Virt&l-Comm.3.2012.22

Università degli Studi di Perugia
Dipartimento di Matematica e Informatica

Dottorato di Ricerca in Matematica e Informatica per l'Elaborazione e la Rappresentazione dell'Informazione e della Conoscenza

XXIII° Ciclo - Settore Scientifico Disciplinare INF/01

A Grid Knowledge Management System aimed at Virtual Research Communities Sustainability based on Quality Evaluation

Dottorando:
Carlo Manuali

Relatore:
Prof. A. Laganà

Coordinatore:
Prof.ssa G. Coletti

Anno Accademico 2009/2010

ISSN: 2279-8773

Acknowledgements

I thank my friends, close and faraway, decreased in quantity though increased in quality as time goes along.

I thank my supervisor, through which I decided to continue to believe in a ethically open, true public and scientifically competitive University.

I thank my wife Martina, who every day resize my pride with her smile.

I thank my parents, for giving me the life.

I thank God for all that.

Ringraziamenti

Ringrazio i miei amici, vicini e lontani, allo scorrere del tempo diminuiti in quantità ma aumentati in qualità.

Ringrazio il mio relatore, grazie al quale ho deciso di continuare a credere in una Università eticamente aperta, decisamente pubblica e scientificamente competitiva.

Ringrazio mia moglie Martina, che ogni giorno ridimensiona il mio orgoglio con il suo sorriso.

Ringrazio i miei genitori, perchè mi hanno dato la vita.

Ringrazio Dio per tutto questo.

to my Father, for the third time.

a mio Padre, per la terza volta.

ISSN: 2279-8773

"Would you tell me, please,

which way I ought to go from here?"

"That depends a good deal on where you want to get to",

said the Cat.

L. Carroll (1832-1898)

Alice's Adventures in Wonderland

ISSN: 2279-8773

Abstract

In the present Thesis work a new Grid Knowledge Management System specifically designed for managing Virtual Research Communities is presented. In particular, a new intelligent Grid Framework named GriF has been developed and validated to run scientific applications on the EGI Grid by considering as a study case quantum reactive scattering codes.

In addition, a new Grid Credit System named GCreS has been designed and prototypized. GCreS evaluates QoS (Quality of Services) and QoU (Quality of Users) by exploiting the GriF architecture and some new sensors in order to foster the Grid Sustainability.

Sommario

Nel presente lavoro di Tesi viene presentato un nuovo sistema per la gestione della conoscenza in Grid specificatamente disegnato per la gestione delle Comunità di Ricerca Virtuali. In particolare, è stato sviluppato e validato un nuovo Framework intelligente di Grid denominato GriF per l'esecuzione di applicazioni scientifiche sulla Grid di EGI, considerando come caso di studio i codici quantistici di scattering reattivo.

Inoltre, è stato progettato e prototipizzato un nuovo sistema di crediti di Grid chiamato GCreS. GCreS valuta la QoS (Qualità dei Servizi) e la QoU

ISSN: 2279-8773

(Qualità degli Utenti) facendo leva sull'architettura di GriF e su alcuni nuovi sensori al fine di favorire la Sostenibilità della Grid.

Contents

ISSN: 2279–8773

ISSN: 2279–8773

ISSN: 2279-8773

ISSN: 2279–8773

List of Figures

ISSN: 2279-8773

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

ISSN: 2279-8773

List of Tables

ISSN: 2279-8773

ISSN: 2279-8773

Chapter 1

Introduction

Grid empowered computational experiments which have become nowadays indispensable for science advances are typically performed in an open Grid infrastructure (as is the case of the European Grid Initiative, or EGI [1]) in which the work done by specific Virtual Organizations (VO)s, Heavy User Communities (HUC)s and/or Virtual Research Communities (VRC)s is crucial to organize, operate and harmonize the efforts spent and the results obtained. Accordingly, the possibility of simplifying and harmonizing the work carried out by computational scientists when evaluating their results within the Grid is of paramount importance. To achieve this goal, we have designed a Grid Knowledge Management System (GKMS) aimed at enhancing VRCs Sustainability by leveraging on Quality Evaluation.

In this respect, after illustrating the main characteristics of EGI and of its

ISSN: 2279-8773

communities in chapter 2, a detailed description of the steps forward made by the Computational Chemistry on the field of Grid Computing is provided in chapter 3.

Then, in chapter 4 GriF [2, 3], a new Collaborative Grid Framework designed and developed by us according to the Service Oriented Architecture (SOA) model, is illustrated. In GriF information related to the users behavior and to the characteristics of their job runs in Grid can be gathered together and combined with the information extracted from the middleware. GriF is also proposed as a tool for the Science Gateways of the Chemistry and Molecular Innovation Science and Technology (CheMIST) community in order to facilitate massive calculations on the Grid and improve scientific cooperation among users utilizing Web and Grid Services fostering Collaboration and promoting Quality.

For this reason, new parameters and Grid sensors targeted to better evaluate the Quality of Users (QoU) and the Quality of Services (QoS) of Grid organizations, are presented in chapter 5.

A preliminary study (based on a Quantum Chemistry testbed) on the profiling of the users of a Grid organization paving the way for a systematic analisys of the work carried out within a Grid community targeted to the enhancement of its Sustainability via the development of a GKMS, is illustrated in chapter 6. To this end, GCreS (a Grid Credit System architecture

ISSN: 2279-8773

model relying on GriF designed and developed by us to the end of rewarding the contributions of the members of the Grid communities in terms of services and activities) is also presented together with some examples of its application. Conclusions and directions for future work are outlined in chapter 7.

Introduzione

Gli esperimenti computazionali potenziati mediante l'implementazione in Grid, divenuti al giorno d'oggi indispensabili per i progressi della scienza, vengono in genere eseguiti in una infrastruttura Grid aperta (come nel caso della European Grid Initiative, o EGI [1]) in cui il lavoro svolto da specifiche Virtual Organizations (VO)s, Heavy User Communities (HUC)s e/o Virtual Research Communities (VRC)s risulta essere cruciale al fine di organizzare, gestire e armonizzare gli sforzi compiuti ed i risultati ottenuti. Di conseguenza, la possibilità di semplificare e armonizzare il lavoro svolto dagli scienziati computazionali e quindi di valutare i loro risultati all'interno della Grid risulta di fondamentale importanza. Per raggiungere questo obiettivo, abbiamo progettato un Grid Knowledge Management System (GKMS) volto a migliorare la sostenibilità delle VRCs facendo leva sulla valutazione della

ISSN: 2279-8773

Qualità.

A questo proposito, dopo aver illustrato le principali caratteristiche di EGI e delle sue comunità nel capitolo 2, viene fornita nel capitolo 3 una descrizione dettagliata dei passi in avanti compiuti da parte della Chimica Computazionale nel campo del Grid Computing.

Quindi, nel capitolo 4 viene illustrato GriF [2, 3], un nuovo Framework Collaborativo di Grid da noi progettato e sviluppato secondo il modello di una Service Oriented Architecture (SOA). In GriF le informazioni relative al comportamento degli utenti ed alle caratteristiche delle loro esecuzioni di calcoli in Grid possono essere raccolte e combinate con le informazioni estratte dal middleware. GriF inoltre si propone come strumento per i Science Gateways della comunità di Chemistry and Molecular Innovation Science and Technology (CheMIST) al fine di facilitare calcoli massivi in Grid e di migliorare la cooperazione scientifica tra gli utenti mediante Web e Grid Services in grado di favorire la Collaborazione e promuovere la Qualità.

Per questo motivo, nuovi parametri e sensori di Grid mirati ad una migliore valutazione della Qualità degli Utenti (QoU) e della Qualità dei Servizi (QoS) delle organizzazioni di Grid vengono presentati nel capitolo 5.

Nel capitolo 6 viene illustrato uno studio preliminare (basato su un esempio di Chimica Quantistica) sulla profilazione degli utenti di una organizzazione di Grid che apre la strada ad una analisi sistematica del lavoro svolto all'interno

ISSN: 2279-8773

di una comunità di Grid mirata al miglioramento della sua sostenibilità attraverso lo sviluppo di un GKMS. A tal fine, viene presentato anche GCreS (un modello architetturale di un Sistema di Crediti di Grid basato su GriF, da noi progettato e sviluppato al fine di premiare i contributi dei membri delle comunità di Grid in termini di servizi e attività) insieme ad alcuni esempi di applicazione. Le conclusioni e diversi orientamenti per il futuro vengono descritti nel capitolo 7.

ISSN: 2279–8773

Chapter 2

The Advent of Grid Computing

2.1 The Evolution of Distributed Computing.

The continuous increase in computing power together with the spread of high-speed broadband networks and the growth of Internet have changed the way in which developed societies manage information and related services. Accordingly, geographically distributed resources, such as storage devices, data sources and supercomputers, can be exploited by different users around the world as a single and unified resource because of their interconnections. As a result, a new paradigm is emerging in which computing is offered as an utility by third parties and the user is billed only according to resources

ISSN: 2279–8773

consumption. This service-oriented approach taken by organizations offering a large variety of services has been shown to be scalable and flexible [4].

The idea of distributing resources within computer networks is not new. It dates back to remote job entry on mainframe computers and the initial use of data entry terminals. It was further developed first with minicomputers, then with Personal Computers (PC)s and two-tier client-server architecture. However, while the PC offered more autonomy on the desktop, the trend is now moving back to a client-server architecture with additional tiers, yet with the server not being in-house.

Distributed computing has been boosted not only by the improvement in computer component technology but also by the evolution in communication protocols. Networks were created by IBM in 1974 using the Systems Network Architecture (SNA) and evolved in 1976 using the ITU-T's X.25 [5]. These enabled large-scale public and private data networks were gradually replaced by more efficient or less complex protocols, notably the well-known TCP/IP. At present broadband networks extend worldwide the geographical reach of distributed computing thanks to the client-server relationship that can extend across borders and continents.

