GRID RESOURCE ONTOLOGY: A KEYSTONE OF SEMANTIC GRID INFORMATION SERVICE

O. Pospishnyi
National Technical University of Ukraine “Kyiv Polytechnic Institute”,
37, Peremohy Ave., Kyiv 03056 Ukraine
pospishniy@kpi.in.ua

Abstract: This work presents a set of ontologies to facilitate the adoption of semantic technologies in Grid middleware and to provide a solid foundation for development of semantic Grid information services. The scope of the proposed ontologies includes a description of Grid resources and functionalities, interrelationship of all Grid components, subsystems and services, definition of administrative and usage policies, etc. The work also presents a set of tools to populate the provided Grid resource ontologies with real-data instance assertions by interrogating a traditional Grid information service and discusses the benefits of such representation of Grid resources.

Keywords: semantic grid, information, grid resource ontology, semantic technologies, middleware.

Introduction

Grid computing already proved to be an effective and powerful instrument for modern data-intensive science and engineering. The idea was simple, yet very powerful - to integrate geographically dispersed computing resources from multiple administrative domains through Grid middleware and to provide a simple and uniform access method to those resources.

Due to the fact, that all Grid components are unique with respect to their hardware and software composition, all Grid systems are inherently extremely heterogenic. That is why a robust, scalable and reliable information system is a cornerstone Grid component. It acts as a nervous system, providing characterization, aggregation and indexing of available Grid resources.

All Grid clients perform resource discovery through querying the information system and rely on its resource accounting functionality during matchmaking and brokering procedures. Unfortunately, due to the fact that widely-used Grid middleware such as Globus, gLite, Unicore and ARC relies only on inflexible attribute-based resource matching, a lot of capable resources are often overlooked or incorrectly allocated. Difficulties arise due to the considerable complexity of adopted information models. Numerous types of Grid resources are described by hundreds of highly technical parameters, making it a formidable task for the user to navigate and comprehend such models. This complexity is known to be one of the main reasons for incomplete or malformed queries to information services, resulting in many Grid job failures.

Meanwhile, ontologies and tools that are being developed for Semantic Web offer a possibility to express knowledge about a particular domain and to infer the implicit information from it. Exploiting such methods in Grid computing allows for better resource management and enables middleware to work more intelligently and efficiently.

This initiative has led to the emergence of Semantic Grid vision. It is an extension of the traditional Grid in which information and services are given well-defined meaning through machine understandable descriptions which helps in making intelligent scheduling decisions with a high degree of automation (Roure et al., 2003).

During the past few years lots of effort and research have been placed in the development of Semantic Grid technologies. However, all previous works share some common drawbacks. They either: rely on semantically-weak RDF/RDFS models; ad-hoc generate Grid resource ontologies based on data retrieved from external sources or they do not consider further extension of ontologies, their reuse and collaborative refinement.

We try to address mentioned drawbacks in our vision of Semantic Grid information service (Pospishnyi and Stirenko, 2012) and Grid resource ontology presented here is one of our key elements.
In this paper we focus on the Grid resource ontology (GRO) that provides a solid foundation for semantic Grid components and opens up new opportunities to qualitatively improve modern Grid infrastructure, making it smarter and simpler to use.

The purpose and scope of GRO

Grid resource ontology (GRO) presented in this paper is a conceptual information model for Grid entities described using Web Ontology Language. It is designed to be a top-level ontology for Semantic Grid information services, providing a description of core Grid resources at the conceptual level. Moreover, although GRO could be used stand-alone, it is expected to be extended by other domain ontologies that would capture additional background knowledge about Grid resources and their usage. In this case GRO provides grounding and shared foundation for all such ontologies.

In order to achieve mentioned purpose we base our ontology on GLUE Schema (Andreozzi, 2007, 2009). The Grid Laboratory Uniform Environment (GLUE) Schema was developed to provide an abstract model for Grid resources, describing the capabilities of the resource providers. It also provides mappings to concrete schemas (LDAP, SQL, XLS) that can be used in various Grid Information Services. As of now, almost all widely-used Grid middleware products rely on GLUE Schema and its support is considered to be almost mandatory.

