A Core Grid Ontology for the Semantic Grid *

Wei Xing Marios D. Dikaiakos
Department of Computer Science
University of Cyprus
CY-1678 Nicosia, Cyprus
{xing, mdd}@cs.ucy.ac.cy

Rizos Sakellariou
School of Computer Science
University of Manchester
M13 9PL Manchester, UK
rizos@cs.man.ac.uk

Abstract

In this paper, we propose a Core Grid Ontology (CGO) that defines fundamental Grid-specific concepts, and the relationships between them. One of the key goals is to make this Core Grid Ontology general enough and easily extensible to be used by different Grid architectures or Grid middleware, so that the CGO can provide a common basis for representing Grid knowledge about Grid systems, including Grid resources, Grid middleware, services, applications, and Grid users. The Core Grid Ontology is designed and developed based on a general model of Grid infrastructures, and described in the Web Ontology Language OWL. Such an ontology can play an important role in building Grid-related Knowledge bases and in supporting the realization of the Semantic Grid.

1 Introduction

In the Semantic Grid, Grid-related information and services are given a well-defined meaning, better enabling computers and people to work in cooperation [9]. Ontologies are among the key building blocks for the Semantic Grid. They define and determine the concepts, vocabularies of Grid entities, resources, capabilities and the relationships between them, with which any kind of content can become *meaningful* by the addition of ontological annotations.

A number of recent efforts have focused on Grid-related ontologies [5, 15, 6, 13, 8]. However, to the best of our knowledge, no existing ontology can be suitable for representing a Grid system. Most ontologies proposed so far, are Grid sub-domain specific, and have been developed for special purposes. Thus, they can be used for only certain Grid sub-systems. For instance, the Virtual Organization ontology (VOO) is developed for Grid Virtual Organization management [5]; it defines what a virtual organization is, especially but not exclusively in the context of Grid computing. VOO classes are mainly about policies and goals of a VO. Several important Grid concepts, such as Grid middleware, Grid application, and Grid resources, are not included. Certainly, this ontology is not applicable for representing Grid systems. Therefore, we design and develop the Core Grid Ontology for representing Grid systems, including fundamental aspects of a Grid system, such as Grid entities, Grid users, Grid applications, Grid resources, Grid middleware, etc. In order to make the CGO general, open and extensible, we design the CGO based on a general model for Grids, which is compatible to major Grid infrastructures [3, 4, 2, 12, 7]. Also we implement this ontology with the Web Ontology Language OWL as the description language for representing the CGO concepts and their relationships [14]. This paper presents our work on the design and development of the Core Grid Ontology.

The main problem for building an ontology for Grids is that there is currently a multitude of proposed Grid architectures and Grid implementations, which 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 of, or interests about 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, and relationships. One of our main goals is to make this Core Grid Ontology general enough and easily extensible to be used by different Grid architectures or Grid middleware, so that the CGO can provide a common basis for representing Grid knowledge about Grid systems, including Grid resources, Grid middleware, services, applications, and Grid users.

The remainder of this paper is organized as follows. In Section 2, we present the design and development of the

*This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)
0-7695-2585-7/06 $20.00 © 2006 IEEE

Core Grid Ontology. We illustrate how to build a Grid knowledge base using CGO with examples in Section 3. Finally, we conclude our paper in Section 4.

2 The Core Grid Ontology

A key design goal when proposing a Core Grid Ontology is to make it extensible and general enough to be used by, or incorporated in different Grid systems and tools. To address this challenge, we build an abstract, generic model of Grids as the object of our Core Grid Ontology (CGO). Key concepts of the CGO are derived from this model. Since the Grid model is an abstraction of different Grid architectures, it ensures that the concepts of the CGO are general enough and suitable for different Grid infrastructures and Grid middleware. We adopt the Web Ontology Language OWL as the description language for representing CGO concepts and the relationships among them [14]. OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web. Given the fact that OWL are W3C recommendations, the Core Grid Ontology is thus open, and compatible with other systems.

