



The ACGT technical architecture: Final Specification

Project Number: FP6-2005-IST-026996

Deliverable id: D 3.4 (with D11.5 incorporated)

Deliverable name: The ACGT technical architecture: Final Specification

Submission Date: 02/04/2009

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	
Document name:	The ACGT technical architecture: Final Specification (Merged with "Finalized ACGT security architecture")
Document type (PU, INT, RE)	INT
Version:	0.7
Submission date:	
Editor: Organisation: Email:	Juliusz Pukacki PSNC pukacki@man.poznan.pl

Document type PU = public, INT = internal, RE = restricted

ABSTRACT

This deliverable presents final vision of ACGT infrastructure architecture. At first key design concept are described, and the global view of the architecture picture.

Then the detailed view is presented, with description of the most of the components developed within ACGT and the interactions between them. There is also a chapter concerning physical deployment of the component in the ACGT testbed.

KEYWORD LIST: System Architecture, Service Oriented Architectures, Architecture Design Patterns, Grid, Distributed Computer Systems, Globus, Gridge Toolkit

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	27.02.2009	Draft	J. Pukacki
0.2	4.02.09	Draft	J. Pukacki
0.3	4.07.09	Draft	J.Pukacki
0.5	11.06.2010	Draft	J.Pukacki
0.6	13.07.2010	Draft	Brecht Claerhout
0.7	19.07.2010	Draft	J.Pukacki

List of Contributors

- Alberto Anguita, UPM
- Brecht Claerhout, Custodix
- Erwin Bonsma, Philips
- Stefan Castille, Custodix
- Johan Karlsson, UMA
- Luis Martin, UPM
- Thierry Sengstag, SIB
- Stelios Sfakianakis, FORTH
- Dennis Wegener, FhG

Contents

EXECUTIVE SUMMARY.....	5
1. OVERVIEW.....	6
2 KEY CONCEPTS.....	7
3 ACGT FINAL ARCHITECTURE.....	9
3.1 DESIGN VIEW.....	9
3.2 LAYERS DESCRIPTION.....	9
3.4 SERVICES ROLES.....	11
3.5 IMPLEMENTATION VIEW.....	12
3.5.1 <i>Components of User Access layer</i>	14
3.5.2 <i>ACGT specific services</i>	18
3.5.3 <i>Advanced Grid Services</i>	21
3.5.4 <i>Common Grid Services</i>	24
3.6 DEPLOYMENT VIEW.....	29
3.6.1 <i>Components deployed on each machine of the testbed</i>	29
3.6.2 <i>Collective infrastructure services</i>	31
3.6.3 <i>ACGT specific services</i>	31
3.6.4 <i>User interface applications</i>	31
4 SUMMARY.....	31
REFERENCES.....	33
APPENDIX A - ABBREVIATIONS AND ACRONYMS.....	34

Executive Summary

ACGT is an Integrated Project (IP) funded in the 6th Framework Program of the European Commission under the Action Line “Integrated biomedical information for better health”. The high level objective of the Action Line is the development of methods and systems for improved medical knowledge discovery and understanding through integration of biomedical information (e.g. using modelling, visualization, data mining and grid technologies). Biomedical data and information to be considered include not only clinical information relating to tissues, organs or personal health-related information but also information at the level of molecules and cells, such as that acquired from genomics and proteomics research.

ACGT focuses on the domain of Cancer research, and its ultimate objective is the design, development and validation of an integrated Grid enabled technological platform in support of post-genomic, multi-centric Clinical Trials on Cancer. The driving motivation behind the project is our committed belief that the breadth and depth of information already available in the research community at large, present an enormous opportunity for improving our ability to reduce mortality from cancer, improve therapies and meet the demanding individualization of care needs.

This deliverable describes the final architecture of the integrated Grid enabled technological platform built during the ACGT Project.

1. Overview

This deliverable presents the final specification of the ACGT environment¹.

The architectural design process was an evolutive one. In deliverable D3.1, the ACGT initial architecture was introduced. A top-level overview of the ACGT infrastructure and the general functionality it should provide to the end users was given. From there the architecture further evolved.

As a main design pattern for the ACGT architecture the layered concept was chosen. For this, the different layers of the architecture were defined and the required functionality was decomposed into those layers.

The layers themselves are described in the context of the general functionality they should provide. The layered structure determines the communication rules between different components. Thanks to this layered approach, it was possible to assign the work of each technical workpackage to the appropriate design scheme. The WP4 workpackage was responsible for providing the grid infrastructure that consists of two major layers: the Common Grid Infrastructure layer and the Advanced Grid Services layer.

The other technical workpackages have been building their software based on the provided interfaces to the grid environment.

In the same way, the common security requirements were determined in the initial phase, so that all implemented services in the next phases could take the security policies of ACGT into account (across layers).

An important step in the architectural design has been the definition of the actual components (services) and their placement on the overall architecture picture with all dependencies among them.

There were no rigid rules for component implementations and the role of components in the ACGT environment. The ACGT environment was designed as a set of loosely coupled services, offering their interfaces to others. Each component can be used as separate entity, but binding them together based on some scenario provides more advanced functionality.

The architectural design process has revealed the importance of the leading usage scenarios for successful development. Thanks to the elaborated scenarios, it has been possible to efficiently detect required components and dependencies between them.

¹This document is in fact the merger of D3.4 “The ACGT technical architecture: Final Specification” and D11.5 “Finalized ACGT security architecture”.

2 Key concepts

There are two important concepts behind ACGT architecture design. The first one is “layering” and the second one is the “Service Oriented Architecture (SOA)” concept.

They were used together in a process of design and development of the ACGT infrastructure components. In particular cases, the final architecture is a compromise between those two ideas.

Layering is one of the most popular architectural design patterns. It structures software, so that it can be decomposed into groups of components each residing at a particular level of abstraction. It also helps to split responsibilities of the different groups of components in the system.

Initially layered architectures were used to divide application presentation features from application logic and database access (three-tier model). The highest level layer in that model is used as an interface for the end user that translates tasks and results to something the user can understand.

The Logic layer coordinates the application, process commands, makes logical decisions. It also moves and processes data between two surrounding layers. Within the Data Layer all the information required by the application is stored and retrieved from storage databases or filesystems.

Such a classical approach as described above can be used as a reference point for more complex applications and systems. It is possible to extend the basic model by spiting particular layers into additional ones depending on the characteristics of the concrete problem. The actual meaning of the layers is thus determined by the modelled environment.

For the ACGT infrastructure the layers denote groups of components performing some operations more ore less specific to the bio-medical background of the project. The upper layers (closer to the end user) are strongly connected to the terms and ideas from the biology and medicine areas. They are using domain specific languages, try to benefit from ontology developed within the project, and stay in general distant from the technical infrastructure - computers, networks, storages, etc.

The lower layers were design to be universal in the context of operations performed. Their main role is to provide access to physical resources in the most convenient way. Components of those layers are not aware of problem specific vocabularies, they are not using ontology descriptions, and can be easily replaced by different components with similar functionality if necessary.

There are some principles behind the layered design. An important one is to implement the services in the same layer using the same or similar technology, so that the communication between components within one layer can be based on the same mechanisms. It makes the development process much easier and makes it possible to introduce guidelines for software developers. This rule is especially useful for the layers that have a lot of newly developed components. In the case where existing infrastructure is heavily re-used, it is not so significant.

The other rule connected with layering is maintaining clear layer separation in the terms of communication between layers. It means that components of one layer can only communicate with the components of the layer directly below. This makes the architecture much more clear and better structured. Of course some exception to this rule are accepted. For instance some logical functionality can be used by the components of all layers, and it is pointless to replicate the same element throughout all layers.

Nowadays, one major trend in system design is Service Oriented Architecture (SOA). It defines software infrastructure as a set of loosely coupled software services that support the requirements of the business processes defined by the users. In a SOA environment, resources on a network are made available as independent services that can be accessed without knowledge of their underlying platform implementation. A service-oriented architecture is not tied to a specific technology. It may be implemented using a wide range of interoperability standards but the most important set of standard in the context of SOA is Web Services (WS). The key issue is that all services that building some environment are independent from each other, they publish their interfaces and that is enough for any other entity in the system to invoke their methods (meaning 'use' the service).

At first sight it is difficult to imagine how the SOA concept coexists with the layering paradigm. The fact is that they operate on different levels of abstraction. Layering concerns functionality of services and their distance to the end user on the one hand and physical resources on the other hand, SOA rather describes the communication pattern.

The two concepts together constitute the logical framework and provide clear guidelines for service developers. For newly developed services, first of all, the location within the layers must be defined. This decision is based on the role of the service, e.g. if the service will be used directly by the end user, or if it provides access to physical resources. When this is determined, the development of the service should follow SOA rules: the service should be an independent entity accessed through a well defined interface. It should work as a standalone component or provide functionality cooperating with the other services.

