D14.17 “Yearly Report”  
(2\textsuperscript{nd} year of the project)

Author(s): Jarek Nabrzyski  
Title: Yearly Report  
Subtitle: Period I-XII.2003  
Work Package: 14  
Lead Partner: PSNC  
Partners: ALL  
Filename: GridLab-14-D.17-0001-Yearly_Report  
Version: 1.2  
Config ID: GridLab-14-D.11-0001-1.0  
Classification: IST

Abstract: This document is a GridLab’s yearly report for the 2nd year of the project.

Project Manager: Jarosław Nabrzyski  
Institute of Bioorganic Chemistry PAS  
Poznań Supercomputing and Networking Center  
ul. Noskowskiego 12/14  
61-704 Poznań, Poland  
Phone:+48 61 858 2072, Fax: +48 61 852 5954  
Email: naber@man.poznan.pl
# Table of Contents

1 Technical Achievement ................................................................................................. 3
   Introduction.................................................................................................................. 3
   Main Achievements of the Second Year of the Project ............................................... 4
   Software Integration ................................................................................................... 7
   Technical Problems ...................................................................................................... 7
   Quality Assurance Issues ............................................................................................. 7
   The GridLab Scenario .................................................................................................. 7

2 GridLab Software 1st Release ..................................................................................... 14
   WP1: Grid Application Toolkit .................................................................................. 14
   WP2: Cactus Grid Application Toolkit ....................................................................... 15
   WP3: Triana ................................................................................................................ 15
   WP4: Portals ............................................................................................................... 19
   WP5 Testbed .............................................................................................................. 21
   WP6 Security .............................................................................................................. 23
   WP7 Adaptive Components ....................................................................................... 25
   WP8 Data Management and Visualization ................................................................. 26
   WP9 Resource Management ..................................................................................... 27
   WP10 Information Services ....................................................................................... 29
   WP11 Monitoring ....................................................................................................... 30
   WP12 Access for Mobile Users ............................................................................... 31
   WP13 Dissemination ................................................................................................. 34
   WP14 Project Management ....................................................................................... 34
   Summary ..................................................................................................................... 35
1 Technical Achievement

Introduction

The main goal of the GridLab project [http://www.gridlab.org] is to bring both new and legacy applications to the emerging Grid world. It will provide new, innovative tools, services and methodologies for efficient development of grid enabled applications. The GridLab software will enable researchers and developers worldwide to develop new generations of engineering and scientific applications that can harness the power of the Grid.

The GridLab project produces various application-oriented Grid services for both end users and developers. Therefore, it covers the whole range of Grid capabilities as required by various applications, such as dynamic resource brokering, monitoring, data management, etc. These services abstract lower-level, very complex Grid functionalities and hence make the development of Grid-aware applications easier.

All these services can be accessed via the more abstract Grid Application Toolkit (GAT) APIs. In a nutshell, the GAT provides a set of very useful mechanisms to plug-in applications to various capability providers, in particular GridLab services, but also other specific libraries, tools, etc. Thus, applications can utilize service discovery at runtime, making use of whatever services are available including different implementations of the same service. This enables end users and application developers to easily develop and run powerful applications on the Grid, without having to know in advance what the runtime environment will provide.

In the first year the project was focused on the survey of the existing Grid technology, the selection of the appropriate middleware tool kits, the refinement of the project plans on the basis of the early experience with the GridLab testbed, the definition of the project architecture, the acquisition of the application user requirements as well as the establishment of the project management, technical and administrative infrastructure. Although not much implementation took place we have successfully participated in the Supercomputing 2002 conference and in spite of a limited number of workpackages we demonstrated the prototype versions of the GridLab software.

At the end of the first year the project successfully passed a very difficult and critical first Review. All partners were active and contributing to the project workplan.

The project also succeeded in making a very important contribution to the establishment of international standards by having the project manager as one of the original founders of the Global Grid Forum and many project leaders chairing the GGF working and research groups. The number of GridLab’s GGF chairs grew in the second year of the project.

In the second year the project struggled with the integration of the software. Actually, INTEGRATION was the key word for this period. The two project meetings, in Eger
(April) and in Olomouc (October), were devoted to the integration and refinement of the main GridLab scenario (see section “The Scenario” below).

Similarly, the overall project management and administration was reinforced due to an internal budget transfer to hire a second project administrator as agreed with the EC and the establishment of a quality control group drawn from the individual WP technical resources.

**Main Achievements of the Second Year of the Project**

- **Successful deployment of middleware for use by the application groups**

  A series of successively more sophisticated and reliable software has been deployed on testbeds covering major European sites in a continuous manner throughout the year. In November the pre-release middleware and GAT were made available to the application groups. Based on the first experiments the final release was delivered in December.

  The AEI numerical relativity group has started to use the GridLab software, including portal access, GRMS, data management, mobile user notification support, and security is under integration.

- **GridLab middleware being exploited by other projects and production facilities.**

  GridLab’s software is being exploited by many other projects. In this respect GridSphere is the most widely used software. The current base of users exceeds the number of 1000+ downloads. Several big projects are using GridSphere as their main portal development framework. These projects include:

  - The Grid Infrastructure Project is a three-year (2002 - 2005) project involving, in conjunction with the Multi-Scale Modeling Project, several Canadian National Research Council (NRC) institutes in the development of scientific applications and middleware for smooth and effective executing of applications on the Grid.
  - The GEON (GEOscience Network) research project is responding to the pressing need in the geosciences to interlink and share multidisciplinary data sets to understand the complex dynamics of Earth systems. To rise to this challenge, we have formed a coalition of IT researchers, representing key technology areas, and Earth Science researchers, representing a broad cross-section of Earth Science sub-disciplines. The need to manage the vast amounts of Earth science data was recognized through NSF-sponsored meetings, which gave birth to the Geoinformatics initiative. The
The creation of GEON will provide the critical initial infrastructure necessary to facilitate Geoinformatics and other research initiatives, such as EarthScope.

- GridSphere will be used in the HPC-Europa project as a primary Grid Portal development framework. This work will be done by PSNC. For more info go to: www.gridsphere.org.

- EPhysics Portal Development at the University of Melbourne: Theoretical astrophysics and experimental particle physics are major clients of high performance computing (HPC) facilities worldwide. They are key application drivers for grid technology and distributed computing paradigms and the codes are typically numerically intensive, and must process, access, or generate massive data sets. Virtual observatories (VOs) are one of the leading applications and early adopters of compute and data grid technology. They deliver a new paradigm for doing experimental astronomy, and in particular offer exciting possibilities to completely integrate observational astronomy and theoretical astrophysics. Experimental particle physics data analysis is the key application of grids. The Belle Analysis Data Grid project is developing a global data grid to access and process terabytes of data from the KEK B-meson factory in Japan, searching for the violation of a fundamental symmetry known as Charge-Parity, or CP, violation. The ePhysics program at the University of Melbourne has a number of general and specific goals. As part of a program, the GridSphere Framework has a major and crucial role in its success, including:
  - leading the international development of data description (metadata) and resource description standards for theoretical astrophysics,
  - developing and extending the existing WebApps with GridSphere into a system which can configure, submit and monitor a range of important theoretical astrophysics simulation and particle physics analysis codes on a high-end, production level,
  - integrating end-product visualization and analysis tools into the GridSphere portal, including the distributed data volume renderer (DVR); and using the developed system to undertake computationally intensive, key, astrophysics and particle physics science projects that have been identified by partner institutes, both in Australia and overseas.

