Marko Niinimäki

Grid Resources, Services and Data – Towards a Semantic Grid System

DEPARTMENT OF COMPUTER SCIENCES
UNIVERSITY OF TAMPERE

D-2006-1

TAMPERE 2006

UNIVERSITY OF TAMPERE
DEPARTMENT OF COMPUTER SCIENCES
SERIES OF PUBLICATIONS D – NET PUBLICATIONS
D-2006-1, JANUARY 2006

Marko Niinimäki

Grid Resources, Services and Data – Towards a Semantic Grid System

DEPARTMENT OF COMPUTER SCIENCES
FIN-33014 UNIVERSITY OF TAMPERE

ISBN 951-44-6555-5
ISSN 1795-4274

Grid Resources, Services and Data – Towards a Semantic Grid System

Marko Niinimäki
Helsinki Institute of Physics, Technology Programme and
Department of Computer Science, University of Tampere, Finland

January 2006

Abstract

This paper describes some of the many-faceted technologies that are known as the Grid. These technologies have been used create distributed applications and data services in a secure manner.

With existing Grid software, several milestones have been reached. However, most of the applications that run on production Grids are quite specialised and require a lot of expertise to set up, run and maintain. In this paper, we design and implement an easy, generic framework for resource, service and data description directory, and a present a user interface prototype for it.

In this framework, the user is not using "computer programs" in the traditional way, there are only services and data that the services can manipulate for the user.

Keywords: Grid, Grid services, virtual organizations.

1 Introduction

As Foster et al. state in [FKNT04], "Grid systems and applications aim to integrate, virtualize, and manage resources and services within distributed, heterogeneous, dynamic 'virtual organizations'." Catlett in [Cat03] divides Grid systems into three generations. The first generation involved local "metacomputers" with basic services such as distributed file systems and site-wide single sign-on. It was up to programmers to provide distributed applications using these and custom communication protocols. The second generation Grids such as I-WAY [DFP+96], Legion [GW97] and Condor [LLM88] created software services and communications protocols that could be used as a basis for developing distributed applications and services. It is, however, obvious that these new bundles of services and protocols (later called Grid middlewares) were mutually incompatible. Catlett sees the third generation Grids to be based on compatible technologies and architectures, featuring Open Grid Service Architecture (OGSA, see [FKNT04]).

With second generation Grid Systems, the concept of job management is pervasive: the user designs a job description that declares the computing and data resources needed, and a system known as the resource broker uses some heuristics to find an "end

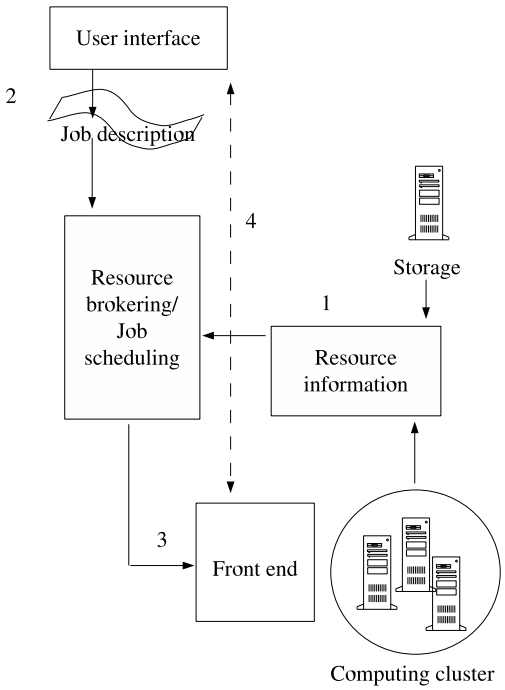


Figure 1: A general architecture of a typical second generation Grid

point" where the job can be run. After a (hopefully) successful completion, the results of the job are stored and the user is notified. As an example of a second generation Grid system, we consider Condor-G [F$^+$02], a combination of Condor batch job management system and Globus Toolkit (see [FK97]). A simplified view of such a system is shown in Figure 1. There, the resources (storage, computing power) register their capabilities in a resource information directory. The user who wants to utilize the resources expresses his request (2) using a job description language. The resource broker matches the resources and the request and the job is send to be executed in a "suitable" location (3). The results are either recovered by the user (4) or "automatically" written to the specified location.

However, the main focus of this paper are Grid technologies beyond job submissions. The important concepts to build on are those of virtual organizations; secure services; resource and service ontologies; and resource and service catalogs. As a novel item of design, we present how to describe data in the Grid context and integrate it with resources and services. Based on these, we present a design and a prototype of a service oriented Grid system that emphasises the semantics of data, resources and services. As far as we know, a coherent, ontological framework like this has not been presented previously, though several of its components have been developed in Grid projects (see below).

The rest of the paper is organized as follows. In Section 2, we provide a back-

ground to existing Grid systems and discuss how to *access* a known Grid resource and service. This section can be read as a survey of basic Grid technologies. In Section 3 we present how to *describe* resources, services and data; more specifically in Section 3.1 we provide an ontology-based description framework for resources, in 3.2 for Grid services, and in 3.3 for data. The main contribution of this paper is presented in Section 4, where a unified design of a resource, service and data directory is presented. In Section 5 we discuss how a user could intuitively use such a platform of services and resources.

2 Related work and background

Related work relevant in this context includes Foster and Kesselman's seminal introductions to the Grid (and applications) in [FK98, FK03]. Specifically, Chapter 21, [SNWN03a], in the latter, and Chapter 4 of [LB05] describe the state of the art of Grid security technologies; they are used as the background of Section 2.1.

The Open Grid Service Architecture (OGSA) has been presented in [FKNT04, KT05, FK03]. OGSA is based on Semantic Web technologies, discussed in [BLHL01, AvH04]. Among these technologies, the Resource Description Framework (RDF) [W3C04c] and Web Ontology Language (OWL) [B$^+$04] are World Wide Web Consortium recommendations for the Semantic Web and previously applied to Grid context for instance in [PCS03]. Applying Semantic Web technologies to Grid environments is commonly known as the Semantic Grid and discussed in e.g. [RJS05].

Grid resource descriptions have been an issue of many studies, e.g. [KBM02]. The resource descriptions are matched with the requirements of jobs by different brokering approaches discussed in e.g. [YSL05, VBW05]. The concept of Grid services in OGSA embraces the management of resources as "Resource management services" and data as "Data services" (see [PTF05]). Research related to discovering Grid resources and services has been done in e.g. [HDH$^+$04, LvS02, TDK03, BGG03]. However, a specifically data oriented approach is seen in Open Grid Service Architecture - Data Access and Integration OGSA-DAI [A$^+$05b].

2.1 Authentication and authorisation in Grids

Authentication (i.e. user identity verification) and authorisation, i.e. deciding whether the user can have an access to a certain resource are essential in distributed applications.

Authentication and authorisation in Grid relies on public key infrastructure (PKI). The best-known example of its use is Grid security infrastructure (GSI) promoted by the Globus Alliance (see [SNWN03b]).

PKI is a set of protocols, services and standards that facilitate the use of public key cryptography by allowing the secure distribution of public keys between communicating parties, and thereby supporting the exchange of sensitive information over an unsecured network such as the Internet [Nie02]. PKI uses two digital keys mathematically related: a public key and a corresponding and unique private key. Given a message, an encryption function (that crypts the message) EK can be easily computed from the public key X, and X is computed from the private key K. X is published, so

that anyone can encrypt messages. If a decryption function DK cannot be easily computed from X without knowledge of K, but readily with knowledge of K, then only the person who generated K can decrypt messages (adapted from [cc04], for details, see [MvOV96]).

In the scope of this paper we discuss PKI as related to digital certificates. In order to do so, we need the concept of digital signature, that we describe following Galwin and Murphy [GM95]:

"A digital signature for an object is created by hashing the object with a one-way hash function and encrypting (signing) the hash value with the private key component of a public/private key pair. The signature is verified by decrypting it with the public key component to expose the hash value and comparing the exposed hash value to a recomputed hash value. If the two hash values match the signature is [..] considered valid."

Currently, X.509 version 3 (recommended by the International Telecommunications Union (ITU) in [IT97]) is the most commonly used PKI standard.