2.1 Building a Grid Model for CGO

The key challenge of building a Grid model is to capture a "right" abstraction for the Grid, which could be used to further specify Grid concepts, relations, and constraints. This abstraction must remain simple and should have a proper level of detail. It should also provide a general view of important aspects of Grids [10].

The Grid can be considered as a collection of Virtual Organizations and of different kinds of resources. Resources are organized and utilized 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 shared goals.

Therefore, we regard a Grid as a constellation of Virtual Organizations (VOs), which includes VOs, users, applications, middleware, services, computing and storage resources, networks, and policies of use. As shown in Figure 1, the proposed model is layer-structured, and is designed around a simple three-layer scheme. The top layer of the model includes multi-VOs, Grid Users, and Applications; Grid middleware and Grid services lie on the middle layer; the bottom layer includes the Grid resources. The Grid fundamental elements of each aspect (appeared in Figure 1(b)) can be *"located"* in a corresponding layer of the proposed model (in Figure 1(a)).

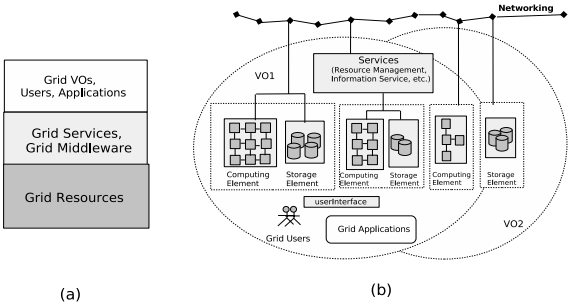

Figure 1. The Overview of the Proposed Grid Model

2.2 Choosing a Data Model and Description Language for the CGO

The Core Grid Ontology is designed to represent the 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 concepts and classes in the Core Grid Ontology [14]. OWL is actually developed as a vocabulary extension of the Resource Description Framework (RDF), namely, it takes RDF data model as its data model [11]. The RDF data model is a directed graph with labeled nodes and arcs; the arcs are directed from one node (subject) to another node (object). The object may be linked to other nodes (e.g. other classes) through properties. One key feature of this data model is that properties in RDF are defined globally, namely, 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. The characteristics of the RDF data model make the ontology easier to extend by adding new classes and properties (slots) into a defined class without any conflict with existing definitions. However, RDF can not describe resources in sufficient detail. For instance, there is no localized range and domain constraints in RDF. To avoid the weakness of the RDF, OWL comes with a larger vocabulary and stronger syntax than RDF.

2.3 The Design of the Core Grid Ontology

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 platform-specific extensions to it easily, in order to represent a platform-specific Grid, for instance, *GlobusToolkit4, GridPortal, XSpace, GRAM, MDS, BDII*.

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 [3, 4, 2, 12, 7].

2.3.1 Definition of Core Grid Ontology Classes and the Class Hierarchy

Based on the Grid model described in Section 2.1, 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 Table 1.

To describe a Grid system, *VO, GridMiddleware,* and *GridResource* are three vital, crucial aspects that define distinct features of a Grid according to the abstract model. And *GridUser, GridApplication, GridComponents, and GridService* are associated concepts of basic Grid entities. A *Grid user* registers in a *VO* in order to run a *Grid application*; and a *Grid application* must belong to a *VO* in which it can share the distributed resources of the VO. *Grid middleware* may include some *Grid components/Grid services* that provide functionalities. Each *Grid component* may have one or several *Grid services* together to perform the functions. Finally, *Grid resources* provide computing power, storage capability, network connection to Grids for executing user applications. Therefore, with these core classes we can represent a Grid system by the three aspects: (i) Which VOs does a Grid infrastructure support? (ii) What kind of middleware is it supported? (iii) What resources does it have? Subsequently, we define 24 general classes that correspond to general Grid entities referring to VO, Grid middleware, and Grid resource: *ComputingComponent, StorageComponent, UserInterface, Policy, ResourceMgt, InfoService, JobMgt, DataMgt, SecurityInfra, MonMgt* etc (see Figure 2). These classes represent important generic Grid entities that can be used to describe a Grid with more detailed information, such as: (a) What kind of applications are supported by a VO? (b) What is the policy of a VO? (c) How many and what kind of components does a Grid middleware have? (d) Which services are used to support Grid resource sharing?