GRMS is going to be used by the Polish national CLUSTERIX project, which aims at building a production quality linux-based multicloud grid. CLUSTERIX will be operational in 2005 and right now GRMS functionalities are being mapped vs. the CLUSTERIX requirements. It is already decided that GRMS is going to be used.

GRMS is also being used in another national project called PROGRESS. The project, which is in its deployment phases, decided to migrate from the
original Progress Resource Broker (PRB) to the GridLab’s GRMS. GRMS has much more features and is going to be supported by PSNC.

The Canadian National Research Council has recently shown its interest in using GRMS for their national Grid infrastructure.

The Mercury Monitor is being used for application monitoring in the EU Datagrid as a part of GAMI (Grid Application Monitoring Infrastructure) consisting of GRM/PROVE and R-GMA besides Mercury. It is also used by the P-GRADE Parallel Grid Runtime and Application Development Environment and the Hungarian Supergrid project.

The data management system from WP8 will be used for the national German grid infrastructure.

• **Active participation in the international standards bodies**

Several GridLab members chair or co-chair some of the GGF research and working groups. Moreover, key project members are participating in several EU consultation meetings. GAT, the main GridLab deliverable, is now under the standardization process of GGF.

• **Coordination and cooperation with related Grid projects**

The project continued to collaborate closely with other EU projects such as CrossGrid, DataGrid and others, while participating in the activity of the EU GRIDSTART project. The collaboration aims mainly at using GAT by these projects.

• **Open-source style Software license established**

The open-source software license has been established. GridLab’ software will follow two open source licenses: GridSphere will follow the GTPL-like license and the other software will follow the BSD-like license.

• **Tutorials**

GridLab has prepared and conducted several GAT tutorials. The tutorials have been given during several GGF meetings, HPDC and Supercomputing. The tutorial has also been given via AccessGrid node. Similar tutorials on GridSphere have been prepared and conducted. The tutorials are available at: [http://www.gridlab.org/WorkPackages/wp-1/Tutorials/GGF8/index.html](http://www.gridlab.org/WorkPackages/wp-1/Tutorials/GGF8/index.html) and [https://portal.gridlab.org/gridsphere/gridsphere](https://portal.gridlab.org/gridsphere/gridsphere).
Software Integration

The main driving force for the projects integration work was the Supercomputing demo at the end of November 2003 in Phoenix, US. The testbed group (WP-5) incarnated an Integration Team, which signed responsible for the technical aspects of that demo. the scenario (migration of a running Cactus job on the testbed (*)) involved almost all work packages, that work was quite complex, but also important for the project.

Although the scenario was implemented successfully, the real performance failed due to software misconfiguration, and had to be repeated. However, central pieces of the GridLab software (services and applications) have been demonstrated to work together as planned.

The next step in the integrational work is the preparation of the final implementation of the GAT, and the integration of the GridLab services hooks (adaptors) into that implementation. That work is proceeding very well, and as of the early 2004 we are close to a stable and semi-final implementation of the migration scenario.

Technical Problems

Last year revealed no new technical problems calling for the attention of the technical project management led by the Technical Board. The testbed that is an integration environment for the GridLab software seems to be stable and well defined by now. Testsuite failures were not caused by fundamental technical difficulties, thus they are all cleared up by now.


Quality Assurance Issues

As most work packages produced software prototypes by December 2003, the Technical Board needs to focus on quality assurance over the next quarter.

The GridLab Scenario

The next is a GridLab migration scenario that accelerated the project’s work last year. This main idea was to gather all work packages to facilitate the integration of the software. Having this scenario working, the GridLab software will have all the
interfaces needed to support even more complex scenarios. This scenario will be demoed during the next review in March 2004.

All GAT Application Scenarios fit into the GridLab architecture (Figure 1) which defines a cleanly layered environment providing application-oriented Grid services for users and developers alike, covering the whole range of Grid capabilities as required by applications, such as resource brokering, monitoring, data management etc. These services will abstract lower-level Grid functionality and will hence make the development and deployment of Grid-aware applications easier. The applications located on the highest level of the user space, can access all capability providers they need via the Grid Application Toolkit (GAT) API. The GAT also resides in the user space, providing interfaces to the capability providers in the capability space. The GAT will abstract those services needed by the Grid applications. This way applications can utilize service discovery at runtime, making use of whatever services are available, including different implementations of the same service. This will enable users and application developers to easily develop and run powerful applications on the Grid, without having to know in advance what the runtime environment will provide.

![GridLab Architecture Diagram](image)

**Figure 1. The general GridLab architecture.**

Based on the GridLab architecture described above, we have proposed the first GAT Application Scenario – Migration Scenario. We have decided to develop and
implement the scenario based on the idea of the job migration because of bad performance and using the GAT (Grid Application Toolkit) with all the underlying GridLab services.

In this scenario we assume that a user starts the GAT-enabled application (WP-1) via the portal (WP-4), and gets notified (WP-12) when his jobs starts. Then, the user monitors the GAT application through the portal and discovers that it is performing badly. The user requests the migration process to a system (one of the resource in the GridLab testbed, WP-5) where the GAT application will perform better. The migration process itself involves a set of underlying services provided by WP-7, WP-8, WP-9, WP-10 and the Monitoring System by WP-11.

**Stage I a (Figure 2)**

1) At the first stage, the user (or developer) is responsible for preparing the GAT application for the submission process. For the scenario, we assume that the user communicates with his application via the portal. However, it is also possible to run the migration scenario without the portal access (by invoking appropriate GAT API calls).

2) The portal calls the MDS service to find out where the GRMS service is available.
3) The portal prepares a job description schema (according to GAT application requirements) in a form of the XRSL (a job description language) and calls the GRMS service.
Figure 2. The user starts a job via portal and gets notified when the job starts.

Stage Ib (Figure 3)

Before the user gets notified when his job starts, there are many actions and communication processes among services in the Service Layer. See below for more details:

1) The GRMS service receives a job description with all requirements from the portal (or GAT application).
2) The GRMS service calls the MDS service to get information about all available resources and their parameters to meet the requirements of the GAT application.
3) The GRMS service provides the Adaptive service with a list of candidate sites (resources).
4) The Adaptive services gets statistics from the Monitoring system installed on the specific sites and prepare a set of useful analysis and statistics.
5) The GRMS service takes into account some additional information taken from the Adaptive service during resource management processes.
6) The GRMS service calls the Replication service to get information about the physical location of files required for the GAT Application.
7) The GRMS service calls the File Movement service to move files to the best resource chosen during resource management processes.
8) The GRMS service gives an ID to the GAT Application.
9) The GAT Application communicates with the Monitoring System.
10) The GAT Application register itself in the MDS service.