A X.509 v3 certificate contains information referring to a public key, which has been digitally signed by a Certification Authority (CA). It is represented by a document where the public key is contained – as well as other useful identification information such as the distinguished name (DN) of the entity it identifies, an expiration date and the CA's name (see [Nie02]). The certificate assures any relying party (using the public key) that the associated private key is held by the "correct" remote entity to whom the certificate was created. An example of a X.509 v3 user certificate is shown in Figure 3. A host certificate that identifies for instance a computer is somewhat different; some fields of such a certificate are shown in Figure 4. Some fields of the (self-signed) NorduGrid CA certificate are shown in Figure 5.

A proxy credential allows entity A to grant to another entity B the right for B to be authorised with others as if it were A (see [T$^+$04, W$^+$04]). In the context of GSI, proxy credentials are *proxy certificates*; typically user proxies (created by the user) or delegated proxies (created by services, using the user proxy). A user proxy contains both a public *and* a private key with a subject of "A/proxy"[1], together with A's public key and it is signed by A. A service or other entity receiving the proxy can verify that it is indeed signed by A. Moreover, through A's public key, it can verify that it has been signed by the CA it claims to be. However, since an interceptor can at least theoretically steal the proxy, its validity is typically limited to 12 hours.[2] The authentication process is described in Figure 2, adopted from [The04a], [LB05] and Globus GSI source code. Technically, the method of implementation for carrying out the exchange of certificate information is based on Secure Sockets Layer/Transport Layer Security (SSL/TLS) [Tho00]. The negotiation takes places when the connection is established between the client and the server.

As obvious in Figure 2, the identity of the host computer is based on its IP address and host name. This can be seen as limitation; a novel approach called Host Identity

[1] More recent approaches use an unique identifier instead.

[2] In a typical Unix system case, user proxies are stored in the file system's /tmp directory with a file name indicating the user's id, and they are readable by that user only. However, anyone with super user rights can steal A's proxy and thus claim to be A. Recent standards add restrictions in proxies and thus minimize the possibility that they are abused (see [T$^+$04]).

- To start the authentication process, A gives B his proxy certificate file *containing* the proxy private and public keys and A's public key.
- A's public key is used to validate the signature on the proxy certificate.
- The CA's public key is then used to validate the signature A's certificate.
- Once B has checked out A's certificate, B must make sure that A really is the person identified in the certificate:
  - B generates a random message and sends it to A, asking A to encrypt it.
  - A encrypts the message using his private key, and sends it back to B.
  - B decrypts the message using A's public key.
  - If this results in the original random message, then B knows that A is who he says he is.
- The same operation takes place so that A verifies B's identity. In this case B has a server certificate instead of a proxy. Therefore:
  - B sends A her certificate, A validates the certificate and sends a challenge message to be encrypted.
  - B encrypts the message and sends it back to A, and A decrypts it and compares it with the original.
  - If it matches, then A knows that B is who she says she is.
- Additionally, both parties check that the other party's certificate is valid. Moreover, A checks that B's canonical name (like `pcrship01.cern.ch`) corresponds with the subject of B's certificate.

Figure 2: A mutual authentication process using a proxy certificate of A and server certificate of B.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 527 (0x20f)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: O=Grid, O=NorduGrid, CN=NorduGrid Certification Authority
        Validity
            Not Before: Feb 11 15:53:21 2004 GMT
            Not After : Feb 10 15:53:21 2005 GMT
        Subject: O=Grid, O=NorduGrid, OU=hip.fi, CN=Marko Niinimaki
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Client, S/MIME
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                34:52:66:B3:55:FD:65:79:E7:EA:59:A2:74:28:C0:1E:D7:DD:17:DB
            X509v3 Authority Key Identifier:
                keyid:18:05:C0:FC:0B:D1:B7:3A:F4:65:92:09:FB:59:A1:5F:C7:88:C4:F0
                DirName:/O=Grid/O=NorduGrid/CN=NorduGrid Certification Authority
                serial:00

            X509v3 Subject Alternative Name:
                email:marko.niinimaki@hip.fi
    Signature Algorithm: md5WithRSAEncryption
```

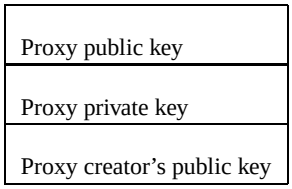
Figure 3: An example of a X.509 user certificate (some fields and values omitted)

```
Issuer: O=Grid, O=NorduGrid, CN=NorduGrid Certification Authority
Subject: O=Grid, O=NorduGrid, CN=host/pcrship01.cern.ch
```

Figure 4: Examples of fields and values in a X.509 host certificate

```
Issuer: O=Grid, O=NorduGrid, CN=NorduGrid Certification Authority
Subject: O=Grid, O=NorduGrid, CN=NorduGrid Certification Authority
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:TRUE
```

Figure 5: Examples of fields and values in a X.509 CA certificate




```
Issuer: O=Grid, O=NorduGrid, OU=hip.fi, CN=Marko Niinimaki
Validity
  Not Before: Nov 18 07:56:23 2004 GMT
  Not After : Nov 18 20:01:23 2004 GMT
Subject: O=Grid, O=NorduGrid, OU=hip.fi, CN=Marko Niinimaki, CN=proxy
```

Figure 6: The structure of a proxy certificate and some lines of a typical proxy

Limited proxy public key (signed by proxy public key) Subject: O=Grid, O=NorduGrid, OU=hip.fi, CN=Marko Niinimaki, CN=proxy, CN=limited proxy
Limited proxy private key
Proxy public key (signed by proxy creator's public key) O=Grid, O=NorduGrid, OU=hip.fi, CN=Marko Niinimaki, CN=proxy
Proxy creator's public key (signed by the CA) O=Grid, O=NorduGrid, OU=hip.fi, CN=Marko Niinimaki
CA's public key O=Grid, O=NorduGrid, CN=NorduGrid Certification Authority

Figure 7: The structure of a delegated proxy certificate

Protocol is discussed in [KGN05, Kar05a] but to our knowledge it has not been applied to Grid contexts.

When the client has been authenticated and authorized by the server, a *proxy delegation* is performed. A delegated proxy file (representing the user) is created by the server software. The structure of a delegated proxy is shown in Figure 7; for details, see [W$^+$04, The04a]. The benefit of the delegated proxy certificate is that the original (user's) proxy, containing the proxy private key is verified but not sent to the server. However, in some approaches like Helsinki Institute of Physics' OpenGrid portal (see [WNN03]) the user's proxy file is transfered quite liberally to be used by grid services. A same kind of proxy store is used by MyProxy software (see [NTW01]). The motivation there is that since data transmissions are encrypted it is unlikely that the proxy file will be stolen by a third party during its transfer. This is the approach we shall use in our implementation in Section 4, too.

The Grid computing field can be characterized as a collection of heterogeneous computing resources that are shared by many individuals and organizations. This has given rise to the concept of "virtual organizations". A virtual organization (VO) is a collection of people in some administrative domain. A user's relationship with his VO is defined by the organization's internal hierarchy. The user can be a part of any number of internal groups in their organization and have multiple roles in many organizations [ACC$^+$03]. A user is authorised to perform tasks or access resources in the Grid according to their VO affiliation and their role(s) within the VO. Virtual Organization Membership Service VOMS is an authentication and authorisation system that allows the addition of VO information in the user's proxy. A VOMS system consists of a user server, a user client, an administration server and an administration client (see [ACC$^+$03]); here it is sufficient to note that a user executes a client "voms-proxy-init" with a parameter specifying the VO (e.g. nordugrid). The client connects to the server, and the server returns the user's proxy with the appropriate VO information. The extension as "raw data" and parsed by voms-proxy-info is shown in Figure 8.

These tools provide us with rather flexible means of authentication and authorisation. With the first versions of Globus middleware, the authentication and authorisation

```
X509v3 extensions:
1.3.6.1.4.1.8005.100.100.5:
....Marko Niinimaki....
....nordugrid://hydra.ii.uib.no:15002
..../nordugrid/Role=NULL/

voms-proxy-info -all
subject   : /O=Grid/O=NorduGrid/OU=hip.fi/CN=Marko Niinimaki/CN=proxy
issuer    : /O=Grid/O=NorduGrid/OU=hip.fi/CN=Marko Niinimaki
identity  : /O=Grid/O=NorduGrid/OU=hip.fi/CN=Marko Niinimaki
type      : proxy
strength  : 512 bits
path      : /tmp/x509up_u1007
timeleft  : 10:08:13
VO        : nordugrid
subject   : /O=Grid/O=NorduGrid/OU=hip.fi/CN=Marko Niinimaki
issuer    : /O=Grid/O=NorduGrid/CN=host/hydra.ii.uib.no
attribute : /nordugrid/Role=NULL/Capability=NULL
timeleft  : 10:08:12
```

Figure 8: Some features of a voms proxy