3 ACGT final architecture

3.1 Design view

The goal of the initial architecture design (D3.1) was to deliver a basic framework for infrastructure developers. It was provided to help all technical partners in ACGT consortium with the development of their software components. At the beginning of the project there was no detailed specification of the required services to be created, so the initial architecture was intended to be a framework - a design view of the future architecture.

The picture below presents that general design view as a starting point for development of ACGT infrastructure.

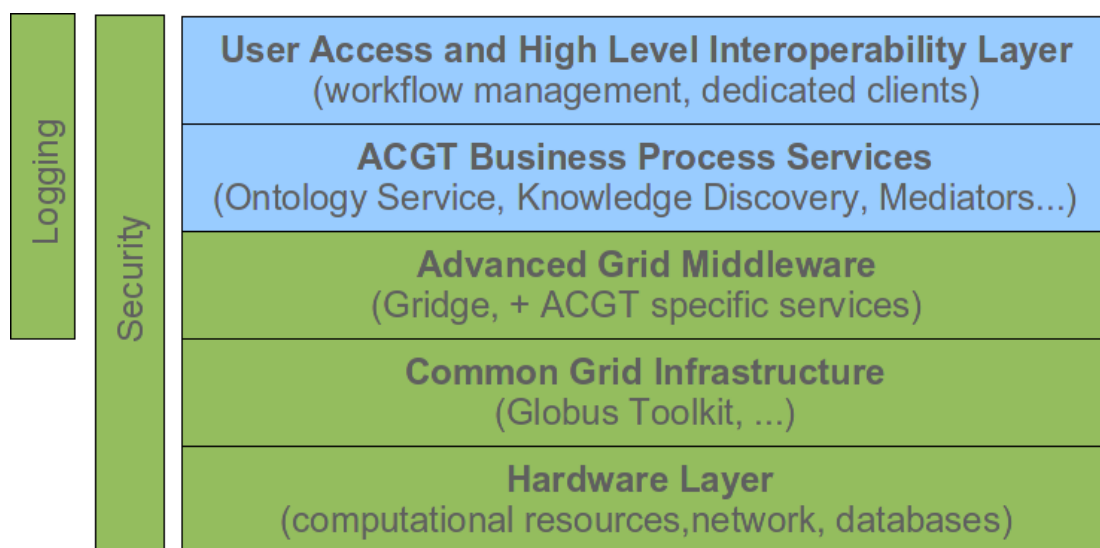


Fig.1. ACGT layered architecture.

There are five horizontal layers presented on the picture. The lower ones are located closer to the physical resources, the top ones closer to the end-users. The mechanism used for providing distributed access to resources is grid technology. The layers at the top are responsible for providing ACGT specific solutions for bioinformaticians and clinicians.

There are two vertical layers. The first one is logging infrastructure and is used by the services regardless their location within architecture. It is very important to have ability to track the activity that in many cases involves different services from different layers.

The other vertical layer is security that constitutes common infrastructure for all components in the infrastructure. It is very important to keep consistent security policies throughout the infrastructure. And also to be able to dynamically manipulate the policies for the complete architecture in the context of virtual organisation management.

3.2 Layers description

As it is marked on the picture in green, the grid technologies are present on the three lower layers. The lowest one is the hardware resource layer where computation is performed and where physical data is stored. The role of the next layer above the hardware resources is to

provide unified, remote access to physical resources. It provides basic functionality required for remote computing and data access:

- job execution and control
- basic authentication and authorization
- file transfer
- databases access
- hardware monitoring
- information about the state of resources (static and dynamic metrics)

The last Grid layer is the Advanced Middleware Layer. It is responsible for providing more advanced mechanisms in the Grid environment. Services in this layer can be described as "collective" because they operate on set of lower level services, to realize more advanced actions - e.g. a metascheduling service that submits jobs to different local queuing systems using the Common Grid Infrastructure remote interfaces.

Functionality provided by this layer can be gathered in a following main points:

- resource management - metascheduling
- data management (database access and file storing and transferring)
- services authorization
- grid monitoring

As it can be seen on architecture picture, the Grid layers are separated from the rest of the system mainly consisting of services that operate on domain specific content.

The advantage of it is clearly visible: based on that Grid platform it is possible to build many different environments for different fields, not only biomedicine.

Grid layers are supposed to provide a standard and secure way for accessing hardware resources of the Grid environment.

The ACGT business process layer consists of components which are not aware of physical resources and the Grid environment. Grid is used as a whole, to perform more abstract actions, and to get information required by end users. Services of this layer are using abstract description of the world defined as some ontology.

There are also services that are able to translate high level descriptions to language understandable by the lower layer (grid). These services provide a higher level of integration of different resources and data and makes them more similar to real word objects.

It is possible do define following functionality for Business Processes Services:

- ontology description provisioning
- execution and control of workflows
- translation of abstract object to resources names
- translation of abstract actions to grid calls
- virtual organization management
- knowledge discovery
- biomedical algorithms

Services of this layer are specific for ACGT environment, but in some cases it is possible to reuse existing ones. For instance it is very important to include analytical services that are

already there, into the ACGT infrastructure. The level of integration also depends on the security requirements. If there are no restrictions in using the service it is possible to skip the integration on the level of authentication and authorization. But there are more problems with introducing security policies for the third party components. They require ACGT compliant wrappers responsible for taking care of access rights management that is consistent with the whole security policy of ACGT.

The User Access Layer contains all application and tools that provide access to the ACGT Environment for end user. There can be a wide variety of software, developed using different technology:

- portals
- standalone applications
- clients dedicated to specific operation
- workflows editors
- visualization tools

Client applications are in the most cases tightly connected to functionality provided by underlying layer. They are also going to be used by different but specific groups of users. Thus it would be difficult to use general tool but there will be need to create dedicated application. These two arguments are showing that User Access Layer is mostly developed by ACGT.

3.4 Services roles

In the ACGT environment there can be a lot of different services. The SOA paradigm defines architecture as a set of loosely coupled software services that support the requirements of business processes and software users. That kind of definition provides just a rough idea of the service without any notion what the functionality of the particular services should be. We would like to build ACGT environment in a SOA manner but it is required to add some semantic, some additional constraints and rules for building the ACGT services.

The services in the ACGT environment can be categorized based on different criteria:

- Position in the ACGT architecture

The goal of a layered architecture is to introduce different abstraction levels for services. Services from different layers operate on different world of terms. Some of them are located near the physical resources using hardware terms, the others are located near end user and should be contacted using language of meta descriptions. The other important fact is that upper layer services are specific to the scientific domain and lower level ones are more general (IT related) and could be used in a generic way by different clients:

- Service of Common Grid Layer
The Services from that layer are used for accessing hardware resources.
- Service of Advanced Grid Layer
Provides more advanced, collective functionality, using lower level services to realize client requests.
- Service of Business Model Layer
These are specific services for ACGT environment. They are closer to the end user, can operate on terms from the biomedical and cancer research domain (meta descriptions, ontology)

- Role in the ACGT environment
It is obvious that services can play different roles in distributed environment. For ACGT we can define two main groups:
 - Infrastructure Services
The services used for building the ACGT environment (middleware). They are performing some more general actions (management related). All of the services from Grid layers belongs to this group. They are responsible for performing general actions like computational jobs scheduling and monitoring, file access and transfer, database access, authorization decisions. The examples of infrastructure services are: Grid Resource Management System (GRMS), Data Management System (DMS), Grid Authorization Service (GAS), etc.
There are some more application specific services that belong to the infrastructure. A good example of this is the GridR Service, responsible for spawning R scripts in the grid environment.
 - Analytical Tool Services
These are the application oriented services dedicated to one kind of activity. They are not performing any general tasks, but are focused on concrete goals. They are mostly located in ACGT Business Model Layer
- Owner/maintainer of the service
 - ACGT Service
An ACGT service is a service that is developed or adopted by the ACGT consortium that has full control over it. The control means possibility to introduce common policies for building such a service, for instance applying the same security mechanisms, the same technology for accessing the service, etc. This services should also be trustworthy - we need to be sure that what the service is supposed to do is in fact doing. Remark: all Infrastructure Services should be ACGT Services.
 - Third Party Service
It is not possible to build everything from scratch. There is a lot of existing tools available freely that can be used without any limitations. We should to define the policy (evaluation procedures) of incorporating such services to ACGT infrastructure. In some cases it means developing additional components - wrappers etc.

3.5 Implementation view

The next step in architecture design is to put real components on the design view, and describe the interactions between components.