Figure 3. Before the user gets notified a set of underlying services are invoked.

Stage II (Figure 4)

1) The user asks the portal for job performance information.
2) The portal calls the GRMS to get information about an ID of the GAT application.
3) The portal checks the MDS service for more information about the GAT application.
4) The portal communicates with the Monitoring system to check some performance information.
5) The portal receives some performance measures from the Monitoring system.
Stage III (Figure 5)

1) The user decides to migrate his GAT application to a better resource. Thus, his asks the portal to migrate the job.
2) The portal calls the GRMS service to migrate the job.
3) The GRMS service calls the MDS service to get information about all available resources and their parameters which meet the requirements of the GAT application.
4) The GRMS service provides the Adaptive service with a list of candidate sites (resources).
5) The Adaptive service gets statistics from the Monitoring system installed on the specific sites and prepare a set of useful analysis and statistics.
6) The GRMS service takes into account some additional information taken from the Adaptive service during resource management processes.
7) The GRMS informs the GAT Application to begin the checkpoint process.
8) The GAT Application checkpoints itself by saving all internal information.
9) The GRMS calls the File Movement service to move files from Resource A to Resource B.
10) The GRMS begins the migration process.
Figure 5. The migration process after a user request.
2 GridLab Software 1st Release

Here we describe the main features of the software delivered within the 1st release.

**WP1: Grid Application Toolkit**

GAT, the Grid Application Toolkit, is the premier application focused grid programming toolkit. A product of WP1, GAT is an application programming toolkit which allows application programmers and their users to spend time accomplishing their goals. GAT relieves the application programmers from the burden of learning each and every new grid technology as they becomes available. GAT supplies a clean, uniform interface to the majority of Grid technologies so that application programmers need only learn one interface to have access to the myriad of current grid technologies. GAT simplifies the life of the developer.

What make GAT unique within the sphere of Grid computing is its flexibility. Through a modular, plugin-architecture GAT facilitates the incorporation almost any grid technology. Expert grid technology providers, such as those who produce Unicore and Globus, write plugins to GAT which deal with the intricacies of their particular technologies so the application programmer does not have to. GAT then presents this functionality to the application programmer through a clean, uniform interface relieving the application programmer from the charge of learning the piebald of different interfaces which constitute grid computing today.

GAT's Features:

- Flexible architecture (Currently undergoing GGF standardization),
- Flexible plugin architecture,
- Support for easy development and integration of third party plugins, so called adaptors,
- Provides an user facing interface for grid file management,
- Provides an plugin interface for grid file management,
- Provides an user facing interface for grid file replica management,
- Provides an plugin interface for grid file replica management,
- Provides an user facing interface for grid data streaming,
- Provides an plugin interface for grid data streaming,
- Provides an user facing interface for grid file streaming,
- Provides an plugin interface for grid file streaming,
- Provides an user facing interface for grid monitoring,
- Provides an plugin interface for grid monitoring,
- Provides an user facing interface for finding grid resources,
- Provides an plugin interface for finding grid resources,
- Provides an user facing interface for reserving grid resources,
- Provides an plugin interface for reserving grid resources,
- Provides an user facing interface for managing grid jobs,
- Provides an plugin interface for managing grid jobs,
- Provides an user facing interface for managing grid "adverts",
- Provides an plugin interface for managing grid "adverts",
- Provides an Event Model which is a clean means of dealing with asynchronous events.

**WP2: Cactus Grid Application Toolkit**

WP2 does not produce almost any code, except integration of the GAT with the Cactus code. Right now, the prototype integration has been done and the Cactus application using the GAT will be demoed during the review. Since WP2 is pretty dependent on WP1, the final integration should follow around month 28-36. However, the integration made so far allow to realize the scenario described below.

**WP3: Triana**

The main goal of WP3 is to define the GAT API in collaboration with WP1 and WP2. Having this in mind, we have developed the GAP interface, a bottom-up Java implementation of the JAT. The GAP is a set of simple a Java interfaces that Java programmer can use to access the underlying Grid middleware. Currently, the GAP is being extended to access the Job submission APIs and Data management APIs from other Gridlab work-packages to extend its functionality into something approximating GAT more precisely. To achieve this goal, the main focus of this year has been to define a production quality application scenario for Triana and have ALL of the Grid functionality required implemented within the GAP interface. Therefore, Triana is currently being applied to implement the inspiral search workflow on a number of resources distributed over our GridLab testbed. An overview of the scenario is shown below:
Briefly, this figure shows the interaction between the client and the various Triana services running on the tesbed. The workflow on the client is relatively simple, simply contains just an input unit, specifying the GPS second of the data to be loaded by one of the worker nodes along with the data length, a processing group node (containing the algorithm) and an output node that processed the results that are returned from the workers. The results contain a minimal amount of data e.g. the GPS second and the correlation ratio of the detected binary and are only sent if something interesting has been detected. Such events, typically of the order of a few per year, are a trivial but important step. On the client side, we are considering the use of various notification schemes upon successful detection e.g. email notification or screen alerts which are readily available within the system.

We have been working on the following sections of the GAP in order to support this integration:

1. **Data Management** - the data will be stored in a decentralised fashion across the GridLab testbed and interfaced through the GridLab data management
services. The data management team have already replaced the communication layer of the gravitational wave IO library for reading/writing data (called the Frame library) which allows geographical transparency by allowing local and remote access to be treated identically using a logical file reference. The user provides a GPS time as the logical filename, which is converted into the file location on the set of distributed resources.

2. Security - currently, security considerations are not built into the GAP interface (unlike the GAT). We intend to integrate the Grid Security Interface based on X.509 certificates into the GAP. This will enable us to contact the secure GridLab services that are currently available to us and deployed on the GridLab testbed.

3. Job Submission - based on the security infrastructure, we will be able to use GridLab services from within a Triana workflow similarly to the way we can invoke Web Services now. This will not only allow us to connect to the GridLab GRMS service (for Globus-based job submission) but also allow us to choreograph job submission workflow for complex submissions e.g. job submission could involve a number of steps: CVS checkout, compilation, service deployment etc.

These services are integrated at the GAP level and therefore extend their current functionality into a broader set of Grid services for application integration.