```
"/O=Grid/O=NorduGrid/OU=hip.fi/CN=Marko Niinimaki" marko
```

Figure 9: An entry in /etc/grid-security/grid-mapfile

process was quite limited; the middleware's "gatekeeper" component gave the user the right to execute programs as a local user based on entries in a specific file in the local (Unix) system, grid-mapfile (see [Theb]). An example of that is shown in Figure 9. There, using a client job submission program, the user whose proxy issuer corresponds to the given line is allowed to execute his program as user "marko". Naturally, the gatekeeper also verifies that the user's CA is known.

A more fine-grained approach utilises access control lists (ACL's). With GridSite server software (see [McN05]), several access types (read, write, execute..) can be defined, and the access to files is per-directory basis. An example in Figure 10 shows a ".gacl" file that controls access to a directory in the server computer. The first entry authorises a person for some access types. However, given dynamic communities, it would become a burden to maintain such access control entries to all potential users. Entries specifying access for VO members are easier to maintain; an entry shown below would allow the users in VO nordugrid (their membership signed by VOMS server hydra.ii.uib.no) to read write and list entries in this directory.

2.2 Grid middleware and other distributed computing approaches

In general, a distributed computing system can be defined (like Özsu and Valduriez in [OV99]) as a number of autonomous processing elements that are interconnected by a computer network and that cooperate in performing their assigned tasks. According to Özsu and Valduriez, the issues that are distributed can be (i) processing logic (ii) functions (iii) data, and (iv) control. Distributing the processing has been the focus of several programming packages that allow the programmer to embed efficient parallelisation in the software, e.g. Parallel Virtual Machine PVM (see [G$^+$94]). Distribution

```
<gacl>
<entry>
<person> <dn>/O=Grid/O=NorduGrid/OU=hip.fi/CN=Marko Niinimaki</dn> </person>
<allow><admin/> <write/> <list/> <exec/> <read/></allow>
</entry>
<entry>
<voms><voms>/O=Grid/O=NorduGrid/CN=host/hydra.ii.uib.no</voms>
  <vo>nordugrid</vo></voms>
  <allow><write/></allow>
  <allow><list/></allow>
  <allow><read/></allow>
</entry>
</gacl>
```

Figure 10: A Grid access control file

according to functions (or functionality) can be seen as assigning specialised tasks to resources that are most suited to process them. Matching tasks and resources is generally seen as the task of a *broker* and implemented for instance as a part of the JXTA architecture (see [Gon01]), or in a more abstract way in the Common Object Request Broker Architecture (CORBA, [Vin97]). Data distribution has been researched in the context of distributed (or federated) database systems (see [OV99]). The distribution of control by collaborating software agents has been addressed in e.g. [Lan98].

Given all these diverse technologies of distributed computing, one can ask what is the role of Grid. This, however, can be seen as an *interface* to distributed data, resources, and services through Grid middleware; or as Grimshaw states in [Gri02], "the objective of Grid middleware is to virtualize resources, provide access, and in general deal with the physical characteristics of the Grid." From the perspective of accessing data, resources and services, we see that the defining feature of Grid middleware and Grid software in general is the use of GSI, discussed in Section 2.1.

3 Grid resources and services

In the context of this paper we consider Grid resources simply as anything a Grid user might be interested in. Grid services are standardized or at least published ways of accessing these resources in the Grid community in question. Thus, disk space is a resource, but a Grid method of putting files or records on a disk, or recovering them, is a service. In a similar manner, executing jobs remotely in a "suitable" computer is a service.

As a simple second generation Grid example, we consider the GSIFTP [Thea, The00] protocol and services that utilise it. GSIFTP is an implementation of FTP (File Transfer Protocol, originally called Bulk Data Transfer Protocol , see [CLZ85]) such that GSI (see Section 2.1) is used for authentication and authorisation. In practice, GSIFTP works as a client-server program; the Globus project uses a modified WU-FTP[3] server and NCFTP[4] client. In practice, the user creates a proxy certificate,

[3]WU-FTP was developed by the University of Washington, see `http://www.wu-ftpd.org`.

[4]See `http://www.ncftp.com/`.

```
 grid-proxy-init
Your identity: /O=Grid/O=NorduGrid/OU=hip.fi/CN=Marko Niinimaki
Enter GRID pass phrase for this identity:
Creating proxy ................................................ Done

 gsincftp  gsiftp://pcrship04.cern.ch
NcFTP 3.0.3 (April 15, 2001) by Mike Gleason (ncftp@ncftp.com).
Connecting to 137.138.250.35...
Server ready
Logging in...
No need for username
Logged in to pcrship04.cern.ch.
Current remote directory is /.
ncftp / > put compute.sh
compute.sh:                                          173.76 kB   51.74 kB/s