GLUE provides a description of core Grid resources at the conceptual level by defining an information model, which is an abstraction of the real world into constructs that can be represented in computer systems. The proposed information model is not tied to any particular implementation and is based on the experience of several modeling approaches being used in current production Grid infrastructures (e.g., GLUE Schema 1.x (Andreozzi, 2007), NorduGrid schema (Konya and Johansson, 2013), Naregi model (Memon, 2013).

GRO is fully compatible with developed GLUE Schemas, providing a new semantically-enhanced OWL mapping for it.

Four version of GRO

Table 1 summarizes all four available version of Grid resource ontology. GRO version 1 and 1e relate to earlier GLUE Schema specification version 1.3 (Andreozzi, 2007). Consequently, GRO 2 and 2e are based on the newer and currently latest GLUE 2.0 Schema (Andreozzi, 2009). Table 1 also mentions Description Logics expressivity that corresponds to the particular version of GRO. Both GRO 1e and GRO 2e were designed to fit the OWL 2 EL language profile and benefit from the application of tractable consequence-based reasoning procedures.

<table>
<thead>
<tr>
<th>Ontology version</th>
<th>GLUE Schema version</th>
<th>DL expressivity</th>
<th>Ontology metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRO 1</td>
<td>1.3</td>
<td>SHIQ(D)</td>
<td></td>
</tr>
<tr>
<td>GRO 1e</td>
<td>1.3</td>
<td>ELD</td>
<td>Classes: 65</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Object Properties: 33</td>
</tr>
<tr>
<td>GRO 2</td>
<td>2.0</td>
<td>ALFQ(D)</td>
<td></td>
</tr>
<tr>
<td>GRO 2e</td>
<td>2.0</td>
<td>LDL</td>
<td>Classes: 68</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Object Properties: 150</td>
</tr>
</tbody>
</table>

All version of Grid resource ontology strongly depend on both, object properties and datatype properties in order to describe Grid entities. Thus any ontology reasoner that is planned to be used with GRO must support OWL datatype expressions.

All GRO ontologies were developed in Protégé ontology editor and verified with real-world instance data which will be discussed later. See Section ‘Availability’ for information on where all mentioned ontologies could be downloaded.
Ontology content and structure

*GRO version 1 and 1e*

Figure 1 provides an overview of overall class structure of GRO 1 ontology.

**GridEntity** is a superclass concept for all core Grid concepts, divided among three main categories: **CoreEntity**, **ComputingResource** and **StorageResource**.

**Site** is a **CoreEntity** that represents an administrative concept used to aggregate a set of resources and services that are installed and managed by the same organization or a set of persons. **Service** is another **CoreEntity** that provides an abstracted, logical view of actual software components that should be formally defined in terms of the messages exchanged between provider and requester.

**ComputingResource** class outlines a model for an abstract view of computing resources at the Grid level. **ComputingElement**, a central concept of this model, aims to describe the computing service that manages computing jobs and offers them a necessary execution environment. The **ComputingElement** has associated a description of its static characteristics (**CEInfo**), a status changing frequently (**CEState**), a general use policy (**CEPolicy**), a set of authorized users or groups (**AccessControlBase**), group-specific attribute values (**VOView**) and jobs running on it (**Job**).

A **Cluster** is an aggregation entity for representing a heterogeneous set of resources (computing nodes belonging to the same cluster may have different hardware and software configuration) managed by a local management system. The **SubCluster** entity depicts a homogeneous set of hosts as regards a selected number of attributes.

**StorageResource** outlines a model for abstracting Grid storage resources. The **StorageElement** is the core concept of this model that identifies the group of services, protocols and data sources responsible for a storage resource. The entities **AccessProtocol** and **ControlProtocol** are used to publish the endpoint protocols related to the **StorageElement**. The **Capability** attribute present in **StorageArea**, **AccessProtocol** and **ControlProtocol** is meant to publish implementation specific hints to a client.

All aforementioned domain concepts and some others are presented on Figure 2. All domain concepts are modeling some aspects of Grid entities, providing some specific descriptions of its components or related minor entities. Because of space limitations, we would not disclose their detailed description although for the most part all domain concepts are straightforward and self-explanatory. See GRO ontology file for specific details.

Lastly, GRO 1 and 1e contains an **Enumeration** class that outlines several support concepts that have no description and act solely as a named enumeration of individuals. Figure 3 lists all enumerative concepts.