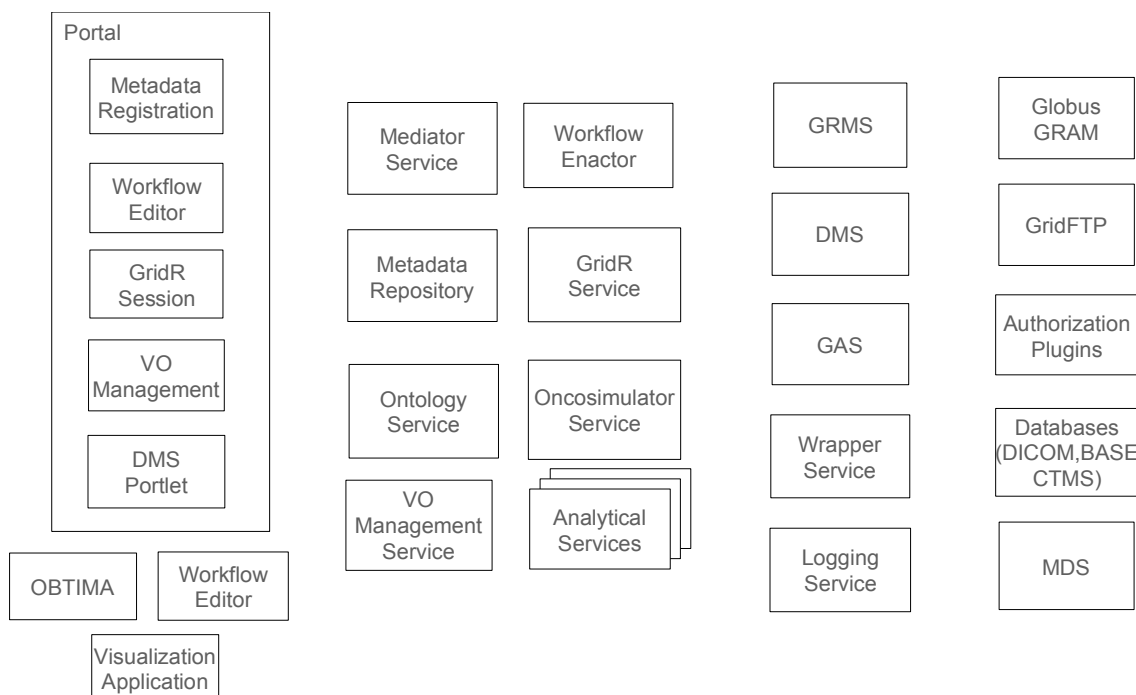


Fig. 2. Implementation view of the architecture.

The picture above presents most of the components building ACGT environment. From the left there are end user tools for accessing the infrastructure. The most important one is the ACGT Portal that consists of several portlets (applications embedded into portal) used for accessing different services from the layer below. Then there are the ACGT specific services responsible for performing advanced scenarios and using grid infrastructure (last two rows) for accessing resources.

Because the ACGT architecture adheres to the SOA paradigm, it is difficult to present the interactions between components in a clear way. On the picture below the lines represent the communication among services:

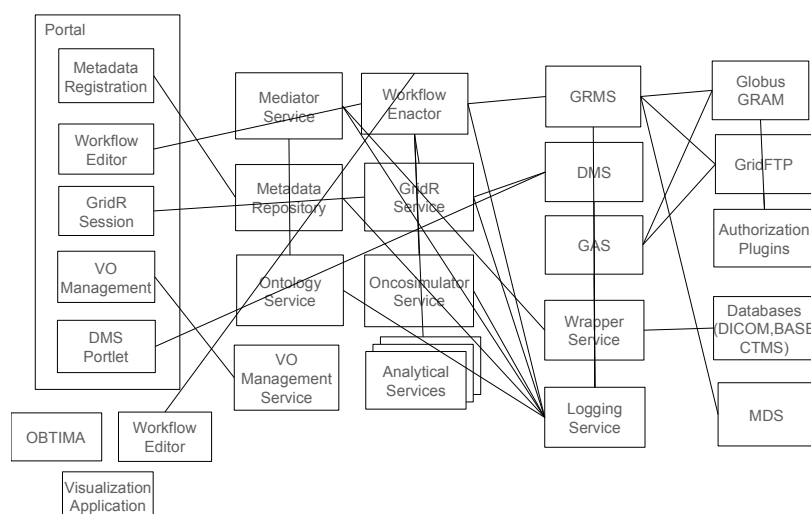


Fig.3. Communication dependencies between component of ACGT infrastructure.

Although the communication patterns on the picture above appear jumbled, the layered communication model can be seen upon closer inspection. The picture below summarises these communication dependencies for reasons of clarity.

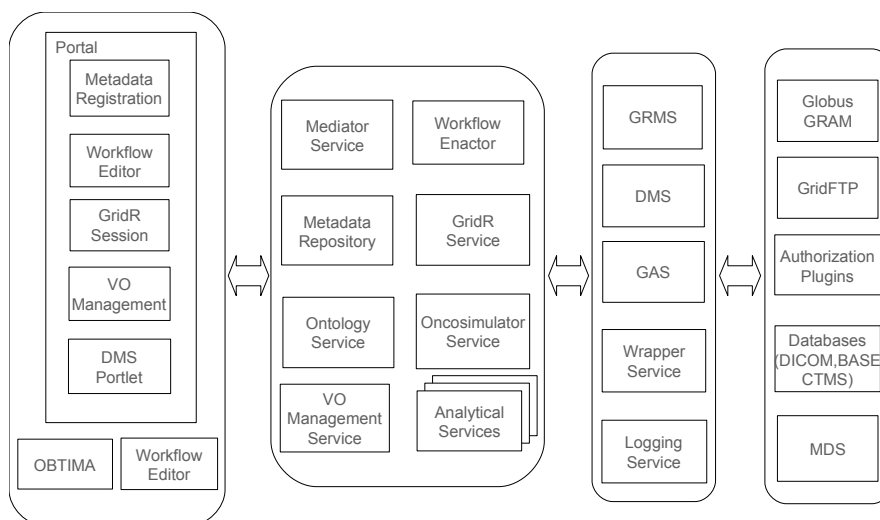


Fig.4. Communication dependencies between layers.

3.5.1 Components of User Access layer

3.5.1.1 ACGT Portal and Portlets

There are two types of applications developed for the end user to access the ACGT infrastructure. The most important type consists of applications under the form of portlets which integrate with the ACGT Portal.

The ACGT Portal is built on the Gridsphere portal framework. GridSphere is an open-source JSR-168 compliant portal framework that is ready to run with a suite of tutorial and example modular web components, called portlets. A portlet is a module of a dynamic web page created using Java. For creating dynamic web pages, JSP (Java Servlet Pages) was developed (JSP combines html with java). A JSP web server is needed to run such applications. As only some parts of the dynamic page actually needs to be modified while doing a set of actions, web page content was redesigned as a set of integrated dynamic JSP modules and a static html content.

JSR (Java Portlet Specification) 168 is one of the main standards in portal development. JSR 168 is a standard API for integrating portlets in portlet containers. It defines a contract between the portlet container and portlets and provides a convenient programming model for portlet developers. The Java Portlet Specification V1.0 was developed under the Java Community Process (having as members experts from leading industry companies) as JSR 168, and released in October 2003. The second version of the standard, Java Portlet Specification V2.0 or JSR286 was released in July 2008 and is in the process of adoption.

The Gridsphere portal framework provides a standard based platform for the easy development of portlets. Portlets are defined by a standard API and provide a model for developing new portal components that can be shared and exchanged by various portlet containers. Gridsphere provides a portlet container, a collection of core portlets and an advanced user interface library that makes developing new portlets easier for application developers.

The most important portlets developed and used in ACGT are:

- Workflow editor

The ACGT Workflow Editor is a graphical tool that allows a user to combine different ACGT services into complex workflows. This tool is accessible through the ACGT Portal and therefore has a web based graphical user interface. It supports searching and browsing of the available services and data sources and their composition through some intuitive and user friendly interface. The workflows created can be stored in a user's specific area and later retrieved and edited so new versions of them can be produced. The publication and sharing of the workflows are also supported so that the user community can exchange information and users benefit from each other's research. Finally the workflow editor supports the execution of the workflows and the monitoring of their enactment status. More detailed information can be found in deliverable "D9.3 – Data and Metadata Management"

- GridR Session

In ACGT, GridR is used as a tool for the remote execution of R code in the grid. More specifically, the task of R code execution is submitted as a grid job to a remote grid machine. The current implementation of the server side GridR components that are related to the grid environment is based on several external software components, namely the GT4 grid middleware, an installation of the R environment on the grid machines which will execute the functions remotely and a GRMS-Server installation from the Gridge toolkit on a central machine in the grid environment that is responsible, for instance, for resource management. On the client side, GridR consists of a set of R functions and involves the Cogkit, which is responsible for proxy generation and data transfer, and a GRMS-Client. The client side part is structured around the components "RemoteExecution" (JobSubmission and JobDescription Generator) and "Locking". The RemoteExecution component is responsible for the execution of R code as a job in the grid environment by transforming the R code to execute into a set of files, creating a job description file in the respective job description language, and submitting the job to the resource management system by the GRMS-client. During this process, the locking component takes care of the consistency of files/variables. An R programming language interface that supports the access to the ACGT services is provided in the ACGT environment. This means that R users and developers will have access to distributed resources in a transparent fashion, as if those resources were local. The complexity of the grid is thus hidden from the user. Again, accessing the ACGT grid environment requires no changes in the core R implementation. In practice, grid access is performed through the call of predefined R functions loaded from a package. R users can make use of the grid technology in a transparent way by passing the functions to be executed in the grid as input to one of those predefined functions (grid.apply) in their local code. TheGridR Session is an applet developed by FhG as a GridR web-based client. The applet was wrapped inside a portlet and made available in the private area of the portal.

