



SIMDAT

Data Grids for Process and Product Development using Numerical Simulation and Knowledge Discovery

Project no.: 511438

Grid-based Systems for solving complex problems – IST Call 2

Integrated project



Deliverable

D.3.2.1 Consolidated report on data access state-of-the-art and roadmap for SIMDAT
&
D.3.2.2 Distributed data repository services for SIMDAT (third version)

Start date of project: 1 September 2004

Duration: 48 months

Due date of deliverable: 01/09/2006

Actual submission date: 23/10/2006

Lead contractor for this deliverable: Intel

Revision: 1.0

**Project co-funded by the European Commission within the Sixth Framework Programme
(2002-2006)**

Dissemination level

PU	Public	X
PP	Restricted to other programme participant (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision history

Date	Version	Author	Modification
28/09/2006	0.1	Michael Krüger	Initial draft outline
05/10/2006	0.2	Mike Boniface, Stephen C Phillips (IT Innovation)	GRIA/OGSA-DAI service part added
06/10/2006	0.3	Dr. Thomas Kentemich	Meteo routing part added
07/10/2006		Michael Krüger	OGSA-DAI wrapper for Auto
09/10/2006	0.4	Michael Krüger	Requirements section updated, Improvements on various sections
11/10/2006	0.5	Michael Krüger	IGOR-FS evaluation added
17/10/2006	1.0	Michael Krüger	Finalisation

Copyright

Copyright © Intel and other members of the SIMDAT consortium, www.simdat.org, 2006.

Table of contents

1	Introduction.....	6
1.1	Purpose.....	6
1.2	Scope of the document.....	6
1.3	Definitions, acronyms and abbreviations.....	7
1.4	References.....	7
2	Technology improvements description.....	9
2.1	GRIA OGSA-DAI Service.....	9
2.1.1	Introduction.....	9
2.1.2	Relevance to Applications	10
2.1.3	Software Architecture	11
3	State-of-the-art surveys and the contribution/position of SIMDAT	15
4	Work provided to the application activities	16
4.1	Meteo activity: VGISC routing mechanism implemented.....	17
4.1.1	V-GISC requirements	17
4.1.2	Routing mechanisms	17
4.1.3	V-GISC specific implementation.....	19
4.1.4	Dynamic node insertion	24
4.1.5	References.....	25
4.2	Auto-1: OGSA-DAI service for TecManager.....	26
4.2.1	Auto-1 PM24 prototype	26
4.2.2	Auto-1 architecture	26
4.2.3	Some of the Auto-1 services	27
4.2.4	Auto 1 data retrieval.....	27
4.2.5	OGSA-DAI activity for Auto 1.....	28
4.3	IGOR-FS evaluation	29
4.4	IGOR-FS Design Scenario.....	29

4.5	Peer-to-Peer	30
4.6	Advantages.....	31
4.7	Properties	31
4.8	Status.....	31
5	Specific and modular requirements documentation.....	32
5.1	Automotive requirements update PM30 & beyond	32
5.2	Aerospace requirements update PM30 & beyond.....	35
5.2.1	Functional Requirements	35
5.2.2	Performance Requirements	36
5.3	Pharmaceutical requirements update PM30 & beyond.....	36
5.4	Meteorology requirements update PM30 & beyond.....	36
5.4.1	Requirements Check List.....	36
5.4.2	Requirements for PM42 as of now	37
6	Roadmap and targets for PM42	40
6.1	Targets for DDRA as of Annex 1	40
6.1.1	M1 (12 months) revisited:.....	42
6.1.2	M2 (18 months) revisited:.....	42
6.1.3	M3 (24 months):	42
6.1.4	M4 (30 months):	43
6.1.5	M6 (48 months):	43
7	Conclusion	44
7.1	Achievements.....	44
7.2	Plans.....	44

1 Introduction

1.1 Purpose

This document represents the deliverables D3.2.1 “Consolidated report on data access state-of-the-art and roadmap for SIMDAT” and D3.2.2 “Distributed data repository services for SIMDAT (third version)” of the Integrated Project IST-2002-511438 (SIMDAT), as specified in the Annex 1-“Description of Work” [1]. It is the combined fifth and sixth deliverable for work package 3 and follows the consolidated requirements report D3.1.1 and the first version of the Distributed Data Repository Access infrastructure D3.1.2 (both after project month 12) and D3.1.3 “SIMDAT Distributed Data Repository Access Infrastructure (Second Version)” and D3.1.4 “SIMDAT Report on SIMDAT Distributed Data Repository Access Infrastructure - evaluation and validation” (both after project month 18).

The intended audience for this document is the application and technology partners within the SIMDAT consortium, as listed in Section 3 of Annex 1 [1] as well as the European Commission Services. SIMDAT partners have a broad range of deep expertise in both application sectors and horizontal technology activities. The document is structured to give a view of the state of the Distributed Data Repository Access (DDRA) activity from the perspective of the technology partners and each of the four application activities as of project month 24 (PM24).

The document presents and discusses the third version of the DDRA infrastructure designed, developed and implemented in conjunction with and on top of the Integrated Grid Infrastructure (as put forward by work package 2). It integrates work with and input from application and technology activities, and is the result of numerous discussions with the respective SIMDAT partners.

1.2 Scope of the document

The main challenge for SIMDAT is to develop and deploy technology and techniques that improve the ability of industrial organizations to collaborate in a flexible and dynamic fashion. This collaboration has to take place at a deep technical level, with applications, databases and resources communicating directly with one another in a controlled and secure fashion. The complex problems to be solved involve multiple data repositories describing many aspects of the product and process development, typically hosted in different departments and at different sites, and not currently linked with each other in a direct fashion. To make the SIMDAT vision reality, these data repositories have to be made accessible in a transparent and reliable way regardless of their location. In some cases, this will also involve replication and synchronization of data repositories. In most industrial sectors like the automotive and aerospace industries, implementations for interlinking the distinct distributed data repositories involved in product design, development and production did not exist prior to the SIMDAT project. The concepts for DDRA services have been developed in PM 1–6 and are reported in Deliverable D3.1.1. An initial implementation of a subset was developed in PM 7–11 and rolled out at the end of PM 11. That implementation has been refined in PM 12-17. In close cooperation with the application activities, detailed feedback on the existing and potential future use cases was harvested, including requirements for the next generation of prototypes, and a list of potential shortcomings of the current DDRA infrastructure. Following that, in PM 18-24, the partners in work packages 2 and 3, basic Grid infrastructure and distributed

data repository access services respectively, have worked together to improve the usability of technologies available to the application activities. As well, the technology partners have cooperated with the application activities to provide custom solutions to fulfill special and immediate requirements.

1.3 Definitions, acronyms and abbreviations

DAIS: database access and integration services

FTP: file transfer protocol

GGF: Global Grid Forum

GRIA: GRID Resources for Industrial Applications (<http://www.gria.org>)

HPC: high performance computing

OASIS: Organization for the Advancement of Structured Information Standards (<http://www.oasis-open.org>)

OGSA: Open Grid Services Architecture (<https://forge.gridforum.org/projects/ogsa-wg>)

OGSA-DAI: Open Grid Services Architecture – Database Access and Integration (<http://www.ogsadai.org.uk>)

RDBMS: relational database management system

SIMDAT: Data Grids for Process and Product Development using Numerical Simulation and Knowledge Discovery

SRB: Storage Resource Broker (<http://www.sdsc.edu/srb>)

VGISC: Virtual Global Information System Centre, a virtual node of a world-wide meteorological information system

VO: virtual organization

WS-I: web service interoperability <http://www.ws-i.org/>

WSRF: web services resource framework
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

XML: extended markup language

1.4 References

1. SIMDAT Annex 1, second version (covering PM 1 to 30)
2. IBM DB2, <http://www-306.ibm.com/software/data/db2/>

3. Oracle, <http://www.oracle.com/index.html>
4. OGSA-DAI Overview, <http://www.ogsadai.org.uk/docs/current/doc/DAIOverview.html>
5. Globus Toolkit Reference, <http://www.globus.org/>
6. Global Grid Forum GGF, <http://www.ggf.org/>

2 Technology improvements description

Complex data repositories are key to the modern design, development, and production of complex products and services. Different data repositories store product design data, results from physical tests, and numerical simulation results characterizing the functional properties of products and processes, as well as knowledge about the design process itself, material properties and development strategies. These data repositories might be a collection of data bases potentially combined with a collection of flat files. The Meteo activity uses some custom database / archive systems that are neither relational database systems nor flat file repositories; these are accessed by using a bespoke request language. Effective correlation of data generated in different departments or at different sites within a global organization is a crucial problem for all industries represented in SIMDAT. This solution requires DDRA services that interact with the semantic definition of the data models and storage formats involved, and enable the retrieval of all relevant information even though the data might be distributed across heterogeneous data repositories.

A first release of the DDRA prototype was derived from existing work, mainly the OGSA-DAI (Open Grid Services Architecture – Data Access and Integration) tools before PM 12. OGSA-DAI specifies a service-based interface for transparent access to data repositories (which can be relational or XML data bases or file systems) across Grid nodes and provides a reference implementation supporting many of the major database engines and packages (IBM DB2, Oracle, MySQL, XIndices, and many more). To accommodate the needs of most of the application activities of SIMDAT for an integrated Grid and DDRA solution, the adaptation and integration of OGSA-DAI with the WS-I based GRIA layer chosen as the Integrated Grid Infrastructure in WP2 was performed within that phase of SIMDAT. The integration solution did require a new GRIA service to be written: the “OGSA-DAI Service”, i.e. a wrapper for OGSA-DAI communicating with OGSA-DAI through direct Java calls, and providing access control through the PBAC mechanisms of GRIA. This GRIA OGSA-DAI service (see below for a more detailed description) has been enhanced significantly during the last couple of months. Generally integration with the WP2 Integrated Grid Infrastructure and support for the higher level grid technologies, i.e. those building on top of the DDRA services was optimized. This enables the application activities to use the available DDRA services in real-world scenarios. Additionally a lot of effort was put into mutually consulting with the application activities. For a few scenarios specially tailored solutions have been made available.

2.1 GRIA OGSA-DAI Service

2.1.1 Introduction

The GRIA OGSA-DAI package is a new application service for GRIA 5 that allows clients to access databases. The OGSA-DAI service can be deployed on its own to provide unmanaged access to databases, or it can be used with the GRIA service provider management package to take advantage of GRIA’s service level agreement and billing capabilities.