In general terms, a distributed system is "a collection of independent computers that appears to its users as a single coherent system" [6]. A common description of distributed systems focused on the importance of consider-

ISSN: 2279-8773

ing aspects such as reliability, fault tolerance and security when going distributed. It is properly rendered by the statement: ”you know you have a distributed system when the crash of a computer you have never heard of stops you from getting any work done” [7]. The main goal of a distributed computing system is to connect users and resources in a transparent, open, cost-effective, reliable and scalable way. Grids[1], Clouds and similar platforms are distributed computing systems sharing different objects including physical (computational power, storage devices and communication capacity) and virtual (operating systems, software and licenses, tasks, applications and services) resources. Typically, a client layer is used to display information, receive user input and communicate with the other layers. Therefore a transparent and network-independent middleware layer plays a mediating role: it connects clients with requested and provisioned resources, balances

[1]The Grid is widely seen as a step beyond the Internet, incorporating pervasive high-bandwidth, high-speed computing, intelligent sensors and large-scale databases into a seamless pool of managed and brokered resources, available to scientists, industry and the man in the street. The name, Grid, draws an analogy between the pervasive availability of electrical power and that of computing and data coupled with mechanisms for their effective use. The potential benefits and social impact of the Grid are enormous. For the scientist, the Grid will enable large-scale scientific collaborations. For industry, it will enable new ways of working. For the man in the street, it will enable unprecedented access to information and services.

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

peak loads between multiple resources and customers, regulates the access to limited resources, monitors all activities and gathers statistics which can be used for system management and quality evaluations. The middleware has to be reliable and always available since it provides interfaces to over- and under-lying layers, which can be used by programmers to shape the system according to their needs. Different resources can be geographically dispersed or hosted in the same data center. Furthermore, they can be interconnected. Regardless of the architecture of the resources, they appear to the user as one entity. The simplest structural unit aggregating and managing Grid resources is called Virtual Organization (or shortly VO) and can make use as well of other resources.

Grid Computing (G-C) enables the sharing, selection and aggregation by users of a wide variety of geographically distributed resources owned by different VOs and is well-suited for solving resource-intensive problems in science, engineering and other human activities. Grids are very large-scale virtualized and distributed computing systems. They cover multiple administrative domains enabling VOs to collaboratively share their resources and create an even larger Grid [8]. Grid technology has emerged from the scientific and academic communities and entered the commercial world leading to distributed systems called Utility Computing (U-C).

The concept of U-C is based on the fact that rather than operating servers

ISSN: 2279-8773

in-house, users (and/or VOs themselves) subscribe to an external U-C service provider paying only for the hardware and software resources they use. U-C relies heavily on the principle of consolidation, where physical resources are shared by a number of applications and users. The principal resources available include, though not being limited to, virtual computing environments (evaluable per hour and data transfer) and storage capacity (evaluable per GB or TB used).

In Cloud Computing (C-C), these services are hosted in a data center and commercialized. Accordingly, a wide range of software applications are offered by providers as a billable service (Software-as-a-Service, or SaaS) and no longer need to be installed on the users PC [9]. For small and medium-sized enterprises, the ability to outsource services and applications not only offers the potential to reduce overall costs, but also can lower the barriers to entry for many processing-intensive activities, since it eliminates the need for the initial investment and the necessity of maintaining dedicated infrastructure.

One main assumption in C-C consists of infinite computing resources available on demand and ubiquitous and unmetered access to broadband Internet. Applications available in the C-C include software suites that were traditionally installed on desktops and can now be found in the Cloud and are, for example, accessible via a Web browser. This paradigm may save license fees,

ISSN: 2279–8773

costs for maintenance and software updates which make it attractive to small businesses and individual users. The continued and successful deployment of C-C presents other challenges, including issues of privacy, security, liability, access and regulation since it operates across borders, and raises jurisdiction and law enforcement issues similarly to those of Internet itself. For example:

- *Reliability and Liability*: as with any other telecommunications service, users expect the Cloud to be a reliable resource, especially if a Cloud provider takes over the task of running "mission-critical" applications and expect as well a clear delineation of liability if serious problems occur. Data integrity and the correctness of results are other facets of reliability. Erroneous results, data loss or alteration due to service disruptions can have a negative impact on the business of the Cloud user. The matters of reliability, liability and QoS can be determined in Service-Level Agreements (SLA)s;

- *Security, Privacy and Anonymity*: It may be the case that the levels of privacy and anonymity available to the user of a Cloud will be lower than that of user desktop applications. To protect the privacy of Cloud users, care must be taken to guard the user's data and applications manipulating that data. Since the physical infrastructure in a distributed computing environment is shared among its users, any doubts about

ISSN: 2279-8773

data security have to be cleared;

- *Access and Usage Restrictions*: The possibility of storing and sharing data in Clouds raises concerns not only about privacy but also about copyright, licenses and intellectual property. Clouds can be accessed at any time, by any user with an Internet connection and from any place. Licensing, usage agreements and intellectual property rights may vary in different participating countries and the fact that the Cloud hides these differences can be subject of problems. Governments will need to carefully consider the appropriate policies and levels of regulation or legislation to provide adequate safeguards for distributed computing.

The key to realizing the full benefits of G-C and C-C may well lie in standardization, particularly in the middleware layer and the area of resource interconnection. In addition to the questions discussed above, the users of G-C and C-C infrastructures are also likely to be concerned with portability and interoperability. Portability is the freedom to migrate data on and off the Clouds of different providers without significant additional effort and switching costs and it should be of clear concern in a standardization process. Furthermore, standardized solutions for automation, monitoring, provisioning and configuration of Cloud and Grid applications need to be found in order to provide interoperability (that is the possibility for the users to em-

ISSN: 2279-8773

ploy infrastructure and services from different providers at the same time).

2.2 The European Projects in Grid Computing.

The European Union has been a major proponent of G-C. Many projects have been funded through the Framework Programme (FP) of the European Commission (EC) [10]. Two of them deserve special mention: the Business Experiments in Grid (BEinGRID) [11] and the Enabling Grids for E-sciencE (EGEE) [12]. The former was a research project partly funded by EC as an Integrated Project under the 6^{th} FP sponsorship program. Started in June 1, 2006, the project lasted 42 months, till November 2009. According to the project fact sheet, its mission was ”to establish effective routes to foster the adoption of Grid Computing across the EU and to stimulate research into innovative business models using Grid technologies”. The latter was based in the European Union although sites in Asia and the United States were also included. It was a follow-up of the European DataGrid (EDG) [13] project starting early 2001 and it has been arguably the largest computing quality Grid for scientists on the planet. This, along with the LHC Computing Grid (LCG) [14], has been developed to support the experiments using the CERN

ISSN: 2279-8773

Large Hadron Collider (LHC) [15]. The LCG project is driven by CERN which needs to handle huge amounts of data, where storage rates of several GB per second (about 10 PB per year) are required. In parallel to this major effort, many regional or scientific Grid activities, projects and other disciplines have contributed to the Grid infrastructure ecosystem available in Europe nowadays. Furthermore, a second pilot application domain has been selected to guide the implementation and certify the performance and functionality of the evolving infrastructure: Biomedical Grids, where several communities are facing the equally daunting challenge of coping with the flood of bioinformatics and healthcare data.

Moreover, the EGEE project community has been divided into 12 partner federations, consisting of over 70 contractors and over 30 non-contracting participants covering a wide-range of both scientific and industrial applications. The EGEE project (also funded by the EC) has put together experts from more than 50 countries with the common goal of building on recent advances in Grid technology and of developing a service Grid infrastructure available to scientists 24 hours-a-day. The project has provided researchers in academic institutions and business enterprises with access to a production level Grid infrastructure independently of their geographic location. The EGEE project has been also focused on attracting a wide range of new users to the Grid. Accordingly, the project's main focus was:

ISSN: 2279-8773

- To build a consistent, robust and secure Grid network attracting additional computing resources;
- To continuously improve and maintain the middleware in order to deliver a reliable service to users;
- To attract new users from industry as well as from science and ensure they receive high-standard training and support.

The EGEE Grid has been built on the EU Research Network GEANT and has leveraged on Grid expertise generated by many EU, national and international Grid projects.

During the third and last phase EGEE has focused its attention mainly on the expansion of the production Grid infrastructure and support to other user communities by adding further computational and data resources. The intention has been that of preparing the migration of the existing production European Grid from a project-based model to a sustainable federated one so as to create multi-national and multi-disciplinary infrastructure based on National Grid Initiatives.

All the three phases of the EGEE project (ended in May 2010) and related European Grid infrastructure projects have rested on three (funding) pillars:

- National or institutional sources for computing infrastructures;

ISSN: 2279–8773

- EC Contributions;
- National funds matching.

Nowadays, the distributed computing infrastructure built and nurtured by the projects DataGrid (2002-2004), EGEE-I, II and III (2004-2010) is supported by the EGI project. Stakeholders of this long-term organization are the National Grid Initiatives (NGI)s, which constitute the different country-wide building blocks of the pan-European Grid. Transitioning the care of the infrastructure from EGEE to EGI is part of the process for ensuring a vibrant and sustained European research platform.

2.3 The present European Grid Initiative.

As mentioned above, one of the main goal of EGI is to establish a sustainable Grid infrastructure for science in Europe in place by the end of EGEE-III and based on the vision of a large pan-European distributed computing and data Grid infrastructure responding to the needs and requirements of the European Research Area (ERA) [16] community. ERA provides a unified framework for the selection and realization of the best scientific projects avoiding the multiplication of similar parallel efforts in all member states, leading to large synergies, economies of scale and a dimension for European research to remain globally competitive. As a matter of fact, a pan European

ISSN: 2279-8773

e-infrastructure is required to support the research projects of the ERA in many research disciplines and to enable them to easily share a range of national resources (compute, storage, data, instruments) and ease their efforts to attain a global dimension.

The basis of EGI are the NGIs established in each participating European country which provide at national level the services for a seamless, shared and uniform access to a variety of computing resources, ranging from PC clusters to sites also operating supercomputers and all sorts of scientific archives. EGI is composed of the NGIs and a central (mostly coordinating) part, called EGI Organization (EGI.org) [17], as shown in Fig. 2.1.

