A Grid-Enabled Digital Library System for Natural Disaster Metadata*

Wei Xing[1], Marios D. Dikaiakos[1], Hua Yang[2], Angelos Sphyris[3], and George Eftichidis[3]

[1] Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus
{xing, mdd}@ucy.ac.cy
[2] Department of Computer Science, Xian Institute of Post and Telecommunications, 710061 Shannxi, China
yanghua@xiyou.edu.cn
[3] Algosystems SA, 206 Syggrou Ave, 176 62, Kallithea (Athens), Greece
{asphyris, geftihid}@algosystems.gr

Abstract. The need to organize and publish metadata about European research results in the field of natural disasters has been met with the help of two innovative technologies: the Open Grid Service Architecture (OGSA) and the Resource Description Framework (RDF). OGSA provides a common platform for sharing distributed metadata securely. RDF facilitates the creation and exchange of metadata. In this paper, we present the design and implementation of a Grid-based digital library for natural-disaster research metadata. We describe the EU-MEDIN RDF schema that we propose to standardize the description of natural-disaster resources, and the gDisDL Grid service-based architecture for storing and querying of RDF metadata in a secure and distributed manner. Finally, we describe a prototype implementation of gDisDL using the Jena RDF Library by HP and the Globus 3 toolkit.

1 Introduction

European R&D projects and other related activities focusing on Natural Hazards and Disasters (earthquakes, floods, forest fires, industrial hazards, landslides, and volcano eruptions) produce results in the form of explicit or tacit knowledge represented by reports, project deliverables, data-sets derived from field work, interesting training and dissemination materials, etc. These artifacts are usually published and described in Web sites maintained by project partners during the duration of the respective projects. Following project completion, however, project teams dissolve and Web-site maintenance and support gradually fade out. Hence, general-purpose search engines are used to search for past-project results. Nevertheless, search-engine query results provide large numbers of unrelated links. Furthermore, hyperlinks pointing to potentially useful material do not come with references or additional links to adequate information describing

* Work supported in part by the European Commission under the EU-MEDIN project (grant agreement no. EVG1-CT-60003).

the "value" of identified resources. Consequently, the identification of and access to useful information and knowledge becomes very difficult. Effectively, valuable knowledge is lost and it is practically impossible to find and take advantage of it.

To cope with this situation and to make research artifacts in the field of natural disasters widely and easily available to the research community, the European Commission has commissioned to Algosystems S.A. the development of a thematic Web portal to support the storage and retrieval of metadata pertaining to results of research in natural disasters [3]. Project-related metadata can be inserted via a Web interface to a back-end database. Interested researchers can use the "EU-MEDIN" portal to query the database and search for project-artifacts. This approach, however, is based on a platform-specific solution. A change in database technology or a need for upgrading the hardware platform or the operating system may dictate the transformation and storage of metadata in a different data model, format, and database system, hence incurring significant cost. Furthermore, the existence, update, and maintenance of the particular site is guaranteed only during the life-cycle of the project that undertook the portal's design, development, and maintenance. Also, the functionality of the site requires that end-users have to connect through the Web and register their metadata centrally; this makes it difficult, however, to extract all metadata related to a particular project and possibly submit them to a third party in batch. Last, but not least, there is a need to describe the metadata in a common and open format, which can become a widely accepted standard; the existence of a common standard will enable the storage of metadata in different platforms while supporting the capability of distributed queries across different metadata databases, the integration of metadata extracted from different sources, etc.

In this paper, we present **gDisDL**, a system designed to address some of the problems mentioned above. Our approach comprises:

– A schema for describing project-related metadata in a platform-independent form, using the *Resource Description Framework* (RDF). RDF is a general framework for describing metadata of Internet resources and for processing these metadata; it is a standard of World Wide Web Consortium (W3C). RDF supports the interoperability between applications that exchange machine-understandable information on the Web.
– A digital library system enabling the collection and storage of RDF-encoded metadata in distributed repositories, and the retrieval thereof from remote sites. This library is implemented as a Grid-service architecture comprising a set of Grid services, which allow the storage, management, and query of RDF metadata in a secure and distributed manner. To develop the library we use the Globus Toolkit 3 [18] for programming Grid services and the Jena toolkit [1] for handling RDF data.
– A set of graphical-user interfaces developed in Java to enable authorized end-users to create RDF metadata for natural-disaster research artifacts and to conduct keyword-based searches in RDF repositories.