2.1.1.1 *Added Value*

The GRIA OGSA-DAI service its focuses on providing a secure and managed access to data resources supporting the key requirements for business deployments include:

- **Easy deployment:** The GRIA OGSA-DAI service is provided as a WAR file, ready to be dropped into a standard servlet container such as Tomcat or bundled with enterprise server EAR files;
- **Easy configuration:** The service provides a web-based configuration and management interface;
- **GRIA's standard security features:** Dynamic security policies to control access to databases and database accounts, transport level encryption using SSL, message level security (WS-Security) providing authentication and integrity checks; and
- **OGSA-DAI session security:** The service uses a custom session manager to ensure that a user's session cannot be compromised.
- **SLA based database QoS:** By also linking the GRIA OGSA-DAI service to the GRIA service provider management package, monitoring, constraining and billing for database usage is supported.

2.1.2 Relevance to Applications

The GRIA OGSA-DAI service has been developed to target generic data usage scenarios that may be found in several of the SIMDAT application sectors.

2.1.2.1 *Data Publication and Subscription*

A service provider may have existing databases that they wish to allow clients to access (often on a read-only basis). The pharmaceutical sector exhibits many such databases containing information valuable to scientists such as EMBL or Uniprot. Using the GRIA OGSA-DAI service's web-based interface, a service provider can connect to any database supported by JDBC (including Oracle, DB2, MySQL, etc). Once connected to the database, the service provider chooses which database accounts to expose to which users by way of standard GRIA access control rules. For example, the service provider may choose to expose to all users a database account that provides read access to the majority of a database, but only expose to a certain organisation a database account that provides access to some privileged information.

By linking the OGSA-DAI service to the GRIA service provider management package, the service provider may monitor, constrain and charge for access by using SLAs, in addition to the access constraints already discussed. An SLA can constrain which databases are available to a user, how many subscriptions are allowed and can charge the user on a one-off basis or using a rolling subscription model.

The service may also be useful in this mode as a means of distributing IGOR-FS entry points.

2.1.2.2 *Distributed Project Teams*

The service can provide support for distributed project engineering teams common in the aerospace and automotive sectors. The lead member of a distributed project team can host the OGSA-DAI service and set up a database to hold project data. By using the access control system discussed above, project team members may be added to the database on a read-only or read-write basis depending upon the privileges associated with their role in the team.

2.1.2.3 *Database Service Provision*

The aerospace prototype in the connectivity phase used a distributed workflow, putting result data into a central database. The database was hosted at one of the aerospace partners, but one can

imagine it being hosted at an independent service provider. The GRIA OGSA-DAI service provides a mode of operation whereby a service provider permits clients to create and manage their own databases on the service provider's system. Using SLAs, the service provider can constrain the number of databases each client can create and (optionally) charge for each database creation.

2.1.3 Software Architecture

2.1.3.1 Overview

Figure 1 shows the architecture of the GRIA OGSA-DAI service deployed within a managed configuration. The GRIA OGSA-DAI service consists of both client-side APIs and server side components. The grey boxes in the diagram indicate existing OGSA-DAI software that we have reused. On the client-side the software includes:

- **GRIA OGSA-DAI Management API:** client side API that allows the management (creation, destruction, access control) on database resources;
- **OGSA-DAI API:** existing API provided by OGSA-DAI WS-I 2.2 to invoke “perform” operations on database resources;
- **GRIA OGSA-DAI handler:** AXIS handler that adds the WS-Addressing header to the service requests in accordance with WSRF industrial profile; and
- **WSS4J handler:** AXIS handler implementing WS-Security specification for message level security.

The client also includes a plug-in for the Graphical GRIA Client that adds functionality to the graphical GRIA client, enabling clients to access and manage their databases without having to write their own Java applications. While this plug-in provides a useful way of experimenting with and demonstrating the OGSA-DAI service, real-world applications are likely to use the OGSA-DAI client API.

On the server-side the software includes:

- **WSS4J handler:** AXIS handler implementing WS-Security specification for message level security;
- **Dynamic Policy:** GRIA's dynamic policy component enforcing authorisation decisions based on the GRIA OGSA-DAI management resource model (see below)
- **GRIA OGSA-DAI Service:** provides the web-service interface for managing, accessing and creating databases at a service provider. The service administrator can choose which users are allowed to create and use databases and which RDBMS backend new databases are created on. A set of web pages is provided with the service to configure and manage the system.
- **RDBMS Manager:** supports the dynamic creation of database resources including the required configuration files for OGSA-DAI
- **OGSA-DAI WS-I 2.2:** existing OGSA-DAI WS-I 2.2 libraries
- **GRIA Role Mapper:** allows OGSA-DAI activities to establish the database username and password to be used to the requesting user in the OGSA-DAI security context

- **GRIA Session Manager:** provides access control restrictions on OGSA-DAI sessions so that only the owner of a session (the user that created it) has permission to execute activities on that session.

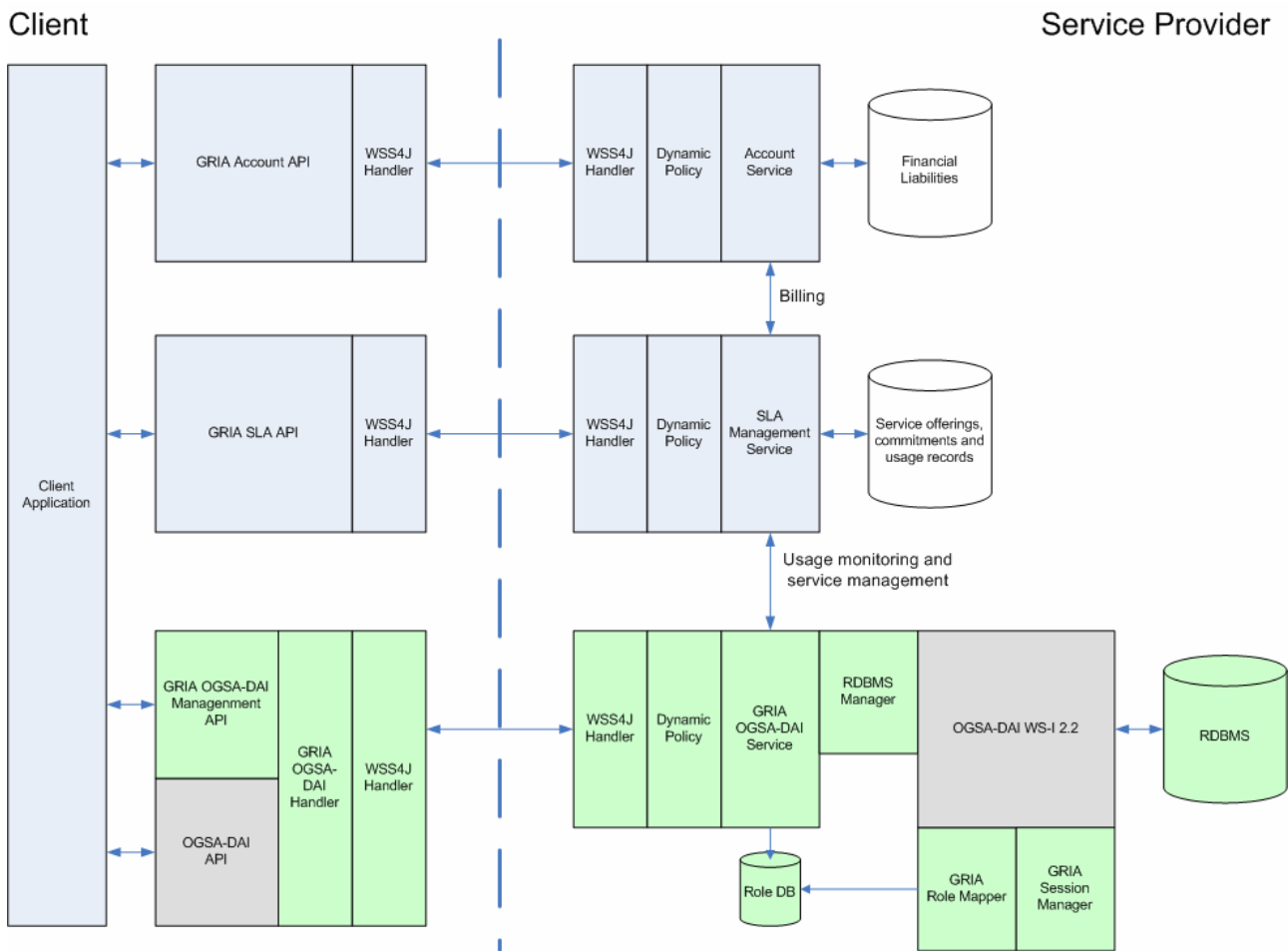


Figure 1: Architecture of the GRIA OGSA-DAI service

In the managed configuration, GRIA OGSA-DAI clients must initially establish a service level agreement that permits them to access OGSA-DAI resources. The SLA defines the service level that the service provider is willing to commit to the client using metrics produced by the OGSA-DAI service (see below). The client provides the SLA when subscribing to a database role or creating a new database at the service provider. The OGSA-DAI service checks with the SLA service that the requested activity is within the terms of the SLA. Once a subscription to a database resource is established the client can then invoke perform operations using the existing client-side OGSA-DAI libraries.

When the service receives a perform request the OGSA-DAI instance is retrieved from the RDBMS manager, the security context is set to the X.509 certificate of the client and perform document passed directly to the retrieved OGSA-DAI instance. OGSA-DAI WS-I then processes the SQL activities within the perform document. The activities call out to the GRIA Role mapper to establish the database username and password to be used for the JDBC connection.

OGSA-DAI does not provide access control on sessions, which is a known issue to the OGSA-DAI developers. This allows one user to establish a session and another to guess the ID and retrieve the data. To overcome this problem the OGSA-DAI service is configured with a GRIA session manager that ensures that only the creator of the session can access the session in subsequent operations using the existing dynamic policy mechanisms provided by GRIA.

2.1.3.2 Resource Model

The GRIA OGSA-DAI service implements a three level resource model, as shown in Figure 2. At the top level is the database (i.e. a database within an RDBMS) which corresponds to a JDBC URI. At the second level are a number of database role resources. Each database role corresponds to an account on the database. This could for instance be an account that has read-only access, an account with write access or one that is restricted to just a few tables—the RDBMS is considered responsible for enforcing these low-level access rights. The third level of resource is the database subscription. Each one of these links a particular GRIA user with a database role. Each database role may have many subscribers.