By design, no instances of **Enumeration** concepts are available in GRO. They are created later, when ontology is being populated with resource assertions, forming a Grid resource knowledge base. For more details see next section.
Each concept of the ontology from the GridEntity and DomainConcept hierarchy a detailed description is given with the help of restrictions on object and data properties.

![Fig. 2. Domain concepts hierarchy](image)

**GRO version 2 and 2e**

Figure 4 introduces the main entities of GRO 2 and 2e ontologies. They capture the core concepts relevant in the modern Grid environment and could be used to derive specialized information models.

The **Entity** class is an abstract root concept that holds metadata (such as ID, creation date and validity) for all derived concepts. An **Extension** class provides a general mechanism to add key/value pairs to arbitrary **Entity** when suitable specific attributes are not present.

The **Location** class is introduced to provide a simple way to express geographical information about a certain **Domain** or **Service**. The **Contact** class, on the other hand, enables the establishment of communication with a person or a group responsible for a particular **Domain** or **Service**. The **Domain** is an abstract concept used to model and identify a collection of actors that may play certain roles in a Grid system. Its derived subclasses **AdminDomain** and **UserDomain** are used to represent a hierarchical administrative structure and Virtual organization (VOs), respectively.

The **Service** class enables the identification and characterization of Grid capabilities, providing an abstracted, logical view of actual software components that participate in the creation of Grid functionalities. A **Service** concept aggregates the following modeling concepts: **Endpoint**, **Share**, **Manager** and **Resource**. The **Endpoint** class models a network location that can be contacted to access certain functionalities based on a well-defined interface. The **Share** class captures a concept of a constrained usage...
of Grid functionalities and resources. The Manager concept provides an abstraction for a software component that locally manages one or more resources. And finally, the Resource class is an abstract entity introduced to identify and model hardware capabilities.

Further, ComputingService, ComputingEndpoint, ComputingManager and ExecturionEnvironment are specialized derived concepts from abovementioned abstract classes that represent Grid computational capabilities. Similarly, StorageService, StorageEndpoint, StorageManager and DataStore are specialized concepts representing Grid storage capacities.

The Activity class represents units of work which are submitted to Service via Endpoint. Its only derivative specialization is the ComputingActivity class, which is used to describe Grid jobs, its properties and state as seen by the local batch system, together with some Grid-level information.

The Policy concept is introduced to model statements, rules or assertions that define the correct or expected behavior of Grid entities. Two specializations, namely AccessPolicy and MappingPolicy are available to introduce associated policies for Endpoint and Share entities, respectively.

Because of space limitations, we leave out remaining ontology concepts either because they are self-explanatory or serve to model some particular aspects of computational or storage Grid services. All GRO concepts are heavily annotated, providing sufficient amount of information for their understanding.

Furthermore, GRO 2 and 2e rely on enumerative datatypes similar to Enumeration concepts in GRO 1. Figure 5 lists all such concepts, subclasses of DataTypes entity. Unlike in GRO 1, versions 2 and 2e contain 178 predefined individuals that instantiate mentioned enumerations.

**Reasoning with Grid Resource Ontology**

GRO ontologies provide only a set of terminological axioms, or so called TBox. To be of any practical use, ontology needs to be complemented by a set of instance assertions that will represent individual physical resources – ABox. Together ABox and TBox constitute Grid resources knowledge base, which can further be used by Grid clients.

For these purposes we have developed and made available a software module that could populate an ABox of Grid resource ontology, by referring to an actual Grid information service. Separate software modules were created for GRO version 1/1e and GRO 2/2e. The generation of resource knowledge base proceeds as follows:

1. Establishing a connection with a specified Grid information service (i.e. top-BDII) using an LDAP protocol.
2. Loading a specified Grid resource ontology file using the OWLAPI library.
3. Creating a new ontology file which would contain all of the ABox assertions. Linking it to the specified GRO ontology using ontology import axiom.
4. Reading all necessary data from the Grid information service and populating ABox ontology with the individual assertions. Performing all necessary data interpretation, conversion and validation.
5. Closing all connections and flushing buffers.