ngsub -c pcrship01.cern.ch -f test.job
Client middleware: nordugrid-0.5.33
Submitting xrsl: &(executable="/bin/echo")
(arguments="track1")(jobName="ngtest")(stdout="stdout")
(executables="compute.sh")(inputfiles=("compute.sh"
"gsiftp://pcrship04.cern.ch/compute.sh"))
Job submitted with jobid gsiftp://pcrship01.cern.ch:2811/jobs/2831111340415291997069207
```

Figure 11: File transfers and job submissions using ARC

initiates the connection to the server using the `gsincftp` client program. The mutual authentication proceeds as in Figure 2.

The NorduGrid consortium's Advance Resource Connector (ARC, see [E+06]) is a second generation Grid middleware that uses the GSIFTP protocol for job submissions in addition to file transfers. A file transfer and a job submission to a specific computer running ARC is shown in Figure 11. There, a user transfers a file (compute.sh) in a file server using GSIFTP, and, in another computer, starts a job that utilises the file. In the latter case, the software component (called Grid Manager in ARC) collects the input files, starts the job and supervises its execution. Its ability to recover input files is naturally based on the fact that it can use the user's delegated proxy, as explained in Section 2.1.

For resource descriptions, ARC uses a system based on Lightweight Directory Access Protocol LDAP [HS95] and Monitoring and Discovery System (MDS, originally called Metacomputing Directory Service in [F+97], see [Thec]). However, in [NTN05b], we have outlined Grid resource description based on Resource Description Framework (RDF) ontologies. In Section 3.1, we provide a short summary. In Section 3.2, we concentrate in (third generation) Grid services. We assume, naturally, that access to services is controlled by GSI.

3.1 Grid Resource Descriptions

In [And04] DataTAG's GLUE project's goals are described as follows. "The GlueSchema activity aims to define a common conceptual data model to be used for grid resources monitoring and discovery." GLUE 1.1 divides Grid into computing (Computing Element, CE), storage (Storage Element, SE) and network. Slightly surprisingly,

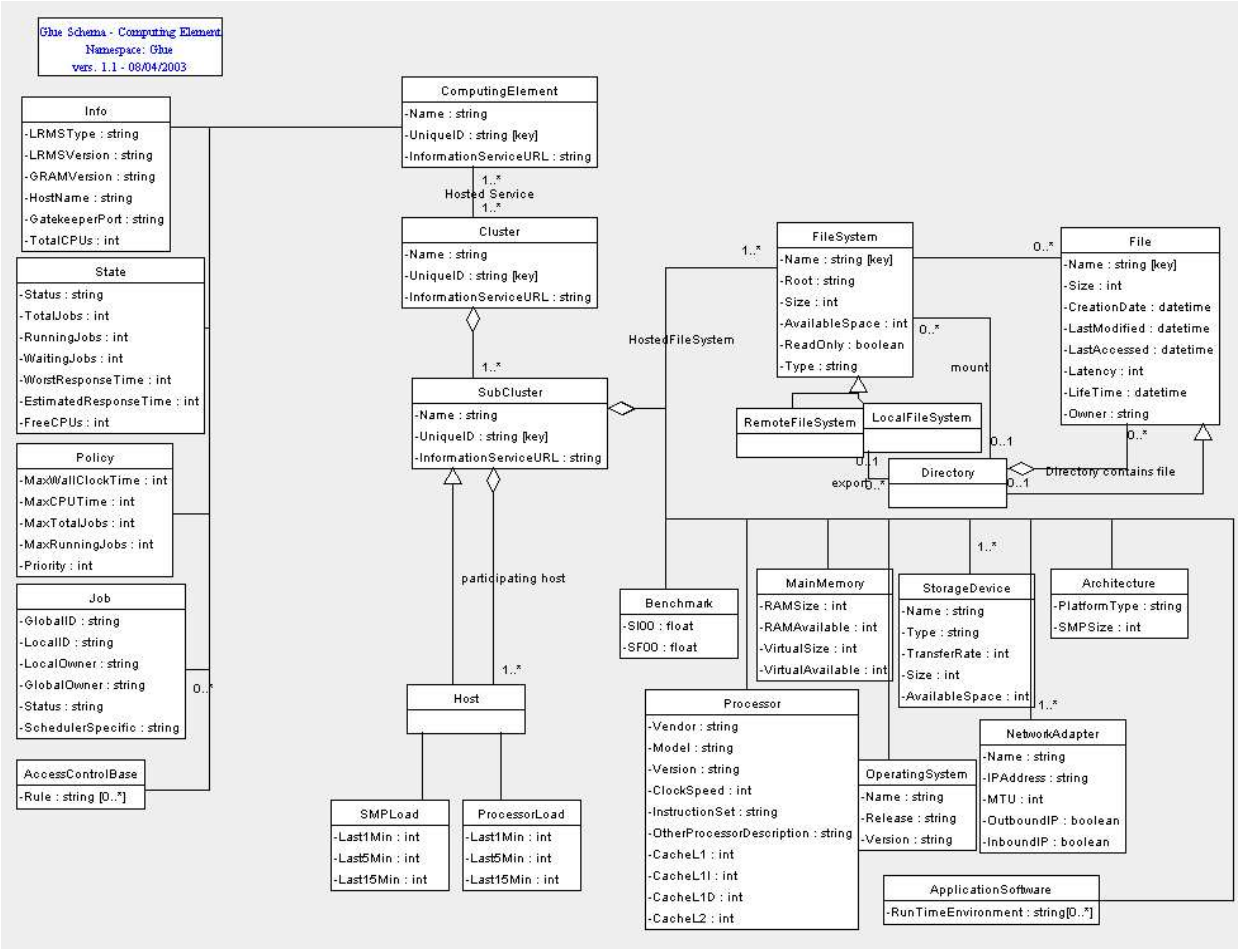


Figure 12: GLUE CE description

there is no separate concept of a “principal” (a user, a group of users or a virtual organization).[5] Figure 12 shows GLUE’s CE description in UML[6] class diagram format.

GLUE is meticulously mapped into several implementational resource presentation formats, including an LDAP schema and a relational database schema.

In [NTN05b], we have created a formal ontological description of GLUE using the World Wide Web Web Consortium’s Web Ontology Language OWL that is based on RDF (see [B$^+$04]). More specifically, with RDF, the most basic method of representation is stating that something (a subject) has a ”property” (a predicate) whose value is something (an object, see [W3C04a]). The subject - property - value triples can express simple facts about individuals like “this document (object) has an author

[5]The current edition in [A$^+$05a] adds a “Access Control Base” element for this purpose.

[6]UML stands for Unified Modelling Language, see [The03b].

(property) whose value is N.N.", but RDF introduces the use of Uniform Resource Identifiers (URIs) that are used as references in these descriptions to (i) network-accessible things, such as an electronic document, an image, a service, or a group of other resources; (ii) things that are not network-accessible, such as human beings, corporations, and books in print; and (iii) abstract concepts that do not physically exist, such as the concept of a "creator" (see [W3C04a]). Using URI references, the above expression could be paraphrased "document `http://wiki.hip.fi/xml/example` has property `http://purl.org/dc/elements/1.1/creator` whose value is `http://www.cs.uta.fi/henkilosto/henkilo.php?uid=csmani`". A set of URI references specified for a specific purpose (e.g. for expressing creators and creation dates of documents) is commonly refered to as a vocabularium and technically often presented as a XML namespace (for details, see [BHL99]). The RDF Schema language (RDFS, see [W3C04b]) has been specifically designed for expressing vocabularies, and OWL is an extension of the schema language. RDFS itself has a rich set of data modelling constructs like classes, their sub-classes and properties (relations with other classes), but OWL further extends RDFS by adding restrictions (like cardinality constraints) to these constructs; for details, see [B$^+$04]. In general, we call a description that uses OWL constructs an *OWL ontology* or *OWL schema*.

The benefit of this presentation is that it allows search operations using high-level query languages like RQL and RDQL [KAS$^+$02, Sea04] instead of implementation specific LDAP queries. Moreover, the form of expressing these resources can be matched with service and data descriptions that are expressed using RDF, too. A part of this OWL ontology is shown in Figure 13, where "glue" is a namespace designed for presenting GLUE elements. An instance satisfying this schema in Figure 14.

The full ontology (the source of the excerpts) is available in `http://wiki.hip.fi/xml/ontology/glue.xml`.

RDF/OWL descriptions can be queried using RQL, an SQL based query language and can operate with concepts of RDF and OWL, i.e. classes, properties, etc. The syntax of RQL reminds SQL with its SELECT, FROM, WHERE clauses. The example query in Figure 16 returns all Unix computers. In practice this means that the user can specify that he wants to execute the job in a computer with an Unix operating system, his job description is appended with the query in Figure 16, and the answer will contain computers with Linux OS's since Linux is Unix (as specified in Figure 15).

3.2 Grid Service Descriptions

According to Baker et al. in [BAFB05], the most important Grid standard to emerge recently is the Open Grid Services Architecture (OGSA), which "aims to define a common, standard, and open architecture for Grid-based applications."

As shown in Figure 17, OGSA is a layered approach where "Grid Services" (OGSA Services) make use of Web Services. These, in turn, have been defined by [W3C04d] as follows: "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL)[7]. Other systems interact with the

[7]WSDL stands for Web Service Description Language.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    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:dc="http://purl.org/dc/elements/1.1/"
    xmlns:glue="http://wiki.hip.fi/xml/ontology/glue.xml"
    xmlns:base="http://wiki.hip.fi/xml/ontology/glue.xml"
    xmlns="http://wiki.hip.fi/xml/ontology/glue.xml">
..

  <owl:Class rdf:ID="OperatingSystem">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="version"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="name"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="release"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

Figure 13: Excerpts of GLUE CE presented in OWL

```
<rdf:Description rdf:about="LinuxWoody">
   <rdf:type rdf:resource="http://wiki.hip.fi/xml/ontology/glue.xml#OperatingSystem" />
   <glue:version>"3.0"</glue:version>
   <glue:name>"Linux"</glue:name>
   <glue:release>"Debian"</glue:release>
</rdf:Description>
```

Figure 14: A part of an instance

```
<owl:Class rdf:ID="Linux">
   <rdfs:subClassOf>
     <owl:Class rdf:ID="Unix" />
   <rdfs:subClassOf>
</owl:Class>
```

Figure 15: Stating that Linux is Unix

```
SELECT ?x
WHERE (?x <rdf:type> <glue:OperatingSystem>),
      (?x <glue:name> ?y) AND
      (?y eq "Unix")
USING glue for <http://wiki.hip.fi/xml/ontology/glue.xml>,
rdf for <http://www.w3.org/1999/02/22-rdf-syntax-ns#>;
```

Figure 16: A query retrieving Unix computers

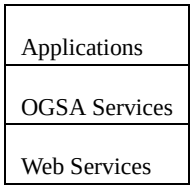


Figure 17: The upper layers of OGSA

```
<wsdl:definitions targetNamespace="https://grid.cs.uta.fi:8443/Services"..
<wsdl:message name="rdfqueryRequest">
<wsdl:part name="proxy" type="xsd:string">
<wsdl:documentation>the secret parameter.. we need the proxy</wsdl:documentation>
</wsdl:part>
<wsdl:part name="rdffile" type="xsd:string">
<wsdl:documentation>The name of the rdf file to be used.</wsdl:documentation>
</wsdl:part>
<wsdl:part name="queryfile" type="xsd:string">
<wsdl:documentation>The name of the query file to be used.</wsdl:documentation>
</wsdl:part>
</wsdl:message>

<wsdl:message name="rdfqueryResponse">
<wsdl:part name="rdfqueryReturn" type="xsd:string">
<wsdl:documentation>an answer to your query</wsdl:documentation>
</wsdl:part>
</wsdl:message>