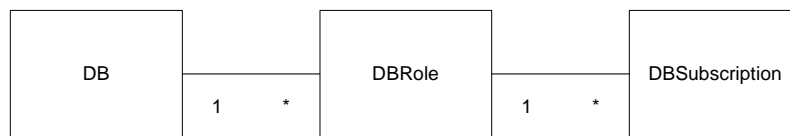


Figure 2 Hierarchical resource model

2.1.3.3 Metrics

For an application to be managed by the GRIA SLA service, it must request and report usage using “metrics”. A metric is a URI that identifies something the SLA service can record the usage of. The SLA service constrains the provisioning and use of the application according to the service’s defined capacity and the client’s SLA, both of which are written in terms of the same metrics. The SLA service is also able to charge for the use of metrics, if required.

The GRIA OGSA-DAI service reports three main metrics, corresponding to the resources discussed in the previous section:

1. <http://www.gria.org/sla/metric/activity/ogsadai/db>
2. <http://www.gria.org/sla/metric/activity/ogsadai/role>
3. <http://www.gria.org/sla/metric/activity/ogsadai/subscription>
4. <http://www.gria.org/sla/metric/resource/subscriber>

For example, if a client requests the creation of a new database, the SLA Management service will:

- check the client’s SLA to see how many databases the service provider is committed to provide;
- check the existing usage records to see how many databases the client is using; and
- if they have not reached the SLA limits the SLA management service will respond permitting the application service to continue.

In addition, a service provider can define custom metrics. This is useful in the situation where the service provider has several databases or database roles and wants to charge for or constrain them differently. For example, the service provider may want to make read-only access to a database free while charging for read/write access. Any database or database can role have custom metric assigned to it. These custom metrics can then be used in the SLAs to differentiate between the different resources. When a user subscribes to a database role that uses a custom metric, the user generates one usage report for the standard subscription metric (<http://www.gria.org/sla/metric/activity/ogsadai/subscription>) and one for the custom metric (e.g.

<http://bioinformaticsdata.com/metric/db/UniProt/v3.2.1>). This is different from subscribing to a role without a custom metric, where the user would only generate a usage report for the standard subscription metric and differentiated pricing or constraints cannot be used.

3 State-of-the-art surveys and the contribution/position of SIMDAT

Between PM12 and PM24 the field of data access and management has seen steady, but gradual progress. The OGSA-DAI package did see two significant new releases, which did roll out important functionality for the WS-I binding: multiple, concurrent requests and sessions are now supported, simplifying the use and increasing the performance of OGSA-DAI for scenarios that have a large number of small, fine-granular accesses. Also, security has been beefed up, with message-level security available for all client/server communication. New activities are provided for the handling of transient (in-memory) data and for a large set of data transformations. OGSA-DAI can now interface with special data transfer mechanisms that use bundled or multiplexed transfers. These improvements are all relevant for SIMDAT since they provide required functionality, add flexibility to write OGSA-DAI extensions (activities) to implement new capabilities, and enable performance improvements. SIMDAT work package 3 partners are cooperating with the OGSA-DAI team on various issues. Specifically it is planned to hold a workshop to further enable partners to use the DDRA services available through GRIA and OGSA-DAI. On the other hand, of course, feedback from the application scenarios of SIMDAT is returned to the OGSA-DAI development team.

In the EGEE project, the LHC Grid data management and replication components have achieved a stable state, indicating that now is the right time to do an in-depth investigation as to the role of these within SIMDAT, and possible ways to integrate them into the set of DDRA components, and with the Integrated Grid Middleware, which is not based on Globus (the ancestor of gLite). SIMDAT technology partners are planning to do this investigation within the upcoming period and cooperate with the application partners to evaluate the applicability of the LHC Grid data technologies within the SIMDAT scenarios.

The P2P file system Igor-FS as developed by the University of Karlsruhe under the auspices of the Pharma activity is now available in its first, reasonably complete implementation. It was designed and implemented to support the publish/subscribe scenario, where a small number of publishers roll out massive amounts of information and update it regularly. Igor-FS provides end-to-end security (all data is encrypted, with cleartext only available to the publisher and the subscribers), opportunity to locally cache data and thus enhance the aggregate bandwidth with guaranteed global consistency, and integrates well into the Linux OS platform. Outside of the Pharma application activity, there are currently no plans to take up Igor-FS, although partners did express interest to evaluate Igor-FS outside of SIMDAT.

Standardization of functionality and interfaces that support well-known Grid use cases has progressed quite well, with recommended standards for Byte-I/O and (soon) distributed access to relational and XML databases (DAIS-R and DAIS-X) finished. The OGSA-DMI working group is looking at how to select appropriate transport protocols between the two endpoints of a data transfer.

4 Work provided to the application activities

By virtue of SIMDAT being organized in matrix like fashion communication and collaboration between technology and application oriented partners is especially direct and effective. Thus, within the application activities a number of special requirements and challenges have arisen that have been addressed through the work of the DDRA work package. Namely, consultancy and/or engineering help has been provided especially to:

- The Meteo activity, for whom a routing mechanism has been developed
- The Auto-1 activity, that has been supported with a custom solution to exploit OGSA-DAI features
- The peer-to-peer file system IGOR-FS developed by Pharma activity partners, which has been evaluated for potential use in the other application activities

Please see below for more detailed information on these topics.

Additionally a lot of effort was put into mutually consulting with the application activities. Generally improved integration with the basic Grid infrastructure and support for the higher level grid technologies, i.e. those building on top of the DDRA services was provided. This enables the application activities to use the provided DDRA services in real world scenarios.

4.1 Meteo activity: VGISC routing mechanism implemented

4.1.1 V-GISC requirements

4.1.1.1 Message routing

V-GISC (Virtual Global Information System Centres) is a distributed information system based on metadata catalogues and data repositories holding the actual observations. The metadata are updated when new observations are entered to the system. In order to increase the performance metadata are synchronized between the catalogue nodes. The detailed synchronization mechanism is explained in a Meteo work package document.

The meteorological organizations participating in the V-GISC are expecting between 20 and 50 catalogue nodes forming the virtual data grid. Moreover it is expected that the nodes are not fully connected via a public network but will be hidden in the DMZ behind firewalls. The current prototype design expects a fully connected network. It is expected that up to four connections will be open per site in order to connect to neighbour nodes.

This meant that a high level message routing between the nodes exposing the lookup functions via web services had to be implemented. Otherwise it would be necessary to update the system routing tables and firewall configurations quite frequently. This is especially true when nodes become temporarily unavailable and alternative message paths have to be used.

Due to the high importance of the data, the logical information network must be self-organizing, self healing and fault tolerant.

The current communication between the nodes is implemented with a document oriented interface. XML documents are passed between service and client either via REST or SOAP messages.

The problem described here is the counterpart to the technologies used in peer-to-peer networks, where a logical topology is layered over a physically fully connected network.

4.1.1.2 Dynamic node insertion

Due to the tight restrictions imposed by administrative constraints, it would be most desirable that the system supports automatic node insertion and re-routing. Because changing system and firewall configurations should be avoided a mechanism based on limited knowledge of the single node is desirable. Therefore the algorithms designed for ad-hoc networks and next-neighbour routing seems to be most suitable.

4.1.2 Routing mechanisms

4.1.2.1 Overview

Routing is used to select paths in a network and sending logically addressed data via intermediary nodes. Routing is usually part of the layer 3 in the OSI model and often performed by specialized hardware (routers). Interfacing with this layer would require administrator access to the network infrastructure, which is not applicable in the V-GISC context.

4.1.2.2 Routing protocols

Static routing, using fixed tables can be used in small networks that do not change frequently. In larger networks, changing frequently, dynamic routing is used. Dynamic routing uses routing tables automatically constructed, based on information carried by protocols, and allows the network to be nearly autonomous in avoiding network failures and blockages.

There are two main classes of dynamic routing protocols.

4.1.2.2.1 Distance vector algorithms¹

Distance vector algorithms use the Bellman-Ford algorithm. When this is used, the link between each node in the network is assigned a number, the *cost*. Nodes will send information from point A to point B via the path that results in the lowest *total cost* (i.e. the sum of the costs of the links between the nodes used).

The algorithm is very simple. When a node first starts, it only knows of its immediate neighbors and the direct cost to them. (This information, the list of destinations, the total cost to each, and the *next hop* to send data to get there, is the routing table, or *distance table*.) Each node, on a regular basis, sends to each neighbor its own current idea of the total cost to get to all the destinations it knows of. The neighboring node(s) examine this information, and compare it to what they already 'know'; anything which represents an improvement on what they already have, they insert in their own routing table. Over time, all the nodes in the network will discover the best next hop for all destinations, and the best total cost.

When one of the nodes involved goes down, those nodes which used it as their next hop for certain destinations discard those entries, and create a new routing table. This is then passed to all adjacent nodes, which then repeat the process. Eventually all the nodes are updated, and will then discover new paths to all the destinations which are still *reachable*.

The distance-vector routing protocol assumes a network connected through several routers, each of which is connected to two or more computer networks. Each network may be connected to one or more routers.

The description below describes a very simple distance-vector routing protocol:

1. In the first stages, the router makes a list of which networks it can reach, and how many *hops* it will cost. In the outset this will be the two or more networks to which this router is connected. The number of hops for these networks will be 1. This table is called a routing table.
2. Periodically (typically every 30 seconds) the routing table is shared with other routers on each of the connected networks via some specified inter-router protocol. These routers will add 1 to every hop-count in the table, as it associates a hop cost of 1 for reaching the router that sent the table. This information is just shared in between physically connected routers ("neighbours"), so routers on other networks are not reached by the new routing tables yet.
3. A new routing table is constructed based on the directly configured network interfaces, as before, with the addition of the new information received from other routers. The hop-count is used as a cost measure for each path. The table also contains a column stating which router offered this hop count, so that the router knows who is next in line for reaching a certain network.

¹ This chapter and the following is copied from the excellent article in en.wikipedia.org

4. Bad routing paths are then purged from the new routing table. If two identical paths to the same network exist, only the one with the smallest hop-count is kept. When the new table has been cleaned up, it may be used to replace the existing routing table used for packet forwarding.
5. The new routing table is then communicated to all neighbours of this router. This way the routing information will spread and eventually all routers know the routing path to each network, which router it shall use to reach this network, and to which router it shall route next.

Distance-vector routing protocols are simple and efficient in small networks, and require little, if any management. However, they do not scale well, and have poor convergence properties, which has led to the development of more complex but more scalable link-state routing protocols for use in large networks.

One of the protocols is the “Ad-hoc On-demand Distance Vector” (AODV).

4.1.2.2.2 Link state algorithms

When link state algorithms are used, the fundamental data each node uses is a map of the network, in the form of a graph. To produce this, each node floods the entire network with information about what other nodes it is connected to, and each node then independently assembles this information into a map. Using this map, each router then independently determines the best route from it to every other node.