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; 2) has some GridUsers registered; 3) has some GridResource accessible; 4) has a VOManager support; 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 2)registeredVO 3)gridEntry*
GridApplication	*An application that can run on Grids, complied with VO policies.*	*1)hasName 2)registeredVO 3)neededLib 4)coService*
GridMiddleware	*Software that provide transparent access to distributed Grid resources such as processing, network bandwidth and storage capacity.*	*1)hasName 2)releaseVersion 3)architectureType 4)hasComponent: hasInfoSys, hasSecurityInfra, hasResourceMgt, hasScheduler, hasMon 5)requiredService*
GridComponent	*A collection of Grid services and interfaces, which can provide access to Grid resources.*	*1)hasID 2)installedSoftware 3)runningService 4)requiredService*
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 2) hasPort 3)requiredService 4)status*
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) hasID: it can be identified in the Grid environment; 2) belongToVO: it must support at least one VO.*

Table 1. The CGO core Classes (1)

After describing general features of a Grid system, we need platform specific details about a Grid system in order to represent a Grid concretely. We introduce Grid platform specific classes to represent the entities of a specific Grid architecture. For instance, the MDS information service of Globus-2 is represented by the class MDS, which is a subclass of InfoService. Similarly, class BDII (the information service of EGEE) is a subclass of InfoService, representing the information service used in EGEE Grid [7]. Using the platform specific classes, a Grid system can be represented in a *consistent and meaningful* way.

In order to make CGO general and extensible, we intend to provide a class "framework" for representing a Grid system, instead of having a complete set of classes and properties of Grids. Thus, users can extend the CGO by adding their classes and properties on a "required-to-have" basis. Any required details of a specific Grid system can be represented by introducing new Grid architecture specific classes. Consequently, classes of the CGO can together establish a constructional foundation to represent any Grid entities in Grids, and Grids as well.

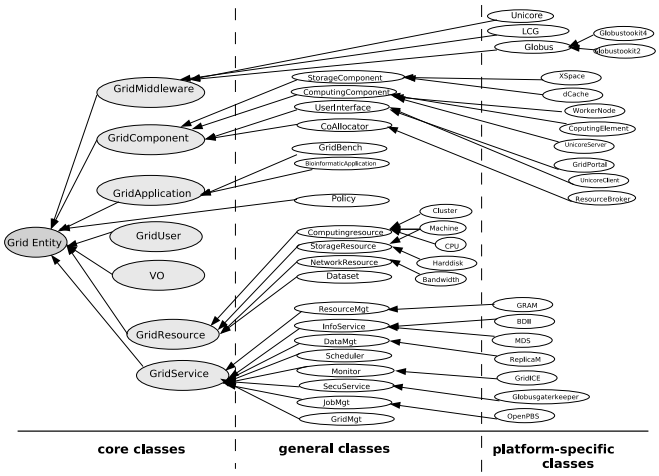

Figure 2. The Overview of the Core Grid Ontology Classes

2.3.2 The Properties of the Core Grid Ontology

In order to represent the relationships and constraints among the ontology classes, we define properties that provide the semantic meaning for the Core Grid Ontology classes. Properties of the CGO are practically defined according to the constraints of the CGO classes, For instance, the constraints of the VO class (See Table 1) are:

```
1) hasID;
2) has some GridUsers registered;
2) has some GridResource accessible;
3) has a VOManager support;
4) has policy; including: a)policy of the VO;
                          b)policy on resources;
                          c)policy on users.
```

Consequently, we can define four properties of the CGO, which are:

```
<owl:ObjectProperty rdf:ID="hasID">
  <rdfs:domain rdf:resource="#GridEntity" />
  <rdfs:range  rdf:resource="#URI" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#registeredUser">
    <rdfs:domain rdf:resource="#VO"/>
    <rdfs:range rdf:resource="#GridUser"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasPolicy">
    <rdfs:range rdf:resource="#Policy"/>
    <rdfs:domain rdf:resource="#VO"/>
  </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="#requiredSevice">
    <rdfs:subPropertyOf rdf:resource="#withService"/>
    <rdfs:domain rdf:resource="#GridEntity"/>
    <rdfs:range rdf:resource="#GridService"/>
</owl:ObjectProperty>
```