<wsdl:operation name="rdfquery" parameterOrder="proxy rdffile queryfile">
<wsdl:documentation>rdf query executes sesame and returns the response</wsdl:documentation>
<wsdl:input message="impl:rdfqueryRequest" name="rdfqueryRequest" />
<wsdl:output message="impl:rdfqueryResponse" name="rdfqueryResponse" />
</wsdl:operation>
```

Figure 18: Part of the wsdl for an RDF query service

Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." An example of a WSDL service is shown in Figure 18.

Web services, however, are built on http/https protocols that are stateless (see [FGM+99], for a description about states and automata in general see e.g. [AU95]). Foster & al emphasize in [FFG+04] that "Even those Web service implementations commonly described as stateless frequently allow for the manipulation of state, i.e., data values that persist across, and evolve because of, Web service interactions." As an example, Foster & al mention an online airline reservation system that must maintain a state concerning flight status, reservations made by specific customers, and the system itself: its current location, load, and performance. Web service interfaces that allow requestors to query flight status, make reservations, change reservation status, and manage the reservation system must provide access to this state. In order to standardize the access to systems, Foster & al propose a Web Service Resource Framework (WSRF) approach. In practice this approach simply models the resource (like an airline reservation pool) in a way that a web service can describe it. Consequently, the requestor (the user of the service) can discover the resource and use standardised operations to query and manipulate the states. This is the added value of Grid Services, expressed as follows in [The03a]: "..while Web services successfully implement applications that manage state today, we need to define conventions for managing state so that applications discover, inspect, and interact with stateful resources in standard and interoperable ways."

Standards for OGSA and WSRF are designed and promoted by the Global Grid Forum (see e.g. [KT05], OGSA is a trademark of the Global Grid Forum).

In our prototype (see Section 4), the service description is quite primitive, mainly because the services we have developed are so simple that they can be rather exhaus-

```
<process:SimpleProcess rdf:ID="RdfProcess"/>
<grounding:WsdlInputMessageMap rdf:ID="wsdlinputrdf">
<grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>Rdf</grounding:wsdlMessagePart>
</grounding:WsdlInputMessageMap>
<profile:ServiceCategory rdf:ID="Queries">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>A simple category for queries</rdfs:comment>
<profile:categoryName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>queries</profile:categoryName>
<profile:code rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>1</profile:code>
</profile:ServiceCategory>
<grounding:WsdlInputMessageMap rdf:ID="wsdlinputproxy">
<grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>PROXY</grounding:wsdlMessagePart>
..
```

Figure 19: Part of the OWL-S instance for an RDF query service

tively described by a category of a service (e.g. "Queries") and the WSDL operation description as in Figure 18. We have generated simple mappings from WSDL (and their locations) to OWL-S Web Service ontology language (see [MBL+03]) to demonstrate the possibility of using ontology descriptions in service directories (see Section 4). An example of the OWL-S instance corresponding with the WSDL is shown in Figure 19.

3.3 Data Descriptions

As a missing link between resources and services, we present here an ontology-based framework for describing data for distributed applications. This means that for structured data, the format of the data should be made explicit. In [NTN05a], we have proposed the following design:

- The data can be distributed and provided by independent parties, but for a data repository (an XML file or a data base with a query interface), a description of the format is provided.
- There is an ontology that provides us with canonical names or mappings for each data repository.

The "data ontology" can be specific or general. It seems feasible to assert that the most general ontology for structured data expresses that there are (named) subjects, predicates and object as with RDF (see Section 3.1 and [W3C04c]). On the other hand, domain specific OWL ontologies determine the use of named entities (classes and subclasses) and their properties in a given domain; an ontology of countries may define that there is an entity named Country, with the attributes ISO-code (a string), and GeographicalLocation (an entity of type Geographical Location). Following this example, each independent data provider can have a country database with almost any find of structure they like, for as long there is a mapping from their structure to the domain ontology. A part of a mapping file for a simple data set is shown in Figure 20. There,

```
<ontologymap datasource="year1980.xml">
<tuple name="trade_data/FactRow">
<column name="value" mapclass="TradeFactRow" mapproperty="hasMeasure"/>
<column name="year" mapclass="TradeFactRow" mapproperty="hasYear"/>
<column name="product" mapclass="TradeFactRow" mapproperty="hasProduct"/>
<column name="orig" mapclass="TradeFactRow" mapproperty="hasExportCountry"/>
<column name="dest" mapclass="TradeFactRow" mapproperty="hasImportCountry"/>
</tuple>
</ontologymap>
```

Figure 20: An ontology mapping

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://wiki.hip.fi/xml/data#"
  xml:base="http://wiki.hip.fi/xml/data">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Data">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >a simple class for data
</rdfs:comment>
  </owl:Class>
  <owl:DatatypeProperty rdf:ID="data_mapping">
    <rdfs:domain rdf:resource="#Data"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="data_nickname">
    <rdfs:domain rdf:resource="#Data"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="data_location">
    <rdfs:domain rdf:resource="#Data"/>
  </owl:DatatypeProperty>
</rdf:RDF>
```

Figure 21: Data ontology

the column names `value`, `year`, `product`, `orig`, and `dest` are mapped into properties `hasMeasure`, `hasYear`, `hasProduct`, `hasExportCountry` and `hasImportCountry` of class `TradeFactRow` in our domain specific OLAP ontology (see [NTN05a]). A grammar (expressed using XML Schema, see [W3C99]) for ontology maps is provided in `http://wiki.hip.fi/xml/ontology/ontomapping.xsd`.[8] Technically, transforming the data into a domain specific ontology is carried out using the ontology maps and the XSLT transformation language stylesheets (see [Cla99]).

In practice, a data source could be seen as a data producing service, and described the same way as services in Section 3.2. However, given the special need of domain ontology mapping, we have provided a simple data ontology format shown in Figure 21.

[8] It should be noted that the grammar allows a "conversion" attribute used for instance in converting Euros to US dollars. This is not used in our case since all the currency values are expressed in USD.

4 Application: Resource, Service and Data Directories

Interestingly, Baker et al. in [BAFB05] have little to say about finding services in the Grid. However, there are several tools available for data and service directories. For data directories, file replica catalogs like Globus Replica Catalog ([S+02]), Globus Replica Location Service ([C+04]), European Data Grid's Reptor ([GKL+02]) and EGEE's[9] Fireman [EJ05] have been successfull in high energy physics Grids. However, these directories typically allow only mappings between nicknames ("logical filenames") and several locations of the file ("physical filenames"). The OGSA-DAI (Open Grid Service Architecture - Data Access Interfaces) project has worked for a generic Grid database access (see e.g. [A+05b]).

Service directories include the Universal Description, Discovery, and Integration protocol UDDI (see [Kre01]). As described in [LB05], UDDI is an industry standard for service registration and publication. A service provider uses UDDI to advertise the services (as WSDL description) that it makes available. A client uses UDDI to find the appropriate services for its purposes. The "yellow pages" information in UDDI allows organising the services in various categories. Given these properties, UDDI would work well as a Grid service directory, but apparently Grid systems do not usually employ UDDI (see, however [BBG+05]), nor do UDDI solutions normally use GSI authentication and authorisation.

There are other, profoundly Grid style, approaches to the same problem, for instance Globus' Monitoring and Discovery System (MDS) and ARC's Runtime Environments (see [The04b]). In the latter framework, the service provider prepares an environment for the application. These environments have standardized names, for instance "APPS/GRAPH/POVRAY-3.6". The service provider inserts the initialization scripts of such an environment in a directory where the local information system discovers it, and declares that it is available (for details, see [Kon04]). However, the user is supposed to know (or to look up) the standartised name in order to request it for his job.

In our design, the directory contains pointers to descriptions of resources, services and data. They are currently not registered automatically to the directory; instead, a user interface for their registration and use is provided.

The basic architecture for the system is shown in Figure 22. There, services in a GSI-enabled server are expressed in WDSL as in Figure 18. They are registered (1) in the Grid service directory as pairs of the operation (url, in this case `https://grid.cs.uta.fi:8443/Services#rdfquery`; the parameters, this case proxy, rdf-file, queryfile) and the description of the operation. With the client, the user can store files (or other data sources, described using the ontology of Figure 21) in GSI-enabled file storages and register tuples consisting of the file name, description and a nickname in the directory (2). The user can, too, query files and services by nickname in the directory. If the user wants to run a service, he queries the directory by the client and receives a group of service names (3,4). He selects the service he wants and is prompted for the parameters by the client. The service call with the parameters is sent to the Grid server by the client (5). The server recovers the files given as parameters

[9] EGEE is a European Union funded Grid project. The acronym stands for Enabling Grids for E-science.

