A Framework Based on Web Services and Grid Technologies for Medical Image Registration

Ignacio Blanquer[1], Vicente Hernández[1], Ferran Mas[1], and Damià Segrelles[2]

[1] Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera S/N, 46022 Valencia, Spain
{iblanque,vhernand,fmas}@dsic.upv.es

[2] Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas, Universidad Politécnica de Valencia, Camino de Vera S/N, 46022 Valencia, Spain
dquilis@itaca.upv.es

Abstract. Medical Imaging implies executing complex post-processing tasks such as segmentation, rendering or registration which requires resources that exceeds the capabilities of conventional systems. The usage of Grid Technologies can be an efficient solution, increasing the production time of shared resources. However, the difficulties on the use of Grid technologies have reduced its spreading outside of the scientific arena.
This article tackles the problem of using Grid Technologies for the co-registration of a series of volumetric medical images. The co-registration of time series of images is a needed pre-processing task when analysing the evolution of the diffusion of contrast agents. This processing requires large computational resources and cannot be tackled efficiently on an individual basis. This article proposes and implements a four-level software architecture that provides a simple interface to the user and deals transparently with the complexity of Grid environment. The four layers implemented are: Grid Layer (the closest to the Grid infrastructure), the Gate-to-Grid (which transforms the user requests to Grid operations), the Web Services layer (which provides a simple, standard and ubiquitous interface to the user) and the application layer.
An application has been developed on top of this architecture to manage the execution of multi-parametric groups of co-registration actions on a large set of medical images. The execution has been performed on the EGEE Grid infrastructure. The application is platform-independent and can be used from any computer without special requirements.

1 Introduction and State of the Art

The current use of Internet as main infrastructure for the integration of information through web based protocols (i.e. HTTP, HTTPS, FTP, FTPS, etc.), opened the door to new possibilities. The Web Services (WS) [1] are one of the most consolidated technologies in web environments, fundamented on the Web Services Description Language (WSDL). WSDL defines the interface and constitutes a key part of the Universal Description Discovery and Integration

(UDDI) [2], which is an initiative for providing directories and descriptions of services for e-business. WSs communicate through the Simple Object Access Protocol (SOAP) [3], a simple and decentralized mechanism for the exchange of typed information structured in XML (Extended Mark-up Language).

As is defined in [4] a Grid provides an abstraction of resources for sharing and collaborating through different administrative domains. These resources can be hardware, data, software and frameworks. The key concept of Grid is the Virtual Organization (VO) [5]. A VO is defined as temporal or permanent set of entities, groups or organizations that provide or use resources. The usage of Grid Computing is currently in expansion. This technology is being introduced in many different application areas such as Biocomputing, Finances or Image Processing, along with the consolidation of its use in more traditional areas such as High Energy Physics or Geosciences.

In this process of development, many basic middlewares such as the different versions of Globus Toolkit [6] (GT2, GT3, GT4), Unicore [7] or InnerGrid [8] have arisen. At present, Grid technologies are converging towards Web Services technologies. The Open Grid Services Architecture (OGSA) [9] represents an evolution in this direction. OGSA seems to be an adequate environment for obtaining efficient and interoperable Grid solutions, some issues (such as the security) still need to be improved. Globus GT3 implemented OGSI (Open Grid Service Infrastructure) which was the first implementation of OGSA. OGSI was deprecated and substituted by the implementation of OGSA by the Web Services Resource Framework (WSRF) [10] in GT4. WSRF is totally based in WSs.

Although there exist newer versions, Globus GT2 is a well-stablished batch basic Grid platform which has been extended in several projects in a different way in which GT3 and GT4 have evolved. The DATAGRID project [11], developed the EDG (European Data Grid) , a Middleware based on GT2, which improved the support of distributed storage, VO management, job planning and job submission. The EDG middleware has been improved and extended in the LCG (Large Hadron Collider Computing Grid) [12] and Alien Projects to fulfil the requirements of the High Energy Physics community. Another evolution of the EDG is gLite [13], a Grid Middleware based in WS and developed in the frame of the Enabling Grids for E-sciencE (EGEE) [14]. gLite has extended the functionality and improved the performance of critical resources, such as the security, integrating the Virtual Organisation Membership System (VOMS) [15] for the management of VOs. VOMS provides information on the user's relationship with his/her Virtual Organization defining groups, roles and capabilities.