- Metadata Registration

The user interface to the Metadata Repository is a JSR 168 compliant portlet application that allows a user to browse and edit the Services and Data types stored into the Metadata Repository. It also allows the user to start (enact) a service. The Metadata Registration Portlets v2.0 were developed using the available Gridsphere tools and rely on plain JSP and minimal JavaScript for rendering. These portlets replace the Metadata Registration Portlets designed for the first prototype of the ACGT Portal and presented in D14.2.

- DMS portlet

The Gridge Data Management System is one of the main components of the Gridge Data Management Suite (GDMSuite) - a middleware platform providing a uniform interface for connecting heterogeneous data sources over a network. GDMSuite stands for the backbone of the Gridge environment, on which computational services perform their operations. The Gridge Data Management Suite constitutes a bundle of packages, designed for the creation of a complete and robust data management environment. It is capable of fulfilling even the most stringent enterprise requirements of grid environments in terms of reliability, security and performance.

Data Management System (DMS) is a middleware application, based on the SOA model that determines a loose coupling between reusable components, which are platform-, language-, and operating system-independent. Similar to computing and network resources, DMS provides services to manage and retrieve data files in order to support grid jobs. The computational resources managed by DMS can be described by a metadata scheme, which allows create an abstract, semantic and explorable layer of resources.

Originally, DMS was provided with a stand-alone Web interface called DMS Portal. The DMS Portlet was developed by SIVCO by emulating all the features available in the DMS Portal. Also the web service layer used by the DMS Portal to access the backend applications was ported into the DMS Portlet.

- GRMS portlet

The Gridge Resource Management System (GRMS) is an open source meta-scheduling system, which 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 feedback control architecture, deals with 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, which can take advantage of various low-level Core Services and existing technologies. Finally, the GRMS can be considered as a robust system, which provides abstraction of the complex grid infrastructure as well as a toolbox, which helps to form and adapts to distributing computing environments.

- VO Management portlet

This portlet can be used for defining security policies for resources inside created VOs. It is a frontend to the authorization service of ACGT (GAS) which is the central point for storing security policy and VO definition for the whole environment. The VO Management Portlet provides a better approach to a VO approach to roles and rights, while keeping a good control of the resource access.

3.5.1.2 Other Applications

There are several other applications and components for end-users which do not directly relate to the ACGT portal.

- ObTiMA

ObTiMA is a complete suite for designing and running a clinical trial, it consists of: a complex clinical trial management tool, workflow editor - the same application that is used in portal (can be also executed as a standalone tool) and visualization

applications - miscellaneous applications used for visualization of data generated by other applications.

ObTiMA (Ontology based Trial Management for ACGT) enables the chairman of a clinical trial to set up and manage multi-centric trials from a single interface. This is not a trivial task at all and has eluded many efforts to date, primarily because of the complexity of the task but also because of the incompatible data formats and semantics of systems currently in use at hospitals.

One of the main components of ObTiMA is the CRF Creator, which allows a chairman to capture data definitions for a clinical trial in a standardized way. Another component of ObTiMA is the Trial Outline Builder used for writing the Trial Protocol (essentially the plan of the Trial). With the help of templates given by an integrated Master Protocol the chairman of a trial will be guided through all steps of the development of a new trial which will be more streamlined and allow data from different trial sites and IT systems to be integrated in a seamless and easy to analyze manner.

From a clinical point of view ObTiMA helps to increase the number of clinico-genomic trials by facilitating the workload involved in creating a new trial. More importantly, patients participating in such trials are expected to benefit from new treatment options that will become available.

- CAT

The “Custodix Anonymisation Tool” (CAT) aims to simplify the process of de-identifying and exporting personal data, and is as such part of the ACGT Data Protection Framework. The tool was created to meet the demands laid out in deliverable “D10.4 The ACGT ethical and legal requirements” for exporting pseudonymous data from the (internal) hospital data stores to their anonymous ACGT counterparts (i.e. the ACGT accessible data sources, also physically residing in the hospitals). The tool is innovative in a sense that it offers a generic solution regardless of the type of data to be treated or of de-identification requirements.

CAT consists of a “workbench” and a “wizard”. The “workbench” serves at defining the mechanics (data protection profile) through which data is exported for sharing, the “wizard” allows to easily apply those profiles on various data sources. The workbench allows domain experts and privacy professionals to:

- create a mapping from a specific data format such as flat files (e.g. CSV), imaging data (e.g. DICOM), microarray data, structured data (e.g. XML, databases) to a generic data model
- define the set of actions that should be performed on the generic data model in order to de-identify data (i.e. the data protection profile)

Once that a data mapping and a data protection profile is created in the “workbench”, end users (i.e. physicians) can easily export several data sources at once by using the wizard (logically, this operation can also be automated).

Privacy processing actions such as creating a pseudonym (randomly assigned, through encryption of immutable identifiers, etc.), freetext de-identification, basic encryption, calculation of relative dates (to obfuscate absolute birthdates), etc. are defined towards the internal generic data model. The privacy transformations are provided as a library (also usable by developers through an API), hence the functionality of the tool can be easily extended to suit new requirements.

The big advantage of this approach is that a uniform and auditable data protection strategy can be followed in the different sites partnering in ACGT.

- PKI and credential management frontends

Authentication of services and end-users in ACGT is based on Public Key certificates (X.509). A dedicated Public Key Infrastructure (PKI) was set up for ACGT (see also further). A separate management site for supporting enrollment and de-activation of users and services through the ACGT PKI was provided within the project. Through this administration site, prospective users or new services can be registered (and obtain their main ACGT credentials), credentials can be renewed and revoked. This site is the main management interface for the authorized registration administrators that are assigned to each of the ACGT partner organizations for performing their Registration Authority (RA) duty.

For the end-users, the registration process (i.e. credential generation) is made as transparent as possible through the site. A Java applet handles the key generation (of the private key) and the correct installation of the certificate generated by the ACGT Certification Authority on the end-user's computer (this combination is known to end-users as the "ACGT passport").

Equally credential management for login to the portal (i.e. MyProxy bootstrapping with a delegated credential, known to end-users as the creation of an "ACGT Visa") is made transparent for the end-user through another Java applet that is included in the portal site. The tool greatly improves the user experience with small features such as for example automatically looking for installed credentials on a user hard disk and on plugged in USB sticks.

3.5.2 ACGT specific services

3.5.2.1 Workflow enactor

The Workflow Engine or Enactor is the software component that is responsible for the execution of workflows. In the ACGT platform the workflow enactor is a third party WS-BPEL compliant engine (Apache ODE, <http://ode.apache.org/>).

The workflow enactor is a central component of the architecture in the sense that it needs to interoperate with all the other analytical and bioinformatics services. This is because a "workflow" is a high level program constructed by the user that consists of all the steps that should be executed in order to deliver an output or achieve a larger and sophisticated goal by the composition and orchestration of the existing ACGT services and tools. Therefore communication with the other services through the enactor also validates the interoperability of these services.

The choice of BPEL as the workflow technology, although it offers advantages, such as the standards compliance, maturity, robustness, etc., presented new challenges when incorporated in the Grid-based ACGT architecture. In particular Grid specific technologies such as WSRF and Grid Security Infrastructure (GSI) are not immediately interoperable with the WS-I (<http://www.ws-i.org/>) based BPEL standard. Especially the delegation of credentials that is supported by GSI is of such importance in the ACGT use cases (see requirements for the ACGT Data Protection Framework) that it required the introduction of additional machinery: the so called "Proxy services".