According to the constraints of the core classes in Table 1, we define a set of key properties of the CGO (the full version of the defintion of the CGO properties can be found in [1]) as follows:

```
<owl:DatatypeProperty rdf:about="#hasName">
    <rdfs:domain rdf:resource="#GridEntity"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The name of a Grid entity</rdfs:comment>
  </owl:DatatypeProperty>
```

```
<owl:ObjectProperty rdf:about="#needLib">
    <rdfs:domain rdf:resource="#GridEntity"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Grid application need libiary to run.</rdfs:comment>
    <rdfs:range rdf:resource="#Lib"/>
  </owl:ObjectProperty>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The type of Grid middleware.</rdfs:comment>
    <rdfs:domain rdf:resource="#GridEntity"/>
    <rdfs:range rdf:resource="#GridMiddleware"/>
  </owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="#releaseVersion">
    <rdfs:domain rdf:resource="#GridMiddleware"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The realeased version of a Grid middleware.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:about="#hasComponent">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A Grid has different components.</rdfs:comment>
    <rdfs:domain rdf:resource="#GridMiddleware"/>
    <rdfs:range rdf:resource="#GridComponent"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasPort">
    <rdfs:domain rdf:resource="#GridService"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The port number of a Grid service.</rdfs:comment>
    <rdfs:range rdf:resource="#Port"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="#state">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdfs:domain rdf:resource="#GridService"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The state of a Grid service.</rdfs:comment>
</owl:DatatypeProperty>
......
```

Any other properties can be added on demand. There are two methods to add new properties. One is to extend the key properties, which are regarded as super properties. A new property can be defined as a sub-property of a super property. For instance, the key property *withService* can be extended to two other sub properties: *requiredService* is used to specify the services that are required by Grid applications and Grid components; *coService* is used to define the co-operative relationship between Grid services. Another method is to add a new property directly that can be used for describing a desired feature of classes in the CGO. In CGO, properties are defined globally, that is, they are not encapsulated as attributes in class definitions. Therefore, it is possible to define new properties that apply to an existing class without changing that class. For example, we define that class *CPU* has key properties: *model*, *type*, and *speed*; later, if the information about CPU price is also needed, we can add *pricePerCPUTime* property into the class *CPU*. This makes the CGO flexible and extensible to adopt any new features of Grid entities.

3 Representing Grid Entities using the CGO

Based on the defined classes and properties of the CGO, we can represent any particular Grid entities. For example, the Computing Element, a component of the EGEE infrastructure, can be described as: (a) Computing Element is a Grid component that provides access to computing resources. (b) A Computing Element is comprised of one or more similar machines managed by a JobMgt, and a Scheduler service.

According to the definition, we can first define a new class ComputingElement with its constraints: (1) a ComputingElement must support at least one VO; (2) a Com-

putingElement must contain a SecurityInfra service; (3) a ComputingElement must contain a JobMgt service and Scheduler service. Then, these three restrictions can be described in Description Logics as follows:

$$\textbf{ComputingElement} \sqsubseteq \exists \text{ supportVO } \text{VO} \sqcap \exists \text{ requiredSevice SecurityInfras} \sqcap \exists \text{ requiredSevice (JobMgt} \sqcup \text{Scheduler)}.$$

So, we can describe the class *ComputingElement* by the defined CGO classes in OWL as follows:

```
<owl:Class rdf:ID="ComputingElement">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#JobMgt"/>
            <owl:Class rdf:about="#Scheduler"/>
          </owl:unionOf>
        </owl:Class>
      </owl:someValuesFrom>
     <owl:onProperty rdf:resource="#requiredSevice"/>
    </owl:Restriction>
  </rdfs:subClassOf>
...
</owl:Class>
```