The algorithm used to do this, Dijkstra's algorithm, does this by building another data structure, a tree, with the node itself as the root, and containing every other node in the network. It starts with a tree containing only itself. Then, one at a time, from the set of nodes which have not yet been added to the tree, it adds the node which has the lowest cost to reach an adjacent node which is already in the tree. This continues until every node has been added to the tree.

This tree is then used to construct the routing table, giving the best next hop, etc, to get from the node itself to any other node in the network.'

The primary advantage of link-state routing is that it reacts more quickly, and in a bounded amount of time, to connectivity changes. The disadvantage is that the communication to discover network members occurs continuously. Also, calculation and memory burdens are continuing, and may be too heavy for small computers. The program is fairly large and complex.

One of the protocols also used in ad-hoc networks is the “Optimized Link State Routing Protocol” (OLSR).

4.1.3 V-GISC specific implementation

4.1.3.1 Protocol choice

Due to the requirements of the V-GISC system, the service network is similar to an ad-hoc network. It is natural to select one of the specialized protocols used in such networks for V-GISC. The network size for V-GISC is small to medium (20-50) so that a distance vector routing protocol can be used without disadvantages. For V-GISC it is desirable to base the routing on a “next-hop” base in order to avoid complete routing paths in the message headers. These paths may be difficult when the messages are signed for security reasons. Having only the information for the next hop also nicely integrates with the WS-Addressing mechanism. From the available protocols “Ad-hoc On-demand Distance Vector” looks ideal.

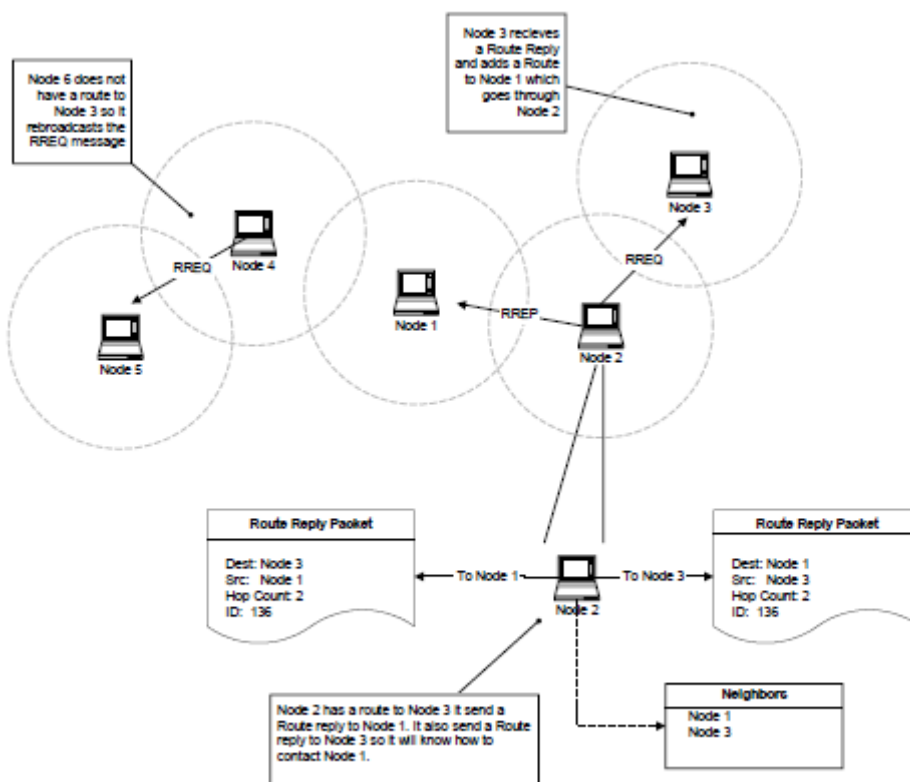
The following short description summarizes the protocol:

The **Ad-hoc On-demand Distance Vector (AODV)** routing algorithm is an algorithm for routing data across wireless mesh networks. It is capable of both unicast and multicast routing. It is a reactive routing protocol, meaning that it establishes a route to a destination only on demand. In contrast, the most common routing protocols of the Internet are proactive, meaning they find routing paths independently of the usage of the paths. AODV is, as the name indicates, a distance-vector protocol.

In AODV, the network is silent until a connection is needed. At that point the network node that needs a connection broadcasts a request for connection. Other AODV nodes forward this message, and record the node that they heard it from, creating an explosion of temporary routes back to the needy node. When a node receives such a message and already has a route to the desired node, it sends a message backwards through a temporary route to the requesting node. The needy node then begins using the route that has the least number of hops through other nodes. Unused entries in the routing tables are recycled after a time.

When a link fails, a routing error is passed back to a transmitting node, and the process repeats.

Much of the complexity of the protocol is to lower the number of messages to conserve the capacity of the network. For example, each request for a route has a sequence number. Nodes use this sequence number so that they do not repeat route requests that they have already passed on. Another such feature is that the route requests have a "time to live" number that limits how many times they can be retransmitted. Another such feature is that if a route request fails, another route request may not be sent until twice as much time has passed as the timeout of the previous route request.



The advantage of AODV is that it creates no extra traffic for communication along existing links. Also, distance vector routing is simple, and doesn't require much memory or calculation. However

AODV requires more time to establish a connection, and the initial communication to establish a route is heavier than some other approaches.

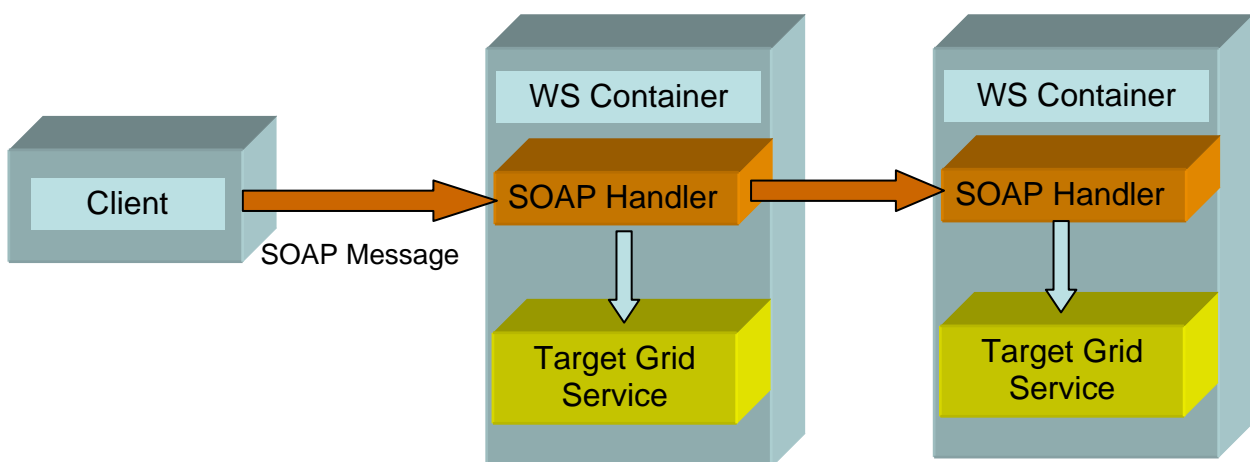
In order to be used at the service layer in V-GISC some adaptations have to be made:

- Usually the routing messages are transported via UDP. This has to be replaced by calling a routing service. This assures a firewall friendly protocol, especially when a SOAP/http binding is used.
- The interface of the routing service has to be defined
- The message format must be defined

4.1.3.2 Message forwarding

This mechanism is strongly dependant on the protocol used.

If the V-GISC services are invoked as grid/web services using a SOAP based mechanism, we can include the routing information into the WS-Addressing header. Most SOAP stacks allow the definition of handlers that intercept the message processing. A special handler can be integrated, that analyzes the WS-Addressing information and forwards the message to the next neighbour instead of processing it. The handler has to contact the routing service for next-hop information.



With pure REST style invocation, we may not have the handler mechanism available. This has to be investigated in more detail. One solution may be, to have a general REST dispatcher that implements the same functionality as the SOAP handler.

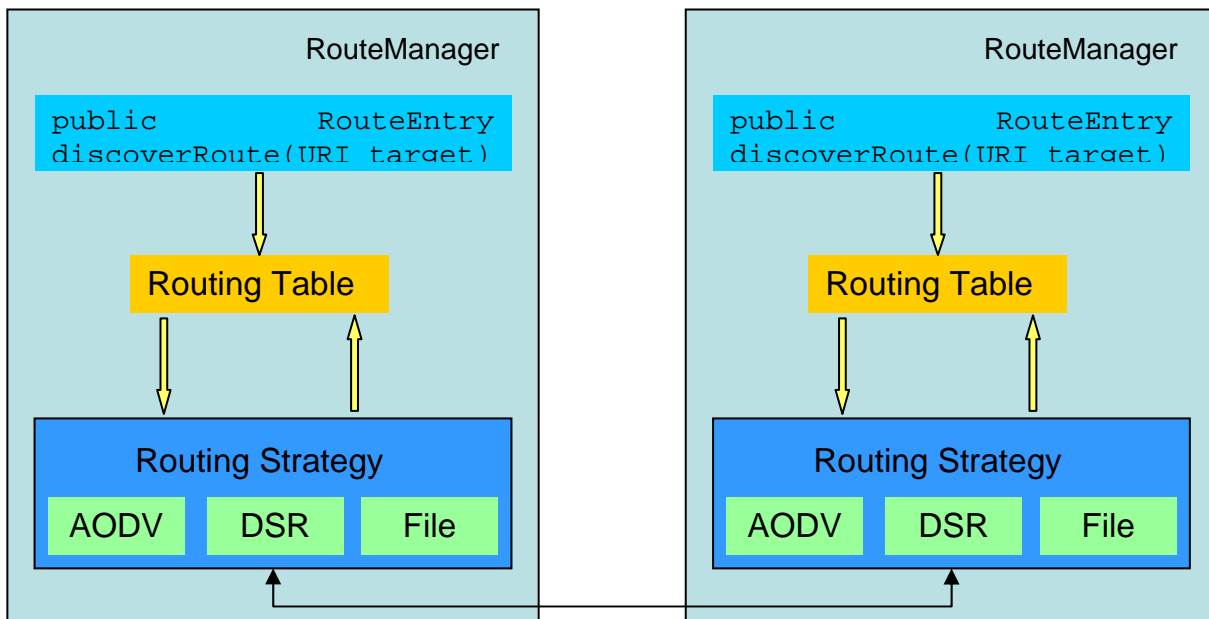
4.1.3.3 RoutingManager

For handling the routing messages a service is needed that builds and updates the routing information (analogous to the daemon process normally used). In addition the routing service must expose an interface for acquiring routing information for the actual V-GISC service calls.