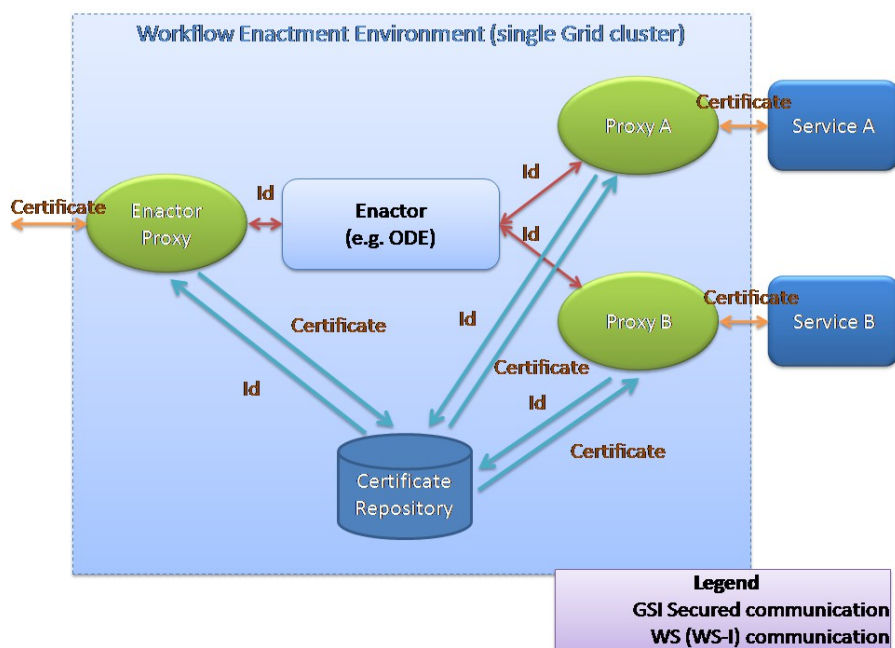


Fig.5. Workflow system architecture

As seen in the image above the enactor does not contact the GSI-enabled ACGT services directly. Instead each "real" ACGT service is mirrored ("proxied") by a corresponding "proxy service" that is confined in a secure workflow environment. The proxy service has the same programmatic interface with the original service and in its implementation it forwards all requests to the service that it mirrors. The BPEL workflows deployed in the enactor do not contain the invocation of the original ACGT services but the invocation of the corresponding proxy services in their places. Therefore the enactor itself is unaware of the whole GRID security framework, it assumes that the services contacted during the workflow execution are the real ones.

This way the Proxy services function as bridges between the BPEL engine and the ACGT Grid services.

3.5.2.2 Metadata Service

The ACGT tool metadata repository handles metadata for the following main tasks:

- Publish (register) tools by service providers.
- Find (discover) tools by service clients.
- Bind (invoke) tools by service clients
- Modify existing tool metadata.
- Retrieve all tool metadata (for metadata browsing tools).

The repository has been implemented in several layers:

- Modular API: This Application Programming Interface (API) integrates different tool repositories and provides discovery/find functionality for tools, data types and functional categories. It uses an access module to integrate with the ACGT repository databases.

- RepoServices API: This API is used internally in the Modular API access layer to the ACGT repository databases. It is described in this section to provide a complete overview of the metadata repository architecture.

3.5.2.3 Semantic Mediator and Mapping Tool Service

The Semantic Mediator is the core component of the ACGT Semantic Mediation layer. It is in charge of accepting queries in terms of the ACGT Master Ontology and translating them into terms of the physical databases included in the integration platform. The Semantic Mediator can be accessed as an OGSA-DAI service, making it available to any terminal connected to the internet. Three different services comprise the Semantic Mediator:

- SemanticMediator: this is the main service offered by the Semantic Mediator (and it is called also as the resource). It offers a SPARQL interface for performing queries in terms of the Master Ontology. The received queries are handled by the Semantic Mediator accordingly to the existing database mappings, so a new query for each -underling data source is produced and their results are properly merged and sent back to the user
- MappingList: this service allows retrieving the content of all mapping files included in the Semantic Mediator. This service is used by the Query Tool.
- updateMappingList: through this service, new mappings can be included in the integration platform. This service is used by the Mapping Tool.

OGSA-DAI services are invoked using perform documents. Each of the three services of the Semantic Mediator requires a different set of arguments, and produces a different type of result. Next subsections depict the details and interface of the three services.

The Mapping Tool plays an important role within the ACGT Semantic Mediation layer, as it offers the possibility of including new data sources in the integration platform. This process involves establishing relations between elements of the schema of the source to be integrated and elements of the ACGT Master Ontology—which acts as global schema for the mediator. These relations are called mappings, and the process of establishing mappings is called mapping process. The mappings are used by the mediator in the task of translating integrated queries into queries for the underlying databases.

The mapping process often requires some degree of expertise on both the domain of the data being mapped and the inner technical characteristics of the mappings being produced. To this end, the Mapping Tool incorporates a series of features aimed at facilitating this task and reducing as much as possible the user's workload. These are listed below:

- Web-based friendly user interface: the interface offers an intuitive approach to all available actions. Access is offered through Internet, avoiding installation requirements at the user terminal
- Extensive and powerful mapping editor: a complete editor allows creating and editing mappings with all necessary detail
- Collaborative environment: the environment offers the possibility of two or more users to concurrently collaborate on the same project

3.5.2.4 GridR Service

The GridR package provides an R programming language interface that supports the access to the ACGT services. This means that R users and developers will have access to

distributed resources in a transparent fashion, as if those resources were local. The complexity of the grid is thus hidden from the user. Accessing the ACGT grid environment requires no changes in the core R implementation. In practice grid access is performed through the call of predefined R functions loaded from a package. R users can make use of the grid technology in a transparent way by initializing the grid environment once (`grid.init`) based on a pre-defined configuration and passing the functions to be executed in the grid as input to one of those predefined functions (`grid.apply`) in their local code.

The GridR service interface supports the execution of user-defined scripts as well as the execution of scripts that had been pre-registered in a repository. In the ACGT platform, the GridR service is implemented as a GSI-secured grid service based on the Globus Toolkit 4 libraries and on the Gridge Toolkit [5]. In detail, the GridR service includes clients to the Gridge data management system (DMS), which is a virtual file system for data organization in grid environments, and to the Gridge grid resource management system (GRMS), which is responsible for grid resource management and scheduling of grid jobs.

The interface to the DMS is based on files; this implies that all input and output data have to be passed to and from GridR by physical files. For this purpose, the GridR service attaches a header to each script which makes the contents of input files accessible in the R session on the execution machine as elements of a predefined R list. The interface for the output is a list of file or directory names that the user can use to export data from the session.

The developer of a parallel GridR script is offered a “directive”-like mechanism for the annotation of the parts of the script that can run in parallel. With the help of these annotations, the GridR service can split the script into parallel or non-parallel sections that are or are not to be run in parallel as grid jobs.

Technically, the GridR service translates the parallelization information into a complex GRMS job description representing the workflow to be performed for the execution of the parallel and non-parallel script sections (e.g., including the dependency of the different script sections or the specification of which files have to be staged in and out), and submits it as GRMS job.

3.5.2.5 Oncosimulator Service

The Oncosimulator is an advanced tool which is able to simulate the response of tumours and affected normal tissues to therapeutic schemes based on clinical, imaging, histopathologic and molecular data of a given cancer patient. It aims at optimizing cancer treatment on a patient-individualized basis by performing *in silico* (on the computer) experiments of candidate therapeutic schemes.

The initial implementation of this application consisted of standalone code that can be run as a batch job on a personal computer or as on hardware resources of a grid infrastructure. The component of ACGT infrastructure responsible for running the application on the grid resources is GRMS (Gridge Resource Management System). There was however the requirement to provide access to the Oncosimulator functionality via a web service interface. The Oncosimulator Service is the components fulfilling this requirements.

3.5.3 Advanced Grid Services

3.5.3.1 Resource Management

The component responsible for resource management within the Gridge Toolkit is GRMS (Gridge Resource Management System). It is an open source meta-scheduling system, which allows developers to build and deploy resource management systems for large scale distributed computing infrastructures. GRMS is based on dynamic resource selection, mapping and an advanced scheduling methodology, combined with a feedback control

architecture, and deals with dynamic grid environment and resource management challenges, such as for example load-balancing among clusters, remote job control or file staging support. 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 which can take advantage of various low-level core services and existing technologies. Finally, GRMS can be considered as a robust system which provides abstraction of the complex grid infrastructure as well as a toolbox which helps to form and adapts to distributing computing environments.

GRMS is a central point for resource and job management activities and tightly cooperates with other services responsible for authorization, monitoring, and data management to fulfil the requirements of the applications. The main features of GRMS are job submission, job control (suspending, resuming, cancelling), the ability to chose "the best" resource for the job execution using a multi-criteria matching algorithm, support for job check-pointing and migration, support for file staging, storing information about the job execution, user notifications support, workflow jobs support etc.

GRMS has been designed as an independent set of components for the resource management processes which can take advantage of various low-level core services as, e.g., GRAM [3], GridFTP [3] and the Gridge Monitoring System, as well as various grid middleware services, e.g., the Gridge Authorization Service (GAS) and the Gridge Data Management Service (DMS). All these services working together provide a consistent, adaptive and robust grid middleware layer which fits dynamically to many different distributing computing infrastructures. The GRMS implementation requires the Globus software [3] to be installed on the grid resources, and uses the following core Globus services deployed on the resources: GRAM, GridFTP, and MDS (optional). GRMS supports the Grid Security Infrastructure by providing GSI-enabled web service interfaces for all clients, e.g. portals or applications, and thus can be integrated with any other compliant grid middleware.

