



Metadata Management

Metadata Information Providers¹

Deliverable	D2.3
Authors	Working Group Metadata Management
Editors	Mikael Höggqvist, Frank Breitling, Hans-Martin Adorf, Tobias Scholl, Michael Braun, Thomas Radke
Date	April 3, 2007
Document Version	1.0.6
Current Version	1.0.6
Previous Versions	

A: Status of this Document

Deliverable D3 of working group 2.

B: Reference to project plan

Third deliverable of working group *Metadata Management*.

C: Abstract

¹This work is part of the AstroGrid-D project and D-Grid. The project is funded by the German Federal Ministry of Education and Research (BMBF).

D: Change History

Version	Date	Name	Brief summary
1.0.6	3.04.2007	Frank Breitling	Corrected page width
1.0.5	30.03.2007	Michael Braun	Corrections by Robert Tucker
1.0.4	23.03.2007	Frank Breitling	changed Robotic Telesopes and Job Monitoring
1.0.3	21.03.2007	Michael Braun	Merge of subsection "ProC job monitoring" into "Grid Job Monitoring"
1.0.2	20.03.2007	Michael Braun	Minor corrections
1.0.1	20.03.2007	Frank Breitling	Added an illustration for the information flow for job monitoring as requested by Harry. A short discussion of foreseen improvements and the current limitations is added.
1.0.0	19.03.2007	Mikael Höggqvist	Spell-check and final release.
0.10.0	08.03.2007	Michael Braun	Extension of subsection "Grid Resource metadata" by generic XSLT, minor changes to "Robotic telescopes" subsection and conclusion.
0.9.0	21.02.2007	Mikael Höggqvist	Made minor revisions and added references to the Introduction. Prepared document for project draft release.
0.0.14	21.02.2007	Michael Braun	Contribution by Hans-Martin Adorf integrated, conclusion added
0.0.13	21.02.2007	Tobias Scholl	Revision of the data stream management section.
0.0.12	20.02.2007	Frank Breitling	Modified the RDF graph of STELLA-I. Revision of the telescope section.
0.0.11	20.02.2007	Michael Braun	Included references for MDS and OwlMap.
0.0.10	20.02.2007	Thomas Radke	Added figure showing the Service Monitor in action.
0.0.9	19.02.2007	Frank Breitling	Move grid monitoring, added XSLT description for robotic telescope + RDF Graph figures + reference to D 5.3.
0.0.8	18.02.2007	Thomas Radke	Added section on Cactus Metadata Management.
0.0.7	16.02.2007	Thomas Radke	Added section on Grid Service Monitoring.
0.0.6	16.02.2007	Tobias Scholl	Revised section about producers for the data stream management system.
0.0.5	15.02.2007	Michael Braun	Transferred list of producers to subsections, added content to subsection "Grid Resource metadata", subsection on Robotic telescopes extended.
0.0.4	12.02.2007	Tobias Scholl	Added the Data-Stream management producer.
0.0.3	12.02.2007	Mikael Höggqvist	Added introduction, Information Producer section and contribution by Hans-Martin Adorf.
0.0.2	12.02.2007	Frank Breitling	Added section on grid monitoring and robotic telescopes.
0.0.1	29.01.2007	Mikael Höggqvist	Template.

E:

Contents

Abstract	1
Change History	2
1 Introduction	4
2 Information Producers	4
3 Available Information Producers	4
3.1 Grid Resource metadata	5
3.2 Grid Service Monitoring	5
3.3 Grid Job Monitoring	7
3.4 Cactus Metadata Management	9
3.5 Robotic Telescopes	10
3.6 Data Stream Management Producer	11
4 Conclusion	12
References	13

1 Introduction

This document summarizes the different Information Producers developed within AstroGrid-D until March 2007. An Information Producer aggregates metadata to the AstroGrid-D Information Service, described in deliverable D2.1 [7] and implemented in D2.2 [6]. The metadata is mostly related to monitoring of the Grid, ranging from resource monitoring to grid job and specific application monitoring. The rest of the deliverable is organized as follows: section 2 contains a generalized description of an AstroGrid-D Information Producer and section 3 describes the existing Information Producers.