After initialization the routing service reads a configuration file containing the available connections (specified as an URI). It uses this information to obtain routing information from the neighbours.

The RoutingManager keeps a next-neighbour routing table that can be updated by pluggable modules implementing different routing strategies. These may include for example:

- A file based static table (for testing and development)
- An AODV based implementation
- Other routing protocols



The interface to the `RoutingManager` currently consists of a single method.

```
public RoutingEntry discoverRoute(URI targetService) throws Exception;
```

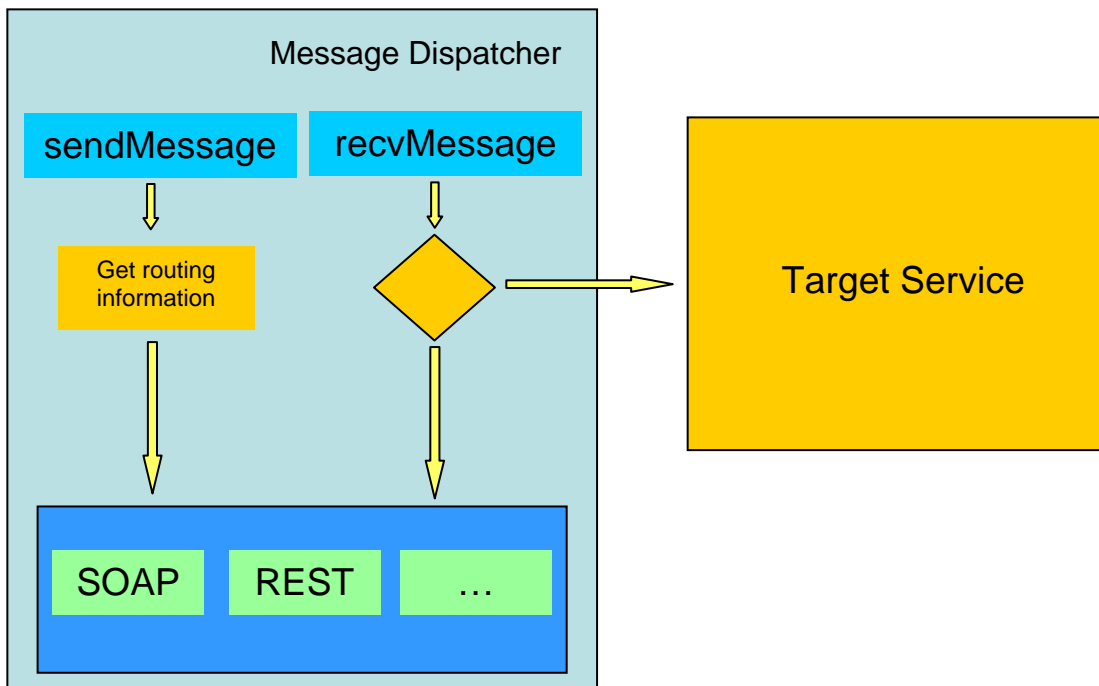
This method is called by the `MessageDispatcher` described earlier in order to discover a route.

The exact interface in the `RoutingStrategy` layer has to be investigated and will be published later.

4.1.3.4 *MessageDispatcher*

The message dispatcher is responsible for analyzing the header of an incoming message and either forwards it to the next hop in the message pass or to invoke the targeted service if it is locally available.

The module is implemented in a way, that it can be integrated either into the handler chain of a SOAP engine (Axis, Xfire) or can be invoked by a separated message processing module when using pure REST services.



Currently the interface consists of two methods:

```
public void sendMessage(String message) throws Exception;
```

```
public void receiveMessage(String message) throws Exception;
```

The `sendMessage()` method is invoked by a local or remote service in order to send a message. The routine will call the `RoutingManager` to discover a route and forward the message.

The `receiveMessage()` method consumes an incoming message and analyzed the WS-Addressing header. If the service is local to this `MessageDispatcher` instance it gets invoked. Otherwise the routine calls the `RoutingManager` in order to discover the next hop and forwards the message.

It is intended, that the dispatcher does not modify the header in order to allow digital signing, but this has to be investigated further.

4.1.3.5 Protocol Binding

In order to support both, SOAP based service invocations and REST based invocations, a document oriented interface was proposed, where the routing messages are represented as XML fragments. Java RMI will not be supported because we cannot associate routing information with RMI calls.

4.1.3.6 Message Header Modifications in Grid-Service Calls

In a SOAP based environment, the target service URI is included in the header anyway and can be analyzed by the service. Essentially WS-Addressing uses the `<wsa:To>` and `<wsa:Action>` elements to address services. The `<wsa:To>` element may contain a logical address that can be translated by the routing mechanisms. In this context WS-Referral may be an option.

A SOAP header example:

```
<s:Envelope xmlns:s="..." xmlns:wsa="...">
  <s:Header>
    <wsa:Action>http://skonnard.com/SubmitClaim</wsa:Action>
    <wsa:To>http://skonnard.com/Claims/Submit.asmx</wsa:To>
    <wsa:From>
      <wsa:Address>http://skonnard.com/main/sub.asmx</wsa:Address>
      <wsa:ReferenceProperties>
        <c:PatientProfile>123456</c:PatientProfile>
        <c:CarrierID>987654</c:CarrierID>
      </wsa:ReferenceProperties>
    </wsa:From>
    <wsa:ReplyTo>
      <wsa:Address>http://skonnard.com/resp/resp.asmx</wsa:Address>
      <wsa:ReferenceProperties>
        <c:PatientProfile>123456</c:PatientProfile>
        <c:CarrierID>987654</c:CarrierID>
      </wsa:ReferenceProperties>
    </wsa:ReplyTo>
    <wsa:FaultTo>
      <wsa:Address>http://skonnard.com/fault/err.asmx</wsa:Address>
      <wsa:ReferenceProperties>
        <c:PatientProfile>123456</c:PatientProfile>
        <c:CarrierID>987654</c:CarrierID>
      </wsa:ReferenceProperties>
    </wsa:FaultTo>
  </s:Header>
  <s:Body xmlns:c="http://example.org/claims">
    <c:SubmitClaim> ... </c:SubmitClaim>
  </s:Body>
</s:Envelope>
```

If V-GISC messages are sent via REST, we may include the <wsa:To> and <wsa: Action> fields in the V-GISC message in order to implement a similar mechanism.

4.1.3.7 Security

In the first phase authenticated https connections are in place for transport layer security. If SOAP based web services are used, the standard WS-Security or WS-SecureConversation mechanisms can be used for message level security.

4.1.4 Dynamic node insertion

Dynamic node insertion consists of two parts:

- Configuring the next neighbour lists of the new node and the neighbours (possibly this requires updating the firewalls)
- Initializing the meta-data catalogue by performing a full update from a parent (this is part of another document)

The first task can either be done manually or the already existing neighbour routing services must expose a method that the joining node can call in order to register. Automatic firewall configuration is not addressed here.

4.1.5 References

- [1] <http://en.wikipedia.org/wiki/Routing>
- [2] http://en.wikipedia.org/wiki/Link-state_routing_protocol
- [3] http://en.wikipedia.org/wiki/Distance-vector_routing_protocol
- [4] http://en.wikipedia.org/wiki/Optimized_link_state_routing_protocol
- [5] <http://en.wikipedia.org/wiki/AODV>

4.2 Auto-1: OGSA-DAI service for TecManager

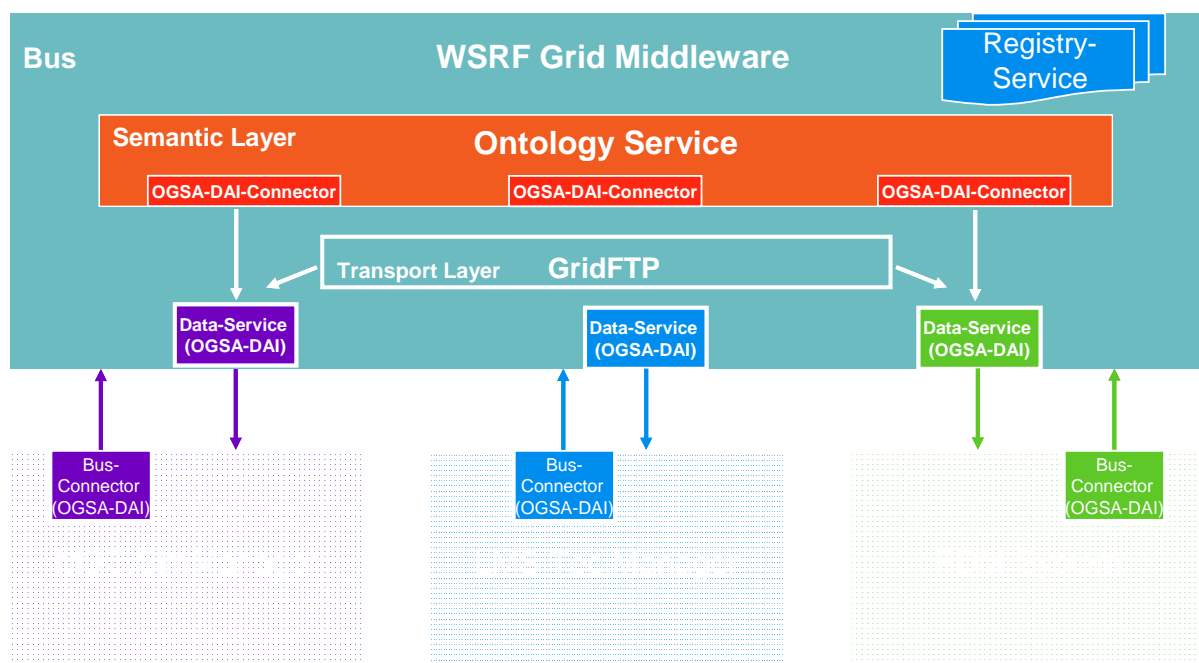
4.2.1 Auto-1 PM24 prototype

The Auto-1 prototype, referred to as Audi SAMD (SIMDAT AutoMotive Demonstrator) prototype, is intended to demonstrate the implementation of Grid technologies for an OEM federation scenario. One of use case scenarios aims at the integration of the different disciplines like CAE and CAT. It should be possible for CAE engineers to browse and use data from a physical test. The initial implementation provided for PM12 with rudimentary coupling components was enriched during last twelve months with essential functionality regarding to the caching and replication functionality released for PM18. This scenario was extended for PM24 by several technological advances, like OGSA-DAI based ontology and data services, and by use of real Problem Solving Environments (PSE's) MSC.SimManager and LMS Tec.Manager. At the same time the focus for the use of PSE's was on the utilization of more realistic data models and dealing with them by the semantic mediation.

