Towards a Virtual Radiological Platform Based on a Grid Infrastructure

Sorina Camarasu[1], Hugues Benoit-Cattin[1], Laurent Guigues[1], Patrick Clarysse[1], Olivier Bernard[1], Denis Friboulet[1]

[1]CREATIS-LRMN,
CNRS UMR5220, INSERM U630, Claude Bernard Lyon 1 University, INSA Lyon,
Villeurbanne France

Abstract

The proposed Virtual Radiological Platform (VRP) project aims at providing realistic multi-modal medical images with 'ground-truth' knowledge. It relies on medical image simulators which often represent heterogeneous and computing demanding applications dealing with large amounts of medical images. In this context, computer grids are interesting architectures providing large computing power and a valuable collaborative framework. In this paper, the analysis of medical imaging applications gridification, based on several experiences, enables us to target key points for a contributory integration of grid architectures within a VRP framework.

Contents

The paper begins with an overview of the Virtual Radiological Platform (VRP) defining its aim, usage and requirements. Chapter 2 briefly describes the two key elements of the grid contribution to such a VRP. Our experience feedback regarding medical application porting on grid infrastructures is summarized in a step-by step gridification approach in Chapter 3. Finally, we discuss problematic issues and draw the conclusions in Section 4.

1 OVERVIEW OF THE VIRTUAL RADIOLOGICAL PLATFORM

1.1 VRP Objective

As illustrated in Figure 1, the VRP aims at providing realistic medical images relying on multi-modal simulators and taking as an input a virtual model. Such a model is a parametric description of an object of interest (organs, body…) in accordance with the simulation process (proton density, diffusion absorption parameter etc.). The following modalities are targeted: Magnetic Resonance Imaging (MRI), Magnetic Resonance Spectroscopy (MRS), Ultrasound (US), XRay and Computed Tomography (CT), Positron Emission Tomography (PET) as well as Radiotherapy and Hadrontherapy.

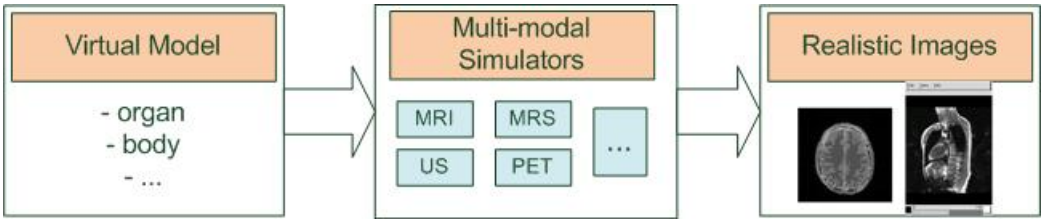

Figure 1 - VRP Overview.

Note that for these modalities a certain number of simulators have already been developed and used: *Simri* [1] which is a 3D MRI Simulator, *Field* [2] for the US simulation, *Gate* [3] and *Sorteo* [4] in the PET field and *ThIS* [5] for Hadrontherapy. In view of the VRP, recent tests have been performed using the specific simulation of MR and PET images of a breathing thorax and beating heart [6].

In a similar context, GEMSS (GRID-enabled Medical Simulation Services) [7] was a two and a half year project which started in 2002 and aimed at enhancing healthcare by bringing a variety of medical computing and resource services into the user's environment and providing access to advanced simulation and image processing services. The project led to the development of a grid middleware integrating a certain number of services. Their architecture was based on standard web services.

Note that GEMSS was mainly oriented towards developing an innovative middleware, whereas we are more oriented towards using existing grid middleware for building an architecture (VRP) allowing for a unified and user-friendly usage of multi-modal simulators.

1.2 VRP Usage

Simulators are particularly useful in the field of medical imaging, as they can serve different purposes. Figure 2 summarizes some of the main functionalities of the VRP:

• Treatment planning: for example *ThIS* [5] is a software dedicated to the Monte-Carlo simulation of irradiations of living tissues with photons, protons or light ions beams for cancer therapy planning [8].

• Theoretical understanding of physical phenomena: a MRI simulator like *Simri* [9] is naturally suited to acquire theoretical understanding of the complex MR technology. It can also be used as an educational tool in medical and technical environments.