These middlewares have been used to deploy Grid infrastructures comprising thousands of resources, increasing the complexity on the usage of the Grid. However, the maturity of these infrastructures in terms of user-friendliness is not sufficient yet. Configuration and maintenance of the services and resources or fault tolerance is hard even for experimented users. Programming Grid applications usually involve a non-trivial degree of knowledge of the intrinsic structure of the Grid. This article presents a software architecture that abstracts the users from the management of Grid environments by providing a set of simple services.

Although the architecture proposed is open to different problems, this article shows the use for the implementation of an application for the co-registration of medical images. This application is oriented to either medical end-users and researchers for the execution of large numbers of runs using different values for the parameters that control the process. Researchers can tune-up the algorithms by executing larger sets of runs, whereas medical users can obtain the results without requiring powerful computers.

The application does not require a deep knowledge of the Grid environments. It offers a high-level user-friendly interface to upload data, submit jobs and download results without requiring to know the syntax of commands, Job Description Language (JDL) data and resource administration or security issues.

2 Motivation and Objectives

The final objective is to provide the users with a tool that could ease the process of selecting the best combination of parameters that produces the best results for the co-registration, and to provide high-quality co-registered images from the initial set of data. This tool will use the Grid as a source of computing power and will offer a simply and user-friendly interface to deal with.

The tool enables the execution of large sets of co-registration actions varying the values of the different parameters, easing the process of transferring the source data and the results. Since Grid concept is mainly batch (and the co-registration is not an interactive process due to its long duration), it must provide with a simply way to monitor the status of the processing. Finally the process must be achieved in the shorter time possible, considering the resources available.

2.1 Pharmacokinetic Modelling

The pharmacokinetic modelling of the images obtained after a quick administration of a bolus of extracellular gadolinium chelates contrast can have a deep impact on the diagnosis and the evaluation of different pathogen entities.

Pharmacokinetic models are designed to forecast the evolution of an endogenous or exogenous component on the tissues. To follow-up the evolution of the contrast agent a sequence of MRI volumetric images is obtained at different times following the injection of contrast. Each of these images comprises a series of image slices that cover the body part explored. Since the whole process takes a few minutes, images are obtained in different break-hold periods. This movement of the patient produces artefacts that make images directly incomparable. This fact is even more important in the area of the abdomen, which is strongly affected by the breathing and the motility of the organs.

The study of pharmacokinetic models for the analysis of hepatic tumours is an outstanding example of the above. A prerequisite for the computation of the parameters that govern the model is the reduction of the deformation of the organs in the different images obtained. This process can be performed by co-registering all the volumetric images with respect to the first one.

2.2 Co-registration

The co-registration of images consists on aligning the voxels of two or more images in the same geometrical space by using the necessary transformations to make the floating images as much as possible similar to the reference image.

In general terms, the registration process could be rigid or deformable. Rigid registration only uses affine transformations (displacements, rotations, scaling) to the floating images. Deformable registration enables the use of elastic deformations on the floating images. Rigid registration introduces fewer artefacts, but it can only be used when dealing with body parts in which the level of internal deformation is lower (e.g. the head). Deformable registration could introduce unrealistic artefacts, but is the only one that could compensate the deformation of elastic organs (e.g. in the abdomen).

Image registration can be applied in 2D (individually to each slice) or in 3D. Registration in 3D is necessary when the deformation happens in the three axes.

2.3 Post-processing

Although the co-registration of images is a computationally complex process which must be performed before the analysis of the images, it is not the only task that needs high performance computing. Extracting the parameters that define the model and computing the transfer rates for each voxel in the space will require large computing resources. The platform implemented has been designed to cope with following post-processing in the same way.

3 Architecture

The basic Grid middleware used in this architecture is the LCG, developed in the LHC Computing Grid Project, which has a good support for high throughtput executions. A four-layered architecture has been developed to abstract the operation of this middleware. The registration application has been implemented on top of this architecture.

Medical data is prone to abuse and need careful treatment in terms of security and privacy. This is even more important when the data has to flow from different sites. It is crucial both to preserve the privacy of the patients and to ensure that people accessing the information are authorised to do so. The way in which this environment guarantees the security is explained in detail in Section 3.2.