The remaining of this paper is organized as follows. In section 2, we introduce Semantic Web technologies and Grid that used in our work and review related work. Section 3 describes matadata representation of the gDisDL system. We present the design challenges and architecture of the gDisDL system in section 4. Finally, we conclude our paper in Section 5.

2 Background

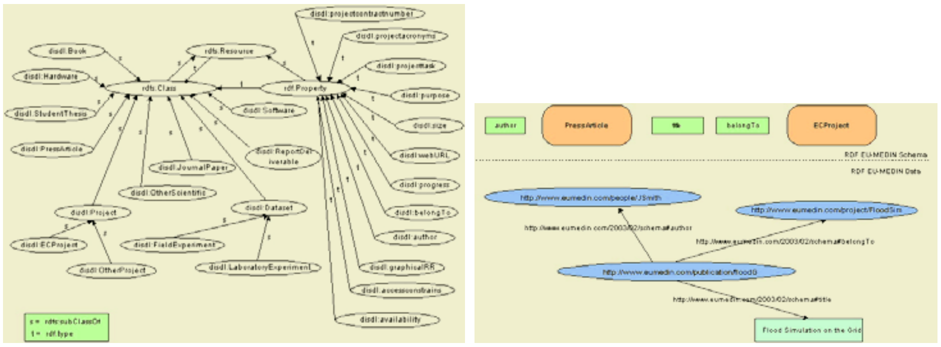
Resource Description Framework (RDF): The Resource Description Framework (RDF) is a language mainly used for representing information about resources on the World Wide Web [15]. In particular, it is intended for representing metadata about documents or other entities (e.g, Web resources, software, publications, reports, image files etc), such as the title, author, modification date, copyright, and licensing information. Although originally intended to be used on Web resources, RDF is capable of representing information about things that can be identified on the Web, even when they cannot be directly retrieved from the Web [15]. This capability makes RDF an appropriate metadata schema language for describing information related to the various results and outcomes of natural-disaster research. RDF is intended for situations in which information needs to be processed by applications, rather than only being displayed to people. RDF provides a common framework for expressing information and can thus be exchanged between applications without loss of meaning. Since it is a common framework, application designers can leverage the availability of common RDF parsers and processing tools. The ability to exchange information between different applications means that the information can be made available to applications other than those for which it was originally created [15].

Jena: Jena is a Java toolkit for building Semantic Web applications [1]. The *Jena Java RDF API* is developed by HP Labs for creating and manipulating RDF metadata. It mainly comprises:1)The "Another RDF Parser" (ARP), a streaming parser suitable for validating the syntax of large RDF documents. 2)A persistence subsystem, which provides persistence for RDF metadata through the use of a back-end database engine. It supports RDQL queries. 3)The RDF query language (RDQL), which is an implementation of an SQL-like query language for RDF. Jena generates RDQL queries dynamically and executes RDQL queries over RDF data stored in the relational persistence store.

Open Grid Service Architecture: Grid supports the sharing and coordinated use of diverse resources in dynamic, distributed "Virtual Organizations" (VOs)[**?**]. The Open Grid Services Architecture (OGSA) is a service-oriented Grid computing architecture, which an extensible set of Grid services that may be aggregated in various ways to meet the needs of VOs[**?**]. OGSA defines uniform Grid service semantics and standard mechanisms for creating, naming, and discovering Grid services. Web service technologies, such as XML, SOAP, WSDL, UDDI,etc., are adopted to build up the Grid services infrastructure. A Grid service is a Web service that provides a set of well-defined interface and that follows specific conventions [**?**]. The interface and behaviors of all Grid services is described by GWSDL[18].

3 Metadata Elicitation

Metadata is structured data about data. The goal of the gDisDL system is to support the storage and retrieval of metadata that pertain to a variety of results derived from



(a) Class Hierarchy for the EU-MEDIN RDF Schema

(b) Example of an EU-MEDIN RDF Schema

Fig. 1. EU-MEDIN RDF Schema

research in natural disasters (earthquakes, floods, forest fires, industrial hazards, landslides, volcano eruptions, etc). To this end, we need a metadata language defined in a common and open format, that will: (i) Promote the standardization of natural disaster metadata, while at the same time allowing future extensions; (ii) Enable the storage of metadata in different platforms according to a common, standard schema; (iii) Support the interoperability of different metadata repositories; in particular the specification of queries and the execution thereof upon different metadata databases.