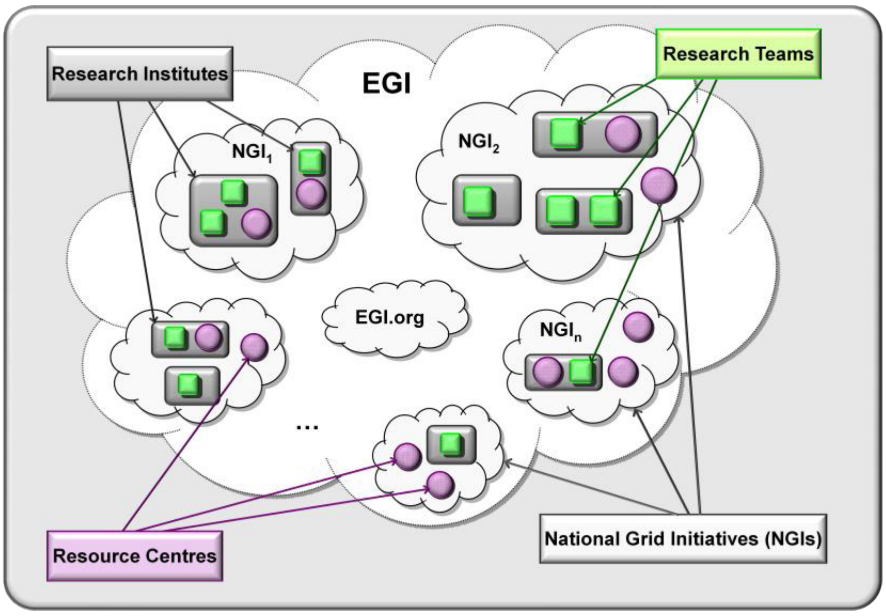


Figure 2.1: **EGI and the NGIs.**

ISSN: 2279–8773

The central pillar of EGI are its users, the scientists. A scientist is affiliated to a Research Institution (RI). Scientists from various scientific disciplines organize their scientific work in Research Teams (RT)s. To use the Grid, RTs establish through the RIs one or more VOs according to their needs. VOs refer to a dynamic set of individual and/or research groups defined around a set of resource-sharing rules and conditions. Typically, all VOs share a specific application domain as well as common concerns and requirements. They may, however, vary in size, scope, duration, sociology and structure. VOs often are (as in EGEE) international and linked to multiple funding agencies. For this reason, they bear specific representations of their needs in EGI.

The tasks of the NGIs are:

- Authentication of individual users;
- Allocation of project or discipline collaboration members to VOs in case of resource sharing;
- Allocation of computing resources to VOs for members' use;
- Authorization to VOs for running jobs, storing and retrieving data;
- Distribution and scheduling of computing jobs, workflows, data retrieval and access requests to authorized computing resources;

ISSN: 2279–8773

- Monitoring of stored data by users and of submitted and processed jobs;
- Accounting of users and VOs in allocation and usage of computing resources;
- Reporting to each NGI on their resource allocation to VOs as well as on the use of those resources by individual users so as to enable NGIs and national funding bodies to account funds usage against VOs' research results;
- Managing in a coordinated fashion software updates and hardware upgrades while continuing the service.

Coherence between the NGIs for the benefit of the international user community is ensured by EGI.org. Although committed to ensure the continuity of service for current users of the European Grid infrastructure, EGI is not just a continuation of EGEE (or of other infrastructure projects in which direct agreements between resource providers and research institutions define resource allocations). EGI is, in fact, built on the NGIs of the member states which own the resources and provide the local user communities. EGI.org will not provide IT resources although it will enable a coherent access, interworking and accounting between national Grid infrastructures. Accordingly,

ISSN: 2279-8773

the EGI.org key role is to ensure pan-European Grid coordination and integrity, aiming at standardization wherever possible. As a matter of fact, EGI.org links existing NGIs and actively supports the set-up and initiation of new NGIs where none exist. Within the EGI partnership, EGI.org and the NGIs work together to operate and further develop a sustainable pan-European Grid infrastructure, enabling optimal sharing of computing and data resources. Furthermore, the relationships between them are governed by the subsidiarity principle, meaning that tasks that are more effectively performed at the national or regional level should be left there.

Main functions needed for accomplishing the primary goal of EGI are:

1. *Operations and Security*: the services required to provide a large scale production quality Grid for science in Europe;

2. *Middleware*: ensuring maintenance, support and interoperability of the middleware currently deployed on the e-Infrastructure;

3. *User Community Services (UCS)*: providing the essential interface and basic support for existing and new user communities;

4. *External Liaison Functions*: providing dissemination as well as the link to other Grids and related organizations.

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

According to item 1, the operations and security function includes those EGI services needed to ensure optimal functionality of the pan-European infrastructure and the overall seamless effective interoperation of national and regional Grids. In addition, a common authentication trust domain is required to persistently identify all Grid participants. In a European e-Infrastructure, coordination will be required on security policies and operational security to support and coordinate the work of teams drawn from the NGIs.

According to item 2, it is necessary to further develop the underlying middleware for the European Grid. In particular, the middleware consortia ARC [18], gLite [19] and UNICORE [20] propose to foster the convergence of their current solutions into a Unified Middleware Distribution (UMD), similar to what the Virtual Data Toolkit (VDT) in the US is for the Open Science Grid (OSG). Moreover, UMD is a pragmatic way to coordinate at the European level the current independent and parallel developments avoiding duplication of effort. Accordingly, UMD will contain the necessary high-quality middleware components satisfying the strict policies, interoperability standards and quality criteria defined by the EGI.org Middleware Unit and endorsed by the Middleware Coordination Board (MCB).

The set of services included in UMD expands and evolves according to the requirements of European research communities and the operational needs of the resource providers. Components from sources other than ARC, gLite

ISSN: 2279-8773

and UNICORE can also be part of UMD if requested by users or NGIs, following the same rules agreed by the MCB. UMD will also deliver stable documented interfaces that will enable the development and the contribution of additional higher level services by third parties.

According to item 3, EGI supports the European communities using the e-Infrastructure by offering them collaborative support in the following areas:

- Gathering requirements from the user communities and providing efficient channels for their representation to the official bodies;
- Carrying out a review process to integrate useful external software not part of the core middleware distribution(s);
- Establishing Science Gateways that expose common tools and services (e.g. Workflow Engines, Web and Grid Services) in a transparent and user-friendly manner to user communities in the various disciplines;
- Establishing technical collaborations with the large European research infrastructure projects (e.g. ESFRI [21]) in support of users of the European research organizations;
- Providing linking services for collaborating projects, to streamline information management tasks and ensuring continuity of service between project cycles (e.g. maintenance of repositories, Frequently Asked Ques-

ISSN: 2279–8773

tions (FAQ)s and Wikis);

- Maintaining a European Grid Application Database that allows applications to be registered, permitting people to search for similar applications and contact the authors for guidance;
- Organizing European events such as the User Forum meetings and topical meetings for specific user communities;
- Providing services for new communities (e.g. front-desk services and VO creation counseling);
- Ensuring that the user communities and Grid administrators are provided with high-quality documentation and training services.

All this activities are carried out mainly by the NGIs in the context of a structured network of UCS as described in the next section.

According to item 4, a small team executes the dissemination activities of the EGI.org focusing on content production and coordinating activities. Technical and specific services are preferably bought from third parties.

For the successful launch of EGI initial co-funding by EC has been necessary. The major purpose of this co-funding has been to bring all the NGIs together, not to substitute for national funding which is the basis of EGI financial stability and Sustainability. Full Sustainability of the EGI opera-

ISSN: 2279-8773

tions should be eventually achieved using national funding only, helped by the expectation that effort to operate the Grid can be gradually decreased thanks to streamlining and automation. Moreover, in order to maintain the competitiveness of the European knowledge-based economy it is essential to ensure a steady stream of scientific and technological breakthroughs from research. This will provide the innovations to be possibly exploited in order to drive the economy, as recommended by the Lisbon Strategy agreed by EU leaders in 2000. To this end, Grid infrastructures offer the possibility to researchers involved in a project to work seamlessly together within a country and also across countries. This allows them to share the different Information and Communication Technology (ICT) resources allocated to the project in a managed way still allowing their owners to keep control over them while ensuring their usage in an optimal way.

2.4 The Heavy User Communities (HUC)s.

To support science and innovation a new operational model for e-Science is needed both for coordinating the infrastructure and for delivering integrated services that cross national borders. In this respect, the EGI-InSPIRE project (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe) [22] is a collaborative effort involving more than 50 institutions

ISSN: 2279-8773

(including 37 NGIs) in over 40 countries. Its mission is to establish a sustainable European Grid Infrastructure. Accordingly, the ultimate goal of EGI-InSPIRE is to provide European scientists and their international partners with a sustainable, reliable e-Infrastructure that can support their needs for large-scale data analysis. This is essential in order to solve the big questions facing science today and in the decades to come. The four-year project is funded by the 7^{th} Working Package (WP), will support Grids of High-Performance Computing (HPC) and High-Throughput Computing (HTC) resources and is ideally placed to integrate new Distributed Computing Infrastructures (DCI)s such as Clouds, supercomputing networks and desktop Grids and to benefit the user communities within the ERA. As a matter of fact, EGI-InSPIRE will collect user requirements and provide support for the current and potential new user communities (e.g. ESFRI projects). EGI-InSPIRE will support as well the current heavy users of the infrastructure, such as High Energy Physics (HEP), Computational Chemistry (CC) and Life Sciences (LS), as they move their critical services and tools from a centralized support model to one driven by their own individual communities. The objectives of the EGI-InSPIRE project are:

- The continued operation and expansion of todays production infrastructure by transitioning to a governance model and operational infras-

ISSN: 2279–8773

tructure that can be increasingly sustained outside of specific project funding;

- The continued support of researchers within Europe and their international collaborators that are using the current production infrastructure;
- The support for current heavy users of the infrastructure in Earth Science, Astronomy and Astrophysics, Fusion, CC and Materials Science Technology, LS and HEP as they move to sustainable support models for their own communities;
- Interfaces that expand access to new user communities including new potential heavy users of the infrastructure from the ESFRI projects;
- The support of mechanisms to integrate existing infrastructure providers in Europe and around the world into the production infrastructure, so as to provide transparent access to all authorized users;
- The development of processes and procedures to allow the integration of new DCI technologies (e.g. Clouds) and heterogeneous resources (e.g. HTC and HPC) into a seamless production infrastructure as they mature and demonstrate value to the EGI community.