One of the main featurres for GRMS is that it perform remote job control and management in the way that satisfies users (job owners) and their applications requirements. All user requirements are expressed within an XML-based resource specification document and sent to GRMS as SOAP requests over GSI transport layer connections. Simultaneously, resource administrators (resource owners) have full control over owned resources on which the jobs and operations will be performed by an appropriate GRMS setup and installation. GRMS together with the core services reduces operational and integration costs for administrators by enabling grid deployment across heterogeneous (and maybe previously incompatible) cluster and resources. Technically speaking, GRMS is a persistent service within a Tomcat/Axis container. It is written completely in Java so it can easily be deployed on various platforms.

3.5.3.2 Data Management

Data storage, management and access in the Gridge environment are supported by the Gridge Data Management Suite (DMS). This suite, composed of several specialized components, allows building a distributed system of services capable of delivering mechanisms for seamless management of large amounts of data. It is based on the autonomic agents pattern using the accessible network infrastructure for communication. From the point of view of external applications, DMS is a virtual file system keeping the data organized in a tree-like structure. The main units of this structure are meta-directories, which allow creating a hierarchy over other objects and metafiles. Metafiles represent a logical view of data regardless of their physical storage location.

The Data Management System consists of three logical layers: the Data Broker, which serves as the access interface to the DMS system and implements the brokering of storage

resources, the Metadata Repository that keeps information about the data managed by the system, and the Data Container, which is responsible for the physical storage of data. In addition, DMS contains modules which extend its functionality to fulfil common enterprise requirements. These include the fully functional web based administrator interface and a Proxy to external scientific databases. The Proxy provides a SOAP interface to the external databases, such as for example those provided by SRS (Sequence Retrieval System).

The Data Broker is designed as an access point to the data resources and data management services. A simple API of the Data Broker allows to easily access the functionality of the services and the stored data. The Data Broker acts as a mediator in the flow of all requests coming from external services, analyses them and eventually passes them to the relevant module. The DMS architecture assumes that multiple instances of the Data Broker can be deployed in the same environment, thus increasing the efficiency of data access from various points in the global Grid environment structure.

The Metadata Repository is the central element of the Gridge distributed data management solution. It is responsible for all metadata operations as well as their storage and maintenance. It manages metadata connected with the data files, their physical locations and transfer protocols that could be used to obtain them, with the access rights to the stored data and with the metadescriptions of the file contents. Currently each DMS installation must contain a single instance of the Metadata Repository, which acts as a central repository of the critical information about the metacatalogue structure, user data and security policy for the whole DMS installation.

The Data Container is a service specialized in the management of physical data locations on the storage resources. The Data Container API is designed in such a way that it allows easy construction and participation in the distributed data management environment of storage containers for different storage environments. The Data Containers currently available in the DMS suite include a generic file system Data Container, a relational database Data Container and a tape archiver Data Container. The data stored on the various storage resources can be accessed with one of the many available protocols including such as GASS, FTP and GridFTP.

The Proxy modules are services that join the functionality of the Metadata Repository allowing to list the available databanks, list their content, read the attached metadata attributes and to build and execute queries, and of the Data Container to provide the data using the selected data transfer protocol. Such Proxy container are highly customized towards the specific platform they are working with to allow building complex queries and executing operations on the found entries.

3.5.3.3 Security Services

One of the primary elements of the ACGT security infrastructure is the authorization service called GAS. The Gridge Authorization Service (GAS) is an authorization system which can act as the standard decision point for all components of a system. Security policies for all system components are stored in GAS. Using these policies GAS can return an authorization decision upon the client request. GAS has been designed in such a way that it is easy to perform integration with external components and it is easy to manage security policies for complex systems. The possibility to integrate with the Globus Toolkit and many operating system components makes GAS an attractive solution for grid applications.

In general, an authorization service can be used for returning an authorization decision upon the user request. The request has to be described by three attributes: user, object and operation. The requester simply asks if the specific user can perform the operation on the specific object. Obviously, the query to an authorization service can be more complex and the answer given by such service can be complicated, too. One of the services which can work in such scenario is the Gridge Authorization Service (GAS). GAS has been designed in

a form which enables many possible applications. GAS can communicate in many ways with other components. By using the modular structure of GAS it is easy to write a completely new communication module. The GAS complex data structure can be used to model many abstract and real world objects and security policies for such objects. For example, GAS has been used for managing security policies: for many Virtual Organizations, for services (like Grid Resource Management Service, Mobile Services and other) and for abstract objects like communicator conferences or computational centres. These and many other features give a possibility to integrate GAS with many existing solutions. Such integration can be very important, because it raises the security level of the existing solutions and makes it possible to use the newest security technologies.

The main goal of GAS is to provide a functionality that would be able to fulfil most authorization requirements of grid computing environments. GAS is designed as a trusted single logical point for defining security policies for complex grid infrastructures. As flexibility is the key requirement, it is to be able to implement various security scenarios, based on push or pull models, simultaneously. Secondly, GAS is considered as independent of specific technologies used at lower layers, and it should be fully usable in environments based on grid toolkits as well as other toolkits. The high level of flexibility is achieved mainly through the modular design of GAS and usage of a complex data structure which can model many scenarios and objects from the real world. It means that GAS can use many different ways for communication with external components and systems, use many security data models and hold security policy on different types of storage systems. These features make GAS attractive for many applications and solutions (not only for those related with grids). GAS has to be the trusted component of each system in which it is used and it brings about that the implementation of GAS was written in ANSI C. This choice makes GAS a very fast and stable component which uses not much CPU power and little amount of memory. The main problem of many authorization systems is their management. It is not easy to work with a complex system in a user-friendly way. Based on many experiences and the end user comments together with GAS, the GAS administration portlet (web application) is provided, which makes management as easy as possible. Flexibility of this solution gives users a full possibility of presenting only these security policies which are important for them.

3.5.4 Common Grid Services

3.5.4.1 GRAM

Grid Resource Allocation Management (GRAM) is a component of the Globus Toolkit responsible for job execution. Grid computing resources are typically operated under the control of a scheduler which implements allocation and prioritization policies while optimizing the execution of all submitted jobs for efficiency and performance. GRAM is not a resource scheduler, but rather a protocol engine for communicating with a range of different local resource schedulers using a standard message format. Rather than consisting of a monolithic solution, GRAM is based on a component architecture at both the protocol and software implementation levels. This component approach serves as an ideal which shapes the implementation as well as the abstract design and features.

There are two implementations of the GRAM service, there is the "web services and "pre-web services" Unix server suite to submit, monitor, and cancel jobs on Grid computing resources. Both systems are known under the moniker "GRAM", while "WS GRAM" refers only to the web service implementation.

Job management with GRAM makes use of multiple types of service:

- Job management services represent, monitor, and control the overall job life cycle. These services are the job-management specific software provided by the GRAM solution.

- File transfer services support staging of files into and out of compute resources. GRAM makes use of these existing services rather than providing redundant solutions; WS GRAM has further refactored some file transfer mechanisms present in pre-web service GRAM.
- Credential management services are used to control the delegation of rights among distributed elements of the GRAM architecture based on users' application requirements. Again, GRAM makes use of more general infrastructure rather than providing a redundant solution, and WS GRAM has continued this refactoring to better separate credential management at the protocol level.

For WS GRAM, the Globus Toolkit software development environment, and particularly WSRF core, is used to implement distributed communications and service state. For pre-web service GRAM, the "gatekeeper" daemon and GSI library are used for communications and service dispatch.

WS GRAM utilizes WSRF functionality to provide for authentication of job management requests as well as to protect job requests from malicious interference, while pre-web service GRAM uses GSI and secure sockets directly. The use of GRAM does not reduce the ability for system administrators to control access to their computing resources. The use of GRAM also does not reduce the safety of jobs users run on a given computing resource. To protect users from each other, jobs are executed in appropriate local security contexts, e.g. under specific Unix user IDs based on details of the job request and authorization policies. Additionally, GRAM mechanisms used to interact with the local resource are design to minimize the privileges required and to minimize the risks of service malfunction or compromise. A client may delegate some of its rights to GRAM services in order to facilitate the above functions, e.g. rights for GRAM to access data on a remote storage element as part of the job execution.

Additionally, the client may delegate rights for use by the job process itself. With pre-web service GRAM, these two uses of rights are inseparable, while WS GRAM provides separate control for each purpose (while still allowing rights to be shared if that is desired). WS GRAM provides an "at most once" job submission semantics. A client is able to check for and possibly resubmit jobs, in order to account for transient communication errors without risk of running more than one copy of the job. Similarly, pre-web service GRAM provides a two-phase submission mechanism to submit and then commit a job to be run.

While many jobs are allowed to run to their natural completion, GRAM provides a mechanism for clients to cancel (abort) their jobs at any point in the job life cycle. WS GRAM provides for reliable, high-performance transfers of files between the compute resource and external (GridFTP) data storage elements before and after the job execution. Pre-web service GRAM can also stage with GridFTP systems but with less flexible reliable-transfer logic driving its requests. GRAM supports a mechanism for incrementally transferring output file contents from the computation resource while the job is running. WS GRAM uses a new mechanism to allow arbitrary numbers of files to be transferred in this fashion, while pre-web service GRAM only supports incremental transfer of the job's standard output and error streams.