• Acquisition Protocol Definition/Optimization: in the case of MRI for example, because of the complexity of the in-vivo MRI acquisitions, the MRI protocol definition and optimization are extremely difficult, but they become accessible through a simulator

• Assessment tool for image analysis methods: simulators produce realistic images that can be used to evaluate the results of image processing algorithms as the 'ground truth' is available through the virtual model of anatomical structures

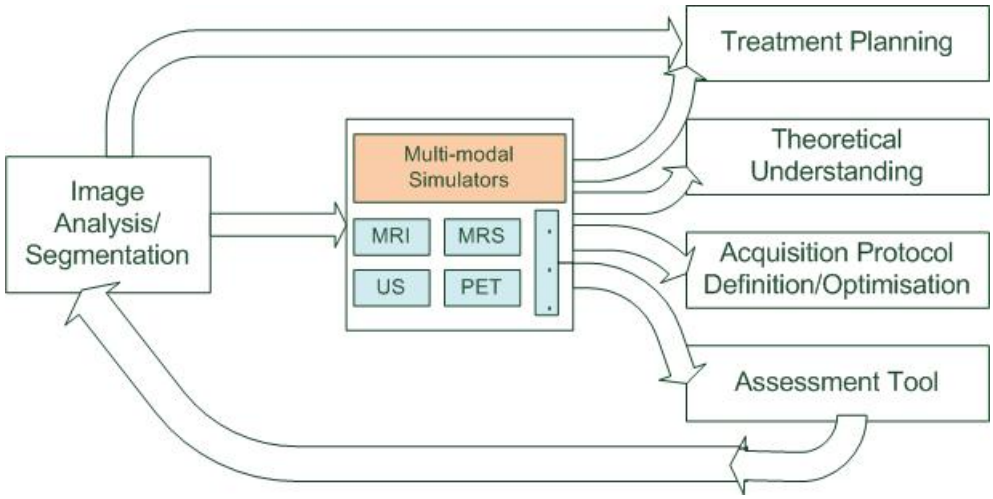

Figure 2 - VRP Use Cases.

At a larger scale, we can imagine a complete simulator of a Virtual Physiological Human [10], i.e. a multi-scale human modeling for a preventive and predictive personalized health. This kind of project is only possible if existing heterogeneous medical imaging applications and especially simulators can be brought together and rendered interoperable on a same platform. Such high level objective is the long-term goal of the FP7 NoE VPH project.

1.3 VRP Requirements

Simulators interoperability and easy plug-in of new simulators are the main requirements for the VRP. In the last few years, a certain number of simulators have been used and developed inside the medical community. Their usage is unfortunately restricted because of their non-interoperability and of their heterogeneous requirements (dependencies, interface, parameters etc.).

Another important aim is making the simulators and the data accessible to everyone. Such a platform should allow users to run simulations without having to install them on their own computer.

Moreover, the platform should have access to distributed architectures providing important computing and storage resources. This will lead to a faster execution for time consuming applications and to a

discharge of the user work station. Parallel and distributing computing will also ensure scaling capabilities, i.e. the possibility of supporting a large number of users and jobs simultaneously. However, the parallel and distributed computing platform has to be transparent to the end user, who does not need to acknowledge the complexity of the underlying infrastructure.

In the next section we will explain how grids can meet these VRP requirements.

2 GRID CONTRIBUTION TO VRP

As presented previously, the VRP has to be able to offer both a collaborative platform and computing resources for the medical image simulators execution. These two requirements can be met with the help of grid infrastructures exploited in various medical and non-medical projects. The next two paragraphs will illustrate this through the example of several medical imaging projects using grid infrastructures for building collaborative platforms and for intensive computing.

2.1 Example of collaborative platforms

In the context of breast cancer, the *MammoGrid* project [11] developed a distributed platform for a community of radiologists. They used the grid infrastructure in order to support collaborative medical image analysis, to enable radiologists share standardized mammograms, compare diagnosis etc. Thus different sites from Udine, Geneva, Cambridge and Oxford contributed at creating an international mammogram database like illustrated in Figure 3. In their implementation a lightweight Grid middleware solution (called *AliEn*) is used. They also developed a set of services called the MammoGrid Services (MGS) for managing mammography images and associated patient data on the grid [12].

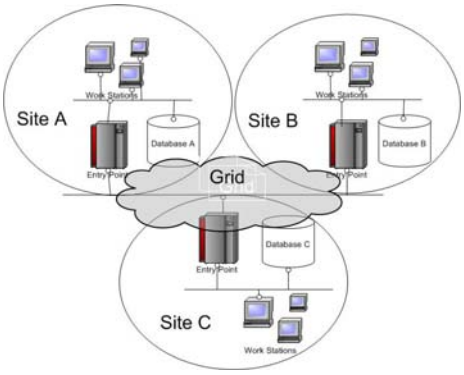

Figure 3 – Illustration of the distributed platform developed within the MammoGrid Project.