ISSN: 2279–8773

Accordingly EGI.org, coordinator of the project, brings together partner institutions established within the community to provide a set of essential human and technical services that enable secure integrated access to distributed resources on behalf of the community. This process is particularly targeted to research activities with the goal of constituting VRCs meant to establish homogeneous aggregations of user clusters, VOs and research institutions representing the driving force for the establishing of Grid technologies in a given research domain. Indeed, the EGI production infrastructure will be addressed to support these structured international user communities devoted to research activities which will be formally represented within EGI at both technical and strategic level.
Within the EGI-InSPIRE project, WP6 (Services for the Heavy User Community) is providing continued support for former activities of EGEE during their transition to a sustainable model within by:

- Supporting the tools, services and capabilities required by different HUCs;
- Identifying the tools, services and capabilities currently used by the HUCs that can benefit all user communities and to promote their adoption;
- Migrating the tools, services and capabilities that could benefit all user

ISSN: 2279-8773

communities into a sustainable support model as part of the core EGI infrastructure;

- Establishing a sustainable support model for the tools, services and capabilities that will remain relevant to single HUCs.

Some of the potential results from this activity include: dashboards customized to specific VOs, workflows and schedulers bridging different DCIs, support of Message Passing Interface (MPI) [23], frameworks for managing collections of jobs on DCIs, services for accessing relational data resources, secure data storage and visualizations tools. Accordingly, WP6 identifies the following shared services and tools:

- *Dashboards*, in particular EDMS [24] that provide a generic framework to monitor sites and their services within a VO using tests specific to that community. As a matter of fact, Dashboards have emerged from within the HEP community and are now being adopted by the LS community to monitor their resources;

- *Applications*, in particular GANGA [25] and DIANE [26] tools that are both part of the EGEE RESPECT programme [27] which recognizes software that builds on top of the gLite platform. Although initially developed for the HEP community, these tools have now gained traction

ISSN: 2279-8773

Virt&l–Comm.3.2012.22

in other communities as they provide simple environments to manage large collections of tasks;

- *Services*, HYDRA [28] and GReIC [29] are services that have emerged from a single community that show potential for adoption in others. The former allows an encryption key to be securely stored on distributed servers in order that storage elements can be used to store confidential data which is critical for the medical community securely. The latter provides uniform relational and non-relational access to heterogeneous data sources and is currently being used to support bioinformatics and Earth Observation Systems;

- *Workflows and Schedulers*, these tools are critical in integrating complex processes, generally involving multiple data sources and different computational resources, as needed within many disciplines. In particular, SOMA2 [30] is a Web-based workflow tool used for computational drug design and general molecular modeling while TAVERNA [31] is used extensively by the bioinformatics community. Moreover, the combination of the Kepler workflow engine [32] and the Migrating Desktop platform [33] are used by the Fusion community to run workflows requiring visualization and interactive access on gLite and UNICORE-enabled resources. In addition, for simpler workflows

ISSN: 2279–8773

and meta-scheduling scenarios the GridWay system [34] is used by the Fusion community. In this respect, special efforts are provided to maintain the integration of these tools with the different systems;

- *MPI*, support for parallel computing applications are critical for many user communities but the integration of this capability into the general infrastructure has been difficult. This task will focus on the improvement of the core services and software needed to support MPI, while engaging with two representative user communities (CheMIST and Fusion) to ensure that the offered support meets their requirements.

ISSN: 2279-8773

Chapter 3

A Grid Approach to Computational Chemistry

3.1 METACHEM and GRIDCHEM COST Actions.

The birth and initial evolution of the CC community to G-C has started within COST [35]. COST is an intergovernmental framework for European Cooperation in Science and Technology allowing the coordination of nationally-funded research on a European level, contributing to reduce the fragmentation in European research investments and opening ERA to worldwide cooperation. The goal of COST is to ensure that Europe holds a strong

ISSN: 2279–8773

position in the field of scientific and technical research for peaceful purposes, by increasing European cooperation and interaction in this field. This research initiative makes it possible for the various national facilities, institutes, universities and private industry to work jointly on a wide range of Research and Development (R&D) activities organized under the form of Actions.

Chemistry is a central basic science domain with a distinguished pedigree and success in Europe (23 of the 55 Nobel Laureates in Chemistry awarded since 1960 are European). Moreover, Europe has a very strong industry bearing a distinguished harvest of successes (8 of the 10 largest chemical companies in the world are based in Europe). Accordingly, in order to maintain and even to improve this position of the European Chemical Science and European Chemical Industry, it was natural for European chemists to join COST and organize themselves to elaborate a strategic scientific scheme for basic collaborative research in Chemistry. The Chemistry domain in COST was named CMST.

Owing to the progress of European scientists (especially physicists) in using the Grid for developing science knowledge and technologies in the late 90's, the CMST decided to start an Action devoted to the application of such technology to chemical knowledge and research. The implemented Action was D23 "METACHEM: Metalaboratories for complex computational applications in Chemistry" that was later followed by D37 "Grid Computing in

ISSN: 2279-8773

Chemistry: GRIDCHEM".

The main objective of D23 (ended in 2005) was the exploitation of the potentialities of meta and Grid Computing for developing computational applications, connecting the scientific know how distributed among several research laboratories and sharing the related computer resources. The sharing of machines on the network has been boosted by the dramatic development of Grid technologies in recent years and by the emphasis given by FP6. The further added value of D23 has been the building of the so called European Meta Laboratories (clusters of geographically distributed Laboratories working in a co-ordinated way on a common project by sharing manpower, hardware and software) and fostering innovative solutions for chemical applications and a new paradigm for collaborative research thank to the use of Grid computer systems.

As a result, progress has been made in assembling computational procedures for the a priori determination of complex molecular structures and processes in order to face the needs for innovations in several scientific and engineering fields. Progress has also been made in developing shared models for ab initio data in order to enhance its reuse and interoperability and in building Grid-based simulators (like for example SIMBEX [36]). An additional focus of the D23 activities has been Grid-based e-learning. For these activities, D23 has gathered the collaboration of 46 research groups from 19 different countries

ISSN: 2279-8773

paving the way for the launch of the subsequent more Grid technology biased COST Action D37.

The main objective of D37 (final evaluation meeting held on January 20-21, 2011) has been the preparation of the ground for gathering and launching the COMPCHEM VO, (described in detail in the next section) using a subset of the distributed computing infrastructures of the Grid of EGEE. The areas of application covered by D37 are concerned with new methodological approaches to molecular sciences, innovative materials science, molecular biology and environmental Chemistry computational applications. The presence of Chemistry on the Grid has significantly impacted also the development of middleware and the design and layout of Grid infrastructures for molecular sciences.

3.2 COMPCHEM and TELCHEM Virtual Organizations.

The open nature of the above mentioned distributed projects has increasingly prompted the exploitation of Grid technologies and collaborative computing among geographically dispersed scientific laboratories having complementary know-how and interests. This form of cooperative aggregation, originally

ISSN: 2279-8773

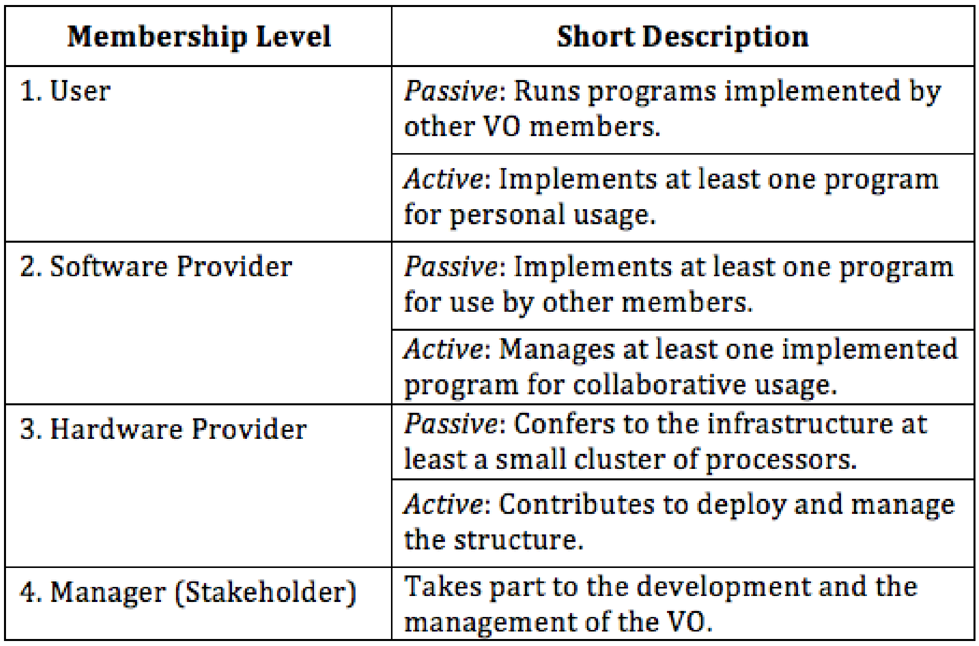
called Metalaboratory (as in the already mentioned Action D23), has gone more recently under the name of VO, as quoted in the previous chapter. In a VO a dynamic collection of individuals, institutions and resources are regulated according to shared standards so as to utilize unique authentication, authorization, resource access, resource discovery and other advanced computational challenges. Another definition of VO reads, in fact, as "a corporate, not-for-profit, productive organizational entity which uses telecommunication tools to enable, maintain and sustain member relationships in distributed work environments". Critical management dimensions are those applying to the spatial (physical distance between members), temporal (working hours scheduling) and configurational (activity inter-connections) aspects of member relationships in the work environment. Moreover, a VO comprises a set of independent organizations which share resources and skills to achieve their missions, yet not limited to an alliance of for profit enterprises.

As already mentioned, as a follow up of Action D23 a CC VO named COMPCHEM has been registered and its members have begun to operate on the production Grid of the EGEE project. At present, 30 Grid sites support COMPCHEM with their computing facilities for a total of about 10000 processors.