2 Information Producers

The AstroGrid-D Information Service, Stellaris, represents metadata using the Resource Description Framework (RDF) [8], from W3C. In order to use Stellaris as Information Service, an Information Producer (IP), must therefore provide metadata in this format. One of the benefits of using RDF is that there exist several converters from other widespread formats (EXIF, Bibtex, OAI-PMH, ...), see [2] for an overview, and many existing vocabularies/schemes that can be adopted when creating a specific IP. Other useful tools for working with RDF can be found at W3C's RDF site [3]. The following steps are important to consider when implementing an IP, many of which will be given as examples in section 3.

Design the schema. Metadata is a way to bring additional meaning to a resource (file, job, compute cluster, etc.). In RDF a resource has properties, where a property describes the resource in some way. For example the sentence "The car has a color with value blue.", says that the property color has value blue. With the RDF Schema [1] language it is possible to define classes, sub-classes and relations between properties for a certain RDF vocabulary. If the metadata is available in another format already (such as XML) a translation to an RDF vocabulary is necessary.

Generate RDF. When the schema for the metadata has been defined, instances of the schema should be generated. This is typically done by using existing tools listed in [3]. The RDF format supports many different serialization formats, where the most common is RDF/XML. Examples and tips on this step are described in section 3.

Interact with Stellaris. Stellaris is accessible using normal HTTP, why most HTTP clients and libraries (cURL, wget, etc.) supporting GET, PUT, POST and DELETE is sufficient for creating, uploading, retrieving and deleting RDF-based metadata. In order to extract specific metadata, a SPARQL [11] endpoint is provided. SPARQL is an RDF-specific query language.

3 Available Information Producers

List of what has been produced so far within the project. This should include

- Producer purpose

- What vocabulary is used? (Describe here or give a link)
- How is RDF produced? (directly, through translation, libraries, templates) Practical part that is useful for others writing RDF producers
- Experience with the information service. Describe how your setup works, how you include the information service in your system.

3.1 Grid Resource metadata

Status data on Grid resources using the Globus Toolkit are provided by the Monitoring and Discovery Service (MDS). Information about each computing element (CE) - a cluster or a workstation - is contained in MDS using the Grid Laboratory Uniform Environment (GLUE) schema. The current implementation of MDS in GT 4.0.x uses GLUECE schema version 1.1. The schema is filled with reasonable data by information providers like the Ganglia Information Provider which should be configured on all AstroGrid-D resources. More information about MDS can be found in [14].

In order to provide this status information to the AstroGrid-D information service Stellaris, we established a periodically applied translation chain (cron job) from MDS' XML schema into RDF containing all the information from the individual GLUECE sub-schemes in the MDS. As the first step, the XML data are retrieved from the AstroGrid-D WebMDS-Service. Usually, the XML output of MDS contains several copies of a GLUECE schema per individual computing element in conjunction with each available queue etc. By applying an XSL-Transformation, a new XML file is extracted which contains the collection of unique GLUECE schemes only (one per computing element).

The translation from XML to RDF is performed using the XML2RDF routine from the OwlMap package [9].

The obtained RDF document is uploaded to Stellaris where it replaces the document of the previous application of this translation chain. Actual status information about computing resources can be retrieved using SPARQL queries.

Most recently, we succeeded in developing a short but generic XSL transformation to translate arbitrary XML files into the RDF/XML format by introducing a generic namespace treatment into the XSL transformation developed for the RTML-to-RDF translation. This new and generic XSL transformation has been used in order to translate the entire AstroGrid-D-MDS information into RDF/XML, thus establishing an additional and exhaustive method to provide MDS information to Stellaris.

Currently, this complete set of MDS information in Stellaris is updated once per hour. Due to the comparably large amount of RDF/XML data (about 1.5 MB) the update process lasts for almost one minute. We are currently working on methods to accelerate the upload process to enable a more frequent update of MDS information.

3.2 Grid Service Monitoring


In order to monitor the status of basic Grid services on all known AstroGrid-D resources, a grid monitoring script (written in Perl) was developed in work package WP-I "*Integration of Compute*

and Data Resources into GACG". This script

1. iterates over a given list of Grid resources (which can be specified by the user or automatically queried from an AstroGrid information service),
2. runs predefined tests for a set of Grid services on each resource,
3. as a result displays the status for each Grid service test (passed/failed/timeout). In the case of a failed test, a log-file is placed in the current working directory showing the stderr messages of the test performed.