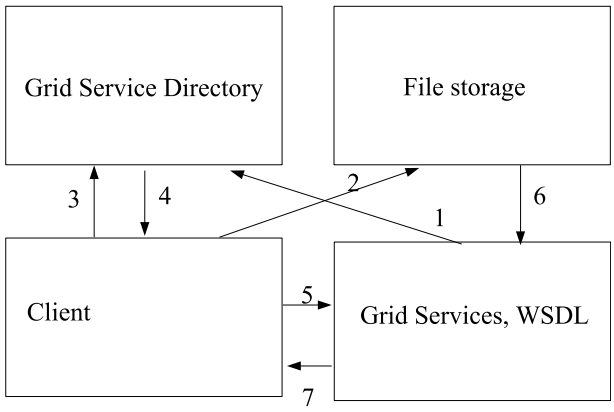


Figure 22: Architecture

(6) and the results are returned to the user (7).

As an example, we consider the `rdfquery` service of Figure 18. The implementation is related to on-demand OLAP[10] cube construction methodology presented in [NTN05a]. Our aim is to support the construction as efficiently as possible under some limiting circumstances, namely that

- The data to be queried is divided into mutually exclusive files.
- The data in each of the files conforms with our RDF ontology.
- Each of the data files can be queried independently (a consequence of the previous items) and the results can be combined mechanically in a file that can be further processed for loading into an OLAP database.

The data in question represents world trade of all types of products between all countries during the 1980's. Our example RDQL query (in Figure 24) recovers tuples $\langle$ value of trade, exporting country, importing country, product, the main group of product, year $\rangle$ such that the importing country is France, the exporting country is Finland and the product's main group is wood industry related. We have implemented the query facility using Sesame RDF engine [BKvH02]. Sesame's performance is most satisfactory, but complex queries with large input data are very demanding. We were unable to run the query using a data source that combines all the input (about 5 million RDF-lines). However, using a source file representing one year of trade, the result can be computed in about 30 minutes.[11] The `rdfquery` operation is mainly a simple wrapper for Sesame such that it transfers the query and input file into a local "staging

[10] OLAP stands for On-Line Analytic Processing.

[11] We utilized a double processor AMD Opteron at 1400 MHz, 2 GB RAM.

```
universe = java
executable = x.class
arguments = x --query y --data z
output = job.out
error = job.err
transfer_input_files = x.class, y, z, ..
jar_files = ..
when_to_transfer_output = ON_EXIT
log = job.log
queue
```

Figure 23: A Condor job submission file

area" and executes the query by calling Sesame. In addition to just launching the command the service software runs it in a local cluster. Condor software (see [LLM88]) has been used for local distributed computing inside the cluster. Figure 23 presents the command file that the service launches for Java program "x" with parameters "–query y –data z".

The `rdfquery` service is declared in a service and data directory that has its own interface for storing and retrieving information. All these operations are WSDL service calls. For communication with the directory, the following operations are provided (we omit the WSDL for brevity):

- `send_and_register_file(string buffer, string filename, int part, int allparts, string nickname, string description)`: sends a potentially very large file in many pieces. The nickname must be unique.
- `register_url(string url, string nick, string category, string description)`: for registering and describing external files and services.
- `get_url_by_nick(string nick)`: returns an url that matches the nickname.
- `get_by_description(string searchstring)`: returns files or services that match the description.
- `get_by_category(string searchstring)`: returns services that match the category.
- `get_categories()`: returns the categories of services.

The user uploads the data files, and defines their nick names, by using the `send_and_register_file` operation. In our example, each of the files representing one year of trade between all countries has been uploaded with nick names "trade1980", "trade1981" etc. The query file, shown in Figure 24 is, likewise, uploaded with a nick name "q6". The user then proceeds to search the service and executes it as in Figure 25. There, the user's proxy file is transfered automatically and it is used by the server to transfer the files.[12]

[12]This implementation should be replaced by a proper proxy delegation (see Section 2.1) or a MyProxy server [NTW01] in the future.

```
select ?value, ?ecountry, ?icountry,
       ?product, ?maingroup, ?year
where (?product  rdf:type schema:Product)
(?fr  rdf:type schema:FactRow)
(?product  schema:hasMainGroup ?maingroup)
     (?ecountry rdf:type schema:Country)
     (?icountry rdf:type schema:Country)
     (?fr schema:hasExportCountry ?ecountry)
     (?fr schema:hasImportCountry ?icountry)
     (?fr schema:hasProduct ?product)
     (?fr schema:hasYear ?year)
     (?fr schema:hasValue ?value)
and (?ecountry eq schema:FI)
and (?icountry eq schema:FR)
and (?maingroup eq schema:P2)
using schema for
  <http://wiki.hip.fi/xml/ontology/olap.owl#>
```

Figure 24: A query file

```
perl register.pl
1 - query by a description
2 - register a file
3 - query by nick
4 - register a service
5 - run a service
6 - get service categories
Please enter a number (1,2,3,4,5,6) and hit enter. Anything else to quit.
5
Please enter a string in the service description: rdf
1: service: http://grid.cs.uta.fi:8080/Services.wsdl#rdfquery(PROXY,rdf,query) description: rdf query
Please enter the number of the service:
1
Please select input for parameter rdf
Please enter a nick: trade1982
Please select input for parameter query
Please enter a nick: q6
If you want to save the response in a file, enter filename, otherwise enter.
trade1982res.xml
```

Figure 25: Client execution

By starting several client programs, the user can execute the queries for each year in parallel. The results can be recovered and combined by the user (or another Grid service).

Several improvements are being planned in the system. Currently, the system does not make a proper difference between a service (e.g. `rdfquery`) and a resource that provides (e.g. computer grid.cs.uta.fi acting as a front end of a cluster). Implementing this will provide a natural framework for a more intelligent resource broker that unifies the technologies of resource and service description and discovery as follows:

- The user finds a service that he wants to execute (and has the rights to use) as in Figure 25.
- There can be several instances of the service provided by different resources, or the service can be “unbound”, i.e. a set of Java classes that can be executed in

any resource providing a suitable Java environment, enough processing capacity and disk space etc.

- Based on the location or requirements of the service (expressed as an instance of the service ontology), the resource information (expressed as instances of the resource ontology) and the location of the data, the resource broker finds an optimal place to run the service.
- The service is run and the results stored for further processing or returned to the client.

The brokering mechanism should take into account that some services are data intensive and it is profitable to carry out the computation as close to the data source as possible; this is the approach used, for example in GridBlocks (see [KNWN04]). On the other hand, some services are CPU intensive and use only minimal amounts of data. Broker designs in general are discussed in e.g. [YSL05, VBW05].

5 Conclusions and future work: A flexible client for semantic Grid computing

In this paper we have discussed Grid technologies including authentication and authorisation; ontology-based Grid resource and service description; and designed a framework and a prototype for finding and using services in the Grid.

The framework described here allows the possibility of creating user interfaces where the user can interactively select the data and the service that operates the data. The services utilize authentication and authorisation (discussed in Section 2.1) in two ways. On one hand, the service directory can be customised to show certain services to authorised users only. On the other hand, the service implementations themselves are accessible to users based on their Grid identities (proxy subject or virtual organization membership).

The design is based on the client in Section 4 and the design of GridBlocks software in [KNWN04]. We find this approach very user friendly and easy to understand for Grid newcomers, too. The Grid resources and services present themselves to the user in a manner that would enable to user to ignore quite a many of the technical concepts in conventional computing. For instance, the user would not need to know where the data and services are. Putting it even more radically, the user can forget the concept of "computer program"; he can operate completely on the level of data and services that process data.

Figure 26 shows the basic user interface, and its variant for a mobile phone (see [Kar05b]). The basic user interface (in the process of development) can be used in the following way:

- The user creates a proxy certificate using the buttons in the upper row.
- The editing area provides the user with basic word processing capabilities and "placeholders" of Grid services and data. The data and services are accessible to the user by pressing the right hand side mouse button.

- With the click of the mouse button, a context menu of services is shown to the user. There, the client contacts the service directory and recovers the services that are accessible to the user. The services are grouped in categories.
- The user browses the services by category or searches them by key words. Once the user has located a service, he will be prompted for input – the data that the service utilises. The user provides the data.
- The service with the appropriate data is started in a remote server. A placeholder is created for the results or their visualisation.
- The user can either wait for the processing of the computation or save the current document in a Grid storage server. Either way, once the processing is finished, the results are shown in the place of the placeholder.