3.5.4.2 MDS

The Monitoring and Discovery System (MDS) is a suite of web services to monitor and discover resources and services on Grids. This system allows users to discover what resources are considered part of a Virtual Organization (VO) and to monitor those resources. MDS services provide query and subscription interfaces to arbitrarily detailed resource data and a trigger interface that can be configured to take action when pre-configured trouble conditions are met. The services included in the WS MDS implementation (MDS4), provided with the Globus Toolkit 4, acquire their information through an extensible interface which can be used to: query WSRF services for resource property information, execute a program to acquire data, or interface with third-party monitoring systems.

Grid computing resources and services can advertise a large amount of data for many different use cases. MDS4 was specifically designed to address the needs of a Grid monitoring system – one that publishes data that is available to multiple people at multiple sites. As such, it is not an event handling system, like NetLogger, or a cluster monitor on its own, but can interface to more detailed monitoring systems and archives, and can publish summary data using standard interfaces.

MDS4 includes two WSRF-based services: an Index Service, which collects data from various sources and provides a query/subscription interface to that data, and a Trigger Service, which collects data from various sources and can be configured to take action based on that data. The Index Service is a registry similar to UDDI, but much more flexible. Indexes collect information and publish that information as resource properties. Clients use the standard WSRF resource property query and subscription/notification interfaces to retrieve information from an Index. Indexes can register to each other in a hierarchical fashion in order to aggregate data at several levels. Indexes are “self-cleaning”; each Index entry has a lifetime and will be removed from the Index if it is not refreshed before it expires. Each Globus container that has MDS4 installed will automatically have a default Index Service instance.

By default, any GRAM, Reliable File Transfer Service (RFT) or Community Authorization Service (CAS) service running in that container will register itself to the container’s default Index Service. The Trigger Service collects information and compares that data against a set of conditions defined in a configuration file. When a condition is met, or triggered, an action takes place, such as emailing a system administrator when the disk space on a server reaches a threshold.

In addition to the services described above, MDS4 includes several additional software components, including an Aggregator Framework, which provides a unified mechanism used by the Index and Trigger services to collect data. The Aggregator Framework is a software framework used to build services that collect and aggregate data. Services (such as the Index and Trigger services) built on the Aggregator Framework are sometimes called aggregator services, and have the following in common:

- They collect information via Aggregator Sources. An Aggregator Source is a Java class that implements an interface (defined as part of the Aggregator Framework) to collect XML-formatted data
- They use a common configuration mechanism to maintain information about which Aggregator Source to use and its associated parameters (which generally specify what data to get, and from where). The Aggregator Framework WSDL defines an “aggregating service group entry type” that holds both configuration information and data. Administrative client programs use standard “WSRF Service Group registration mechanisms” to register these service group entries to the aggregator service.
- They are self-cleaning – each registration has a lifetime; if a registration expires without being refreshed, it and its associated data are removed from the server.

MDS4 includes the following three Aggregator Sources:

- the Query Aggregator Source, which polls a WSRF service for resource property information,
- the Subscription Aggregator Source, which collect data from a WSRF service via WSRF subscription/notification,
- the Execution Aggregator Source, which executes an administrator-supplied program to collect information.

Depending on the implementation, an Aggregator Source may use an external software component (for example, the Execution Aggregator Source uses an executable program), or a WSRF service may use an external component to create and update its resource

properties (which may then be registered to an Index or other aggregator service, using the Query or Subscription Aggregator Source). This set of components are referred to as Information Providers. Currently, MDS4 includes the following sources of information: Hawkeye Information Provider, Ganglia Information Provider, WS GRAM, Reliable File Transfer Service (RFT), Community Authorization Service (CAS) , any other WSRF service that publishes resource properties.

3.5.4.3 GridFTP

GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet file transfer protocol. A set of protocol features and extensions defined already in IETF RFCs was selected and added a few additional features to meet requirements from current data grid projects.

GridFTP functionality includes some features that are supported by FTP extensions that have already been standardized (RFC 959) but are seldom implemented in current systems. Other features are new extensions to FTP.

- Grid Security Infrastructure (GSI) and Kerberos support: Robust and flexible authentication, integrity, and confidentiality features are critical when transferring or accessing files. GridFTP must support GSI and Kerberos authentication, with user controlled setting of various levels of data integrity and/or confidentiality. GridFTP implements the authentication mechanisms defined by RFC 2228, "FTP Security Extensions".
- Third-party control of data transfer: To manage large datasets for distributed communities, authenticated third-party control of data transfers between storage servers must be provided. A third-party operation allows a user or application at one site to initiate, monitor and control a data transfer operation between two other sites: the source and destination for the data transfer. The GridFTP based implementation adds Generic Security Services (GSS)-API authentication to the existing third-party transfer capability defined in the FTP standard.
- Parallel data transfer: On wide-area links, using multiple TCP streams in parallel (even between the same source and destination) can improve aggregate bandwidth over using a single TCP stream. GridFTP supports parallel data transfer through FTP command extensions and data channel extensions.
- Striped data transfer: Data may be striped or interleaved across multiple servers, as in a DPSS network disk cache. GridFTP includes extensions that initiate striped transfers, which use multiple TCP streams to transfer data that is partitioned among multiple servers. Striped transfers provide further bandwidth improvements over those achieved with parallel transfers. The GridFTP protocol extensions was defined to support striped data transfers.
- Partial file transfer: Some applications can benefit from transferring portions of files rather than complete files: for example, high-energy physics analyses that require access to relatively small subsets of massive, object-oriented physics database files. The best that the standard FTP protocol allows is transfer of the remainder of a file starting at a particular offset. GridFTP provides commands to support transfers of arbitrary subsets or regions of a file.
- Automatic negotiation of TCP buffer/window sizes: Using optimal settings for TCP buffer/ window sizes can dramatically improve data transfer performance. However, manually setting TCP buffer/window sizes is an error-prone process (particularly for nonexperts) and is often simply not done. GridFTP extends the standard FTP command set and data channel protocol to support both manual setting and

automatic negotiation of TCP buffer sizes for large files and for large sets of small files.

- Support for reliable and restartable data transfer: Reliable transfer is important for many applications that manage data. Fault recovery methods are needed to handle failures such as transient network and server outages. The FTP standard includes basic features for restarting failed transfers that are not widely implemented. GridFTP exploits these features and extends them to cover the new data channel protocol.

3.5.4.4 Security Services

- PKI Services

As previously mentioned, ACGT security relies on PKI. The ACGT PKI is a commercial grade PKI implementation (with respect to the followed security practices) which follows the X.509 standards and is composed of several interdependent modules. The service is not specific for GRID infrastructures, but rather supportive to the Common Grid Infrastructure.

The Certificate Authority (CA) module is the central component that issues and signs certificates for end-users and services. The ACGT CA is based on the well known EJBCA (<http://www.ejbca.org>) software. It is not directly accessible by end-users, but is used by the end-user administration site (Registration Authority front-end discussed earlier) and other PKI services.

The OCSP (Online Certificate Status Protocol) service and CRL (Certificate Revocation List) distribution point are services that allow checking of the revocation status of certificates. This information is essential in restricting access on ACGT resources to authorized users (and services) only. The OCSP service allows for real-time revocation status checking for each certificate that is encountered during credential validation while a CRL is valid for a longer time.

- MyProxy

Complex and long running tasks require that the end-users are able to delegate credentials to software agents (such as for example the workflow enactor), so that these agents can act on behalf of the user while he is offline. Delegation means that a user “transfers” his access rights (typically for a restricted period of time) to another actor (service).

Delegation in ACGT is mainly provided through X.509 proxy credentials. Proxy credentials are basically certificates signed by the end-user’s certificate instead of a dedicated Certificate Authority. By issuing such a certificate the end-user delegates his rights to a specific service or person.

MyProxy is an online credential repository supporting this form of delegation. A dedicated ACGT MyProxy service has been deployed and configured to allow only certificates generated by ACGT approved CAs. Although delegation in ACGT is not restricted to Myproxy based delegation (cf. deliverable D11.2), the delegation is always bootstrapped by the MyProxy service at the portal level.

3.6 Deployment view

the deployment view of the architecture focuses on the actual instances of components installed and used in the testbed and all the possibilities of deployment of different components.

It is possible to classify the types of services with regards to deployment requirements:

- Components deployed on each machine of the testbed
- Collective infrastructure services
- ACGT specific services
- User interface applications

3.6.1 Components deployed on each machine of the testbed

The components of this kind are mostly responsible for remote access to physical resources (computational nodes or databases). The most important part of the infrastructure providing access to resources is the Globus Toolkit. It provides mechanisms for remote application submission, monitoring of the resource availability and its parameters, transferring files, etc.