We consider artifacts derived from natural-disaster research projects as resources whose properties will be represented in RDF. To specify the metadata for those artifacts, we conducted a detailed requirements analysis and came up with a classification containing 17 distinct resource classes: "EC project," "Event," "Journal paper," "Software," "Student Thesis (MSc or PhD)," "Other scientific paper," "Report/deliverable," "Web site," "Press article," "Book," "Other project," "Field experimental dataset," "Laboratory experimental dataset," "Spatial digital dataset," "Hardware," "Media presentation," and "Unclassified activity." The properties of and the relationships between class instances were defined accordingly.

We used the RDF Schema [9] to describe the identified set of RDF classes, properties, and values. The resulting schema is called EU-MEDIN RDF schema and represents our proposed metadata standard for natural-disaster research resources. Figure 1(a) shows the class hierarchy of the EU-MEDIN RDF schema.

Below, we give an example extracted from the EU-MEDIN RDF schema. This excerpt of our schema includes two classes, *Press article*, and *EC Project*, and three properties, *author, belongTo* and *name*. Using those classes and properties, we can describe in RDF the following piece of knowledge: "John Smith wrote a Paper, which belongs to FloodSim project. The paper's title is "Flood Simulation on the Grid" in RDF. Figure 1(b) shows part of the schema definition and the RDF description of the example, presented as an RDF graph.

4 gDisDL System Design

gDisDL system is a Grid service-oriented system, which consists of the following components: the *Data Aggregator*, the *Searcher*, the *gDisDL Store*, and the *Data Manager*. As shown in Figure 2, the *gDisDL* system comprises a number of geographically distributed *gDisDL* nodes. Each node should includes a *Data Aggregator*, a *Searcher*, and a *gDisDL Store*. The *Data Aggregator* collects, validates, and encodes metadata in RDF; the *Searcher* is designed for querying RDF metadata; the *gDisDL Store* is used to store RDF metadata; the *Data manager* manages the RDF metadata maintained in the gDisDL stores. Finally, the *Editor* and the *Searcher GUI* are client tools enabling users to interacting easily with gDisDL through a graphical-user interface.

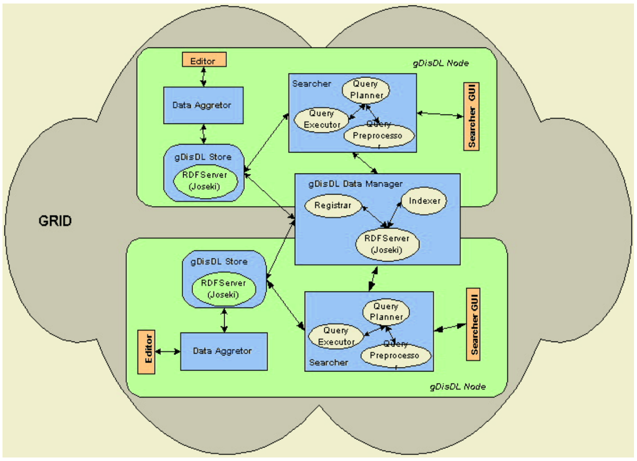


Fig. 2. The Architecture of the gDisDL System

4.1 Design Goals

The main design issue of the gDisDL system is how to share RDF metadata efficiently and securely in a distributed manner. To address this challenge, we adopt the Open Grid Service Architecture (OGSA). To this end, we design gDisDL as a set of Grid services, which are defined according to the Open Grid Service Infrastructure (OGSI) specifications.

Another design challenge is how to encode and store metadata in RDF. Currently, most RDF-based systems ask users to feed RDF metadata directly [4]. Therefore, users have to encode resource metadata into RDF syntax manually, a process which is inconvenient and difficult. To cope with this problem, we designed and implemented the *DataAggregator*, a component which generates the RDF syntax automatically and subsequently stores the RDF-encoded metadata in RDF storage.

Another challenge is the storage and query of RDF metadata. Typically, an RDF database can be used to store RDF metadata, and an RDF query language can be used to express queries and execute them on the database. In the EU-MEDIN use-case scenario, however, most projects do not contribute large bodies of metadata; also, metadata updates are not very frequent. Therefore, a database system would be an overly expensive solution for our system requirements. Moreover, we would have to choose one among several existing RDF databases and query languages [4, 1, 17, 13], and integrate it with gDisDL. Thus, the system would depend heavily on the chosen database system. Currently, the default behavior of gDisDL is to store RDF metadata in plain files. In order to make gDisDL more open and extensible, we designed the *Searcher* component as a "query translator" that processes user requests and generates queries according to the back-end storage used in each gDisDL site. Thus, the back-end storage system remains transparent to the user and any RDF database systems can be deployed with gDisDL.