Crucial to the success of a VO is the development of a Sustainability plan [37] especially in fields like basic research and science education in which funding

ISSN: 2279-8773

is scarcer than for science and technology applied research. For this purpose, COMPCHEM has adopted the policy of offering to its members the possibility of trading their support to the VO activities in return for a free use of the VO resources. Accordingly, the entry level of COMPCHEM offers to the researcher the possibility of implementing a code just for personal use (see Table 3.1 where the level articulation of the COMPCHEM VO has been sketched).



Membership Level	Short Description
1. User	*Passive*: Runs programs implemented by other VO members.
	Active: Implements at least one program for personal usage.
2. Software Provider	*Passive*: Implements at least one program for use by other members.
	Active: Manages at least one implemented program for collaborative usage.
3. Hardware Provider	*Passive*: Confers to the infrastructure at least a small cluster of processors.
	Active: Contributes to deploy and manage the structure.
4. Manager (Stakeholder)	Takes part to the development and the management of the VO.

Table 3.1: **Levels of Membership in COMPCHEM (*an updated version of the membership levels originally described in ref. [37] and of the subsequent Table originally published in ref. [38]*).**

However, the entry level of COMPCHEM is meant to be a kind of pre-membership situation in which it is possible to launch and run jobs even

ISSN: 2279-8773

without getting the VO membership certificate (further details will be given later). Permanence at this level is expected to have a limited temporal extension necessary for screening the laboratories on their real willingness to operate on a Grid platform. Already at this level, in fact, a minimum amount of competences necessary to restructure the code to run in a distributed way and exploiting the advantages of using a Grid platform needs to be acquired. In return, one gets the advantage of running the code on a large number of processors and more easily interacting with the codes of other users of the VO. As already mentioned, a user may become an actual member of the COMPCHEM VO only after committing him/herself to open the code implemented on the Grid to a shared usage by other members of the VO (from this level onwards one can start claiming a preferential treatment from the organization). This implies the validation of a stable version of the code and the assemblage of all the necessary Graphical User Interfaces (GUI)s for its friendly usage by other researchers. It also implies software maintenance and user support. It may also require the commitment to confer to the Grid additional hardware (especially for those suites of codes which need special devices) after a negotiation with the Management Committee (MC) of the VO about the relevance of such a commitment to the strategic choices of the VO. Obviously, the conferring of both software and hardware to COMPCHEM will take place gradually due to the time needed to validate

ISSN: 2279-8773

the software and to gridify the applications. The status of COMPCHEM member may lead to further levels of involvement. VO members, in fact, are welcome to take care of maintaining the local software and segment of Grid hardware (particular attention is needed for the conferring of software, either commercial or not, bearing special constraints like the payment of fees, since in this case, commercial, legal and financial aspects are better dealt centrally).

The foundations of such an organization rest, therefore, on the fact that any contributed software needs to be structured following a Service Oriented approach, which is the basic concept of the SOA. SOA expects that every application has to be designed to support interoperable machine-to-machine interaction over a network. In addition, all of the programs (defined as Web Services) have to expose one or more interfaces, described in a machine-processable format called Web Services Description Language (WSDL) [39], to interact with the other ones. The interaction takes place in a manner prescribed by Simple Object Access Protocol (SOAP) [40] and its messages are typically conveyed through the HTTP protocol with an eXtensible Markup Language (XML) [41] serialization in conjunction with other Web-related standards.

The main purpose of a Web Service approach is to provide some functionalities implemented by a user (which could be an institution, an organization

ISSN: 2279-8773

as well as a person) on behalf of all the other users. In particular, the user providing the appropriate software to implement a particular service is the Provider while the user wishing to make use of a provider's Web Service is the Consumer. Consequently, the monitoring activities associated with the application of a Web or a Grid Service approach allow a possible regulation of the internal and external activities of the VO through Credit assignment and redemption. This means that members should get Credits (also called "terms of exchange credits", or *toec* [42]) for the resources they make available to COMPCHEM and can redeem them either by acquiring services from the VO or by trading Credits for financial resources.

In this respect, education is a typical example of activities in which there is a high demand for sharing resources and developing collaboration, but scarce funding. Accordingly, there is a dramatic need for establishing active communities able to stimulate the growth of knowledge in a fashion transversal to economic, cultural and geographic areas. Unfortunately, in the developed areas limited interest and infrastructures are available for this purpose. At the same time, in developing economies, other services are believed to be more basic than education when considering the allocation of limited resources. However, since knowledge is a commodity that needs mainly brain-ware (and not heavy industrial plants) and the contemporary society intensively exploits knowledge products (for which increasing development and deployment of in-

ISSN: 2279-8773

formation and communication technologies is made in any case) most of the tools necessary for educational purposes are often produced by ICT developments in other fields. As a matter of fact, it has been already found in the past (as in the case of the Web and Internet) that educational activities have a high capacity of reusing and recycling knowledge technologies designed for other purposes.

Following this line, the exploitation of the innovative features of Grid paves also the way to new Teaching and Learning (T&L) distributed forms of collaboration among Chemistry teachers and students and to the sharing, as well, of manware (teachers and students themselves), educational tools and contents (called Learning Objects (LO)s or T&L Units). To proceed towards the structuring of a new VO in education, several other aspects like members/services agreement, usage policies and Sustainability issues have to be considered as well as a proper restructuring of all the applications already developed in this domain. This means that COMPCHEM itself could act as an incubator for a new T&L dedicated VO (called TELCHEM, Teaching & Learning in CHEMistry, in [42]).

After all, various types of activities bearing different levels of virtuality and already implemented in COMPCHEM can be borrowed for T&L purposes. The first level of virtuality is, in fact, that of the nano world where virtual refers to atomic and molecular objects as well as to related tools and exper-

ISSN: 2279-8773

imental procedures dealt as real objects at a macro level. The second level of virtuality is that of laboratories, T&L Units and classes (e.g. of human dimension virtual objects). These are sometimes merely based on text documents though, as already mentioned, multimedia technologies are becoming increasingly popular and have made the virtual educational objects more and more realistic thanks to hypertexts, videos, audios, animated materials and rendering devices which already need synergistic efforts of different experts. The third level of virtuality of key relevance here is that of the organizational dimension, in which the term "virtual" refers to the organization (that is to geographical and juridical aspects) which are, at the moment, the least developed ones.

The TELCHEM VO will have the mission of taking care of the whole process in which knowledge is imparted either by a tutor, or a knowledge provider or an expert (of any kind) to a student (or, more in general, to a knowledge recipient who at a certain point becomes active and starts a learning process of which he/she defines (and perhaps designs and implements) the necessary tools). In all cases, on-line tutoring should satisfy some requirements like broadband Internet access, audio microphone and speaker, a shared screen on which the student and the tutor can write (it is very important for effective teaching and it can be done through the use of collaborative software commonly called whiteboards or dashboards), a digital pen mouse (for writ-

ISSN: 2279–8773

ing, drawing and highlighting text or dealing with mathematical equations) rather than a webcam or digital video camera for physical demonstrations or high-quality visual feedbacks.

Tutoring services typically offered on-line by TELCHEM should be based on the integration of all software components of e-Learning (instructional, demonstrative, collaborative and individual research) in which instructions and demonstrations are provided by tutors, collaboration is administered through on-line discussion or chat group with peers while individual research is developed through materials made available separately from the on-line instructional components. This requires also a video and audio real time service. Video should essentially provide visual feedback to learning that should be accompanied by audio for ease-of-use (without additional phone bills and the inconvenience of dealing with two separate devices). Accordingly, typical on-line audio based on Voice Over IP (VOIP), which is a mature Internet technology, will have to be employed.

Furthermore, teachers need the ability to understand a subject well enough to convey its essence to a new generation of students: the goal here is to establish a sound knowledge base on which students will be able to build new acquisitions as they are exposed to different life experiences. Accordingly, some courses will be designed for teachers to provide them with the necessary skills to operate in this environment. Moreover, course materials will be

ISSN: 2279–8773

certified and archived in a common shared and distributed digital library or databases.

As a matter of fact, a notable initiative that could be associated with TEL-CHEM is that of the Working Group 5 of the European Chemistry and Chemical Engineering and Education Network (EC2E2N) [43] project, funded by the EC aimed at developing a Virtual Campus for Chemistry and Chemical Engineering. EC2E2N is a three-year project that, in conjunction with the European Chemistry Thematic Network Association (ECTNA) [44], investigates the exploitation of modern networking and computing technologies to support T&L activities for Chemistry and Chemical Engineering higher education, to develop relationships among the Universities of the project based on the model of a Community of Practices (CoP) for education by adopting networked open and distance digital technologies), to design a system for a distributed management of the specific features of Chemistry and Chemical Engineering T&L practices to integrate local front teaching with Web-based activities, to identify the best way of making shareable the educational material regardless of the place of origin (distributed shared repositories).

The link of educational CoPs with the Grid and VOs becomes obvious if one considers the computing power necessary to empower the molecular simulations embedded into several Chemistry LOs, the network infrastructure needed to operate in a coordinated fashion and the developing of suitable

ISSN: 2279-8773

complex multilevel virtual realistic environments indispensable for building virtual experiences. Such link is also obvious due to the need of educational CoPs for a collaborative environment to produce suitable T&L material, shared repositories and various telematic activities all meant to facilitate the interaction between students and teachers.

An important feature of TELCHEM will be the focus on e-tools (both open-source and proprietary software although preference will be given to the former). Among the open source software there will be G-LOREP [45], the knowledge management system tool being developed to handle distributed repositories.

Another important open source software targeted to make TELCHEM sustainable, will be the object of this Thesis work through the development of a community economy whose main asset will be R&D in T&L. As a matter of fact, in TELCHEM R&D in T&L will be considered as a resource in itself independently of its short term productivity (though not independently of its efficiency and efficacy). For this reason, R&D in T&L will not only be treated as a specially protected area for free circulation of ideas and innovation but it will also be financially supported regardless of the revenues it can generate. Contributions to R&D are often evaluated on the basis of regular internal reports and articles in international scientific journals. This will apply also to TELCHEM though particular emphasis will be put on the stimulation

ISSN: 2279-8773

and creation of new ideas in T&L as well as on innovation. To this end, the design and development of tools evaluating the cost of implementing and validating further research evolution that starts from the existing products is favored. In this respect, *toecs* from a specific budget will be assigned on the basis of the trust that other members of the community give to R&D activities as a key manifestation of the community value. This can be used as an incentive to stimulate particularly valuable contributions (including the knowledge gained with failures, failure recovery and retrieval of lost information) typical of research activities. Specific Credits will also be assigned to education, training, dissemination and marketing aimed at making a VO a true knowledge company.