Recent Advances:

- Web Services. Created a GAP binding to Web Services, which allows web services to be graphically connected within Triana. This is the first step; the second, being developed now, will GSI-enable this implementation so that we can talk to other Gridlab Web Services to integrate Job Submission etc within Triana.
- A method of communicating with mobile devices via email (deliverable 3.6 –done). The two Triana units, SendMailMsg and RecvMailMsg are self contained units that perform sending and receiving email messages respectively. This will be used in the demo
- Developed P2PS (Simplified) – a lightweight P2P implementation based on the ideas of JXTA and P2P. This will be used to communicate with the distributed services.
- Bug fixes: Many major GUI bug fixes have been completed on the basis of the feedback from users.

Our main focus has been on providing bindings (i.e. Gridlab adapters) for Triana to map to heterogeneous sets of the middleware for the interaction with Gridlab services and other types of middleware. Here is a list of the software produced by WP3.

- Triana is a graphical Problem Solving Environment (PSE) that enables scientists to graphically program applications. Users write programs by dragging components called units or tools from toolboxes, and dropping them
onto a scratch pad, or workspace - it has not been written within the scope of GridLab but we have provided many plugins described below, to enable the Grid interaction.

- **GAP Grid Application Prototype Interface** (GAP Interface) is a generic high-level interface that provides a subset of the GAT functionality. It is a lightweight Java implementation that provides abstract advertising, discovery and communication within the Grid environment. It has three bindings described below:
  - **GAP Binding 1: JXTA** We have written an abstract interface to the Jxta P2P middleware, called JxtaServe. This implements the GAP interface to provide GAP services within a Jxta environment.
  - **GAP Binding 2: P2PS** - we have written a lightweight Peer-to-Peer middleware called P2PS (P2P Simplified), based on XML advertisements and messaging. With P2PS, you can write P2P style applications, without the complexity of other similar architectures such as JXTA and it has been designed to be independent of any implementation language or computing hardware. A number of groups are using P2PS, although we have not officially released the source code on the web site (we will though do so in May this year).
  - **GAP Binding 3: Web Services** We have created a GAP binding to Web services, called WServe. This provides a generic interface to discover and interact with Web Services. Services are described using WSDL, put into the UDDI registry. We have also integrated the Web Services Invocation Framework WSIF for communication via SOAP.

- **Triana Extensions:** We have written a number of utilities that have been incorporated as Triana units. For example, we have written an interface to SMTP and POP/IMAP to allow Triana units to send and receive emails. We have written BPEL4WS readers and writers for Triana taskgraphs so that we can integrate with standardised workflows.

- **Application support:** We have collaborated with the GEO 600 project to create an application scenario that uses Triana to search for coalescing binary star systems emitted from gravitational wave detectors. The significant application will be used for the GridLab dissemination by demonstrating the use of the GAP and GridLab services for a production of Grid scientific application run.
WP4: Portals

GridSphere, the premier product of the GridLab Portals Work Package (WP4), is a Portlet JSR (Java Specification Request 168) compliant Grid portal framework. The Portlet JSR was originally inspired by the Jakarta JetSpeed open-source portal project and has since gained wide-acceptance in the JCP (Java Community Process). Already, all the major portal vendors, including IBM, Oracle, Sun, and BEA, have released Portlet JSR complaint portal frameworks. Currently, there is only a handful of open-source projects which claim to be Portlet JSR compliant. These include Pluto, the Portlet JSR reference implementation released by Sun, LifeRay, and GridSphere.

What makes GridSphere unique is that it is the only open-source Portlet JSR project that was designed for the needs of Grid portal developers. Yet, GridSphere's modular design means that it supports general Web portal development. In fact, the GridSphere web application is just a portlet container that provides a core set of portlets for administering and customizing a portal and a portlet development framework we call the Action Portlet Model. The GridSphere portlet container is designed to manage other portlet web application repositories (WARs). This means new portlets can be dynamically added by deploying a portlet WAR and registering it with the GridSphere portlet container. In this way, GridSphere can be Grid-enabled simply by downloading the Grid Portlets web application from the GridSphere Project web site and deploying it wherever GridSphere is installed.

GridSphere's Features

GridSphere is a feature-rich and robust portal framework. Here is a list of the features GridSphere provides:

- JSR-168 compliant portlet container,
- Support for the easy development and integration of "third-party portlets" as web application repositories (WARs),
- Provides the Action Portlet Model, a cleaner and more effective way for handling portlet action events,
- Provides JSP tags that, when used with the Action Portlet Model, offer a complete solution for building JSP-based portlets,
- Flexible XML based portal presentation description can be easily modified to create customized portal layouts,
- Built-in support for basic Role Based Access Control (RBAC) separating users into guests, users, admins and super users,
- Persistence of data provided using Hibernate for RDBMS database support.
- Provides a basic component model, called Portlet Services, for sharing business logic across portlet web applications,
- Integrated Junit/Cactus unit tests for complete server side testing of portlet services including the generation of test reports,
Grid Lab

- Provides a set of core portlets that include functionality for login & logout, user and access control management, portlet subscription and customizable portlet layout,
- Localization support in the Portlet API implementation and GridSphere's core portlets support English, French, German, Czech, Polish, Hungarian and Italian,
- Open-source and 100% free!

Grid Portlet's Features

The Grid Portlets web application builds upon the GridSphere framework to support the needs of Grid portal developers. Its features include:

- Support for GSSAPI-GSI credential upload and retrieval, multiple credentials per user, and single sign-on with Grid credentials,
- Provides a high-level resource management model that makes it easy to administrators to add resources to resource registry as well as to discover resources with information services, such as those based on the Globus Metacomputing Data Service,
- Provides high-level API for job submission that supports multiple implementations, including job submission to Globus Gatekeepers with Java CoG and the Grid Resource Management System (GRMS),
- Provides high-level API for file browsing and file replication that supports multiple implementations,
- Provides a task API and model, upon which job submission and file tasks are built,
- Provides portlets that make use of the Grid Portlets API,
- Provides sample portlets for demonstrating how easy it is to use OGSA services within GridSphere.

Future Directions

We are continually looking for new ways to enhance the GridSphere and Grid Portlets web applications. Future work will include turning the Action Portlet Model into a fully even-driven component-based model that will make it easy to develop and share visual components, we will call Action Beans, across portlets and web applications. We are also going to add more support for workflow and task management within the Grid Portlets framework, as well as to make it possible to automatically advertise portlet services as Web services. The end result will be a one-stop environment for developing Grid-enabled Web services and portlets for using them!
More Information

More information about GridSphere can be found on our project website at http://www.gridsphere.org. Also visit http://www.gridlab.org/WorkPackages/wp-4/index.html to learn more about how GridSphere fits into the vision of the GridLab project.

WP5 Testbed

The GridLab Testbed group is responsible for both, taking care of properly working of the infrastructure, including coordination of all installations on remote sites and for monitoring the testbed. In order to efficiently proceed with these operations WP5 has developed the Grid testbed monitoring tools, which in addition to the Grid monitoring provide information that can be used by the GRMS system. This information is more stable and getting it is much faster comparing to MDS information.