After that, we can generate instances of class ComputingElement. In the CY01-LCG2 Grid node of the EGEE Grid infrastructure [7], we have a computing element named ce101.grid.ucy.ac.cy. We describe the ce101 based on the definition of the class ComputingElement. From the information service (i.e. BDII), we can retrieve the information that the *GridMiddleware* of the ce101 node is *LCG*; it supports three VOs, i.e. ATLAS, BioMed, LHCB; and the JobMgt service is *openPBS*; the Scheduler service is *MAUI*. Besides, we can fetch the information about totalCPU from the openPBS server. Finally, we can create an instance of the class ComputingElement as follows:

```
<ComputingElement rdf:ID="ce101.grid.ucy.ac.cy">
 <hasName xml:lang="en">ce101.grid.ucy.ac.cy</hasName>
 <hasID rdf:resource="#IP_CE101_UCY"/>
 <belongToVO rdf:resource="#Biomed"/>
 <belongToVO rdf:resource="#SEE"/>
 <belongToVO rdf:resource="#Dteam"/>
 <installedSoftware rdf:resource="#Scientific_Linux_303"/>
 <installedSoftware rdf:resource="#LCG_2.6.0"/>
 <runningServices rdf:resource="#openpbs_ucy"/>
 <runningServices rdf:resource="#maui_ucy"/>
...
</ComputingElement>
```

4 Conclusions and Future Work

In this paper, we presented our work towards building a Core Grid Ontology (CGO). We first introduced an abstract model of Grid. After that, we designed the CGO 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. 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 support knowledge-based queries. Since the ontologies/knowledge will be stored in multi-Grids enviroment, we need a suitable OWL query language and distributed query mechanism to query those distributed knowledge efficiently.

References

[1] CGO OWL. http://grid.ucy.ac.cy/grisen/cgo.owl.

[2] European DataGrid. http://eu-datagrid.web.cern.ch/eu-datagrid/.

[3] Globus Toolkit. http://www.globus.org/toolkit/.

[4] Unicore Grid. http://www.unicore.org.

[5] P. Alper, O. Corcho, I. Kotsiopoulos, P. Missier, S. Bechhofer, D. Kuo, and C. Goble. S-OGSA as a Reference Architecture for OntoGrid and for the Semantic Grid. The 3rd GGF Semantic Grid Workshop, GGF16, 2006.

[6] J. Brooke, K. Garwood, and C. Goble. Semantic Matching of Grid Resource Descriptions. In M.D.Dikaiakos, editor, *Proceedings of Second European Across Grids Conference (AXGrids 2004), LNCS 3165*, pages 240–249, Nicosia, Cyprus, 2004. Springer-Verlag.

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

[8] C.Wroe, C.Goble, M.Greenwood, P.Lord, S.Miles, J.Papay, T.Payne, and L.Moreau. Automating Experiments Using Semantic Data on a Bioinformatics Grid. *IEEE Intelligent Systems*, 19(1):48–55, 2004.

[9] D. De Roure, N. R. Jennings, and N. R. Shadbolt. The Semantic Grid: Past, Present and Future. In *Proceedings of the IEEE*, volume 93(3), ISSN:0018-9219, pages 669–681. IEEE, March 2005.

[10] M. Dikaiakos and A. Artemiou. Navigating the grid information space: Design and implementation of the ovid browser. Technical Report TR-2004-07, Department of Computer Science, University of Cyprus, December 2004.

[11] G. Klyne and J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation, February 2004.

[12] J. Marco and et al. First Prototype of the Crossgrid Testbed. In *Proceedings of First European AcrossGrids Conference (AXGrids 2003), LNCS 2970*, pages 67–77, Santiago de Compostela, Spain, 2003. Springer-Verlag.

[13] S. Miles, J. Papay, V. Dialani, M. Luck, K. Decker, T. Payne, and L. Moreau. Personalised Grid Service Discovery. *IEE Proceedings Software: Special Issue on Performance Engineering*, 150(4):252–256, 2003.

[14] P. Patel-Schneider, P. Hayes, and I. Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax*. World Wide Web Consortium, February 2004.

[15] R. Stevens, A. Robinson, and C. Goble. myGrid: Personalised Bioinformatics on the Information Grid. In *11th International Conference on Intelligent Systems in Molecular Biology*, 2003.