The test results obtained by the script are translated into a RDF/XML document using an RDF schema namely the `astrogrid-health-monitor-schema` ispecifically designed for this purpose. The RDF/XML document is finally uploaded to an AstroGrid-D information service: for this a routine in the Perl script opens a TCP/IP socket connection to the machine and port where the external information service is running, and sends the serialised RDF/XML string as data in a HTTP PUT operation following the standard web protocol.

After the service monitoring results have been uploaded to the information service, they can then immediately be queried from a *Grid Service Monitor* web-page (see figure 1 for results from an example status query) which is available on the AstroGrid-D intranet (<http://mintaka.aip.de:8080/lenya/intranet/live/grid-status/grid-service-monitor.html>).



DN	Date	GridResource	Status
/C=DE/O=GermanGrid/OU=FZK/CN=Frank Breitling	20.02.2007 11:33:47	Supergrid.aei.mpg.de	failed
/C=DE/O=GridGermany/OU=Leibniz-Rechenzentrum/CN=Tobias Scholl	20.02.2007 11:39:38	Supergrid.aei.mpg.de	passed
/O=GermanGrid/OU=AEI/CN=Robert Engel	20.02.2007 11:30:29	Supergrid.aei.mpg.de	passed
/O=GermanGrid/OU=AEI/CN=Thomas Radke	16.02.2007 16:23:04	Supergrid.aei.mpg.de	passed
/C=DE/O=GermanGrid/OU=FZK/CN=Frank Breitling	20.02.2007 11:33:47	a01.hlrb2.lrz-muenchen.de	failed
/C=DE/O=GridGermany/OU=Leibniz-Rechenzentrum/CN=Tobias Scholl	20.02.2007 11:39:38	a01.hlrb2.lrz-muenchen.de	failed
/O=GermanGrid/OU=AEI/CN=Robert Engel	20.02.2007 11:30:29	a01.hlrb2.lrz-muenchen.de	passed
/O=GermanGrid/OU=AEI/CN=Thomas Radke	16.02.2007 16:23:04	a01.hlrb2.lrz-muenchen.de	passed
/C=DE/O=GermanGrid/OU=FZK/CN=Frank Breitling	20.02.2007 11:33:47	astar.aip.de	passed
/C=DE/O=GridGermany/OU=Leibniz-Rechenzentrum/CN=Tobias Scholl	20.02.2007 11:39:38	astar.aip.de	passed
/O=GermanGrid/OU=AEI/CN=Robert Engel	20.02.2007 11:30:29	astar.aip.de	passed
/O=GermanGrid/OU=AEI/CN=Thomas Radke	16.02.2007 16:23:04	astar.aip.de	passed
/C=DE/O=GermanGrid/OU=FZK/CN=Frank Breitling	20.02.2007 11:33:47	astrodata01.gac-grid.org	passed

Figure 1: Results from a status query for the GSISSH service as shown on the *Grid Service Monitor* webpage

Different predefined queries are available as SPARQL queries:

- Service Monitor metadata information

- hostname of client machine where the test was run from
- date/time of the test
- client user login name
- client user's certificate distinguished name
- status of successful tests for individual Grid Services
 - GSISSH
 - GSISCP
 - GRAM
- status of failed tests for individual Grid Services
- full status overview of all Grid Service tests
- full status of individual Grid Services with error log for failed tests

Both the Grid Service Monitoring script and the RDF schema used to describe the service monitoring metadata can be downloaded from AstroGrid-D SVN via

```
svn checkout svn://svn.gac-grid.org/software/monitoring
```

More documentation can be found on the intranet page <http://mintaka.aip.de:8080/lenya/intranet/live/workpackages/wg1/grid-infrastructure/monitoring.html>.

3.3 Grid Job Monitoring

Grid job monitoring has been developed to provide a first graphical user interface to the status of submitted grid jobs. Monitoring is accomplished using the web service interface of Globus. For each job (submitted with the "o" option) the Grid Resource Allocation and Management (GRAM) of Globus returns an endpoint reference (EPR) as handle to the job state. Monitoring uses the EPR to retrieve the state of a running job from GRAM. This information is then filled in a RDF template. This RDF template contains:

- job id
- job name
- current job state
- host name
- user name
- start time
- stage in time
- start time of program execution

- stage out time

After a state has been received and the RDF template has been updated, it is uploaded to the information service through its HTTP interface using the command line tool “cURL”. Currently the job submission and monitoring is done from the same host as illustrated in Fig. 2. A disadvantage of this concept is that the submission host is required to be online for monitoring. This disadvantage can be avoided if the monitoring is delegated to the same host where the job is executed. The provision of information directly from the execution host is indicated by the black dashed arrow which replaces the information flow back through the submission host as indicated by the red arrows.