3.1 Layers

As it has been mentioned in previous sections, the development of this framework has been structured into four layers, thus providing a higher level of independence and abstraction from the specificities of the Grid and the resources. The following sections describes the technology and the implementation of each layer. Figure 1 shows the layers of the proposed architecture.

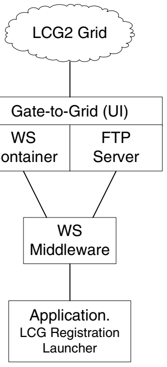


Fig. 1. The proposed architecture

Grid Layer. The system developed in this work makes use of the computational and storage resources being deployed in the EGEE infrastructure along a large number of computing centres distributed among different countries. EGEE currently uses LCG, although there are plans for migrating to gLite. This layer offers the "single computer" vision of the Grid through the storage catalogues and workload management services that tackle with the problem of selecting the rightmost resource. Figure 2 shows a sample LCG2 Grid structure.

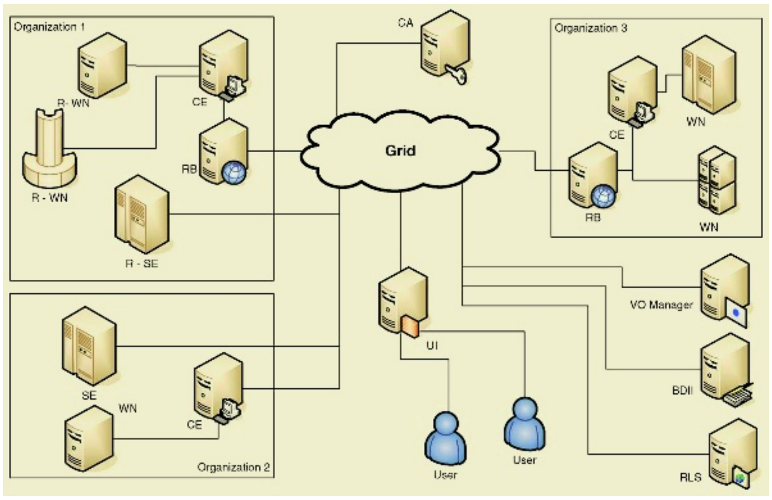


Fig. 2. A LCG2 Grid sample scheme

The LCG2 Grid middleware comprises the following elements:

- IS-BDII: Information Service - Berkeley Database Information Index, this element provides the information about the Grid resources and their status.
- CA: Certification Authority, the CA signs the certificates from both resources and users.

- CE: Computing Element, it is defined as a queue of Grid jobs. A Computing Element is a farm of homogeneous computing nodes called Worker Nodes.
- WN: Worker Node, is a computer in charge of executing jobs.
- SE: Storage Element, a SE is a storage resource in which a task can store data to be used by the computers of the Grid.
- RC, RLS: Replica Catalogue, Replica Location Service, are the elements that manage the location of the Grid data.
- RB: Resource Broker, the RB performs the load balancing of the jobs in the Grid, deciding in which CEs the jobs will be launched.
- UI: User Interface, this component is the entry point of the users to the Grid and provides a set of commands and APIs that can be used by the programs to perform different actions on the Grid.

The Job Description Language (JDL) is the way in which jobs are described in LCG. A JDL is a text file specifying the executable, the program parameters, the files involved in the processing and other additional requirements.

A description of the four layers of the architecture is provided along the following subsections.

Gate-to-Grid Layer. The Gate-to-Grid Layer constitutes the meeting point between the Grid and the Web environment that is intended to be used as user interface. In this layer there are WSs providing the interaction with the Grid similarly as if the user were directly logged in the UI. The WSs are deployed in a Web container in the UI which provides this mediation.

The use of the Grid is performed through the UI by a set of scripts and programs which have been developed to ease the task of launching executions and managing group of jobs. The steps required to execute a new set of jobs in the Grid are:

1. A unique directory is created for each parametric execution. This directory has separated folders to store the received images to be co-registered, the JDL files generated and the output files retrieved from the jobs. It also includes several files with information about the jobs, such as job identifiers and parameters of the registration process.
2. Files to be registered are copied to a specific location in this directory.
3. For each combination of parameters and pair of volumes to be registered, a JDL file filled-in with the appropriate values is generated.
4. Files needed by the registration process are copied in the SE and registered in the RC and the RLS.
5. The jobs are submitted to the Grid through an RB that selects the best available CE according to a predefined criteria.
6. Finally, when the job is done and retrieved, folders and temporal files are removed from the UI. The files registered in the SE that are no longer needed are also deleted.