Distributed data read access is now available via loose integration implemented through OGSA-DAI. The client and server-side connectivity to the grid middleware is established for PSEs, the data transfer service is set up through OGSA-DAI. The PSE's enhancement towards OGSA-DAI has been done in the first step for LMS Tec.Manager. The enhancement for MSC SimManager will follow until PM30. The transfer of mass data between MSC SimManager and LMS Tec.Manager based on OGSA-DAI is implemented.

4.2.2 Auto-1 architecture

The Auto deliverables describe the main aspects of architecture in depth. With this section here only a short summary of those aspects is presented. Additionally the components available at PM24 are outlined. The following picture illustrates the view on the proposed architecture by exposing the components available.



The bus builds the spine of the architecture according to the SOA-oriented view. It is based on grid compliant middleware. The following points summarize responsibilities which the bus has:

1. Providing communication between the clients and services
2. Registration of all services
3. Logging & Failure notification
4. Handling of Authorisation/Authentication

For the PM24 prototype Tomcat Axis and Globus Toolkit were used as middleware. Primarily the focus of the PM24 prototype was on the first point. The remaining responsibilities of the middleware will be added more and more in the coming phases. The systems that participate in the communication via middleware should define firstly stubs for the services they need to be able to send the requests to them. Secondly they can provide one or more services for other systems, e.g. data services if they want to share their data. For all provided services, their properties can be exposed. If a system provides a data service, then the object model of the underlying data should be offered as a property which is needed for semantic integration.

In the PM24 prototype the communication between two systems was established: MSC.SimManager with the role of a client providing appropriate stubs for the middleware, LMS Tec.Manager with the role of a data service provider providing data access via OGSA-DAI and exposing its own data model described in OWL, RDF and F-Logic.

4.2.3 Some of the Auto-1 services

Ontology service

For PM24 prototype an ontology service has been extended to fit into the Grid Middleware by utilizing the OGSA-DAI framework. Besides, it has been enriched with a stub for the data service of LMS Tec.Manager. It has been implemented as a built-in for Ontobroker. As proposed in the automotive requirements deliverable this stub has utilized the query mechanism specific to the target system, in PM24 prototype it is LMS Tec.Manager. In the upcoming phase the same procedure will be applied for MSC SimManager as well.

Data Services

For the PM24 prototype LMS Tec.Manager has been used as data provider according to the use case description. For this purpose an OGSA-DAI complaint data service has been created. This service has introduced a transparent query mechanism for LMS Tec.Manager for use in a grid environment abstracting from LMS Tec.Manager specific functionality. This mechanism provides an interface for metadata and mass data queries, which can be used as template for creation of the OGSA-DAI data services for other systems, like MSC SimManager. It is planned to provide the first implementation of MSC SimManager data service by PM30.

4.2.4 Auto 1 data retrieval

Two data retrieval methods have been introduced proposing a way for communication a client system can choose. Depending on the demand for semantic mediation a client system can decide either to utilize the ontology service for query processing or directly ask a data service of the target system. For the PM24 prototype the assumption has been made that two systems MSC.SimManager and LMS Tec.Manager use different data models. This has caused a need for the semantic mediation. Therefore the method for data retrieval via Ontology service has been chosen.

4.2.5 OGSA-DAI activity for Auto 1

The use of OGSA-DAI provided some challenges to the Auto-1 partners with regard to its deployment outside of the widely used servlet container like Apache Tomcat. The current tools for creation of application specific OGSA-DAI activities require quite some manual steps from the unexperienced user. This makes the process of creation of a new activity cumbersome and error-prone. Here the DDRA partners provided a significant amount of support to the process of activity creation.

LMS TecManager is an object wrapper for data bases. A wrapping service was needed to provide the Ontoprise tool Ontobroker with access to that data. For that a specifically designed and fitted OGSA-DAI activity has been written.

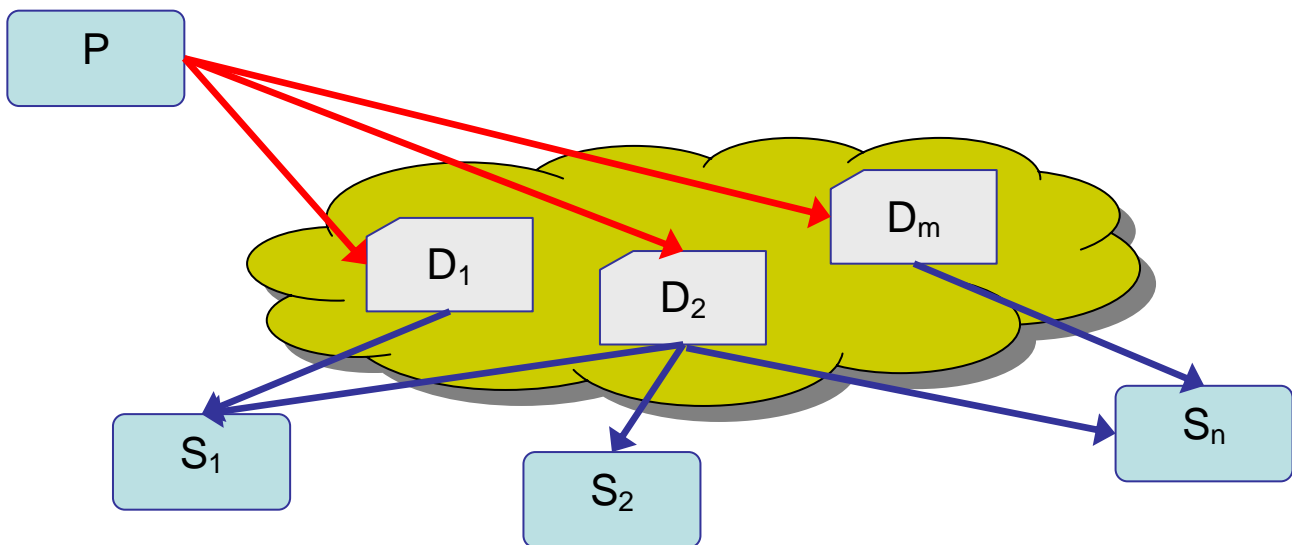
4.3 IGOR-FS evaluation

This chapter aims to give an overview of the file system IGOR-FS as it has been developed by SIMDAT partner University of Karlsruhe. In some detail we concisely present the IGOR-FS design objectives and capabilities, clarify development schedule and development options, and as well aim to foster discussion on how to use IGOR-FS in SIMDAT application activities. Feedback from the other, i.e. non-pharma, application activities has been gathered on where they think IGOR-FS might/should be used and what gaps need to be closed as of now.

4.4 IGOR-FS Design Scenario

IGOR-FS is designed as follows:

- A Publisher P makes large collection of datasets D_1 to D_m available. Here data is updated regularly. Updates tend to be small compared to complete data volume.
- Subscribers S_1 to S_n access subset of data; they need to use the latest available versions while they are assumed to be working on a set small compared to complete data volume.
- Data must be protected against unauthorized access.



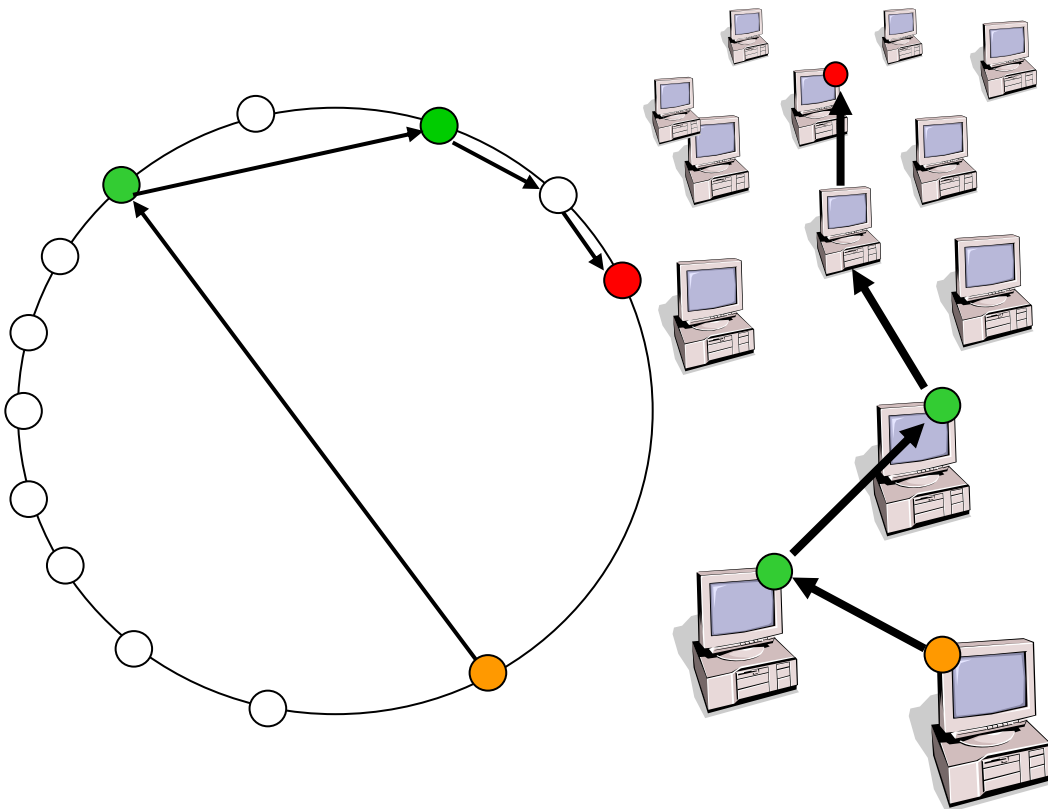
IGOR-FS was designed following a number of objectives:

- Be application-neutral and transparent
- Reduce data volumes to be transferred or stored (for each subscriber: transfer “just the changes”, store recurring data only once)
- Accommodate scale-up (system scales with number of subscribers)
- End-to-end protection of data (keep data secure and allow authentication)
- Support commercial subscription models

4.5 Peer-to-Peer

IGOR-FS is a peer-to-peer file system. That means:

- Distribute blocks across publisher(s) and subscribers
- Impose ring structure with shortcuts for block lookup
- Intermediate nodes transport encrypted contents, can cache blocks locally
- Consistency is easy, file updates create new instances of all affected blocks and the directory chain
- Updates are made visible by distribution of new root block id and key



As a file system IGOR-FS supports:

- Posix I/O functionality and interfaces
- Store and transmit all data as encrypted blocks (use hash function as encryption key, allow authentication of blocks)
- Blocks are identified with their hash value
- Directories contain references and keys for file blocks
- File blocks contain data, or list of references to other file blocks (with keys)
- Knowledge of root block (and encryption key) gives access to all files

4.6 Advantages

With IGOR-FS the cache consistency problem is solved, i.e. caching can be handled locally while there is a speed up for local and remote access. Data updates create new blocks.