![Diagram showing GRO 2 and 2e enumerative datatypes](image)

**Fig. 5.** GRO 2 and 2e enumerative datatypes

We tested our KB generation software on the LHC Computing Grid (WLCG), the most ambitious Grid system to date, which serves to carry out experiments at the Large Hadron Collider. Application is not only limited to this particular Grid infrastructure and can be used to import data from any other Grid system that has BDII- or MDS-based information service.

As soon as we establish resource knowledge base we can interrogate it, applying wide range of benefits inherent to the semantic technologies.

*Firstly*, we can bring in some additional background knowledge into a querying process, greatly simplifying and empowering it. Because Grid is an inherently diverse multi-organizational system, it is reasonable to allow some way of introducing additional domain-specific knowledge to the information system. This would allow clients to interact with the Grid environment through a domain-aware context, relying on some additional knowledge, presented inside his Virtual organization (VO).

Reflect upon the following example. Consider a Grid virtual organization working in the field of high energy physics. Populated GRO ontology could provide some basic view on the available Grid resources (directly, or through semantic information service). Although, it might be cumbersome for the members of this organizations to perform resource discovery relying only on universal resource model, presented by the GLUE and GRO. For these reasons we suggest a formalization of relevant domain knowledge in the form of ontologies and extension of GRO-based resource knowledge base with their help. For our hypothetical virtual organization such additional ontologies could contain descriptions of particular experiments activities, instruments or software used within this VO, in context of the required Grid resources or services.

*Secondly*, applying similar ontology extension mechanism, we can introduce additional common knowledge and arbitrary user assertions to semantic information service. This should considerably simplify user’s interaction with the Grid system, providing a set of definitions that would hide complex informational model of Grid resources. For example, consider the following assertions:

### UK_Site ≡ Site ⊓ hasLocation.([Location ⊓ hasName.[pattern .*], UK])

### Idle_CE ≡ ComputingElement

- ⊓ hasState.((CState ⊓ hasRunningJobs.(= 0) ⊓ hasWaitingJobs.(= 0)
  ⊓ hasFreeJobSlots.(> 0)) ⊓ hasState.((CState ⊓ hasStatus . Production))

### x64_Cluster ≡ SubCluster ⊓ describedBy.(hasPlatformType.(= "x86_64")

- describedBy.(hasRAMSize.(≥ 4096, ≤ 8192))

---

**UK Site**

- **Site**
  - **hasLocation**
    - **Location**
      - **hasName**
        - **pattern**
          - **."*"**
          - **UK**
They introduce new user-friendly concepts such as “Grid site located in UK”, “Idle Computing Element” and “Hi-end x64 Cluster” that hide all unnecessary underlying complexity. Of course, such ontologies could be quite extensive and can be shared among Grid users and virtual organizations.

Thirdly, we consider applying an informational complement method to augment Grid resource knowledge base. With its help we can deduce additional, implicit or undefined information about Grid resources themselves, applying ontological knowledge to the already available information. This way it becomes possible to clarify Grid resource characteristics, judging by what is already known about them.

And finally, we view ontologies as a powerful validation mechanism for both client queries and resource advertisements. This would allow semantic information services to check incoming client queries and resource assertions for any logical inconsistency. This method could potentially significantly reduce the amount of misleading information about Grid resources presented in the system and could inform users about their malformed or pointless inquiries, providing some reasonable explanations for it.

Of course, none of the abovementioned benefits could be achieved without effective and reliable reasoning tools. It is clear, that for the most Grid systems GRO knowledge base could grow to the point, where it could not be effectively processed by the state of the art tableaux reasoners. Such drawbacks motivated us to introduce modified version of Grid resource ontologies, namely GRO 1e and GRO 2e, that could benefit from tractable consequence-based reasoning procedures. Both of the mentioned ontologies are related to the OWL 2 EL profile, making it possible to perform reasoning tasks in time that is polynomial w.r.t. the size of the ontology.

For high-performance processing of GRO knowledge bases we modified ELK reasoner (Kazakov et al., 2011) to introduced sufficient support for datatype expressions. For additional information see (Pospishnyi, 2012).

Availability
All versions of Grid resource ontologies are freely available online as OWL files in RDF/XML format: http://grid-ontology.googlecode.com/
Knowledge base generation software and all their source codes are also available online and could be freely modified and used by everyone: https://grid-ontology.googlecode.com/svn/trunk/

References