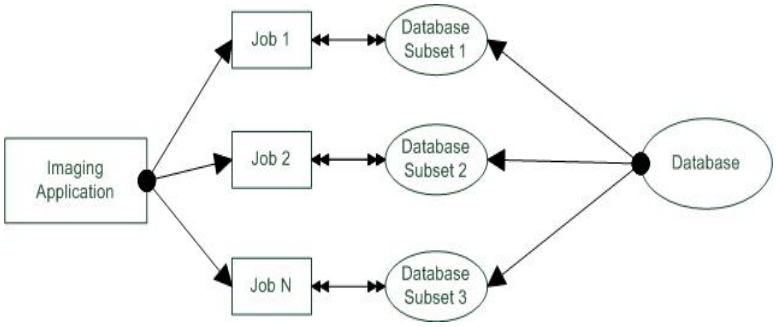

Figure 4 - Database and processing partitioning.

Still on breast cancer, the MAGIC-5 Project [13] also used a dedicated *AliEn* Server for their database architecture allowing for images to be acquired in any site available to the Project. Data are stored on local resources and recorded on a common service, known as Data Catalogue, together with the related information (metadata).

2.2 Intensive computing

Grids are also extremely valuable for their computing resources. In the case of a VRP, a certain number of medical imaging simulators are needed. Grid computing parallelism allows then for important speed-up for such computing intensive applications. We can refer to it as *processing partitioning*. As illustrated in Figure 4, processing partitioning can be coupled with *database partitioning* for higher efficiency of the medical applications which deal with large amounts of data (e.g. medical image indexing and retrieval).

If the database is partitioned in bags of images to be analyzed, each bag can be processed by a single computing job. If one bag represented one image so that all images were processed in parallel, then the execution time would be the maximum of the execution times of each image processing [14]. However, for large databases this hypothesis is not realistic, as thousands of processors cannot be available simultaneously on a shared grid infrastructure concurrently exploited by a large number of users.

Processing partitioning is studied in [14]. The strategy for choosing the granularity of the tasks submitted aims at reducing both the total execution time and the number of submitted sub-jobs (for infrastructure optimization). A probabilistic model is proposed and validated based on experimental results. A more detailed study on submission strategies optimization can be found in [15]. This paper studies the influence of various parameters as for example execution site, resource broker or day of the week. It shows that the grid latency is closely related to the choice of a resource broker or a computing element and that information concerning these two parameters may be both sufficient and accessible for job submission and system performances optimization.

3 Experience Feedback on Application Porting

A large number of medical imaging applications including simulators have been implemented without having taken into account a possible future grid usage. However, we have nowadays the possibility of using distributed architectures for these applications if we succeed in adapting them for grid usage, which comes within the scope of the VRP. This section summarizes our experience feedback regarding medical application porting on grid infrastructures in a step-by step gridification methodology. This methodology will also serve for future work needed for integrating new applications into our grid oriented VRP.

After several years of experience with porting applications on distributed architectures (starting with the European DataGrid Project in 2001), we can say that the gridification process requires several layers of adaptability:

3.1 Basic adaptability – successful execution

The application has to be able to run on the grid environment, i.e. operating system, file system, shared resources etc. The configuration of the distant grid nodes is often very little known to the grid user and does not necessarily correspond to the user's local configuration. Therefore, the basic adaptability can refer both to the distant grid node environment and to the application to be executed on it.

The distant environment often needs a certain 'customization' before job execution. The customization can vary from variable definition to file downloading from accessible sources. These operations can be done through a shell script that launches the application after the necessary environment customization. Depending on the used platform and on the user's access rights, there will be a certain number of restrictions that may not allow for all the changes the users would like to bring in order to run his/her application. In this case, he will have to customize the application.

This may be the case for applications requiring a certain number of libraries that are not present on the distant nodes. *ThIS* [5] for example is based on the *Geant4* [16] toolkit for the simulation of the passage of particles through matter. *Geant4* is used in different application areas, mainly in high energy, nuclear and accelerator physics, but also in medical and space sciences. *ThIS* uses a certain number of the *Geant4* libraries which may be installed on some of the grid nodes, but not necessarily on all of them. Moreover, *Geant4* can build shared libraries to which *ThIS* can be linked. The solution we adopted in order to achieve a basic adaptability for *ThIS* was to re-build all dependencies without shared libraries and then compile *ThIS* by linking it statically to the required libraries.