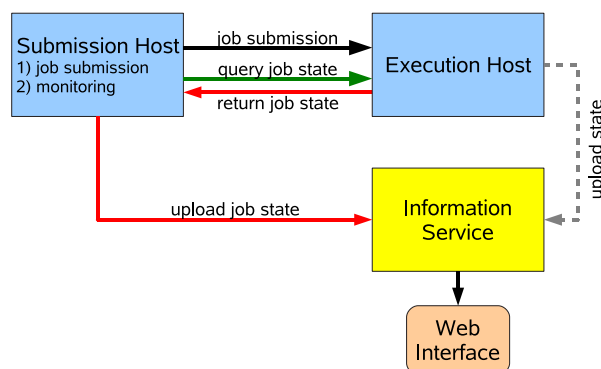


Figure 2: Illustration of information exchange for job monitoring. The dashed arrow indicates the foreseen improved information flow which will replace the route back through the submission host which is indicated by the red arrows.

The grid job monitoring has been first developed for the DYNAMO use case and found to be very useful during test runs. The necessary software is available through the AstroGrid-D web site [10]. However, the code can be easily adopted by other use cases of AstroGrid-D.

A first adoption has been made for the use case ProC. Similar to the processing for the use case DYNAMO, the metadata is produced as follows; the EPR of the job is returned by the job submission command. The job status is periodically monitored using the 'globus-job-status' command based on the returned EPR. The system 'date' command is applied to generate the corresponding time stamp. Some of the metadata is sent along with the job submission as a parameter. Some other information is hard-coded into the respective ProC script that submits the individual workflows. This metadata is encoded in a template RDF file. SED replace commands are used in order to replace the place-holders in the template by the real information.

From the information service the information can be retrieved through SPARQL queries or more conveniently through a special web interface, the Grid-Timeline. Pointing a web browser to this interface each job is represented by a “timeline”, i.e. a horizontal bar which is proportional to the run time. Color codes for the different job states provide an intuitive overview of the progress of submitted grid jobs. Additional information, such as job ID, name of the user and name of the job, can be obtained through a mouse click on a jobs timeline.

3.4 Cactus Metadata Management

For the Cactus use case a specific application scenario was developed to automate the procedure of regular Cactus tests and allow users to conveniently monitor the status and history of Cactus test simulations. This scenario was realised using AstroGrid-D technology: (1) the information service developed in work package WP-II for storing and managing application-specific metadata, and (2) the GridSphere portal framework provided by work package WP-VII to build a Cactus User Portal as a standardised web-based user interface to access and query application-specific metadata. In work package WP-VI a `CACTUS INTEGRATION TESTS` module for generating the metadata and a `CACTUSRDF` portlet for presenting the metadata were developed.

The `CACTUS INTEGRATION TESTS` module consists of a collection of Perl scripts functioning as wrappers around the Cactus test suite mechanism. They automate the execution of individual interdependent unit tests and are responsible for extraction and evaluation of test results from available log-files and the translation of these results into an RDF document. For this purpose, an RDF schema was designed describing the following items of information for a given integration test:

- a descriptive name identifying this test
- the exact date/time of the test
- the hostname of the machine the test was run, plus the total number of processors used
- the login name of the user who ran the test
- the configuration options and thornlist used to build a Cactus executable
- the status results (`succeeded/failed`) and log-files for each individual unit test:
- for the test suites, also the names and a summary of `passed/failed` tests

The resulting RDF document is serialized into RDF/XML and finally uploaded by the `CACTUS INTEGRATION TESTS` module to an external AstroGrid-D information service to store and archive the integration test results.

Closely related to the generation of Cactus metadata on the application side is its presentation through a human-machine interface in the form of a web-based Cactus user portal. Such a portal, based on the GridSphere portal framework, has been deployed by work package WP-VII; it is available online under <https://portal.cactuscode.org>.