The different Grid Services are offered through the scripts and programs aforementioned. These programs work with the UI instructions in order to ease the tasks for job and data management. The access to these programs and scripts is remotely available through the WSs deployed in the UI. The copying of the input files from the computer where the user is located to the UI computer is performed through FTP (File Transfer Protocol).

The most important WSs offered in the UI are:

InitSession. This service is in charge of creating the proxy from the Grid user certificates. The proxy is then used in the Grid environment as a credential of that user, providing a single sign-on for the access to all the resources.

GetNewPathExecution. As described before, jobs launched in a parametric execution (and not yet cleared) will have in the UI their own group folder. This folder has to be unique for each group of jobs. This service will get a unique name for each job group and it will create the directory tree to manage that job execution. This directory will store the image, logs, JDLs and other information files.

Submit. The submit call starts an action that carries on the registration of the files from the UI to the SE, creates the JDLs according to the given registration parameters and the files stored on the specified directory of the UI. It finally submits the jobs to the Grid using the generated JDL files.

GetInformationJobs. This service gets information about the jobs belonging to the same execution group. The information retrieved by this call is an XML document with the job identifiers and the associated parameters.

CancelJob. This call cancels a single job (part of an execution group). The cancellation of a job implies the removal of the registered files on the Grid.

CancelGroup. This service performs the cancellation of all the jobs launched to the Grid from a group of parametric executions. As in the case of CancelJob, the cancellation of jobs implies the removal of its associated files from the SEs. Moreover, in this case the temporal directory created on the UI is also removed when all the jobs are cancelled.

GetJobStatus. This service informs about the status of a job in the Grid, given the job identifier. The normal sequence of states of a job is: submitted, waiting, ready, scheduled, running, done and cleared. Other possible states are aborted and cancelled.

PrepareResults. This service is used to prepare the results of an execution before downloading them. When a job finishes, the resulting image, the standard output and the standard error files can be downloaded. For this purpose the *PrepareResults* service retrieves the results from the Grid and stores them in the UI.

The executable must exist in the UI system and it has to be statically compiled so that it can be executed without library dependencies problems in any machine of the Grid. The implemented registration of this project is based on the Insight Segmentation and Registration Toolkit (ITK) [16] software library. ITK is an Open Source software library for image registration and segmentation.

Middleware Web Services Layer. The Middleware Web Services Layer provides an abstraction to the use of the WSs. The abstraction of the WSs Layer has two purposes. On one hand, to create a unique interface independent from the application layer, and on the other hand to provide methods and simple data structures to ease the development of final applications.

The development of a separate software library for the access to the WSs will ease future extensions for other applications that share similar requirements. Moreover it will enable introducing optimizations in this layer without necessarily affecting the applications developed on top of this layer.

Moreover, this layer offers a set of calls based on the Globus FTP APIs to perform the data transferring with the Gate-to-Grid Layer. More precisely, the abstraction of the WSs lies, in first place, on hiding the creation and management of the necessary stubs for the communication with the published WSs. In second place, this layer manages the data obtained by the WSs by means of simple structures closer to the application. From each of the available WSs in the Gate-to-Grid layer there exists a method in this layer that gets the information in XML given by the WSs and returns that information in basic types or structured objects which can be managed directly by the application layer.

Application Layer. This layer is the one that offers the graphical user interface which will be used for the user interaction. This layer makes use of functions, objects and components offered by the middleware WS layer to perform any operation on the Grid.