Another possible solution is to copy the shared libraries on the nodes together with the executable or to create the necessary environment for compiling our application on the target execution node.

3.2 Intermediate adaptability – application parallelization

Grids are particularly efficient for intensive computing applications that can be parallelized. This is often the case for medical image applications. However, parallelization is sometimes not taken into account at the conception phase of the application. We will consider the example of *Simri* and *ThIS* which are two simulators parallelized in two different ways. *Simri* uses an MPI (Message Passing Interface) implementation, whereas *ThIS* deals with Monte-Carlo simulations.

MPI parallelization is transparent to the end user: task parallelization and message passing between the tasks executing on different processors is handled automatically if the application has been developed using one of the MPI implementations like for example LAM/MPI or MPICH. MPI is often used on computer clusters, where users have access to multiple processors simultaneously. *Simri* is a typical example of application that has been modified in order to be parallelized with LAM/MPI [17]. It can now be executed on a single PC, as well as on parallel machines, on a small internal cluster and even on larger infrastructures like the EGEE Grid.

The experience of using both infrastructures for *Simri* showed that MPI is more adapted for cluster usage than for grid usage. A first issue when using grids concerns the previous adaptability point: the different computing elements which can be distributed geographically all over the world and have different administrators may not support the needed MPI implementation. A second issue is that usually the same application cannot be split between different computing sites. This means its parallelization is limited to one structure and cannot benefit from the free resources elsewhere. This limitation could be overcome with wrappers and we can then speak of a higher layer of adaptability.

ThIS is a *Geant4* based software dedicated to the Monte-Carlo (MC) simulation of irradiations of living tissues with photons, protons or light ions beams for cancer therapy. For reliable results, a large number of particles are simulated, leading to intensive computing. However, this large number of particles needed for one complete simulation can be split into sub-simulations with fewer particles which are thus less computing intensive. The condition to observe when splitting one simulation is that each sub-job must use a different random seed number. This allows the sub-simulations to be statistically independent and to be thus run concurrently. The parallelization in this case is achieved naturally by simulating fewer particles concurrently on multiple processors. For example, if we want to simulate 50 M particles, the simulation is divided into 50 sub-jobs, each with 1 M particles to simulate. However, additional tools are needed for job splitting and result merging.

In conclusion, the parallelization technique depends essentially on the application. For independent MC particle simulations the splitting and merging are straightforward and therefore they can be done with external generic tools as presented in the next section. On the contrary, in the case of the MRI simulation

splitting and merging requires image partitioning and reconstruction which cannot be achieved easily with the same tools. MPI parallelization solves this problem and renders the splitting and merging process transparent to the user, which explains our choice for the parallelization technique. Even if MPI may not have been conceived for usage on grid architectures, in practice a strong MPI demand has been observed within the biomedical community of the EGEE project. Their experience shows that for a large number of biological applications MPI is the most accessible parallelization technique, allowing for the code to be executed without any further effort on different architectures: parallel machines, clusters and grids.

3.3 Advanced adaptability – wrappers/descriptors

Advanced adaptability is meant to help the user take full advantage of the grid resources and application parallelism by bringing a certain degree of automation in the process of job parallelization and result retrieval. This can be achieved with additional tools like wrappers or even new software layers.

3.3.1 Parallel job submission, monitoring and retrieval

The previous section presented two means allowing for applications to be executed on multiple processors concurrently. However, both have their own limitations that can be overcome with additional tools.

In the case of MPI, the different processes of a same job could only execute on the processors of a same cluster. One of the topics covered by the Interactive European Grid is advanced MPI usage on the *int.eu.grid* infrastructure [18]. In this field, they developed several tools in order to enhance workload management features for MPI, as well as a cross broker allowing for cross-site MPI.

A natural parallelization like the one used in the case of *ThIS* can benefit from automatic handling of job splitting and result merging. The usefulness of his approach has been acknowledged so that today there are several tools that have been designed to implement it. The Java Job Submission (JJS) [19] tool for example can be used for easier job submission, monitoring and result retrieval. It hides most of the grid complexity from the user and also allows for simultaneous submission of similar sub-jobs called parametric jobs. However it does not take into account actual job splitting and merging. *Ganga* [20] is another interesting tool which includes a job splitter and merger. Moreover, it allows users to interact with multiple and different computing backends in a very similar way through the *Ganga* interface.