Another important design consideration is security. The security of the gDisDL is mainly concerning two aspects: one is accessing distributed RDF metadata secretly; another is that the shared RDF metadata should be protected, for instance, restricting the users who have not the right to access the information. We adopt the Grid security infrastructure (GSI) in the *gDisDL*, which uses the security mechanisms such as authentication and authorization to secrete the shared RDF metadata.

4.2 gDisDL Components

The Data Aggregator. The *Data Aggregator* encodes information provided by end-users for EU-MEDIN resources into RDF. In particular, the Aggregator:

1. Gets the information describing an EU-MEDIN resource from the *Editor*.
2. Validates the provided information with respect to the EU-MEDIN RDF schema, taking into account resource classes, class properties, restrictions, and data types.
3. If the information is deemed valid, the Aggregator will generate a unique URI to identify the resource. The URI contains two parts: one is the location of the gDisDL system that the user uses (e.g. the domain name); the other is the time when the RDF metadata was generated.
4. If the data is not valid, the Aggregator sends an error message to the Editor, and ends the process.
5. The Data Aggregator transforms the validated data together with the created URI into an RDF graph, which is a collection triples, each consisting of a subject, a predicate and an object [14]. The RDF metadata of the resource is thus created.
6. The RDF metadata is encoded in RDF/XML format and saved in a gDisDL store.

The gDisDL Store. *gDisDL Stores* are the repositories used to save RDF metadata created by the Data Aggregator. A gDisDL Store, by default, stores metadata for similar EU-MEDIN resources (i.e, resources belonging to the same EU-MEDIN class) into the same RDF file. For example, all RDF metadata describing Journal Papers are kept in one file, all RDF metadata describing Data Sets are kept in another file, and so on. Thus, when we look for metadata about Journal Papers, we can search into local or remote RDF files dedicates to Journal-paper metadata. In order for a gDisDL Store to make its contents available on the Grid, it has to register with one gDisDL Data Manager. Furthermore, the Store has to notify this Data Manage about updates in its contents. The Store's contents are published through Jena's Joseki Web server [1, 2].

The Searcher. The *Searcher* is the component responsible for the metadata search process. It comprises three subcomponents: the Query Pre-processor, the Query Planner, and the Query Executor. The Query Pre-processor receives and validates the user's requests using a mechanism similar to that of the Data Aggregator. The Query Planner works together with a Data Manager to prepare a query plan; the Query Executor executes the query according to the query plan.

A typical searching procedure begins with the Query Pre-processor that receives the request from the client. It checks the data of the request according to the RDF Schema, translates it into an RDF triple model, and passes the result to the Query Planner. The Query Planner then makes a query plan by consulting the gDisDL Data Manager. The query plan specifies what kind of back-end storage system should be queried, e.g. a gDisDL Store or an RDF database; in the first case, it also provides the address of the target *gDisDL Store*. Eventually, the Query Executer performs the real query according to the query plan. It can search the RDF metadata looking through the RDF file in target gDisDL Stores.

The query of the gDisDL system is based on the RDF triple model. Namely, the query is a a single triple pattern, with optional subject (parameter "s"), predicate (parameter "p"), and object (parameter "o"). Absence of a parameter implies "any" for matching that part of the triple pattern. The *Searcher* handles RDF data based on the triple (Subject, predicate, Object). According the user's input parameters, the *Searcher* will make a query and locate the RDF file that may contain the desired RDF metadata from the gDisDL store; then it explores all the RDF triple statements, e.g., the name or URI of the resources ("s"), their properties ("p"), and their values ("o"), compares them with the input parameters, and retrieves the matched RDF triples. if the back-end store is an RDF database system, the Query Planner can translate the user's input (i.e. the resource type, the property, values of the properties) into a proper RDF query language format.