The developed tool has the following features available:

- Parameter profile management. The management of the parameters allows creating, modifying and removing configurations of parameters for the launching of multi-parametric registrations. These registration parameters are defined as a rank of values, expressed as by three values: initial value, increment and final value. The profiles can be loaded from a set of templates or directly filled-in before submitting the co-registrations according to these parameters.
- Transferring of the volumetric images that are going to be registered. The first step is to upload the images that are going to be registered. The application provides the option to upload the reference image and the other images that will be registered. These files are automatically transferred to the UI to be managed by the Gate-to-Grid layer.
- Submission of the parametric jobs to the Grid. For each combination of the input parameters, the application submits a job belonging to the same job group. The user can assign a name to the job group to ease the identification of jobs in the monitoring window.
- Job monitoring. The application offers the option to keep track of the submitted jobs for each group.
- Obtaining the results. When a Grid execution has reached the state *done*, the user can retrieve the results generated by the job to the local machine. The results include the registered image, the standard output and standard error generated by the program launched to the Grid. The user can also download the results from a group of jobs automatically.

3.2 Security

When dealing with medical data, the security of information being managed is very important to guarantee the privacy of the patients. Among the personal data, medical data is especially critical due to the implications that can have the release of the information for its owner.

The use of Grid technologies in health implies actions that might put the privacy of medical data in risk. Processing medical data in a distributed environment implies the transference of the data to remote resources. Although transferences can be performed through secure protocols, the data could be accessed by users with the sufficient privileges in the remote systems. However, in the specific action of this work, the information related to the patient is removed from the headers of the images and only the raw images are sent, identified through a coded name that do not share any information with the patient data. So the privacy in not comprised by anonimizing the images.

Regarding the access to the system, the different layers of the architecture defined in the Section 3 have different approaches in the implementation of the security. Basically, it can be divided into two parts: One part is related to the WS environment and other part will deal with the Grid-LCG environment. In both cases secure protocols are used for the communication.

In relation to the scope of the security of WSs, the architecture defines a client layer (middleware WS) for interacting with the system. For WSs, the SOAP protocol is used on top HTTPS, which is based in Secure Sockets Layer (SSL). The HTTPS protocol guarantees the privacy of the data and the use of digital certificates guarantees authentication of user.

The Grid middleware used (LCG), provides a native infrastructure of security Grid Security Infrastructure (GSI). GSI is also based in SSL. Before accessing any resource of the Grid, a proxy must be created from the client certificate, which should have been. The certificate is duly signed by the Certificate Authority (CA). Each resource of the Grid establishes a mapping of the Distinguish Name (DN) obtained from the proxy generated, to a resource user. Finally each deployed resource in the Grid is certificated by a valid CA.

The security in the different environments of the architecture (Web and Grid) has already been defined. The following lines try to explain how the connection between the Grid and Web security environments is performed. In the described architecture, the Gate-to-Grid is the common point between the Web and Grid environment. A mapping system of the Web users (through Web user certificate) has been implemented in the Gate-to-Grid layer that associates the Web users with the Grid users. For each user, a Grid proxy is created from its Grid user certificate.

4 Results

The first result of this work has been the *LCG Registration Launcher* tool, which has been developed on top of the architecture described in this article. Figure 4 shows two screenshots of the application, one showing the panel for uploading

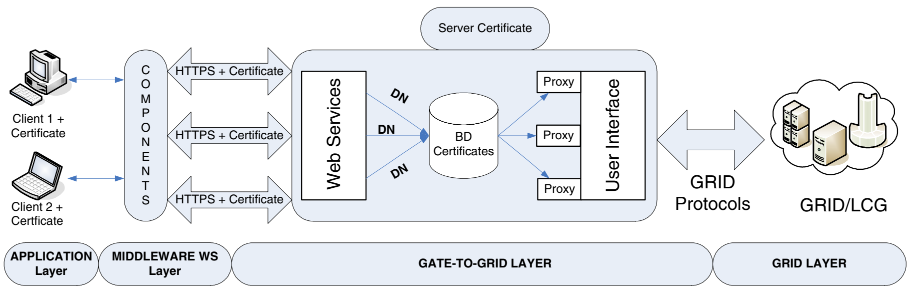
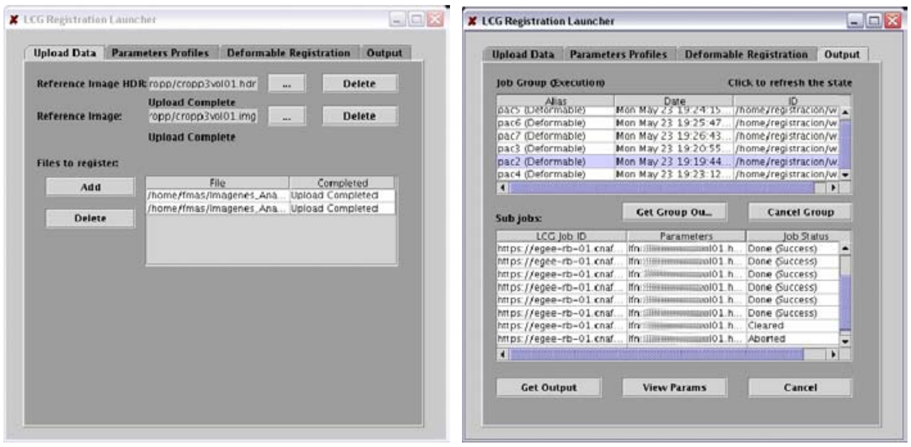