IGOR-FS offers ideal system scaling, that is more subscribers means more cached data.

Only data updates have to be transferred. Working can be cached at each subscriber.

IGOR-FS has built-in end-to-end security. The intermediate nodes only see encrypted data.

4.7 Properties

With IGOR-FS there is a need for a secure key distribution scheme. (There is a study how to do that with GRIA!)

With IGOR-FS there is a different approach to access control. The access cannot be limited to parts of filesystem easily; access cannot be revoked once given. As an alternative access can be limited access to old data by not distributing keys for updates.

Coordinated updates are required: Every change in a file results in a change to the root block, thereby makes key distribution necessary.

Performance implications of encryption are unclear. Anyway they are limited to the endpoints, i.e. the publisher and the subscriber.

4.8 Status

Current prototype exploits FUSE Linux technology. IGOR-FS is a mount and use file system with full transparency to application. It is a single writer instance and has no crossing ability for firewalls/NAT routers.

PM 36 prototype preliminary planning is for multiple writers, timely usage of hot data, retrieval of old versions. There will be distinguishing between original and derived data to improve caching. It is planned to enable access to backup versions. There are many other possible extensions which include the support for crossing of firewalls (IGOR-FS gateway), the switch to asymmetric encryption to allow per-user access control, control access to subtrees of the file system, and the integration into MS Windows ...

4.9 Conclusion

This overview of the file system IGOR-FS has been delivered to SIMDAT technology and application activity partners of SIMDAT. Feedback from non-pharma, application activities has been gathered on where they think IGOR-FS might/should be used and what gaps need to be closed as of now. A number of proposals have been put forward to University of Karlsruhe. It seems that as-of-now the technology has some promise and generates vague interest, but only might be applicable to the use cases only later on.

5 Specific and modular requirements documentation

This chapter contains a brief update of the requirements as they have been put forward by the application activities of SIMDAT and assembled in the consolidated requirements report D3.1.1. For the implementation and test plans for PM30 please refer to the respective documents or chapter.

Below we list detailed descriptions of the requirements as they have been agreed upon in conjunction with the respective application activities.

5.1 Automotive requirements update PM30 & beyond

PSE's enhancement towards OGSA-DAI

Name	OGSA-DAI Services for MSC SimManager		
Req. Id	R-AU-PM30-1		
Application Activity	Automotive		
Prototype(s)	Auto-1 PM30		
Date Created	2006-03-22	Priority	High
Created By	Walter Wirch	Technology component	Distributed Data Access, Grid Infrastructure
First Implementation Date	End of February 2007	SIMDAT module targeted	WP 9.2 Tasks 1, 4, 6, 8, 9
Description	<p>The OGSA-DAI enabled versions of LMS Tec.Manager and MSC SimManager will offer data access via a uniform protocol OGSA-DAI by employing specific activities for the access to their physical storages. The OGSA-DAI service of LMS Tec.Manager will be accomplished for PM24 for showcasing the CAE/CAT integration, the OGSA-DAI service of SimManager will be finished at latest for PM30, to be integrated in a common Automotive prototype for PM30.</p>		
Relation to prototype	<ul style="list-style-type: none"> Auto-1 PM24 		
Requested functionality	<ul style="list-style-type: none"> Data access to CAT and later to the CAE data sources of PSE's 		
Validation			

<ul style="list-style-type: none"> Availability of the service and the data access functions of in the services 			
Assumptions			
<ul style="list-style-type: none"> 			

Query translation for ontology services

Name	EL2SPARQL Translation for MSC SimManager		
Req. Id	R-AU-PM42 (originally R-AU-PM24/30-2)		
Application Activity	Automotive		
Prototype(s)	Auto-1 PM42		
Date Created	2006-03-22	Priority	High
Created By	Walter Wirch	Technology component	Distributed Data Access, Grid Infrastructure, Ontologies
First Implementation Date	PM42	SIMDAT module targeted	WP 9.2 Tasks 1,2,4, 6, 8, 9
Description	<p>Through the translation of MSC SimManager's internal query language to SPARQL MSC SimManager will be enabled to query the Ontology Services by means of SPARQL, which is now wide accepted language and has a potential for emerging to the standards later in the semantic web community. In the Ontology Service the sent query will be processed and transformed in the representation which is expected by the target system.</p>		
Relation to prototype			
<ul style="list-style-type: none"> Auto-1 PM24 			
Requested functionality			
<ul style="list-style-type: none"> Translation of MSC SimManager's internal query language to the SPRAQL, agreed language of ontology services in SIMDAT project. 			
Validation			
<ul style="list-style-type: none"> Capability of SimManager to send a SPARQL query to the ontology service. 			
Assumptions			
<ul style="list-style-type: none"> 			

Name	Access to PSE via the GRIA OGSA-DAI service
------	---

Req. Id	AUTO-DDRA-ESI-1		
Application Activity	Automotive		
Prototype(s)	Automotive prototype for PM 30		
Date Created	2006-09-19	Priority	Medium
Created By	ESI	Technology component	GRIA OGSA DAI service
Status <i>finished, first version to be delivered by., obsolete, ...</i>	<i>Submitted</i>		
First Implementation Date		SIMDAT module targeted	WP 9.3
Description	<p>At the moment, the GRIA OGSA-DAI service can be used to access databases with a standard SQL interface through a JDBC connection. Could it be extended to access a PSE's database (like Visual Composer's database) ? PSE's stored data are usually not exploitable directly : there is a an intermediate "business logic" layer on top of the database that interprets these data and present them to the user. The access to a PSE's database is thus done through a native API.</p> <p>If the GRIA OGSA DAI service could be interfaced with PSEs, then we could use its provided "subscription to roles" mechanism as a possible solution to the issue of user rights management when accessing PSEs databases in a grid context</p>		
Relation to prototype			
	<ul style="list-style-type: none"> • 		
Requested functionality			
	<ul style="list-style-type: none"> • Extend the GRIA OGSA-DAI service to all types of data resources 		
Validation			
	<ul style="list-style-type: none"> • PM 30 Automotive prototypes 		
<i>Optional Remarks, here:</i>			
Assumptions			
	<ul style="list-style-type: none"> • 		

5.2 Aerospace requirements update PM30 & beyond

5.2.1 Functional Requirements

Requirement	Description	Priority	Implemented	Remaining
WS-I compliant services	Standards compliant service interface	High	✓	
Identity management services	CA, Revocation Service	High	Partial	Revocation service
Authentication service	Ability to authenticate users (X509 based) Authenticated transactions Ability to handle federated identity for exploitation phase	High	Partial	Federated identity
Authorisation Service	Policy driven access control to resources Dynamic policy management for exploitation phase	High	Partial	Dynamic policy management
Access to legacy applications	Service container for legacy applications including submission access to compute service	High	✓	
Data transfer / access service	Transfer/access of flat files Database access including schema publishing in exploitation phase Policy based access control	High	✓	
Single interface to compute service	Ability to access compute services with different scheduling requirements through single interface Reservation of compute resources Sandboxing runtime in compute service including ability to specify sandbox environment	Medium	Partial	Reservation and sandboxing
Resource discovery	Ability to discover alternate services due to service failure or	Low	×	

	unavailability			
--	----------------	--	--	--

5.2.2 Performance Requirements

Requirement	Description	Priority
Fast data transfer service	Timely transfer of data of the order of Gigabytes comparable to ftp	High
Scalability	The infrastructure needs to be scalable to enable exploitation by at least 100 simultaneous users	High
Manageability	The infrastructure and policy specification management overhead should not grow exponentially with scale	High

5.3 Pharmaceutical requirements update PM30 & beyond

Due to the special situation of the pharma activity there have not been forward concrete requirements to the DDRA workpackage. Nonetheless the workpackage partners continue to consult with the respective application partners and monitor the scenarios for potentially surfacing challenges to the DDRA service technologies.

5.4 Meteorology requirements update PM30 & beyond

The DDRA workpackage partners have closely cooperated with the meteo activity partners to develop a custom solution for the problems at hand in the respective scenario. Of course this cooperation will continue throughout the further project. Below a status update of the meteo requirements is given.

5.4.1 Requirements Check List

Requirement	Description	Priority	Implemented	Remaining
Provide discovery/requesting functionalities	Interfaces for discovering and requesting meteorological datasets	High	Partial	Enhance discovery services (full-text search, ...)
Implement catalogue synchronization	Provide synchronization of the metadata catalogue amongst	High	✓	

	all partners			
Implement data replication/cache	Provide replication of real-time data to ensure high availability	High	X	
Provide a universal interface for meteorological databases	Interfaces to support the integration of meteorological databases in a consistent and uniform way.	High	✓	
Provide virtual database administration tools	Develop user interfaces to ease catalogue maintenance and administration	High	X	

5.4.2 Requirements for PM42 as of now

Name	Data Replication/Caching		
Req. Id	METEO-P3.4		
Application Activity	Meteo		
Prototype(s)	Meteo Prototype for PM42		
Date Created	2006-09-10	Priority	High
Created By	ECMWF	Technology component	DDA
Status	Evaluation		
First Implementation Date	Phase III	SIMDAT module targeted	WP 12.3
Description			
To ensure high availability of some of the data, meteorological observations will be replicated/cached between the National Centres.			

Relation to prototype			
Meteo PM24			
Requested functionality			
Data replication for meteorological real-time data (such as observations)			
Validation			
Critical data available from at least two Nodes			
Assumptions			
The synchronization engine developed might be used for replicating the data			

Name	Data delivery of very large datasets.		
Req. Id	METEO-P3.5		
Application Activity	Meteo		
Prototype(s)	Meteo Prototype for PM42		
Date Created	2006-09-10	Priority	High
Created By	ECMWF	Technology component	DDA
Status	Evaluation		
First Implementation Date	Phase III	SIMDAT module targeted	WP 12.3
Description			
Data requests will generate in cases, very large datasets (several GBytes). Alternative delivery mechanisms such as GridFTP, Peer to Peer transport mechanisms or BBFTP will be evaluated to deliver these datasets using the subscription service or the web portal.			
Relation to prototype			
Meteo PM24			
Requested functionality			
Delivery of large datasets			
Validation			

Transfer of a large data files to the client application.