GridLab Testbed Status Monitoring Tool

- The first version was based on the TeraGrid project Perl script with sequential testing, and then was completely rewritten to overcome inherent limitations,
- Based on centralistic approach (tests are launched from the central site and run every hour),
- User-defined degree of parallelism,
- Full parallelism is not feasible for larger infrastructure (infrastructure DoS), thus the number of simultaneous threads is configurable,
- Core written in Java, configurable using XML,
- Tests are independent of the programming language,
- Globus 2 tests use either Java CoG or a small C wrapper that handles timeouts for hung-up jobs (solves problems with firewalls swallowing TCP SYN packets),
- Supports test dependencies,
- Currently uses configured prerequisites for each test (e.g., tests job submission only if the gatekeeper can authorize),
- Supports tests triggered on-demand these tests use user supplied identity (based on X.509 certificates),
- Clean layered architecture.

Presentation layer

- XML/XHTML interface
  - supports both static and dynamic web pages,
  - static: test results are converted to XML and then transformed to XHTML using XSLT,
  - dynamic: using GridSphere portal supports history browsing,
multilingual interface.

- Web service interface is used as the information service by the Resource Management
  - Almost OGSA compliant (SOAP over GSI-HTTP)

Testing core with pluggable tests

- Globus-2 tests: GRIS, GSI-FTP, Gatekeeper, GSI-SSH, and GIIS,
- Job manager tests: tests all job managers advertised in information services, if they can run both normal and MPI jobs,
- GridLab specific tests
  - check the availability of MPI C and MPI Fortran compilers,
  - check on accepted CAs (whether compliant to GridLab requirements)
  - check whether the required software is installed and really working (C, C++, CVS, F90, GNU make, Perl, Java)
  - check whether grid-mapfile contains all required users
  - check on GridLab Mercury
- GridLab MDS Extensions and MDS web service,
- Storage Layer,
  - Any database supported by JDBC can be supplied; currently based on PostgreSQL database.

Service Tests

Tests of GridLab GSI-enabled web services using the HTTPG protocol
Implementation uses either Java CoG or C with gSOAP tool with GSI plugin Tests whether a webservice is up and responding. The following webservises are being tested:

- Grid Resource Management Service,
- Adaptive service,
- Metadata service,
- Replica Catalog,
- DATA movement,
- DATA browsing,
- Authorization,
- Message Box Service,
- Testbed Status.

Matrix Tests (N-to-N tests)

- Data Movement - tries to transfer a 1MB file between each tuple of machines using Data Movement Service developed by WP-8,
- The test finds firewall and misconfiguration problems, and also provides traffic
for Adaptive service developed by WP-7,
• GSI-SSH - tries to connect using GSI-enabled Secure Shell from each machine
to each machine.

WP6 Security
The Grid Authorization Service (GAS) is developed under the GridLab’s security
workpackage. The main objective of this WP is to provide functionality that would
meet most authorization requirements of grid computing environments. The GAS is
designed as a trusted single logical point for defining security policy for complex grid
infrastructures. Since flexibility is the key requirement, one can implement various
security scenarios, based on push or pull models simultaneously, using the GAS
service.

The GAS is considered as independent of specific technologies used at lower layers,
and it could be fully useable in environments based on Globus (supporting the
compatibility scenario with Globus’ CAS) as well as other toolkits. The high level of
the flexibility is achieved mainly through the modular design of GAS. It is divided
into five logical components, with the main GAS CORE module responsible for
performing authorization decisions based upon defined security policy, which is
maintained as a set of permissions for specific subjects (e.g. user) and objects (e.g.
resource). The remaining components are responsible for: managing security policy,
communication between users/applications/services and the GAS, integration with a
database system where policy information is stored, interaction with other security
solutions such as authentication services etc.

Key features:
• Designed to fulfill specific requirements of grid-based computing environments,
• Trusted a single logical point for managing security policy for virtual
organization,
• Independent of specific technologies applied to build a grid infrastructure,
• Support for different scenarios of using GAS, with a possibility to apply them
simultaneously within a single virtual organization,
• The modular structure allows introducing new modules for communication,
database support, service management and integration with external security
solutions.

Detailed features provided by GAS 1st release (state as of December 2003):
• A multilayered and hierarchical data structure for objects, which can model a lot
of applications into GAS,
• A hierarchical data structure for users and group of users,
• Database support,
  o MySQL database server and MyODBC drivers for accessing database
server are used.

- Optimizing the number of security policy rules by using wildcard functionality and negation of security policy rules,
- Using time or capability limitations for security policy rules,
- Two ways of communications:
  - Its own-designed protocol based on the GSI standard protocol;
  - The gSOAP protocol (gSOAP plugin for GSI).
- Managing the authorization service from a special administration client and from the GridSphere portlet,
- Getting a single authorization decision for the user or service. It is possible to get the authorization decision in three ways:
  - by gSOAP functions;
  - by command from the simple client application or the administration client application;
  - by the security portlet in the GridSphere.
- Generating a part of the security policy for the user in a user-specific form,
- Supporting two security authorization models,
  - RAD (Resource Access Decision),
  - RBAC (Role Based Access Control).
- Integration with Globus 2.2,
- Developed in C (except the security portlet for the Gridsphere which is written in Java).

**Future developments:**

- Extending methods of managing the GAS server and providing a method for the local controlling management of the GAS,
- Providing and extending the scenarios functionality of the GAS to support varied security scenarios and sessions objects for the GAS server users. After the first release, the extension of the scenarios engine will be provided to work with other security scenarios (More Complex Super-Scheduler Scenario, User to User Asynchronous Collaboration Scenario, ...). The more accurate scenarios are described in our deliverables listed at the end of this section,
- Extending methods of generating the security policy by improving and optimizing current solutions, providing methods which can solve deadlock problems,
- Providing a GUI administration client, which makes the GAS server very easy to manage. It is planned to create the GUI client as a portlet into the GridSphere. The current command line interface will evaluate to a more user-friendly interface, that would contain elements such as combo-boxes, edit-boxes etc. and thanks to this make the portlet easier to use. A full Graphic User Interface will be developed to present all the GAS server items in a graphic form,
- Possible integrations with other security solutions if there is enough time to develop, but the GAS will be ready to support the future security technologies
for sure
• Providing the possibility to distinguish between the functionality of the local GAS and the global GAS and accurate scenario for cooperation between local and global GASes,
• Continuing the current and new integration with various GridLab services like GRMS, Mobiles, Monitoring etc.

More info can be found at: https://www.gridlab.org/WorkPackages/wp-6/index.html <http://www.gridlab.org/>, in out mailing list: "security@gridlab.org" and in ours deliverables:
https://www.gridlab.org/Resources/Deliverables/D6.1c.pdf,
https://www.gridlab.org/Resources/Deliverables/D6.2b.pdf,
https://www.gridlab.org/Resources/Deliverables/D6.3.pdf, and

**WP7 Adaptive Components**
The software developed by WP7 consists of adaptive components, the Delphoi service, and the Pythia services with support programs.