Fig. 3. Security scheme of the proposed architecture


LCG Registration Launcher

Upload Data | Parameters Profiles | Deformable Registration | Output

Reference Image HDR: ropp/cropp3vol01.hdr ... Delete
Upload Complete
Reference Image: 'opp/cropp3vol01.img ... Delete
Upload Complete

Files to register:

Add
Delete

File	Completed
/home/fmas/imagenes_Ana...	Upload Completed
/home/fmas/imagenes_Ana...	Upload Completed

LCG Registration Launcher

Upload Data | Parameters Profiles | Deformable Registration | Output

Job Group (Execution) Click to refresh the state

Alias	Date	ID
pac5 (Deformable)	Mon May 23 19:24:15...	/home/registracion/w
pac6 (Deformable)	Mon May 23 19:25:47...	/home/registracion/w
pac7 (Deformable)	Mon May 23 19:26:43...	/home/registracion/w
pac3 (Deformable)	Mon May 23 19:20:55...	/home/registracion/w
pac2 (Deformable)	Mon May 23 19:19:44...	/home/registracion/w
pac4 (Deformable)	Mon May 23 19:23:12...	/home/registracion/w

Sub jobs: Get Group Ou... Cancel Group

LCG Job ID	Parameters	Job Status
https://egee-rb-01.cnaf...	lfn:[redacted]ol01.h...	Done (Success)
https://egee-rb-01.cnaf...	lfn:[redacted]ol01.h...	Done (Success)
https://egee-rb-01.cnaf...	lfn:[redacted]ol01.h...	Done (Success)
https://egee-rb-01.cnaf...	lfn:[redacted]ol01.h...	Done (Success)
https://egee-rb-01.cnaf...	lfn:[redacted]ol01.h...	Done (Success)
https://egee-rb-01.cnaf...	lfn:[redacted]01.h...	Done (Success)
https://egee-rb-01.cnaf...	lfn:[redacted]ol01.h...	Cleared
https://egee-rb-01.cnaf...	lfn:[redacted]01.h...	Aborted

Get Output | View Params | Cancel

Fig. 4. Screenshots of the LCG registration launcher application

reference and floating volumes, and the other one showing the panel for the monitoring of the launched jobs.

The required time to perform a registration of a volumetric image in a PIII at 866 Mhz with 512 MB of RAM is approximately 1 hour and 27 minutes.

By one hand, when launching 12 simultaneous co-registrations using the EGEE Grid resources (for the biomed virtual organization) and giving the RB the ability to choose among the system resources, it took approximately 3 hours and 19 minutes. In this case it must be considered that the resources where shared by other Grid users. On the other hand, when just using resources with a lower degree of utilization, the total time for the 12 jobs was of 2 hours and 53 minutes (this case was using a CE with 20 WNs).

If a the same resources were used in a batch processing approach, using the same resources and running manually the jobs on the computing farm, the computing time would be a 8% shorter. The overhead of Grids is due to the use of secure protocols, remote and distributed storage resources and the scheduling overhead, which is in the order of minutes due to the monitoring policies which are implemented in a poll fashion.