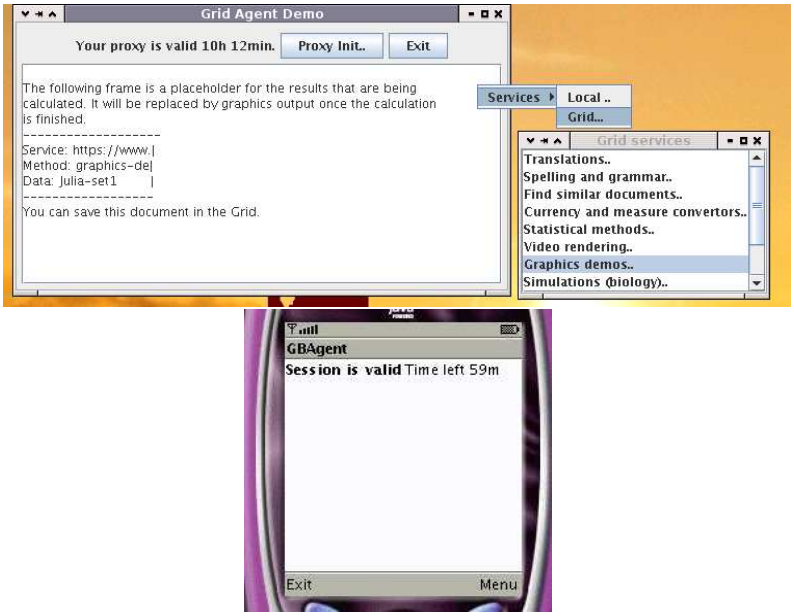


Figure 26: Agent user interface

GridSite software, that is the basis of our server implementation, also supports https communication with a web browser. In order to enable this communication, the user needs to import his certificate in the browser. It should be noticed that this is a plain user certificate (in PKCS format, see [RSA99]) and cannot incorporate VOMS or other proxy extensions. However, the user's full proxy can be made available to the server using a MyProxy server (see [NTW01]) or by transfering the proxy to the server by the client. We have written a client software for our service platform using Mozilla extensions and libraries (see [BKO+02]). The libraries enable the client to call WSDL

functions in the servers, and to format the output for the browser. An example is shown in Figure 27.

Figure 27: A Mozilla client

Acknowledgements

The author is grateful to Mr. Mika Mustikkamäki and Ms. Xiao Wang for their input in a previous versions of Section 2.1, to Dr. Tapio Niemi and Mr. Santtu Toivonen for a previous version of Section 3.1, to Mr. Juho Karppinen for a permission to use Figure 26, and to Dr. Sergio Andreozzi for his permission to use Figure 12 in publication [NTN05b].

References

[A+05a] S. Andreozzi et al. GLUE Schema Specification Version 1.2, Final Specification 3rd Dec 2005. Available at http://infnforge.cnaf.infn.it/glueinfomodel/uploads/Spec/GLUEInfoModel_1_2_final.pdf, 2005.

[A+05b] M. Antonioletti et al. The design and implementation of Grid database services in OGSA-DAI. *Concurrency and Computation: Practice and Experience*, 17(2-4), 2005.

[ACC+03] R. Alfieri, R. Cecchini, V. Ciashini, L. dell'Agnello, A. Frohner, K. Lorentey, and F. Spataro. VOMS an authorization system for virtual organizations. In *Proceedings of the 1st European Across Grids Conference - Santiago de Compostela, Spain, 13-14 February 2003*, 2003.

[And04] S. Andreozzi. GLUE - DataTag Project, work package 4: Interoperability between Grid domains. Available at http://server11.infn.it/datatag/wp4, 2004.

[AU95] A. Aho and J. Ullman. *Foundations of Computer Science*, chapter 10. Freeman, 1995.

[AvH04] G. Antoniou and F. van Harmelen. *A Semantic Web Primer (Cooperative Information Systems)*. The MIT Press, 2004.

[B+04] S. Bechhofer et al. OWL Web Ontology Language Reference. World Wide Web Consortium, February 2004. W3C Recommendation, available at: http://www.w3.org/TR/owl-ref/.

[BAFB05] M. Baker, A. Apon, C. Ferner, and J. Brown. Emerging grid standards. *IEEE Computer*, April 2005.

[BBG+05] S. Banerjee, S. Basu, S. Garg, S. Garg, S. Lee, P. Mullan, and P. Sharma. Scalable Grid Service Discovery Based on UDDI. In *3rd International Workshop on Middleware for Grid Computing - MGC 2005*, Grenoble, France, December 2005.

[BGG03] J. Brooke, K. Garwood, and C. Goble. Interoperability of Grid Resource Descriptions: A Semantic Approach. In *The First GGF Semantic Grid Workshop*, 2003. Available on: http://www.semanticgrid.org/GGF/ggf9/john/.

[BHL99] T. Bray, D. Hollander, and A. Layman. Namespaces in XML. The World Wide Web Consortium, 1999. Available at http://www.w3.org/TR/1999/REC-xml-names-19990114/xs.

[BKO+02] D. Boswell, B. King, I. Oeschger, P. Collins, and E. Murphy. *Creating Applications with Mozilla*. O'Reilly, 2002.

[BKvH02] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *Proceedings of the 2002 International Semantic Web Conference*, 2002.

[BLHL01] T. Berners-Lee, J. Handler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[C+04] A. L. Chervenak et al. Performance and Scalability of a Replica Location Service. In *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04)*. IEEE Computer Society Press, 2004.

[Cat03] C. Catlett. Chairman's forum. *Grid Connections – News and Information for the Global Grid Forum Community*, 1(3), 2003.

[cc04] crypt comments@math.ncsu.edu. "cryptography faq". Available at http://www.faqs.org/faqs/cryptography-faq, 2004.

[Cla99] J. Clark. XSL Transformations (XSLT) Version 1.0 – W3C Recommendation 16 November 1999. The World Wide Web Consortium, 1999. Available at http://www.w3.org/TR/xslt.

[CLZ85] D. C. Clark, M. L. Lambert, and L. Zhang. RFC 969 – NETBLT: A Bulk Data Transfer Protocol. Technical report, Network Working Group, 1985.

[DFP+96] T. DeFanti, I. Foster, M. Papka, R. Stevens, and T. Kuhfuss. Overview of the I-WAY: Wide area visual supercomputing. *International Journal of Supercomputer Applications*, 10(2), 1996.

[E+06] M. Ellert et al. ARC middleware for lightweight computational Grids. To appear, 2006.

[EJ05] EGEE-JRA1. Egee user's guide, glite data management catalog - cli. Technical report, EGEE, 2005.

[F+97] S. Fitzgerald et al. A Directory Service for Configuring High-performance Distributed Computations. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, pages 365–375. IEEE Computer Society Press, 1997.

[F+02] J. Frey et al. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Cluster Computing*, 5(3), July 2002.

[FFG+04] I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, I. Sedukhin, D. Snelling, W. Vanbenepe, and S. Weerawarana. Modeling stateful resources with web services. Technical report, IBM, 2004.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616 – Hypertext Transfer Protocol – HTTP/1.1. Technical report, Network Working Group, 1999.

[FK97] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

[FK98] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[FK03] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2 edition, 2003.

[FKNT04] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The physiology of the Grid. Draft 1.0 as of July 2004, available at http://www.globus.org/research/papers/ogsa.pdf, 2004.

[G+94] A. Geist et al. *PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing*. The MIT Press, 1994.

[GKL+02] L. Guy, P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger. Replica Management in Data Grids. Technical report, European Data Grid, 2002.

[GM95] J. M. Galvin and S. L. Murphy. Using public key technology – issues of binding and protectio. Technical report, The Internet Society, 1995.

[Gon01] L. Gong. Project jxta: A technology overview. Technical report, Sun Microsystems, 2001.

[Gri02] A. Grimshaw. What is a grid? *Grid Today*, 1(26), 2002.

[GW97] A. S. Grimshaw and W. A. Wulf. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1), 1997.

[HDH+04] A. Harth, S. Decker, Y. He, H. Tangmunarunkit, and C. Kesselman. A semantic matchmaker service on the grid. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 326–327. ACM Press, 2004.

[HS95] T. A. Howes and M. C. Smith. A Scalable, Deployable Directory Service Framework for the Internet. Technical Report 95-7, CITI (Center for Information Technology Integration), University of Michigan, USA, 1995.