The actual adaptive components are plug-ins to GAT modules. They all have a basic performance model for their particular task by which they make decisions about behavior adaptation, according to the current status of their resources. Although different adaptive components fulfill widely different tasks, they use similar implementation mechanisms and they all share the need to retrieve resource performance information.

All adaptive components use a unified interface to retrieve the various kinds of resource information: They use a Web-Service interface to the Delphoi service which has been deployed in the GridLab testbed. Delphoi provides short-term predictions for the status of resources. Its name has been inspired by the famous oracle from ancient Greek mythology.

While the Delphoi service is providing the information to the adaptive components, the raw data itself is collected at the individual sites of the GridLab testbed. On all sites we deployed a service called Pythia, named after the goddess through which the Delphoi oracle retrieved its wisdom. Each Pythia observes the status of its resource locally. Pythia can be configured to retrieve various kinds of information, currently it collects network performance data, CPU information (via GridLab's Mercury system) and disk availability.

This overall architecture allows for re-use of adaptation mechanisms in various grid application scenarios. Minimization of intrusiveness of the overall system has been a very important design criterion. While the Pythias constantly collect various properties, they only store this information locally on a size-limited disk space. Only
upon request from an adaptive component, the Delphoi service retrieves the information needed for the given request and delivers the needed performance prediction.

The necessary set of measured network performance properties has been identified in the course of GridLab's WP7. We are actively participating in GGF's network measurement working group (NM-WG). A result of our efforts is an upcoming GGF recommendation on a hierarchy of network performance characteristics for grid applications and services. Besides this fundamental work, we have evaluated and selected the most suitable and least intrusive network measurement tools. Besides measuring network delay directly from Pythia, we deploy the open-source tools Pathrate and pathChirp to measure bandwidth capacity and available bandwidth. We have produced versions of these tools that can be configured according to the needs of individual site administration policies (like available port numbers). We have also ported these tools to all platforms available in the GridLab testbed.

The system consisting of Delphoi services and Pythias implements mechanisms for fault tolerance and extensive diagnostics. The overall system tolerates individual machines outages as well as intruder attacks from the Internet. Delphoi's status www page provides detailed status information of all involved software components.

**WP8 Data Management and Visualization**

Here is the list of features of the software produced by WP8. This WP provides both: Visualization and Data Management services and tools:

**Visualization**

- Visualization of remote HDF5 files (storing Cactus data, medical data),
  - Based on custom ERET extensions for GridFTP,
  - Server-side processing of high-level operations: metadata gathering, dataset read. Designed for HDF5,
  - Using on the striped GridFTP server implementation provided by Globus,
  - Live video streaming from a visualization process,
  - Secure remote steering of the visualization process.

**File Movement**

- Web service providing third party transfers (copy, move, delete) of physical files
  - Uses Globus gass_copy library,
  - Uses credential delegation,
  - Blocking and non-blocking interfaces,
  - Reliable and restartable transfers (using restart markers internally),
  - No dependency on external software,
  - Progress indication (using performance markers internally),
  - Connection caching while copying multiple files,
File Browsing
- Web service providing information about physical files or directories like directory listings, modification time, file size,
- Connection caching,
- Uses credential delegation.

Remote Partial File Access
- Very efficient extraction of subsets from remote files,
- Integrated into Amira for progressive Visualization of remote data,
- Integrated with Triana for access to Geo600 files,
- Based on the Globus GridFTP server.

Replica Catalog
- Implemented in C++ using gsoap,
- Uses credential delegation,
- Provides a POSIX-like interface for browsing/access,
- Reference Implementation for the GAT,
- Bundles the Pythia Adaptive Service and the File Movement Service for efficient replication.

StorageBox Metadata Catalog
- Multiple namespaces for personal annotation and group collaboration,
- Pluggable database backend (with implementation for PostgreSQL),
- Pluggable authorization policies (with implementation of owner model),
- GSI and SSL authentication,
- SOAP interface, compatible with Axis and gSOAP,
- Portable C++ implementation, currently Windows, Linux, Mac OS X.

WP9 Resource Management
The GridLab Resource Management System (GRMS) is an open source meta-scheduling system that allows developers to build and deploy resource management systems for large scale distributed computing infrastructures. The GRMS, based on dynamic resource selection, mapping and advanced scheduling methodology, combined with the feedback control architecture, deals with a dynamic Grid environment and resource management challenges, e.g. load-balancing among clusters, remote job control or file staging support. Therefore, the main goal of the
GRMS is to manage the whole process of remote job submission to various batch queuing systems, clusters or resources. It has been designed as an independent core component for resource management processes that can take advantage of various low-level Core Services and the existing technologies. Finally, the GRMS can be considered as a robust system that provides abstraction of the complex grid infrastructure as well as a toolbox, which helps to form and adapt to distributing computing environments.

The first release of the GRMS is based on the Globus 2.X and uses Globus Core Services deployed on resources. The GRMS supports the Grid Security Infrastructure by providing the GSI-enabled web service interface for all client type, e.g. portals or applications, and thus can be integrated with any other middleware grid environment. The GRMS was entirely developed in Java and thus could be installed on various operating systems and resources. One of the main assumptions for the GRMS is to perform the remote jobs control and management in the way that satisfies users (Job Owners) and their applications requirements. All users requirements are expressed within XML-based resource specification documents and sent to the GRMS as SOAP requests over GSI transport layer connections.

Simultaneously, Resource Administrators (Resource Owners) have full control over resources on which all jobs and operations will be performed by the appropriate GRMS setup and installation. Note that the GRMS together with Core Services reduces operational and integration costs for Administrators by enabling grid deployment across previously incompatible cluster and resources.

The first release of the GRMS (v1.0a) allows Users to use the following functionalities:

- Ability to dynamically choose the best available resource or a queuing system for the remote job execution according to the provided Job Description, GRMS configuration and Administrator preferences,
- Ability to stage-in and stage-out files (input files, output files, stdin, stdout, stderr) required by jobs and users before and after executions according to the provided Job Description,
- Ability to setup environments before and after job execution according to the provided Job Description,
- Ability to submit and control a job remotely,
  - ability to cancel a job,
  - ability to check the job status,
- Ability to check detailed information about a host on which a job is/has been executed,
- Ability to check a list of candidate resources for a job and its Job Description,
- Ability to check a list of jobs submitted by an application user.

Moreover, due to a specific interface for registration of applications in the GRMS, more complex and more dynamic application scenarios are supported now. This functionality is available for all applications, in particular GAT enabled applications that are able to register callback information in the GRMS and then wait for a checkpoint call from GRMS. Once the GRMS’s call is received by the application, the process of checkpointing begins. In this case, it is an application-level checkpoint...
in which the application is obliged to store all information required for restart in one or more checkpoint files (described also in Job Description). Therefore, we have listed two additional functionalities of the GRMS below:

- Ability to register callback information in the GRMS,
- Ability to migrate a job to a better resource according to the provided Job Description.