In work group WP-VI the necessary Cactus metadata management portlet (called `CACTUSRDF`) was developed. The `CACTUSRDF` portlet provides functionality to query Cactus integration test results from an external AstroGrid-D information service and present them in different possible views to the user:

1. a summary view of all most recent integration tests from all test machines, showing the status of all unit tests
2. a detailed view of Cactus test suites for an individual integration test, showing the status of all test suites

3. a history view for an individual Cactus test suite, queried over all available integration tests results on all test machines

For the queries, the user can also specify parameters to restrict the resulting metadata shown in the portal, e.g. by an individual Cactus integration test (identified by its name), by the user who ran the test (identified by the user's login name), or by a specific test machine where the test was run (identified by the hostname).

Internally, each of the different views is implemented as a dynamically generated SPARQL query submitted to the external information service. The `CACTUSRDF` portlet is implemented in Java and uses the Jena RDF framework for all transactions with the AstroGrid-D information service.

Complete documentation for the *Cactus Integration Tests* module and the `CACTUSRDF` portlet is available in [12].

3.5 Robotic Telescopes

The integration of robotic telescopes into the AstroGrid-D infrastructure aims at a global network of astronomic instruments based on grid technology. A first step in this direction is the usage of the grid information service as interface to metadata of telescopes. A standard for metadata of telescopes and observation requests has been developed by the Heterogeneous Telescope Networks (HTM) consortium. Their protocol standard is the Remote Telescope Markup Language (RTML) which has also been adopted by AstroGrid-D. Since the Stellaris information service uses RDF for the storage of information, a transformation into RDF is necessary before the metadata can be upload. Details regarding the transformation as well as the uploading and retrieving of information are given in [5].

The possible contents of metadata for telescopes is defined by the RTML schema. and can be divided into static and dynamic metadata. Typical static metadata contains

- telescope id and name
- location: name, longitude, latitude, height
- camera properties: description, detector, plate scale, filters
- focal length
- focal ratio
- aperture

Typical dynamic metadata will be discussed in [4].

Access to the information is possible through the SPARQL query interface and also through a web browser directed to the Telescope Map, which is part of the Grid Resource Map. This software packages is available through [10].

3.6 Data Stream Management Producer

For the data stream management two information producers have been designed. The first information producer describes the *content providers* in the data stream management system. Content providers publish data streams into the data stream management and provide metadata about the available data streams. The second information producer delivers information about *function providers* which are repositories for re-usable components that can be integrated in data-stream processing tasks. Both information producers follow the same basic principle: they create an RDF model based on an internal data structure. The RDF model is serialized afterwards and then transmitted to the information service.

Content Provider. The information producer is realized as proof-of-concept for creating RDF models in Java using the *Jena* library.² Content providers publish data streams into the data management system. The data streams are characterized by structural information, e. g., a *Document Type Definition (DTD)*, and the *frequency* of the data stream. An example for this information is shown in Figure 3. The content providers are represented by the service URL at the container. Thus clients can directly use this information to interact with a content provider, e. g., to subscribe to this data stream.