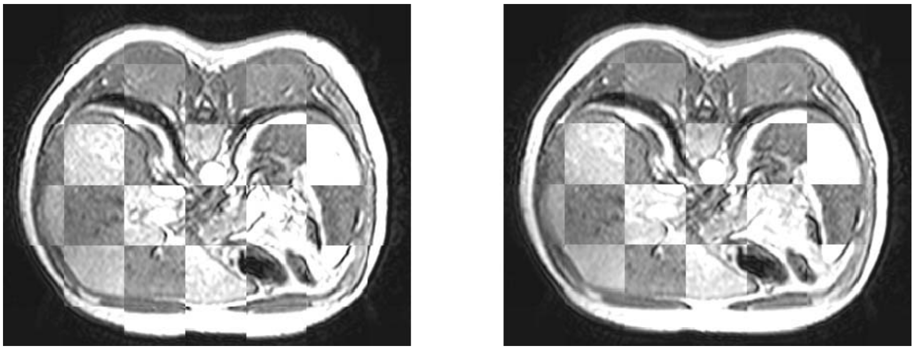


Fig. 5. Results of the co-registration before (left) and after (right)

Regarding the results obtained, Figure 5 shows a tiled composition of two images before the co-registration (left) and after the process (right). The figure clearly shows the improvement in the alignment of the voxels of the image.

5 Conclusions and Further Work

The final application developed in this work offers an easy-to-use high level interface that allows the use of the LCG2-based EGEE Grid infrastructure for image co-registration by Grid-unaware users.

With the use of the tool described in this work, the user achieves a large computational performance for the co-registration of radiological volumes and the evaluation of the parameters involved.

The Grid is an enabling technology that provides the clinical practice of processes with processes that, by its computational requirements, were not feasible with a conventional approach. It also offers a high throughput platform for medical research. The proposed architecture is adaptable to different platforms and enables the execution of different applications changing the user interface.

This work is a starting point for the realization of a middleware focused on the abstraction of the Grid to ease the development of interfaces for the submission of complex jobs to the Grid.

The application developed is a part of larger project. As introduced in Section 2, the co-registration application is a first step of the pharmacokinetic model identification. The next step to be treated will be the extraction of the pharmacokinetic model from a set of acquisitions. For this task, it was necessary to have the co-registration tool that has been developed in this work. Finally, the middleware WS layer will be enlarged to give support to new functionalities related with the extraction of the pharmacokinetic model. Finnally, the application currently supports the Analyze format [17], although the extension to other formats such as DICOM [18] is being considered among the priorities.

Acknowledgements

The authors wish to thank the financial support received from The Spanish Ministry of Science and Technology to develop the project *Investigación y Desarrollo de Servicios GRID: Aplicación a Modelos Cliente-Servidor, Colaborativos y de Alta Productividad*, with reference TIC2003-01318. This work has been partially supported by the Structural Funds of the European Regional Development Fund.

References

1. Scott Short. *Creación de servicios Web XML para la plataforma .NET.* Mc-Graw-Hill, 2002.
2. Universal Description, Discovery and Integratin (UDDI). http://www.uddi.org.
3. Simple Object Access Protocol (SOAP). http://www.w3c.org.
4. Expert Group Report. *Next Generation Grids 2.* European Commission, 2004. http://www.cordis.lu/ist/grids/index.htm.
5. Foster I. and Kesselman C. *The GRID: Blueprint for a New Computing Infraestructure.* Morgan Kaufmann Publishers, Inc., 1998.
6. I. Foster and C. Kesselman. Globus: A metacomputing infraetructure toolkit. *Intl J. Supercomputer Applications*, pages 115,128, 1997.
7. Forschungszentrum. *Unicore Plus Final Report.* Dietmar Erwin, 2003.
8. *InnerGrid Users' manual.* GridSystems, 2003.
9. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002.
10. The Globus Alliance. http://www.globus.org.
11. The DATAGRID project. http://www.eu-datagrid.org.
12. LHC Computing Grid. http://lcg.web.cern.ch/LCG.
13. gLite. Lightweight Middleware for Grid Computing. http://glite.web.cern.ch/glite.
14. Enabling Grids for E-sciencE. http://www.eu-egee.org.
15. Virtual Organization. http://hep-project-grid-scg.web.cern.ch/hep-project-grid-scg/voms.html.
16. Ibanez, Schroeder, Ng, and Cates. *The ITK Software Guide.* Kitware Inc.
17. Analyze 7.5 File Format. Mayo Clinic.
18. National Electrical Manufacturers Association. Digital Imaging and Communications in Medicine (DICOM). 1300 N. 17th Street, Rosslyn, Virginia 22209 USA.