[IT97] ITU-T. Itu-t recommendation x.509. Technical Report ISO/IEC 9594-8:1997, International Telecomunication Union, 1997. Information technology - Open Systems Interconnection - The Directory: Authentication framework.

[Kar05a] N. Karlsson. Enabling multiple hip identities on linux. Master's thesis, Helsinki University of Technology, 2005.

[Kar05b] J. Karppinen. Data Intensive Mobile Grid Services. Master's thesis, Helsinki University of Technology, February 2005.

[KAS+02] G. Karvounarakis, S. Alexaki, M. Scholl, V. Christopides, and D. Plexousakis. RQL: A declarative query language for RDF. In *Proceedings of the World Wide Web conference 2002, May 7-11, 2002, Honolulu (WWW2002)*. IEEE Press, 2002.

[KBM02] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software – Practice and Experience*, (32), 2002.

[KGN05] T. Koponen, A. Gurtov, and P. Nikander. Application mobility with host identity protocol. In *Proceedings of the Network and Distributed System Security Symposium*. The Internet Society, 2005. Available at http://www.isoc.org/isoc/conferences/ndss/05/workshop/koponen.pdf.

[KNWN04] J. Karppinen, M. Niinimaki, J. White, and T. Niemi. Gridblocks – web portal and client for distributed computing. In *Proceedings of ICEIS 2004, 6th International Conference on Enterprise Information Systems*, Porto - Portugal, April 2004.

[Kon04] B. Konya. The nordugrid/arc information system, technical description and reference manual. Available at http://www.nordugrid.org/documents/arc_infosys.pdf, 2004.

[Kre01] H. Kreger. Web services conceptual architecture. Technical report, IBM, 2001.

[KT05] H. Kishimoto and J. Treadwell. Defining the Grid: A Roadmap for OGSA Standards. Available at http://www.ggf.org/documents/GFD.53.pdf, September 2005.

[Lan98] D. Lange. Mobile objects and mobile agents: The future of distributed computing? In *Proceedings of The European Conference on Object-Oriented Programming '98*, 1998.

[LB05] M. Li and M. Baker. *The Grid – Core Technologies*. Wiley, 2005.

[LLM88] M. Litzkow, M. Livny, and M. Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.

[LvS02] S.A. Ludwig and P. van Santen. A Grid Service Discovery Matchmaker Based on Ontology Description. In *EuroWeb 2002 Conference*, 2002.

[MBL+03] D. Martin, M. Burstein, O. Lassila, M. Paolucci, T. Payne, and S. McIlraith. Describing Web Services using OWL-S and WSDL. DAML-S Coalition Working Document, October, 2003.

[McN05] A. McNab. The gridsite web/grid security system: Research articles. *Softw. Pract. Exper.*, 35(9), 2005.

[MvOV96] A. J. Menezes, P. C. van Ooschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[Nie02] H. Niederreiter. *Coding theory and cryptology*, volume 1 of *Lecture Notes*. National University of Singapore, 2002.

[NTN05a] T. Niemi, S. Toivonen, and M. Niinimaki. Utilising Semantic Web in Data Integration for OLAP. In *Proceedings of the VLDB Workshop on Ontologies-based techniques for Databases and Information Systems*, 2005.

[NTN05b] M. Niinimaki, S. Toivonen, and T. Niemi. Modelling, searching and utilising grid resources. In *Proceedings of the 15th European-Japanese Conference of Information Modelling and Knowledge Bases, Tallinn, Estonia*, May 2005.

[NTW01] J. Novotny, S. Tuecke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. In *10th IEEE International Symposium on High Performance Distributed Computing*, 2001.

[OV99] M. T. Özsu and P. Valduriez. *Principles of Distributes Database Systems*. Prentice Hall, 2 edition, 1999.

[PCS03] L. Pouchard, L. Cinquni, and G. Strand. The earth system Grid discovery and semantic web technologies. In *Semantic Web Technologies for Searching and Retrieving Scientific Data, ISWCII*, 2003.

[PTF05] A. Polleres, I. Toma, and D. Fensel. Modeling services for the semantic grid. Technical report, Digital Enterprise Research Institute (DERI), 2005. Dagstuhl Seminar Proceedings 05271 Semantic Grid: The Convergence of Technologies.

[RJS05] D. De Roure, N. R. Jennings, and N. R. Shadbolt. The semantic grid: past, present, and future. *Proceedings of the IEEE*, 93(3), 2005.

[RSA99] RSA Laboratories. Pkcs 12 v1.0: Personal information exchange syntax. Technical report, RSA, 1999.

[S+02] H. Stockinger et al. File and Object Replication in Data Grids. *Cluster Computing*, 5(3), July 2002.

[Sea04] A. Seaborne. RDQL - a query language for RDF. Technical report, W3C Member Submission, 2004. Available at http://www.w3c.org/Submission/RDQL/.

[SNWN03a] F. Siebenlist, N. Nagaratram, V. Welch, and C. Neuman. Security for virtual organizations: federating trust and policy domains. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2003.

[SNWN03b] F. Siebenlist, N. Nagaratram, V. Welch, and C. Neuman. Security for virtual organizations: Federating trust and policy domains. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd edition, 2003.

[T+04] S. Tuecke et al. RFC 3820 – Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. Technical report, Network Working Group, 2004.

[TDK03] H. Tangmunarunkit, S. Decker, and C. Kesselman. Ontology-based resource matching in the grid the grid meets the semantic web. In *1st Workshop on Semantics in Peer-to-Peer and Grid Computing*, 2003.

[Thea] The Globus Alliance. GSIFTP Tools for the Data Grid. Available at: http://www.globus.org/toolkit/docs/2.4/datagrid/deliverables/gsiftp-tools.html.

[Theb] The Globus Alliance. ”gt 2.4 gsi: Managing the grid-mapfile”. Available at http://www.globus.org/toolkit/docs/2.4/gsi/grid-mapfile_v11.html.

[Thec] The Globus Alliance. Gt information services: Monitoring & discovery system (mds). Available at http://www.globus.org/toolkit/mds/.

[The00] The Globus Project. GridFTP – Universal Data Transfer for the Grid. White paper, available at: http://www.globus.org/toolkit/docs/2.4/datagrid/deliverables/C2WPdraft3.pdf, 2000.

[The03a] The Globus Alliance. The WS-Resource Framework. Available at http://www.globus.org/wsrf/, 2003.

[The03b] The Object Management Group. Unified modelling language specification. Technical report, OMG, 2003. Available at http://www.omg.org.

[The04a] The Globus Alliance. GT 4.0: Security. Available at http://www.globus.org/toolkit/docs/4.0/security/, 2004.

[The04b] The Nordugrid Consortium. Runtime environment registry. Available at http://www.csc.fi/grid/rer/, 2004.

[Tho00] S. A. Thomas. *SSL & TLS Essentials: Securing the Web*. Wiley, 2000.

[VBW05] S. Venugopal, R. Buyya, and L. Winton. A grid service broker for scheduling e-science applications on global data grids. *Journal of Concurrency and Computation: Practice and Experience*, 2005.

[Vin97] S. Vinoski. Corba: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications*, Feb 1997.

[W+04] Von Welch et al. X.509 Proxy Certificates for Dynamic Delegation. In *Proceedings of the 3rd Annual PKI R&D Workshop*. NIST, April 2004.

[W3C99] W3C. XML Schema Requirements – W3C Note 15 February 1999. The World Wide Web Consortium, 1999. Available at http://www.w3.org/TR/NOTE-xml-schema-req.

[W3C04a] W3C. RDF Primer. The World Wide Web Consortium, 2004. Available at http://www.w3.org/TR/rdf-primer.

[W3C04b] W3C. RDF Vocabulary Description Language 1.0: RDF Schema. The World Wide Web Consortium, 2004. Available at http://www.w3.org/TR/rdf-schema.

[W3C04c] W3C. Resource Description Framework (RDF). The World Wide Web Consortium, 2004. Available at http://www.w3.org/RDF/.

[W3C04d] W3C. Web services architecture. W3C Working Group Note 11 February, 2004.

[WNN03] J. White, M. Niinimaki, and T. Niemi. OpenGrid - a web based portal to Grid applications (poster). In *The 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, (CCGrid 2003)*, Tokyo, Japan, May 2003.

[YSL05] C. Yang, P. Shih, and K. Li. A High-Performance Computational Resource Broker for Grid Computing Environments. In *19th International Conference on Advanced Information Networking and Applications (AINA'05)*, 2005.