Finally, the release complies to the following requirements:

- Act on behalf of users (resources) and meet application requirements concerning resources and their environment,
- run and control precompiled batch jobs remotely,
- run and control precompiled MPI batch jobs remotely,
- run Java applications remotely,
- register applications and receive unique JOB IDs,
- checkpoint applications remotely,
- migrate applications remotely,
- store all historic information about job statuses and resources which have been used during a the job submission process,
- contact the Information Service to receive static and dynamic information about resources,
- register GRMS in an Information Service,
- contact an Adaptive Components Service to get additional information about distributed resources and networks,
- stage-in and stage-out files required by jobs before and after executions using Core Services (GridFTP/GASS/FTP) or GridLab Middleware Services (Replica Catalog Service and Data Movement Service).

**WP10 Information Services**

The GridLab MDS has been designed as an extension of the Globus Project MDS, taking into account the requirements for the grid computing suggested by other GridLab WPs. The prototype release includes:

- GridLab MDS schema extension;
- information providers related to the new schema;
- a set of functionalities developed as a web service to publish and retrieve information stored in the MDS.

The information schema that we have extended, takes into account the requirements of the grid computing community very closely; it is not meant to be static but will continue to evolve and will be extended to support additional information that could be required in the future.

The current security policy adopted for the latest release of the GridLab MDS is to allow anonymous binding to the MDS for reading; however, only authorized users are
actually allowed to write data through our MDS web service, and authentication/authorization is based on Globus GSI. The features of the GridLab MDS can be summarized as follows:

- information schema based on LDAP,
- new information providers to extract information related to
  - Services;
  - Web Services;
  - Software;
  - Users;
  - Firewalls;
  - Virtual Organizations;
  - Certification Authorities;
  - Clusters;
- new method to configure LDAP server properly in order to reduce the response time,
- using caching mechanisms.

The release also includes a web service and related clients, to publish and retrieve information stored in the MDS. The main features are:

- support for authentication/authorization mechanisms based on globus GSI;
- support for service or web service register, unregister and discovery;
- extension of allowed character set according to the RFC 1738;
- support for handling of information related to software, virtual organization, firewalls;
- support for generic queries on LDAP server.

**WP11 Monitoring**

The Mercury Grid Monitoring System is developed within the GridLab project WP11. It provides a general and extensible grid monitoring infrastructure. The Mercury Monitor is designed to satisfy specific requirements of the grid performance monitoring: it provides monitoring data represented as metrics via both pull and push model data access semantics and also supports steering by controls. It supports monitoring of grid entities such as resources and applications in a generic, extensible and scalable way. The architecture of the Mercury Monitor is based on the GMA proposed by GGF, and implemented in a modular way with the emphasis on simplicity, efficiency, portability and low intrusiveness on the monitored system.
The input of the monitoring system consists of measurements generated by sensors. Sensors are controlled by producers that can transfer measurements to consumers when requested. Sensors are implemented as shared objects that are dynamically loaded into the producer at runtime depending on configuration and incoming requests for different measurements.

In Mercury all measurable quantities are represented as metrics. Metrics are defined by a unique name (such as, \{em host.cpu.user\} which identifies the metric definition), a list of formal parameters and a data type. By providing actual values for the formal parameters a metric instance can be created representing an entity to be monitored. A measurement corresponding to a metric instance is called the metric value. Metric values contain a time-stamp and the measured data according to the data type of the metric definition. Sensor modules implement the measurement of one or more metrics.

The Mercury Monitor supports both event-like (i.e. an external event is needed to produce a metric value) and continuous metrics (i.e. a measurement is possible whenever a consumer requests it such as, the CPU temperature in a host). Continuous metrics can be made event-like by requesting automatic periodic measurements.

In addition to the functionality proposed in the GMA document, Mercury also supports actuators. Actuators are analogous to sensors but instead of taking measurements of metrics they implement controls that represent interactions with either the monitored entities or the monitoring system itself. Besides monitoring it also facilitates steering.

For more information and to download the Mercury Monitor please consult the GridLab WP11 webpage: [http://www.gridlab.org/WorkPackages/wp-11/](http://www.gridlab.org/WorkPackages/wp-11/)

**WP12 Access for Mobile Users**

The software development in the GridLab Workpackage 12 - Access for Mobile Users - is focused on providing a set of applications that would enable communication between mobile devices, such as cell phones, Personal Digital Assistants or laptops and Grid Services on the other side. This class of applications is represented by clients running on mobile devices, mobile gateways acting as a bridge between clients and grid services as well as additional specialized grid services for mobile users.

The unique possibility of giving access to grid resources for users using relatively “weak” devices is one of the features that distinguish our applications from other grid related projects and workpackages. The other point is the technology chosen for the project: Java 2 Micro Edition - Mobile Information Device Profile (J2ME- MIDP). Applications (midlets) on the client side allow us to develop flexible, possibly off-line
working programs that may be used on a wide range of devices supporting J2ME.

WPs efforts focused on providing the following applications/modules:

- **Message Box Java API**: A short messages database which can be accessed via JAVA API. This API was designed to be used by an external portal for storing, managing and sending notifications in a form of an e-mail, SMS, as well as fax messages to end users. The currently supported features are:
  - one central repository for user messages,
  - storing, retrieving and sending messages,
  - messages structured in folders (adding and deleting folders as well as moving/coping messages between folders is possible),
  - messages can have properties (a set of flag for each message which can be set/reset).

- **HTTP(S) Message Box Server**: This application is used as the first version of a gateway for mobile devices. It is implemented in a form of a servlet-based web application and serves messages stored in the Message Box repository to the mobile client application - Message Reader Midlet.

- **Message Reader Midlet**: The J2ME compliant application for reading user messages served by the Message Box Server or the Mobile Command Center, additionally supports displaying a visualization images in the PNG format; those images can be situated any place on the web; their location comes from URLs parsed from messages). The main features of the Message Reader Midlet are the following:
  - CLDC/MIDP 1.0 compliant multithreaded client, support for HTTP on all devices, HTTPS on a wide range of devices (mainly MIDP 2.0 compliant or particular manufacturer's even for 1.0) – for devices without HTTPS support we will provide a version with security solutions based on the Legion of Bouncy Castle package for J2ME,
  - User friendly interface with the minimal level of difficulty and annoying alphanumeric typing (which is, in the users’ opinion, one of the main limitations of mobile devices),
  - All login data once typed (e.g. server URL, login name ) are stored in the persistent memory of a device for future use (using MIDP Record Management System),
  - The user is able to login into a server, view his/her profile data, view his/her folders and messages served by the Message Box via the Message Box Server or the Mobile Command Center,
  - Displaying of PNG visualizations is possible,
  - Automatic parsing of the content of a message text is performed to highlight any available URLs pointing visualization images,
  - The size of a client midlet together with a splash screen is only about 20KB! (this allows us to use the application even on the low-memory devices),
  - XML documents parsing can be performed using external kXML
package, however, due to memory constraints we use simple text messaging and our own parsing procedures,
- A midlet package can be downloaded directly from WML/HTML pages hosted by GridLab WP12 (downloading and managing applications is performed using Over The Air (OTA) J2ME provisioning).