Grid middleware integrates basic tools for job submission, monitoring and retrieval on the grid. On the EGEE grid infrastructure for example, the Workload Management System (WMS) is used. The WMS is a very useful tool; however it is often not sufficient. The WMS allows for job submission and monitoring ‘manually’ with command lines. This requires a large amount of work for status querying for every job, then manual retrieval and merging of results. Despite its simplicity, the WMS also has a certain number of advanced functionalities among which we can cite the ‘parametric job’ type that allows the automatic submission of a number of very similar jobs through one job description file. However, this advanced feature still requires further improvement on certain aspects like global job status. At present, for example, if one submits N jobs at a time with one job description file for parametric jobs and some of the jobs are done and the others fail, the global job status stays ‘Running’. In conclusion, more sophisticated tools like Ganga or JJS are often needed for more efficient job management.

3.3.2 Middleware compatibility

The multiple computing backend issue brings us to another important aspect that could be handled as an advanced adaptability layer: middleware interoperability, as different technologies not only co-exist but are also changing and evolving continuously. In our case for example, *Simri* has access to the EGEE grid deploying the gLite middleware and to an internal cluster deploying PBS. The same application can be

executed on both platforms due to a home-made tool (a portal) that translates user requests into the right language and deals with job submission, monitoring and retrieval depending on the backend.

3.3.3 Integration into service platforms

Last but not least, advanced adaptability can also refer to wrapping applications for integration into a service platform. On a service oriented platform heterogeneous applications need to be presented in a normalized language. The GEMSS project mentions application descriptors [21]. An application descriptor is an XML document that provides meta-data about an existing application and is usually supplied by the service provider. The information provided in an application descriptor defines the semantics of the operations of a generic application service and enables the service framework to expose an application automatically as a Web service. An application descriptor usually specifies the input/output files, the script for initiating job execution, scripts for gathering status information, and a finish file, which indicates that a job has been finished.

3.4 End-user adaptability

On top of these adaptability layers, there is a user-adaptability layer which is a high level interface allowing users with no grid specific knowledge to use the available services. Grid architectures usually offer transparent access to resources to grid users but they lack a generic user- friendly interface at a final user level. This hinders their use by physicians and researchers who would need secure and easy-to-use portals for submitting their jobs.

This problem has been acknowledged by different communities who have already developed a certain number of portals such as Genius [22], GridSphere [23] etc. A few communities have developed their own portals for specific needs. The *Simri* portal [24] has been developed for example for the *Simri* simulator. This 3 layer architecture portal was developed in Java and PhP. More information on the technical implementation can be found in [25]. Experience shows that the development of a specific portal for a given application is not a tremendous task. However, creating and especially maintaining a new portal for every application rapidly becomes a very time consuming job. The challenge thus becomes conceiving a more generic tool, i.e. a portal easily re-configurable for similar applications.

4 OPEN QUESTIONS AND CONCLUSION

Grids are promising architectures that can bring different solutions to medical image storage and intensive computing problems. Their growing interest is reflected through numerous projects and grids initiatives emerging worldwide. However, grid issues limiting the adoption of existing systems still exist.

In the previous section we tackled a step-by-step approach for medical application porting on grid architectures emerged from our own experience which shows that porting and deploying applications on grid infrastructures is not a straightforward process. Moreover, although this approach is meant to be general, different applications have different structures and requirements making it impossible to have a recipe 100% valid for all applications. The difficulty of the task is therefore increased.

For example, one problem that we still find difficult to overcome and that could be considered as an advanced adaptability issue concerns large input and output file management. Most grid middleware incorporate file catalogues and data management services, but job parallelization often requires multiple

copies of data from storage elements/databases to the working nodes. This problem is particularly important for medical applications.

Last but not least, we noticed that an important element which seems to significantly hinder the growth of the grid user community is a higher user interface. In order to really render a gridified application accessible in practice one has to invest important resources for the user interface and support issues.

The VRP architecture should therefore take into account all these problems and facilitate the integration of medical simulators into the grid environment. Among the main requirements of the VRP platform we can cite the access to grid architectures and distributed data, easy plug-in of new simulators and personalized models, as well as a user-friendly interface. Designing and implementing the VRP's architecture represents thus a very challenging project. At present our team works on designing the first draft of the VRP architecture, which relies on Web services and the XML language.