Assumptions

GridFTP, BBFTP or Peer to Peer technologies will be evaluated.

6 Roadmap and targets for PM42

6.1 Targets for DDRA as of Annex 1

As one of the common denominators of the SIMDAT application activities, access to data that resides on a remote Grid resource or that is distributed across many Grid resources is required. On top of the access functionality, higher-level services like automatic handling of meta and provenance data, synchronization between repositories, coordinated distributed queries, and support for semantic interoperability based on ontologies etc. will be built in the application activities and in the technology components. The access services to be developed in this component will over the full duration of SIMDAT offer this functionality:

- Provide transparent (across different underlying database implementations) access to (relational) databases or flat file repositories residing on local or remote Grid resources, and support a full set of query and database access functions. Distributed read access from n readers will be supported, with distributed write and/or updates being developed in close cooperation with the application activities that need them (in particular, the automotive applications).
- Support attaching metadata and annotations to records that are sufficient for the intended use of provenance data, ontologies and knowledge discover/mining techniques in the respective technology component and application activities.
- Offer middleware-level functionality for coordinated distributed queries spanning a number of database instances across multiple Grid resources, and interface with workflow technology components to facilitate the creation and execution of knowledge mining and management workflows.
- Allow transparent access to files that are indexed using a database, plus directory services for same. There is no need for presenting a consistent “global” file system view across multiple reference databases.
- Introduce access control at the right level of abstraction and granularity (potentially down to the level of records) that are setup through management/administration services. This will be used by VO components to define access and inspection rights for individuals or organizations according to their role in a VO (need to know), and for applications according to their requirements (e.g. a crash simulation needs to be able to access data that may be otherwise unreadable by a VO participant). From experience in the automotive activity, it has become very clear that access control at the level of objects in a CAD/CAE model is more effective than at the level of records, where much of the semantic information is lost.
- Support the introduction of data caching and data replication mechanisms that will improve performance and robustness. Data caching can/should be done in a way abstract to the end-user, governed by local policies. Data replication will need to be driven by global policies that are visible to at least the Grid administrator, but possibly also to the end-user. It is expected that techniques and components from the CERN/EGEE gLite will be viable.

- To lay the groundwork for performance assessment and improvements, identify meaningful benchmarks of the DDRA components and produce performance data to be discussed with the DDRA users.

The first project phase focused on the distributed read and initial update functionality from the first topic, the basic distributed query functionality from the second, the third and fourth topic, and initial support for the fifth topic project. To safeguard the schedule for the first demonstrators, the first prototype was based on an external component – the OGSA–DAI database access package developed by the University of Edinburgh and IBM UK. As part of this prototype, bindings of OGSA–DAI for the chosen Grid infrastructure (GRIA) have been included. Due to time pressure, most of the demonstrators actually used a simpler way of accessing remote databases: a Web Service interface that transmits SQL queries and sends back the result data. After the PM12 milestone, this workpackage cooperated with the application activities to define the best use of the DDRA services based on OGSA–DAI, and to leverage the capabilities of the DDRA services in their applications/demonstrators.

From the experiences up to PM24, it has become clear that information security in industrial scenarios needs to be given prime attention. The academic model of virtualizing databases at the level of SQL queries, raises concerns about protecting data from unauthorized access: each partner is only allowed access to certain parts of a model, which are stored in a highly complex subset of rows/columns. To validate the access rights at the level of raw SQL queries is difficult, and the OGSA–DAI components have to be trusted to not open any backdoors to protected information. One possible solution is to virtualize at a higher level (parts of model), and this WP will work with the application activities (in particular, the automotive sector) to define a workable solution and include the necessary support into then DDRA services (Task 1, Task 2, Task 3).

The Regarding functionality of the fourth DDRA services release, the focal points will be (Task 4, Task 1, Task 3):

- Provide and improve support for the emerging requirements of the “higher–level” technology activities, in particular Workflow (for distributed queries), Ontologies and Knowledge Services.
- Extend the functional envelope to accommodate effective metadata management, and support distributed coherence and synchronization protocols (driven by the Meteorology activity).
- Investigate performance bottlenecks within the DDRA component implementations, discuss their relevance with application activities and drill down on root causes of the significant bottlenecks. Evaluate ways to correct or work around those,
- Work with CERN and the EGEE project to evaluate the data management and replication functionality built into the gLite middleware, and work with application activities to reach agreement on which of the capabilities/components should be integrated into the DDRA release.
- Cooperate with the VO technical activity and the application activity, to accommodate the requirements on AAA, security and trust management within the DDRA components, preferably at the level of OGSA-DAI interfaces. Since the AAA parts of this has been addressed in the Interoperability phase, we will focus on the management of access privileges and the creation of audit trails.

- Assure compliance to and interoperability with emerging standards, such as in the data area of the OGF (DAIS, OGSA-D, GridFTP, OREP, others ...).

Of course, this WP will continue to liaise very closely with WP 2.3 to ensure seamless integration with the Integrated Grid Infrastructure prototypes, and to leverage the capabilities of this infrastructure. Conversely, feedback on the BGI releases will be given, and input/requirements for the ongoing architecture and development work will be given.

After transitioning from initial, possibly proprietary, data access platforms, the application activities will increasingly rely on the basic DDRA services being made available in this component. The work in these areas will profit from having a stable platform available, and from getting support during installation and operation. Emphasis will be put on portability and interoperability (e.g. by relying on de-facto standard interfaces), thereby protect the investments by developers and users throughout SIMDAT. Wherever possible, externally developed components are being re-used to add capabilities.

6.1.1 M1 (12 months) revisited:

Starting from a detailed analysis of the state-of-the-art and a consolidated set of requirements produced in close cooperation with the application sectors, a gap analysis concerning the distributed data repository access (DDRA) components has been produced. From that, a roadmap will have been agreed with the application activities, covering the design of the initial DDRA components (mainly taken from the OGSA-DAI project), and the enhancements to be integrated to cover application requirements throughout SIMDAT. The first version of the DDRA was available, focusing on distributed read access, and integration with the basic Grid infrastructure was in an advanced stage of development. This first version concentrated on distributed read access, and did not address requirements for distributed updates and performance.

6.1.2 M2 (18 months) revisited:

The first and second versions of the DDRA components were available for use by the partners, with the first version already operational. The second version extended the functional envelope to cover general remote/distributed data access, and incorporated requirements emerging from the first set of demonstrators. Assistance (including training events and tutorials) has been given to partners to include the DDRA services in their demonstrator stacks, and to build higher-level services and applications on top of DDRA services.

6.1.3 M3 (24 months):

From experiences with the first two versions, and in particular drawing on an evaluation of the demonstrators, an in-depth evaluation of the applicability and functionality of the first two DDRA releases is available in that it was found that the application activities adapted the available technology not as fast as expected, and therefore an updated roadmap for the will have to be constructed accordingly later on. From a new state-of-the-art analysis (covering in particular the results from fellow FP6 projects), use of suitable outside components will be integrated into the roadmap, wherever advantageous. The third release of the DDRA components has become available; it focuses on supporting the emerging requirements of higher-level technology layers (such as Ontologies and Knowledge Services), will start to address performance and robustness issues, and also accommodate the lessons learned from the first two demonstrators.

6.1.4 M4 (30 months):

The third version of the distributed data access components will be deployed and operational across the application sectors. Support for deploying and using of the components will have been provided to partners, including training and assistance in enabling applications to take advantage of the distributed data access functionality. Again, a detailed evaluation will be produced, with conclusions on the focal points for the roadmap leading up to the final DDRA release.

6.1.5 M6 (48 months):

The final DDRA version will address the performance issues posed by knowledge mining of large-scale distributed databases, and improve compatibility and interoperability with database engines and third-party databases.

7 Conclusion

7.1 Achievements

In PM 19-24 significant effort was put into developing an improved solution for the GRIA OGSA-DAI package, i.e. a new application service for GRIA 5 that allows clients to access databases. The OGSA-DAI service can be deployed on its own to provide unmanaged access to databases, or it can be used with the GRIA service provider management package to take advantage of GRIA's service level agreement and billing capabilities. At the same time a lot of work was spent on developing custom solution for the application activities. Namely a routing mechanism has been written for the Meteo activity, an OGSA-DAI special activity for data accesses in the Auto activity. Additionally an evaluation of the peer-to-peer file system IGOR-FS developed by SIMDAT partner University of Karlsruhe was performed and made available to the other partners to take IGOR-FS into consideration for applicability outside the Pharma activity as well.

Generally integration with the WP2 grid infrastructure and support for the higher level grid technologies, i.e. those building on top of the DDRA services was provided. This enables the application activities to use the provided DDRA services in real world scenarios. Additionally a lot of effort was put into mutually consulting with the application activities.

7.2 Plans

This workpackage will continue the development of Distributed Data Access (DDRA) components integrated with the generic Grid infrastructure services coming out of WP2.2 and WP2.3, and with the Knowledge phase prototypes of the application activities. As with the previous phase, the exact set of functional or operational extensions and enhancements will depend on the evaluation of the PM 24 prototypes. It is clear that focus will again be on supporting the newly emerging requirements from "higher-level" technology activities (workflow, ontologies and knowledge services), facilitating effective management of metadata, and address issues of robustness and performance critical for the deployment to production scenarios. Furthermore, we expect that the problem of managing and administering the (remote) data repositories, complete with specifying access rights at the appropriate level of abstraction and granularity will need attention. Since OGSA-DAI is emerging as a uniform interface to remote and/or distributed data resources across SIMDAT, it would make sense to extend OGSA-DAI by interfaces to the basic management/administration mechanisms needed, and then couple with the Generic Grid Infrastructure, the VO components coming from inside or outside of SimDAT, and the databases used for the prototypes. As an important part of improving performance and robustness, the need and possible implementation of data caching (driven by local policies) and data replication (driven by global policies) will be investigated in depth. The CERN/EGEE data management and replication infrastructure contained in gLite seems now complete and robust enough to consider for inclusion (in part) into SIMDAT DDRA. At this time, the IGOR-FS distributed file system technology developed in the Pharma activity looks to be too low-level (file system vs. data repository) to be a candidate for DDRA integration. Yet, we will closely watch this activity. For the Meteo application activity, Intel will continue to provide key components for the catalogue synchronization and data routing elements. These are prime candidate for wider uptake within

SIMDAT and finally integration into the DDRA releases, contingent on the other application activities putting up requirements comparable to Meteo.