For this reasons, in conjunction with a Web Service approach, also a Collaborative Service aimed at managing a Grid Credit System will be discussed within the present Thesis work. Indeed, the goal of this service is that of considering several elements (such as the users reputation, the quality of the users activities, the results obtained, the kind and the number of resources managed, the support offered to the VO, the stable and/or innovative services carried out, the use of them and the related feedback) to implement one or more policies concerning the *toecs* assignment for stimulating the users activity within the VO.

ISSN: 2279-8773

3.3 The CheMIST community.

To better focus the object of this Thesis work we shall confine our attention to the already mentioned Chemistry and Molecular Innovation Science and Technology (CheMIST) community as it has been described in the recent homonymous proposal to the FP7-INFRASTRUCTURES-2011-2 (1.2.1: e-Science environments) call [46]. The CheMIST community is built around two pan-European Grid VOs (GAUSSIAN and COMPCHEM) constituted during the final years of the EGEE project [47] using the gLite middleware and both counting on about one hundred members. GAUSSIAN is mainly devoted to implementing and maintaining ab initio packages and to providing also support on the Grid to computational chemists carrying out their research using mainly commercial electronic structure software (as is the popular Gaussian package) for predicting in an ab initio fashion the electronic properties of molecules and molecular aggregates. COMPCHEM is mainly devoted to implementing codes and providing support on the Grid to computational chemists carrying out their research by combining structural, dynamical and kinetics studies of chemical processes using components of ad hoc workflows linking ab initio calculations to the experimental signal. The work exploits also cooperation activities of some working groups of the already mentioned Actions D23 and D37.

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

Other two VOs of similar consistency are involved in CheMIST: ENEA and MoSGrid. The two VOs have the characteristics of being operating in a particular context (a technology oriented industrial research center under public ownership using proprietary software the first, a cluster of University researchers using the UNICORE middleware [20] the second).

The core activity of the members of the mentioned VOs are focused on running the Quantum Chemistry and Molecular Dynamics packages (such as Amber, Charmm, CPMD, Gamess, Gaussian, Gromacs, Gromos, NAMD, Quantum Espresso and VASP). At the same time, intention to join the Grid activities of the community has been expressed by other geographically dispersed minor laboratories though they were not active in EGEE. Altogether, this makes a community of more than 500 people which refer to EGI as a coordination body for distributed computing.

The mission of CheMIST is therefore to act as a hub for the Chemistry and Molecular Science community by coordinating computing activities on heterogeneous platforms, providing support to the members, safeguarding related knowledge and technologies and preserving and maintaining as well a set of basic services. The scientific target of the members of the CheMIST community is the study of the properties of matter referable to the characteristics of the nuclei considered as stable (though structured) entities and as (from weak to strong) aggregates of atoms and molecules. The purpose of

ISSN: 2279-8773

such study is to invest the knowledge developed in this way to build new science, innovation, technologies and services for the general social and economic growth. Accordingly, the engagement of the CheMIST community in computation is addressed to a series of research activities, computer codes (either commercial or in-house designed and implemented), development, database construction and management, data rendering and virtual reality handling articulated into a large number of laboratories concurring in carrying out advanced modeling and simulations based on multi-scale and multi-physics approaches to reality starting, whenever is appropriate, from the nano-scale level. This means that there are several important applications (yet not a dominant one as in other scientific domains) and a large number of in-house developed methods and software packages with a high-innovative potential that benefit from the efforts of a large number of researchers and may serve an even much larger number of users.

The success of the CheMIST community is visible in the success of the implementation of quantum Chemistry packages (like the Gamess and Gaussian ones mentioned above), which are at present solid foundations of any determination of molecular structures and properties. Their applications range from materials design to photo-assisted processes rationalization and spectroscopic analytic studies. As a matter of fact, with the advent of the Grid, more of these studies are becoming distributed "flagship applications" even

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

if their natural (and original) computational platforms are supercomputers. Molecular dynamics codes are also another popular pillar of innovative research in computational science and technology. Some of the relevant codes have been implemented on the Grid both out of the family of programs treating the atoms in a classical fashion (like Venus, DL_Poly and Gromacs) and out of the family of quantum programs treating the atoms as waves. Also in this case, the natural and original platform for running quantum scattering calculations are supercomputers because of the high request of nodes memory. Nonetheless, the porting on the Grid has been successfully achieved for some quantum atom diatom programs using either Time Dependent (TD) approaches (like RWAVEPR, MCTDH and FLUSS) or the Time Independent (TI) one (like ABC [63]). Also the corresponding semi-classical programs have been considered for porting on the Grid.

Scattering efficiency of chemical reactions, however, is only one of the important chemical properties. To characterize most of the chemical processes of interest in practical applications thermal reactivity (the so-called rate coefficients) is the input needed (like in kinetics simulations of reactive systems). Most of the realistic modeling applications, however, take these calculations as building blocks of a multi-scale treatment.

For this reason, the prospect is to incorporate them into universal-like molecular simulators (like the so called Grid Empowered Molecular Simulator

ISSN: 2279-8773

(GEMS) based on a Grid workflow linking the Grid versions of ab initio, quantum dynamics and statistical averaging, that will be described in more detail in the next section) that can be used as a computational engine of molecular properties to be assembled with other computational packages to form more complex applications. As a matter of fact, the combination of above described molecular level studies with different scale and statistical treatments generates theoretical predictions of structure-property relationships, size of molecules effects, chain of molecular processes, inter-phase phenomena, interaction with light are all properties of interest among for chemists, materials technologists, engineers, biologists that can be evaluated in reasonable amounts of elapsed time only using G-C.

The main effort assigned to the CheMIST community within the EGI-InSPIRE project is the support to the use of MPI within its members. This is, however, only part of the real effort of the CheMIST community that is heavily involved in contributing to the activities of the HUC board by taking care of the porting to the Grid of the above mentioned packages and in-house developed programs and extending and consolidating as well the library of programs implemented on the Grid. In particular, a proposal called e-CheMIST has been submitted to the FP7 call mentioned above (INFRA-2011-2-1.2.1) to establish an e-Science environment devoted to:

ISSN: 2279-8773

(a) the offering of the most popular Grid tools for supporting HUCs including Dashboards, Science Gateways, Workflows and Schedulers engines at service level;

(b) the development of new tools and softwares at research level.

As to point *b*, e-CheMIST will take care of the development of tools like GEMS and the self assessment electronic tests EChemTest (that are described in the next section of this chapter). It will also take care of the further development of the versatile Framework GriF (see for details the following chapters in which the structuring of VRCs and the development of a Grid Credit System are discussed), a facilitator for the gridification of the programs on the Grid. GriF, take care also of evaluating the quality both of the work carried out by their users and of the services offered by the VO itself while not needing any knowledge of the Grid to submit jobs. Other domain specific new tools and softwares will be subject of research by e-CheMIST (e.g. distributed repositories of LOs, chemical labelers and materials designers) that will not be described here falling outside the scope of the present Thesis work.

As to point *a*, instead, CheMIST shall provide the following specific services:

- Support the activities of its members and of the members of other communities by means of Grid tools and services useful for that purpose

ISSN: 2279-8773

like portals, inter-VO services and inter-application workflows. In particular various workflows engines (like the already mentioned Kepler, Soma2 and Taverna) are considered in order to meet the complexity of several applications showing articulated branches among various suites of codes which often combine nested features of high-performance and high-throughput computing. The considered tools are meant to allow a friendly use of the Grid and to run on a large variety of platforms;

- Host and support VO specific services (like handling licensed software for GAUSSIAN and handling a Credit System for COMPCHEM as it has been already modeled by GCreS that will be described in detail in chapter 6). As to the licensed software handling although some solutions have been already experimented in the final period of EGEE III the problem still needs to be systematically investigated and a general solution to be implemented. As to the support for the development of a Credit System like GCreS (also aimed at fostering the collaborative implementation of CheMIST complex applications), a scheme of objective and subjective evaluation of providers and users activities has been foreseen. Moreover, it will be focused on the (objective and subjective) evaluation of the quality provided and of the suitability of the resources used (as well as of the analysis of the users attitudes) in order

ISSN: 2279-8773

to constitute a base for managing the Credit System that will be implemented. Specific training and support activities will be implemented for the members of the CheMIST community in order to enable them to efficiently implement related Grid applications by lowering the barriers introduced by the continuous and rapid evolution of distributed computer technologies (which remain hard to master for researchers of other fields). This implies also to train CheMIST scientists to design and implement new approaches to the solution of molecular and material problems as well as to develop even more complex (and at the same time more realistic) Grid enabled simulations. Finally, support for the use of a variety of visualization tools will be offered to exploit the Grid capability of providing increasingly realistic means for graphical representations of molecules and nano structures;

- Host Framework services (like Dashboards) for an efficient management of the CheMIST community. This support will be aimed at facilitating an integration of the activities of the different theoretical, computational and experimental laboratories. Support will be also given to the assemblage of collaborative tools for scientific investigation (like GEMS) designed to support the collaborative execution of realistic simulations relevant to either technological or environmental simulations and relevant as well to the integration on the Grid platform of a set of distributed chemical instrumentations and experiments. Such an integration is meant to support the institution of a Europe wide virtual chemical laboratory for either educational or experimental purposes;

- Host services for the Scientific Gateways (in particular, though not solely, for the CheMIST community) focused on the implementation of standardized data formats, the offer of computer applications as Web an Grid Services, the exploitation of graphical application and MPI libraries. The support to the implementation of de facto standards for molecular structural data calculation and representation will be based on ab initio electronic structure outputs generated by different Quantum Chemistry (QC) packages. This will make the assemblage of repositories, tools and workflows more user friendly and will encourage, as well, efforts in predicting, designing and analyzing Molecular and Materials structures and properties using multiple codes, across multiple disciplines and for an open user base. Due to the fact that QC databases (especially those concerned with drugs and Life Sciences) often spread worldwide and are private, facilitating their access on the Grid is of paramount importance and the use of appropriate Grid tools for handling them is necessary. This is for example the case of G-DSE

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

(Grid-Data Source Engine), one of the first tools aimed at integrating database resources and the Grid. Support will also be given to other tools like OGSA-DAI [48], AMGA [49] and GRelC. Another important support will be the one given to MPI, MPICH and other parallel instruments. Several CheMIST applications do in fact get significant benefit from being parallelized (especially if executed on HPC platforms) and therefore represent suitable candidates for exploiting the use of parallelization tools. Support will be given, therefore, to the efforts of building a suitable environment in which libraries of Grid enabled computer codes can be made available (possibly as Grid Services) since they represent the most valuable asset of the CheMIST community;

- Support interoperability among CheMIST applications operating on different middleware and/or needing platforms of different level. This support will be provided for both developed in-house and licensed software. For licensed software, again, no general policy has yet been developed despite the fact that a large fraction of commonly used QC applications are not open source. The problem is of concern not only for the gLite middleware but also for the ARC and UNICORE ones. The fact that also other disciplines depend in some way from commer-

ISSN: 2279-8773

cial and/or licensed software makes developed solutions of benefit also for other communities;

- Support for the transition of services of the CC cluster of EGEE to EGI services. This support will be vital in the first part of the project and will require a substantial communication effort not to loose part of the community. Actually, however, the transition period will be taken as an opportunity for structuring the community along the guidelines of the project.