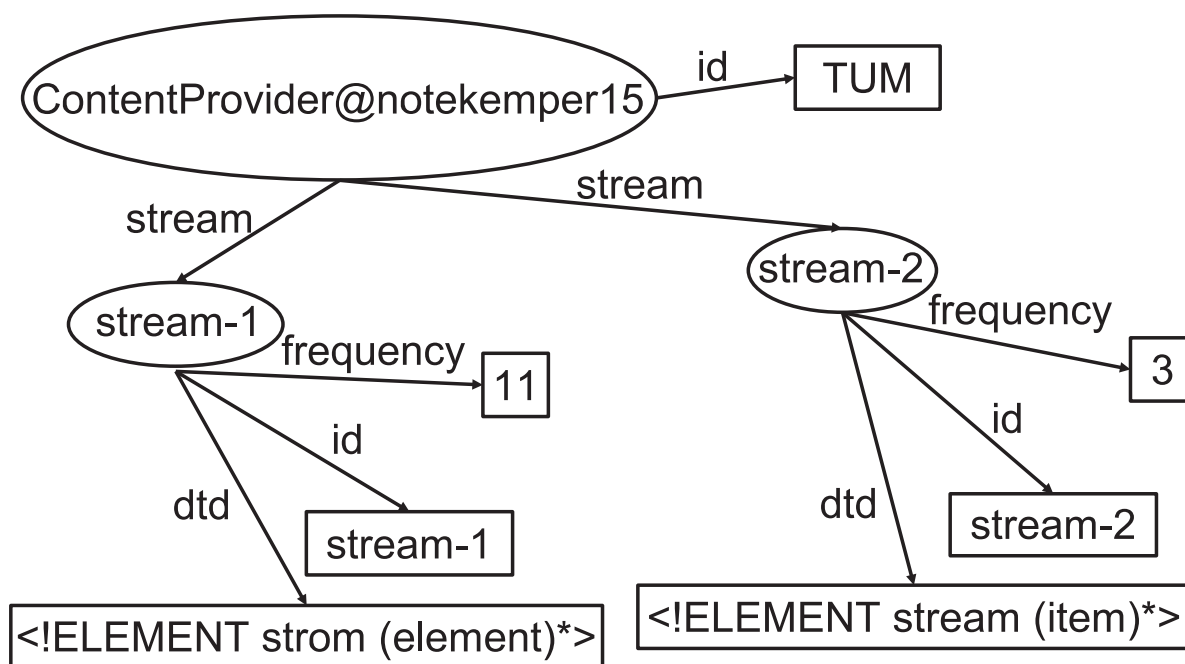


Figure 3: An RDF example graph for the metadata of a content provider.

Function Provider. The data stream management enables researchers in the astrophysics community to share implementations of stream-based data processing modules, so-called *operators*. These are stored in distributed *function providers* easing the maintenance task for the individual author. Authors specify the metadata necessary to reuse their implementation. This includes

²<http://jena.sourceforge.net>

the name of the operator, a detailed description, input parameters, output parameters, configuration parameters, the URL where the archive containing the operator can be downloaded from, the *Distinguished Name (DN)* of the authors' grid-certificate, and several keywords describing the functionality. The certificate information is necessary to sign the operator archive and thus allow the data stream management to verify that the code is from a trustworthy source. A subset of this metadata is replicated at the information service to allow operator discovery based on that information. Having found a reusable operator, researchers can integrate those (usually written by other astronomers) into their own data stream processing tasks based on the information provided by the information service and the function provider. We describe the functionality of this information producer in more detail in the according deliverable by working group 4 [13].

4 Conclusion

Several information producers have already been developed for Stellaris. Some of them are only partly finished - like the information producer for Grid resource metadata where a further speedup of the information upload process is strongly recommended. Work is in progress to improve these individual information producers and their interaction with Stellaris and to develop additional IPs.

F: References / Bibliography

References

- [1] D. Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF schema. <http://www.w3.org/TR/rdf-schema/>, February 2004.
- [2] ESW Wiki. ConverterToRDF. <http://esw.w3.org/topic/ConverterToRdf>, February 2007.
- [3] ESW Wiki. SemanticWebTools. <http://esw.w3.org/topic/SemanticWebTools>, February 2007.
- [4] F. Breitling. Providing Dynamic Metadata of Robotic Telescopes to Stellaris. Technical Report D2.7, AstroGrid-D project, In preparation.
- [5] F. Breitling. Providing Static Metadata of Robotic Telescopes to Stellaris. Technical Report D2.4, AstroGrid-D project, In preparation.
- [6] M. Högvist. Stellaris: The AstroGrid-D Information Service. Technical Report D2.2, AstroGrid-D project, January 2007.
- [7] M. Högvist, T. Röblitz. AstroGrid-D Information Service Requirements Specification and Architectural Design. Technical Report D2.1, AstroGrid-D project, July 2006.
- [8] F. Manola and E. Miller. RDF primer. <http://www.w3.org/TR/rdf-primer/>, February 2004.
- [9] Matthias Ferdinand, Christian Zirpins, and David Trastour. Lifting XML Schema to OWL. <http://vsis-www.informatik.uni-hamburg.de/getDoc.php/publications/204/f%zt-lxs-04.pdf>, <http://fresco-www.informatik.uni-hamburg.de/technology/index.php>, March 2004.
- [10] Members of AstroGrid-D. AstroGrid-D Project Products. <http://www.gac-grid.org/project-products/Software.html>.
- [11] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. <http://www.w3.org/TR/rdf-sparql-query/>, February 2006.
- [12] T. Radke. Prototype Implementation of grid-enabled Monitoring Methods. Technical Report D6.4, AstroGrid-D project, March 2007.
- [13] T. Scholl. Distributed Database Access and Data Stream Management: Distributed Function Providers. Technical Report D4.4, AstroGrid-D project, March 2007.
- [14] The Globus alliance. GT Information Services: Monitoring and Discovery System (MDS). <http://www.globus.org/toolkit/mds/>.