* etc.

One main challenge in developing a Core Grid Ontology is to provide formal definitions and axioms that constrain the interpretation of classes. We describe the concepts and 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 [8, 9, 2, 7, 3].

The CGO is comprised of two key parts: (1) the CGO classes and the class hierarchy; (2) the properties of the CGO classes, which assert the relationships and the constraints among the classes.

### 3.0.1 Definition of Core Grid Ontology Classes and the Class Hierarchy

Based on the Grid model described in Section 3, we start with defining basic concepts. These concepts should correspond to classes that are the fundamental elements or the very important aspects of a Grid. We define 7 core classes of a Grid system from the abstract Grid model. They are: VO, GridResource, GridMiddleware, GridComponent, GridUser, GridApplication, GridService. The definitions of the core classes are explained in Tables 1,2,3. The general classes and architecture-specific classes can be seen in figure 2.

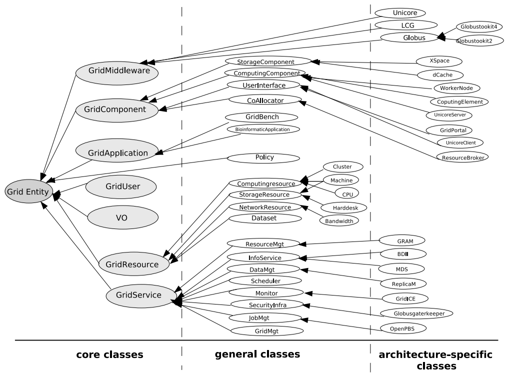


Figure 2: The Overview of the Core Grid Ontology Classes

| *Class* | *Description* | *Constraints* |
|---|---|---|
| *VO* | *a dynamic collection of distributed resources that are shared by a dynamic collection of users from one or more physical organizations.* | *1) has ID;<br>2) has some GridUsers registered;<br>3) has some GridResource accessible;<br>4) has a VOManager support;<br>5) has policy; including: policy of the VO, policy on users, policy on resources;* |
| *GridUser* | *a person who can access to a Grid.* | *1)hasID<br>2)registeredVO<br>3)gridEntry* |
| *GridApplication* | *an application that can run on Grids, complied with VO policies.* | *1)hasName<br>2)registeredVO<br>3)neededLib<br>4)coService* |

Table 1: The CGO core Classes (1)

| *Class* | *Description* | *Constraints* |
|---|---|---|
| *GridMiddleware* | *to provide transparent access to distributed Grid resources such as processing, network bandwidth and storage capacity.* | *1)hasName<br>2)releaseVersion<br>3)architectureType<br>4)hasComponent: hasInfoSys, hasSecurityInfra, hasResourceMgt, hasScheduler, hasMon<br>5)requiredService* |
| *GridComponent* | *a collection of Grid services and interfaces, which can provide access to Grid resources.* | *1)hasID<br>2)installedSoftware<br>3)requiredService<br>4)coService* |
| *GridService* | *a service on a Grid, which is software that carries out some task on behalf of yet another piece of software called a client.* | *1)hasID<br>2) hasPort<br>3)coService<br>4)status* |

Table 2: The CGO core Classes (2)

## 3.1 Defining Core Grid Ontology Properties

In order to represent the relationships and constrains among the ontology classes, we define the properties to provide the semantic meaningful for the Core Grid Ontology. In this paper, we just introduced the properties or slots that reflect the relationships among the

| Class | Description | Constraints |
|---|---|---|
| *GridResource* | *a Grid entity that is employed to fulfill a job or resource request. It could be: (1) all the computers, workstations that make up a Grid; (2) the communication networks connecting those computers; (3) all the data storage connected to a Grid, and the data on them. (4) all the other active components and networks connected to a Grid.* | *1) it can be identified in the Grid environment; 2) it must support at least one VO.* |

Table 3: The CGO core Classes (3)

high level ontology classes. The full properties can be founded in [1]. Table2 shows part of properties in the core Grid ontology.

- hasID: A name or identification of a Grid entity.
- supportVO: A Grid entity (e.g. GridResource, GridApplication) that supports a VO. Every Grid user, Grid application, and Grid resource should belong to at least one VO.
- hasComponent: A property is used to describe the relationship between GridMiddleware and GridComponent.
- hasPolicy: A set of rules for VOs, users, and resources sharing.
- registeredUser: The users that register in a specific VO.
- withService : A service that is needed by a Grid middlware, a Grid component, or a Grid application.
- coService: a sub property of the property withService, which is a co-operative service of other services.
- requiredService: a sub property of the property of withService, a service is required for a specific Grid component, an application, or other services.
- gridEntry: A property that predicates a entry where a user can access a Grid system.
- installedSoftware : The software installed in a Grid system.

### 3.2 Generating the Ontology Instances

After introducing the core Grid ontology, we represent a ComputingElement (CE) of the CY01-LCG2 Grid node within EGEE project to show how the ontology can help generate the knowledge-based information about a CE [3].

In the ontology, we defined constrains of the class ComputingElemen as CE:

$$\begin{aligned} \text{CE}\ &\exists\ \text{runningSevice (gridmanager)},\\ &\exists\ \text{runningSevice (Jobmanager} \sqcup \text{JobScheduler)},\\ &\forall(\text{VO}). \end{aligned}$$

It indicates that *Service* (gridmanager, jobmanager, and job scheduler) run on it. From the definition of the class *Gridmiddleware* and class *LCG*, we then can "*know*" that the jobmanager service is *PBS* and job scheduler service is *MAUI*. The Table 4 shows the CE description in OWL using the Core Grid Ontology.

```
......
  <ComputingElement rdf:ID="ce101.grid.ucy.ac.cy">
    <coService rdf:resource="#SSHD"/>
    <supportVO rdf:resource="#ATLAS"/>
    <supportVO rdf:resource="#SEE"/>
    <supportVO rdf:resource="#ALICE"/>
    <runningSevice rdf:resource="#maui"/>
    <operatingSystem rdf:resource="#Linux"/>
    <runningSevice rdf:resource="#PBS"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A Computing Element located in CY01-LCG2 site in Cyprus</rdfs:comment>
    <coService rdf:resource="#MPI"/>
    <supportVO rdf:resource="#BIOMED"/>
  </ComputingElement>
......
```

Table 4: A CE Instance in Core Grid Ontology

# 4 Conclusion

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 for, among other things, 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.

# References


[1] Core Grid Ontology. http://grid.ucy.ac.cy/grisen/coreonto.owl.

[2] Europe Data Grid. http://eu-datagrid.web.cern.ch/eu-datagrid/.

[3] S. Campana and A. Sciaba M. Litmaath. LCG-2 Middleware Overview. LCG Technical Document. https://edms.cern.ch/file/498079//LCG-mw.pdf.

[4] N.R.Shadbolt De Roure, D.Jennings, editor. *The Semantic Grid: Past, Present and Future*. IEEE, March 2005.

[5] 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.

[6] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Supercomputer Applications*, 2001.

[7] 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.

[8] 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.

[9] Ph. Wieder and D. Mallmann. UniGrids - Uniform Interface to Grid Services. In *7th HLRS Metacomputing and Grid Workshop*. Stuttgart, Germany, April 2004.