Finally, the CheMIST community will provide other more general support to the Grid users in close contact with other EGI instances. These are: general coordination of the activities, validation of the proposed middleware services, fostering of the interoperability of the interdisciplinary codes on various platforms, support inter-domains application development and porting, maintain a front desk, ensure documentation, training and dissemination for the related science gateways. More in detail this implies for the community to support with its own services the current activities of the existing CC VOs, to help them in managing registered users and shared resources, to gather, maintain, consolidate and further develop other VOs, to spread to other domains the knowledge produced by the members of the CheMIST community, to foster interactions, relationships and resource sharing with other VOs, to

ISSN: 2279-8773

increase the number of people and entities using the CheMIST community services, to promote the use of computer resources and Grid technologies by the CheMIST community, to collaborate with EGI operations and middleware development groups to tailor a design and an implementation of Grid tools and services best suited to the necessity of the CheMIST community, to support for and help in the design of scientific projects and funding applications through a valorization of the services provided efficiently and to support its members with training, documentation and maintenance procedures (including the proper handling of trademark and commercial obligations for commercial packages).

3.4 The Grid Empowered Molecular Simulator: GEMS.

The Grid Empowered Molecular Simulator (GEMS) [38, 50] is a key innovative target of e-CheMIST because molecular simulations are the ground on which several modern technological and environmental research advances rely. For this purpose, several computational methods and related computer codes specifically designed to deal with molecular simulations of chemical transformations in an ab initio fashion have been developed in the

ISSN: 2279-8773

past [51–55]. The associated computational problems can be partitioned into a sequence of three computational blocks: INTERACTION, DYNAMICS and OBSERVABLES. This articulation of the problem is illustrated in more detail by the workflow sketched in Fig. 3.1.

INTERACTION is the block in which the ab initio calculations determining at various levels of accuracy the electronic structure [51, 52] of the molecular system are carried out within a Born-Oppenheimer scheme if one does not find (or does not want to use) an existing potential energy routine when available. After performing ab initio calculations (or collecting results available from the literature) the calculated energies (usually called potential energy values) are sometimes used directly as they are produced. Most often, however, ab initio calculations are first performed at the best affordable level of accuracy depending on the availability of an adequate amount of computer time and storage for the set of geometries of the molecular system necessary to describe the considered process. Then the calculated potential energy values are properly refined by calculating a subset of points at higher level of accuracy and adjusted accordingly (also to exclude non converged results and to reproduce some known molecular properties). At this point the resulting values are fitted to an appropriate functional form to generate an analytical Potential Energy Surface (PES). If ab initio calculations are unfeasible one can build an empirical force field.

ISSN: 2279-8773

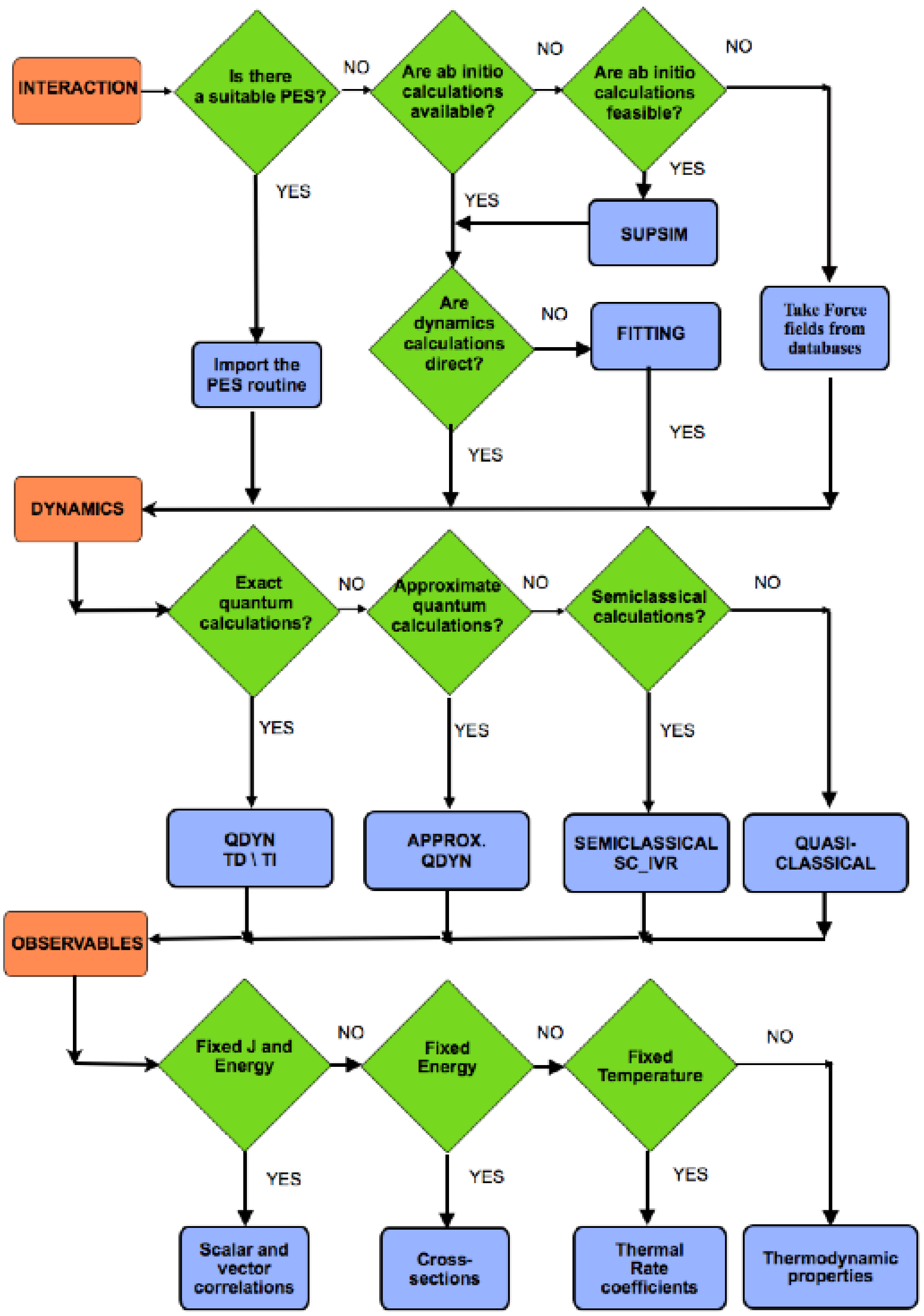


Figure 3.1: **GEMS Workflow.**

The first effort tackled when developing GEMS was the Grid implementation of this first block. The key computational task of this block is the execution of the programs (or suites of programs) carrying out the electronic structure calculations. Our efforts have led to the development of a module called SUPSIM [56] that has been tested for atom diatom systems when using some well known packages (such as GAMESS US [57], DALTON [58] and MOLPRO [59]) implemented at moderately high-level. Higher level implementations of ab initio codes inevitably require the availability on the computing platform of HPC resources (which are presently scarcely available on the EGI Grid platforms). In fact, while some preliminary minor parts of the calculations could be distributed on the Grid, the main components of the iterative procedures are not suited for distributed computing. A minor module of INTERACTION is FITTING that is invoked when the ab initio values need to be fitted to a suitable functional form of the global type [60]. FITTING is definitely less time and memory demanding. Despite that alternative approaches utilizing ab initio valus as such are being developed.

DYNAMICS is the block that carries out the integration of the equations determining the dynamics of the nuclei [53, 55, 61] of the system. In a rigorous approach the problem is dealt by using full dimensional quantum mechanics means which are the method of election for an exact calculation of chemical reactivity. However, although for atom diatom systems the integration of

ISSN: 2279-8773

related differential equations is nowadays routinary when the total angular momentum is zero, this is not true when convergence with it is seeked. This is due to the impressive amount of computer time and memory storage necessary for that purpose that a computational chemist can hardly get. In this case either reduced dimensionality quantum or classical or semi- (or quasi-) classical mechanics methods are used (or even a combination of them and model treatments).

The computational task of this block consists in the distributed execution of the different types of quantum programs (or suites of programs) and is, at present, structured to run in two main modules: the TD and the TI ones. Both modules calculate on the Grid the elements of the reactive scattering **S** matrix (whose square moduli represent the quantum probabilities). In GEMS, the quantum TD method, usually called wavepacket method, is embodied for atom diatom system into the program RWAVEPR [62]. Traditionally, however, the most popular approach to the calculation of detailed reactive properties is the TI one that is embodied in the ABC quantum reactive scattering program.