• **Message Box Web Service:** This application serves messages stored in the Message Box repository to the grid/web services clients. The application is implemented as a web service written in Java and running on a Java based environment. The hosting environment consists of the Tomcat, which is the servlet container and the Apache Axis, which is a SOAP (Simple Object Access Protocol) engine. The interface of this service is defined in the WSDL format (Web Services Definition Language). The communication between the service and clients is done by GSI-enabled HTTP-based protocol called "httpg" implementing transport-level security introduced by the Globus Community. The protocol is essentially equivalent to the "https" one, but additionally it supports delegation of credentials. To provide the transport-level security a part of GT3-core was adopted. The main feature of the service is to provide a secure grid service interface to the Message Box API functions.

• **Mobile Command Center:** This application acts as a gateway for mobile devices, forwarding requests from mobile devices to grid services such as the Message Box Webservice or Visualization Webservice. Besides, this application extends the forwarded requests adding information about a mobile device limitations to enforce the response in a form acceptable for the end device). The Mobile Command Center is going to encompass:
  - HTTP Message Box server functionality (done),
  - Access to GridSphere services (e.g. job management - under development) with the same user/password as for the portal framework (done),
  - Access to specific grid services for mobiles (e.g. the Visualization Service) (under development),
  - Secure information exchanging (under development).

• **Mobile Client Midlet for GridLab** (this is another J2ME compliant application which will be used as a client for GridLab services). Its features include:
  - Message Reader Midlet functionality (done),
  - Steering user applications/simulations together with the MCC Gateway and GridSphere services (to be done),
  - Visualization browsing extended with a zoom, multiple image navigation and 3D visualization support (under development),
  - Storing data for future off-line work (to be done).
We are also planning to add extensions to all our services to make them able to work with other GridLab services (WP6 – Security: Grid Authorization Service - to be done, WP8 - Data Handling and Visualization: Metadata Storage Box service - under development and Visualization service - under development, WP4 – Grid Portals: GridSphere services - under development).

More information about GridLab’s Access for the mobile users workpackage and our applications can be found on our workpackage pages: http://www.gridlab.org/WorkPackages/wp-12/index.html

**WP13 Dissemination**

The project has been disseminated in much more ways than promised in the technical annex. PSNC put its own money into producing several brochures addressing the main GridLab achievements. This year PSNC has started funding a monthly brochure on GridLab main products. The first brochure, featuring GridSphere, appeared in January at the GlobusWorld and Gridstart concetration meetings. More than 2000 copies have been distributed worldwide. A similar brochure for every work package will be distributed every month in 2004. Additionally, during the GGF in Berlin a brochure featuring the GridLab GGF demo will be distributed.

Additionally to the brochures we have attended more than 50 conferences, submitted 40+ papers, gave 70+ invited talks, gave many tutorials and lectures on GridLab. The GridLab software is already being used by many other national and international projects including Poland, Hungary, Germany, Czech Rep., Italy, Spain, UK, Norway, Sweden, Zwitzerland, Canada, US, Korea, Australia and more. The whole dissemination report can be found in the quarterly reports.

**WP14 Project Management**

This year required much more interaction from the management team, especially the Sterring Committee and Technical Board. Teleconferences of the whole GridLab, the Steering Committee or the Technical Board were organized almost every week. Because of two reviews (December 2004 and June 2003) within 6 months a lot of effort has been made to organize the review and addressing the Reviewers’ comments. Two responses to the review were addressed in 2003 and the main outcome of these reviews was the following:

1. GridLab management made clear the responsibilities of the Steering Committee and the Technical Board,
2. More effort has been devoted to the GAT development. Moreover we have started the standardization process of the GAT under the SAGE GGF group. The community response was very good, unlike the response of the GGF Steering Group which seems not to understand the Application community’s needs. We had to struggle with many non-meritoric arguments etc. The group should be finally approved soon.

3. Some other GridLab work started their standardization effort as well. The new group Grid Scheduling Architecture (GSA-RG) with Jarek Nabrzyski as a co-chair will work on the definition of the Grid scheduling components. Many GridLab experiences will be ported to this group. GridLab is also active, in other GGF W/RGs, such as: Application and Testbed WG, Performance and monitoring.

4. GridLab is also very active in the Gridstart project, also with many gridlab persons chairing Technical Working Groups.

5. We devoted a lot of effort to software integration. Two 8-day meetings: in Eger (April) and in Olomouc (October) have been fully devoted to integration.

6. We have finally agreed on the GridLab open source license.

7. The new deliverable has been added: D13.9 Exploitation Plan, and produced in month 24.

8. We have clarified the position of GridLab vs. OGSA/OGSI and GT3.

9. Major edition of Annex1 was done, including many changes in deliverables. Recent approval of changes and some outstanding issues caused some problems with some deliverables to be delivered in time. Actually the software part is progressing very well, but it the paperwork that has not been yet done. These deliverables include D2.1 and D1.4 (AEI).

**Summary**

GridLab has been the leading project for Grid research and development addressed at the application level and enabling applications on the grid the last two years. It has established itself as Europe’s one of the main projects through intense activity by the partners in all domains of Grid technology and exploitation.

The first year of the project was dedicated to the establishment of the necessary organizational structures, requirements gathering and analysis, assembling a large-scale Grid testbed, surveying software toolkits and defining the architecture of the main components.

In the second year, the focus of the project turned towards development and integration but also towards legal issues regarding the open source license and consortium agreement. The project always aimed at the ambitious solutions and scenarios, where the applications take and advantage of dynamically changing grid environment with relatively small effort from user and application developer sides.

Project’s software already in the second year has been used by many other projects. What is important GridLab was always listening to these different external to GridLab groups to provide as generic as possible solutions. This sometimes delayed the
software development and documentation process but, on other hand, allowed to satisfy upfront many realy application communities.

The third year of the project will be called „year of quality and deployment”. The software will undergo the tough quality check procedures, while in parallel the enhancements in software features will continue. What is very important, many deployments in various application and general Grid communities will take place. This will be only possible if we continue our (tradiotional already) intensive integration meetings and also improve, if possible, our management procedures. The user community demands towards GAT and other GridLab software are very high. We have to start keeping a balance between what we can do and what the community wants us to do and safe this large space and bag of experiences and new requirements for the follow up proposal. Our exploitation plans are very ambitious but also we try to build for the future which will start after the project ends.

Regarding the project end date we will propose to extend the project for 3 months, i.e. until 31 March 2005.