The gDisDL Data Manager. The gDisDL Data Manager provides index information about RDF metadata maintained in distributed gDisDL Stores. To this end, it keeps a list of URI's of the stored RDF resources. The index information is used when making a query plan. The gDisDL Data Manager has three subcomponents: an Indexer, which maintains the list of URIs of all the metadata in all the gDisDL nodes; a Registrar for registering gDisDL Stores; the Joseki RDF web server, for retrieving and publishing RDF metadata, i.e., index information. The gDisDL Data Manager receives update notifications from its associated gDisDL Stores and updates its index accordingly. Each gDisDL store should register to the gDisDL Data Manager in order to share the RDF metadata of it.

5 Implementation

We have implemented a prototype of gDisDL. In this section, we provide some details about the gDisDL implementation.

The Grid gDisDL is implemented within the Open Grid Services Infrastructure (OGSI). Globus Toolkit 3 and Jena are the main development tools used in our implementation. GT3 is a software toolkit that can be used to program grid-based applications. It is implemented in Java based on the OGSI specification. GT3 provides lot of services, programs, utilities, etc. Jena is a Java API that can be used to create and manipulate RDF graphs. it comprises object classes to represent graphs, resources, properties, and literals; a graph is called a model and is represented by the model interface. In our implementation, Jena is used as a JAVA RDF toolkit for creating, manipulating and querying RDF metadata.

To implement a Grid service, the first and most important work is to define the interface of the Grid service, namely, specify the service interface in GWSDL [18]. Once the interface is correctly defined in GWSDL, implementing the service using Java and other tools is straightforward. Thus we will focus on describing how the interface is defined and what kinds of operations will be invoked.

5.1 Data Aggregator Grid Services and Interface

The *DataAggregator service* processes the collected information and data from the *Editor* client, encodes it into RDF format, and saves it into gDisDL stores as RDF files. The interface of the DataAggregator grid service is defined in GWSDL as follows:

```
<gwsdl:portType name="DataAggregatorPortType" extends="ogsi:GridService">
  <operation name="retriveInfo">
    <input message="tns:GetInputMessage"/>
    <output message="tns:GetOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="getRDF">
    <input message="tns:GetRDFInputMessage"/>
    <output message="tns:GetRDFOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="validate">
    <input message="tns:ValInputMessage"/>
    <output message="tns:ValOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="saveRDF">
    <input message="tns:SaveInputMessage"/>
    <output message="tns:SaveOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
</gwsdl:portType>
```

Operation/PortType: `retrieveInfo()` is used to get values of the RDF triples from a client; `getRDF()` creates an RDF graph and assigns the values of the triples; `validate()` checks and validates the input data according to the syntax of EU-MEDIN RDF metadata; `saveRDF()` saves the RDF metadata into a file in RDF/XML syntax.

5.2 Searcher Grid Services and Interface

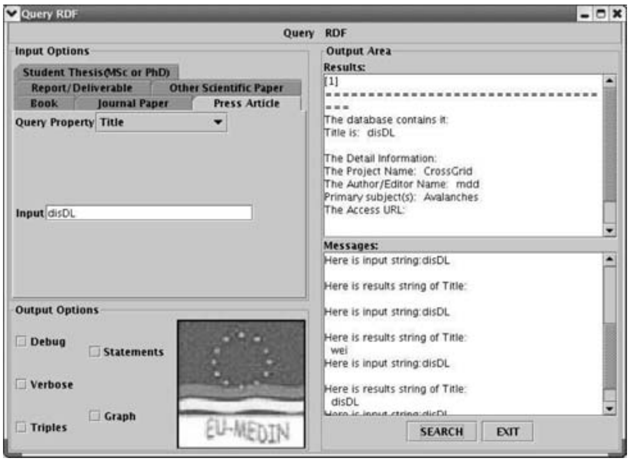
The Searcher grid service is used for receiving and answering user queries about EU-MEDIN resources. There are two cases for the Searcher when searching for RDF metadata. In the first case, the Searcher uses the RDF metadata document `match()` method to search the RDF metadata in a RDF document. In the second case, the search in conducted upon an RDF database, using a database-specific plug-in. Currently we just implemented the first one. The interface is defined as follows:

```
<gwsdl:portType name="SearcherPortType" extends="ogsi:GridService">
  <operation name="preprocess">
    <input message="tns:PreInputMessage"/>
    <output message="tns:PreOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="searchList">
    <input message="tns:ListInputMessage"/>
    <output message="tns:ListOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="match">
    <input message="tns:MatchInputMessage"/>
    <output message="tns:MatchOutputMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
</gwsdl:portType>
```

Operation/PortType: the `preprocess()` operation is used for preprocessing user requests. The `searchList()` gets the remote RDF metadata information from a gDisDL Data Manager. The `match()` operation is used by Query Executor to match RDF triples.