The most important software entities installed on computational resources are:

- GRAM: This Globus Toolkit component is responsible for applications submission and management. GRAM can be configured to run an executable via a simple "fork" mechanism but the recommended configuration for the ACGT grid infrastructure is to interface the local queueing system. The reason for that is the following: the grid resources are not dedicated to the ACGT testbed alone, so it is very important to keep the local resource management policy consistent. The local queueing system deployed on a machine manages usage of the node by different external entities or organizations and keeps the availability of the computational power on a proper level, preventing overloading on the one hand and task starvation on the other.

For the resources joining the ACGT testbed, if there is no queueing system already installed we have recommended the OpenPBS/Torque solution, and provide all required support for it.

- GridFTP: Although GridFTP is independent from the Globus Toolkit, it is compliant with it in a context of technology. It is required to deploy GridFTP server on each node used for computation in ACGT testbed. The ACGT metascheduler is using it to transfer all files required for computation. It is not possible to submit any remote application execution without GridFTP installed on a computational node.

- Grid monitoring agents: these agents are mostly part of the Globus Toolkit and are responsible for providing all available information about the status of a computational node: hardware parameters (number of processors, processors type, operating system etc), dynamic parameters (processors performance, load of the machine, number of tasks in a local queues), software components deployed (applications, libraries).

That information is further used by the metascheduler in a process of mapping tasks to resources on a middleware level (grid scheduling).

- GAS plugins: components developed by PSNC that have to reside on GRAM and GridFTP on each computational node. They are responsible for authorization procedures and for virtual account management.

Before anybody can use any resource the security policy stored in the Grid Authorization Service (GAS) must be checked (authorisation). This is what the plugins are responsible for: they are calling GAS and ask, if a given person is allowed to submit a task or transfer data to the resource. The GAS plugins support the "virtual account" mechanism, which is a convenient way of accessing resources in a Virtual Organization. The idea is based on policies stored in a central authorization service. Based on that policy there is a pool of accounts created and managed by a service in a dynamic way. When a person accesses a resource, and is allowed to do so, he is (temporarily) mapped to one of virtual accounts on a local machine.

- Data Wrappers: The data wrappers are used for accessing different databases that can be used by more advanced mediation tools and distributed queering mechanisms.

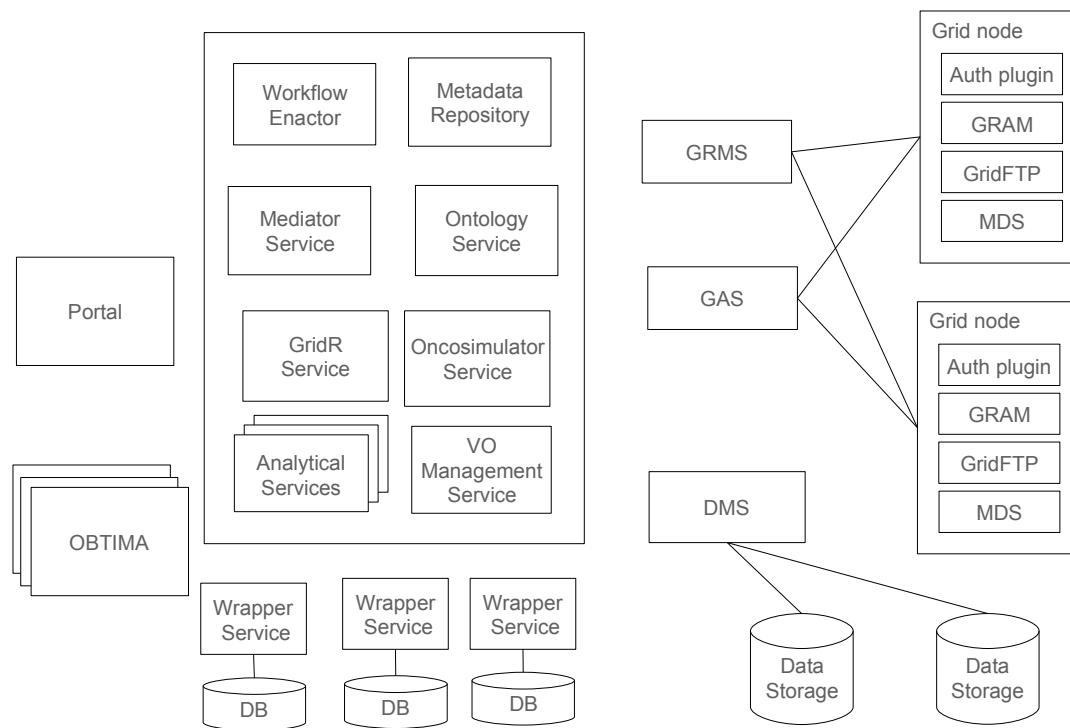


Fig.6. Deployment view of the ACGT architecture.

3.6.2 Collective infrastructure services

This category of components are building the advanced middleware of grid infrastructure. They are responsible for providing a unified interface for resource management (GRMS), data management (DMS), authorization (GAS), etc. The collective services are deployed as central management points in the architecture, meaning one instance of each service for the whole testbed. The reason of having a single instance of these services is the ability to manage the underlying resources. GRMS is able to perform load balancing in the grid environment. DMS deals with the files that can be accessed by different clients, and from different locations. The authorization service keeps the overall policy of accessing different resources in the grid.

The collective services are single points of failure of the infrastructure. There are some ways to decrease the risk associated with this. For the GRMS it is possible to duplicate the deployed service. A backup installation take over when there are problems with main service. Of course this requires implementation of additional logic on the client level - user level tools should be able to switch between different instances of GRMS in a dynamic way.

DMS can provide similar duplication mechanism as GRMS does. In addition to that, it is designed to be able to exploit different data storages that can be located in a distributed locations.

Risks associated with GAS (or the security infrastructure as a whole) are discussed in deliverable D10.8 Risk analysis concerning the data security and data protection framework.

3.6.3 ACGT specific services

There is no limitation to the number of instances of ACGT services in the environment. Each of the service needs to be registered in the repository and can then be used by the end user. It does not matter if the client application invokes instance A or B (running at different locations) of the same functional component. It is thus possible to have a distributed network of services running near the (physical) location of the end user to provide maximum response time for the client.

3.6.4 User interface applications

For the user interface application the number of instance does not matter. Standalone tools can be used by the individual users separately. The portal client is a little different case. Single installation can be used by multiple users, so it is possible to have one instance of portal for the whole environment. But it is possible to have the portal install on multiple sites.

However there is requirement for some portlets to be the only one instance in the testbed. It concerns administration portlets: VO management, GAS administrator portlet, DMS administration portlet.

4 Summary

Within this deliverable an overview of the final architecture of the integrated Grid enabled technological platform which forms ACGT was given.

The approach and principles behind the architect design have been explained. The core components of the ACGT infrastructure have been described in the context of the ACGT layered architecture. And, finally, their deployment within the ACGT testbed has been indicated.

References

- [1] Globus Toolkit <http://www.globus.org>
- [2] Gridge Toolkit <http://www.gridge.org>
- [3] ACGT D4.1 Prototype and report of the ACGT GRID layer
- [4] <http://www.globus.org/toolkit/docs/4.0/>
- [5] <http://www.globus.org/toolkit/docs/4.0/security/key-index.html>
- [6] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems", IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006
- [7] J. Pukacki, M. Kosiedowski, R. Mikołajczak, M. Adamski, P. Grabowski, M. Jankowski, M. Kupczyk, C. Mazurek, N. Meyer, J. Nabrzyski, T. Piontek, M. Russell, M. Stroiński, M. Wolski "Programming Grid Applications with Gridge", Computational Methods in Science and Technology vol. 12, Poznan 2006.
- [8] D. Wegener, T. Sengstag, S. Sfakianakis, S. Rüping, "Supporting parallel R code in clinical trials: a grid-based approach". Accepted for publication at the HiPGCoMB 2008 workshop

Appendix A - Abbreviations and acronyms

<i>API</i>	Application Programming Interface
<i>CA</i>	Certification Authority
<i>CAT</i>	Custodix Anonymisation Tool
<i>CRF</i>	<i>Case Report Form</i>
<i>FTP</i>	File Transfer Protocol
<i>GAS</i>	Gridge Authorization Service
<i>GASS</i>	Global Access to Secondary Storage
<i>GDMS</i>	Gridge Data Management System
<i>GRMS</i>	Gridge Resource Management System
<i>GSI</i>	Globus Security Infrastructure
<i>MDS</i>	Monitoring & Discovery Service
<i>OBTIMA</i>	Ontology based Trial Management System for ACGT
<i>OGSA-DAI</i>	Open Grid Service Architecture-Data Access and Integration
<i>RA</i>	Registration Authority
<i>RFT</i>	Reliable File Transfer
<i>SOA</i>	Service Oriented Architecture
<i>SPARQL</i>	SPARQL Protocol and RDF Query Language
<i>VO</i>	Virtual Organisation
<i>WSDL</i>	Web Service Definition Language