5 ACKNOWLEDGEMENTS

This research work is in the scope of the scientific topics of the FP7 NoE VPH and of the French National Grid Institute (IdG). This work has been partially funded by the EGEE2 project within the European FP6.

Many thanks to David Sarrut for his collaboration on porting ThIS on the EGEE Grid, to Thomas Grimaldi for his contribution on the VRP project and to Fabrice Bellet for his help with application compiling and customization for the grid.

6 BIBLIOGRAPHY

[1] H. Benoit-Cattin, G. Collewet, B. Belaroussi, H. Saint-Jalmes, and C. Odet, "The SIMRI project: A versatile and interactive MRI simulator," *Journal of Magnetic Resonance*, vol. 173, pp. 97-115, 2005.

[2] J. A. Jensen, "Field: A Program for Simulating Ultrasound Systems," *Medical & Biological Engineering & Computer*, vol. 34, pp. 351-353, 1996.

[3] S. Jan, et al., "GATE: a simulation toolkit for PET and SPECT," *Physics in Medicine and Biology*, vol. 49, pp. 4543-4561, 2004.

[4] A. Reilhac, et al., "PET-SORTEO: a Monte Carlo-based Simulator with high count rate capabilities," *IEEE Transactions on Nuclear Science*, vol. 51, pp. 46- 52, 2004.

[5] ThIS Homepage, http://www.creatis.insa-lyon.fr/rio/ThIS

[6] R. Haddad, I. E. Magnin, and P. Clarysse, "A new fully-digital anthropomorphic and dynamic thorax/heart model," in *29th Annual International Conference of the IEEE EMBS*, Lyon, France, 2007, pp. 5999-6002.

[7] The GEMSS Project: Grid-enabled medical simulation services. EU IST Project IST-2001-37153, 2002--2005, http://www.gemss.de/

[8] A. GÓMEZ, et al., "Monte Carlo Verification of IMRT treatment plans on Grid," in *Healthgrid*, Geneva, 2007, pp. 105-114.

[9] Simri Homepage, http://www.simri.org/

[10] Towards Virtual Physiological Human: Multilevel Modelling and Simulation of the Human Anatomy and Physiology – White Paper, November 2005, http://europa.eu.int/information_society/activities/health/docs/events/barcelona2005/ec-vph-white-paper2005nov.pdf

[11] R. Warren, et al., "MammoGrid - a prototype distributed mammographic database for Europe," *Clinical Radiology* 2007.

[12] F. Estrella, et al., "Experiences of engineering Grid-based medical software," *International journal of medical informatics*, pp. 621-632, 2007.

[13] R. Bellotti, et al., "Distributed medical images analysis on a Grid infrastructure," *Future Generation Computer Systems*, 2007.

[14] J. Montagnat, T. Glatard, and D. Lingrand, "Texture-based Medical Image Indexing and Retrieval on Grids," *Medical Imaging Technology*, vol. 25, 2007.

[15] T. Glatard, D. Lingrand, J. Montagnat, and M. Riveill, "Impact of the execution context on Grid job performances" in *International Workshop on Context-Awareness and Mobility in Grid Computing (WCAMG'07)*, Rio de Janeiro, 2007.

[16] Geant4 Homepage, http://geant4.cern.ch/

[17] H. Benoit-Cattin, F. Bellet, J. Montagnat, and C. Odet, "Magnetic Resonance Imaging (MRI) simulation on a grid computing architecture" in *CCGRID* Tokyo, 2003.

[18] The Interactive European Grid Project, http://www.interactive-grid.eu/

[19] Java Job Submission Tool, http://cc.in2p3.fr/docenligne/269

[20] Ganga Homepage, http://ganga.web.cern.ch/ganga/

[21] S. Benkner, I. Brandic, G. Engelbrecht, and R. Schmidt, "VGE - A Service-Oriented Grid Environment for On-Demand Supercomputing" in *Fifth IEEE/ACM International Workshop on Grid Computing*, 2004.

[22] The Genius Portal, https://genius.ct.infn.it/

[23] The Gridsphere Portal Framework, http://www.gridsphere.org

[24] The Simri Portal, https://simri.creatis.insa-lyon.fr

[25] F. Bellet, I. Nistoreanu, C. Pera, and H. Benoit-Cattin, "Magnetic resonance imaging simulation on EGEE grid architecture: A web portal design." in *HealthGrid*, Valencia, 2006, pp. 34-42.