5.3 gDisDL GUIs

Two graphical interfaces are designed and developed to facilitate the end user: the *Editor* and the GUI Client of the Searcher. The *Editor* is a GUI client of the Data Ag-



Query Property Title

Input disDL

```
[1]
===================================
===
The database contains it:
Title is:  disDL

The Detail Information:
The Project Name:  CrossGrid
The Author/Editor Name:  mdd
Primary subject(s):  Avalanches
The Access URL:
```

```
Here is input string:disDL

Here is results string of Title:

Here is input string:disDL

Here is results string of Title:
  wei
Here is input string:disDL

Here is results string of Title:
  disDL
```

Fig. 3. GUI of gDisDL Searcher

gregator Grid service where a user can input information and data about EU-MEDIN resources. Similar to the EU-MEDIN portal, it also provides some forms which can be used to collect information about the EU-MEDIN resources. The Searcher GUI Client of the Searcher Grid Service is needed for users to input the parameters of metadata queries and get results (see Figure 3). The GUI allows users to specify the resource type, the property, values of the properties, etc. The GUI Client also decodes the RDF query results into human-readable form, and displays it on the result window (see Figure3).

6 Conclusions and Future Work

In this paper, we present an RDF-based Grid service approach for organizing and capitalizing European research results in the field of natural disasters. In brief, our approach makes the RDF metadata of the European research results in the field of natural disasters to be shared securely and effectively in a heterogeneous network environment using Grid technology. Then we describe the design and the prototype implementation of the Grid-enable gDisDL system. The RDF-based Grid-enable gDisDL system is a platform independent system which provides good interoperability with other systems. It can store, manage, and query RDF metadata in a secure and distributed manner.

Next step, we will develop a RDF database plug-in of the Searcher in order to integrate RDF databases into our system. A mechanism is also needed for the gDisDL Data managers to exchange indexing information. Furthermore, the gDisDL Data manager will be able to be used as a kind of Cache to retrieve and publish metadata located in the gDisDL stores for improving the query performance.

References

1. Jena - a semantic web framework for java. http://jena.sourceforge.net, 2003.
2. Joseki. http://www.joseki.org, 2003.
3. Eu-medin portal. http://www.eu-medin.org, 2004.
4. S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In *Proceedings of the Second International Workshop on the Semantic Web (SemWeb '01)*, pages 1–13, May 2001. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-40/.
5. Seaborne Andy. An rdf netapi. Technical report, HP Labs, 2002.
6. T. Berners-Lee, R. Fielding, and L. Masinter. *Uniform Resource Identifiers (URI):Generic Syntax.* The Internet Engineering Task Force, August 1998.
7. Tim Berners-Lee. *Notation 3*, 1998.
8. T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau. *Extensible Markup Language (XML) 1.0.* World Wide Web Consortium, third edition, February 2004.
9. Dan Brickley and R.V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0.* World Wide Web Consortium.
10. D. Connolly, F.V. Harmelen, I. Horrocks, D.L. McGuinness, P. Patel-Schneider, and L.A. Stein. *DAML+OIL (March 2001) Reference Description.* World Wide Web Consortium, March 2001.

11. I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke. The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.
12. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 15(3), 2001.
13. G.Karvounarakis, V.Christophides S.Alexaki, and Michel Scholl D.Plexousakis. RQL: A Declarative Query Language for RDF. The Eleventh International World Wide Web Conference (WWW'02), May 2004.
14. Graham Klyne and Jeremy Carroll. Resource Description Framework (RDF): Concepts and Abstract Data Model. Technical report, The World Wide Web Consortium, August 2002.
15. F. Manola and E. Miller (editors). RDF Primer. W3C Working Draft, October 2003. http://www.w3.org/TR/rdf-primer/.
16. P.F. Patel-Schneider, P. Hayes, and I. Horrock. *OWL Web Ontology Language Semantics and Abstract Syntax*. World Wide Web Consortium, February 2004.
17. Andy Seaborne. *RDQL - A Query Language for RDF*. The World Wide Web Consortium (W3C), January 2004.
18. Borja Sotomayor. The globus toolkit 3 programmer's tutorial. Technical report, The Globus Alliance, 2003.
19. 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.