OBSERVABLES is the block that carries out the necessary statistical (and model) treatments of the outcomes of the theoretical calculations to provide a priory estimates of the measured properties of the system. This implies a proper sampling of both initial and final conditions as well as an additional

ISSN: 2279-8773

integration over some unobserved variables. Most often the experimental measurements are not taken as such. In fact, usually, the experimentalists elaborate the value of the measured signal so as to work out of it some quantities easier to compare with theory. Goal of this block is, however, to avoid this step that sometimes implies the use of models and assumptions of questionable nature. Whenever possible, in fact, OBSERVABLES incorporates numerical procedure building the value of the experimental signal by taking into proper account all the characteristics of the experimental apparatus used. In the current version of GEMS, OBSERVABLES considers only the calculation of the reactive scattering data measurable in CMB experiments. GEMS has both the peculiar characteristic of being fully ab initio (for this reason the DYNAMICS block has been founded on full quantum dynamics programs) and clearly service oriented (for this reason it has been modularized and structured as a framework in which high flexibility is offered to the user especially in the last part, OBSERVABLES, made linkable to experiment-based software). This makes GEMS not only an extremely useful tool for molecular investigations but candidates it also as the service of election for the CheMIST community.

ISSN: 2279–8773

Virt&l-Comm.3.2012.22

3.5 The Electronic Tests: EChemTest.

EChemTest (European Chemistry Tests) is an assessment software owned by ECTNA that can be used in order to test the skills and knowledge of students in various fields of Chemistry. The Tests are not designed to confer skills. Yet, they can be used to evaluate the skills of the students at various levels and for various purposes. As an example, tests are used by some Universities as an integrated assessment component in their courses to assign European Credits. An EChemTest session consists of a one hour test made of up to 30 questions of different types, taken at random from a large bank, covering the European Core Chemistry Program at three different levels equivalent to:

1. Ending of compulsory education (General Chemistry 1);
2. Beginning of University studies (General Chemistry 2);
3. Ending of Chemistry Bachelor studies (Analytical, Biological, Inorganic, Organic and Physical Chemistry level 3);
4. Ending of Chemistry related Master studies (at present only Computational, Synthetic and Cultural Heritage).

Related question banks have been prepared following the guidelines of the Core Chemistry Syllabus of the various University Cycles adopting also the Chemistry Eurobachelor® and Euromaster®). Several types of questions

ISSN: 2279-8773

are available for EChemTest sessions: multiple choice, multiple response, numeric, selection, text, graphical ”hot-spot” and problem solving. The software used also permits a detailed analysis of the collected responses to questions.

If the student successfully passes an EChemTest examination in one of the authorized Testing Centers, he/she becomes eligible for being awarded the corresponding Certificate. Each Testing Center is managed by an Educational Manager supported by a Laboratory Technician and assisted by an Educational Supervisor. Tests require a computer with a high-speed Internet connection and a modern browser equipped with a Java Virtual Machine (JVM) [65] running all the interactive sections of the tests. No computing skills are required for the test, apart from being able to use the mouse to point and click, or to use the keyboard to type-in single-word or numeric answers.

The tests, available in various European languages, evaluate recall and understanding of facts, applications of knowledge, evaluation and synthesis of information in chemistry related subjects. They are produced using the commercially available software Question Mark Perception [66]. The text encoding used is UTF-8 [67] for a wider language and character compatibility (so as, for example, to display non-latin characters correctly). When a student successfully passes an EChemTest examination in one of the authorized

ISSN: 2279-8773

Testing Centers, he/she becomes eligible for being awarded the corresponding Certificate.

At the moment, EChemTest tests are used to monitor students of exchange programs, career progression of professionals (e.g. for industrial mobility) and to manage life long learning processes. They can be conducted either in a demo or in a certification mode. In the demo mode the platform can be used to carry out sample sessions of tests to see "how it looks", trial evaluations to understand whether the test is suitable to foster an educational progression, continuous education final hot-evaluations and evaluation of preparation differences between prior and after a teaching sequence or an academic year. As individual "life long learning" self-assessment certification test, EChemTest is used to prove a level of knowledge comparable to an academic level expectation. In the certification mode tests are used to carry out dynamic sessions leading to a final mark for the issuing of European Chemistry Certificates by ECTNA, for the delivering of Credits to students at member institutions and even for awarding professional recognitions. In this case the sessions are monitored and the tests are taken under controlled conditions.

The EChemTest sessions are carried out at the EChemTest Testing Centers officially recognized by the ECTNA. They have to comply with strict requisites related to:

ISSN: 2279-8773

- Context and Logistic (e.g. computer room and maintenance);
- Examination conditions (e.g. secure network and bandwidth availability);
- Examination procedure (e.g. monitoring and ID control).

All this has made EChemTest a suitable tool for setting standards for student mobility programs.
Going into details, the Microsoft Windows Server of the system located at the Testing Center of Perugia [68] has been equipped with the Perception software and related data are stored in a Microsoft SQL server [69] database. Moreover, security is implemented at two different data and network levels. The former makes use of a Raid-1 system in order to guarantee the backup and restore of data. The latter makes use of a local firewall (to protect the server) and of a perimetric firewall (to protect the net of the `unipg.it` domain that hosts the server) which is configured in a way that allows a high personalization on the basis of the IP source/destination address and of the service/protocol requested.
EChemTest drives the user step-by-step starting from the beginning up to the end of each test session. In addition, the entire process, certification included, is completely paperless, self explanatory, self sufficient and is also complemented by demo tests. Up to date EChemTest has registered more

ISSN: 2279-8773

than 4200 users and relies on 8 official Testing Centers located in different countries of Europe[1].

Moreover, specific efforts have been devoted to implement a commercial version of EChemTest for which new technical solutions are being developed also to the end of fostering its Sustainability. In this respect, the centralized three-tier architecture (or part of it) in which the EChemTest system relies at present is being properly transferred to Grid platforms. Accordingly, SOA and Web Services approaches are being adopted in order to offer the EChemTest sessions under the form of Grid Service while the back-end (essentially made by the DataBase Management System, or DBMS, Microsoft SQL Server containing the various EChemTest libraries) is going to be moved on the Grid by using the already mentioned software systems (e.g. GRelC) aimed at integrating and managing database resources on the Grid.

Further developments can be foreseen also for the provision of supplemen-

[1]As from ECTNA Official Documents, the recognized Testing Centers are: CPE Lyon (A. Smith, P. Mimero) in France, University of Perugia (A. Laganà, C. Manuali, N. Faginas Lago) in Italy, Technical University of Vienna (J. Fröhlich, C. Hametner, H. Krebs) in Austria, University of Helsinki (K. Wähälä, T. Hase, J. Koskimies, N. Aremo) in Finland, Aristotle University of Thessaloniki (E. Varella, I. Kozaris, E. Koliarmou) in Greece, Jagiellonian University of Krakow (A. Michalak, A. Kolasa, K. Szczeponek) in Poland, University Complutense of Madrid (F. Gavilanes, J. Alcaraz) in Spain and University of Reading (D. Cardin, C. Cardin, C. Covell) in United Kingdom.

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

tary educational materials that could be prepared and offered within the LOs provided by G-LOREP.

ISSN: 2279-8773

Virt&l–Comm.3.2012.22

Chapter 4

On the Structuring of Virtual Research Communities

4.1 Collaborative Environments in Grid.

A key novelty of G-C of dramatic impact on scientific research organizations is its suitability for collaborative work. Collaborative Environments are a set of tools allowing a group of users to cooperate in order to accomplish a given task, regardless of their geographical dispersion. To cope in general with the complex objectives of Collaborative Environments in Grid, it is necessary to climb to the higher level of abstraction associated with Knowledge Management (KM). KM is, in fact, the conceptual platform necessary to support intelligent process automation and collaborative problem solving

ISSN: 2279–8773

in large-scale science over Internet. KM comprises a range of practices able to represent, identify, create, elaborate and distribute knowledge over multiple environments. The management of scientific knowledge in a distributed environment is a key challenge for domains such as Chemistry, Physics and Mathematics, in which the knowledge is largely spread over the network. Furthermore, there is also a huge potential for KM in other domains like Life Sciences, Health Care, Materials and Nanotechnology.

A first crucial aspect of modern KM is the volume of data handled that may easily amount to the order of TBs and PBs. As a matter of fact, the communities of researchers need to produce, access and analyze large amounts of data (often using sophisticated and computationally expensive techniques). These sets of scientific data are themselves increasingly more numerous (and almost always geographically distributed) than the computing and storage resources available to the related communities. Moreover, efficient and reliable execution of these tasks may require the appropriate management of TB caches, transfer of GBs of data over wide area networks, coordination of data and supercomputer elaborations, performance estimations to guide the selection of dataset replicas, as well as other advanced operations collectively optimizing the usage of storage, networking and computing resources. These requirements cannot be adequately met by the traditional Web tools and need ad hoc hardware and software solutions that only a Grid infrastructure

ISSN: 2279-8773

can offer [42].

A second crucial aspect of KM is the standardization of data. In this respect, it is of fundamental importance to arrange, whenever possible, a common Grid framework able to gather several kinds of information coming from different sources (e.g. log files, user feedbacks) in a proper way in order to proceed with data quality evaluations. At the same time, owing to the fact in that various different tools (e.g. workflow engines, meta schedulers) can be used by the Grid community, a standardization appears to be necessary so as to allow the mentioned tools communicate each other by sharing data and functions. In this respect, as already mentioned in the previous chapter, one of the modern approaches is to refer to SOA and Web Services.

A last crucial aspect of KM considered here stems from the fact that collaboration is also a recursive process in which people work together by benefiting from each other work to the end of achieving common goals [see Appendix A]. In fact, during related activities, VRC users leave (or are incline to leave) tracks about themselves and the actions taken which are useful for evaluating the efficacy of collaboration. In order to capture these information we have developed a high-level Grid Framework (named GriF and illustrated in detail later on in this chapter) that operates when users utilize the Grid.

GriF adopts a Collaborative Filtering (CF) technique to the end of singling out information and patterns imbedded within data collected during user

ISSN: 2279-8773

activities. CF applications typically involve very large data sets. Therefore, to work out automatic predictions (filtering) about the user intention and strategies, GriF is set to collect together with inputs from the systems opinions from other users counting on the fact that who agreed (or disagreed) on the use of a service in the past tends to agree (or disagree) again in the future. To carry out the evaluation, then, CF goes through the following two steps:

1. Looks for users sharing the same rating patterns of the considered user (e.g. the user about which the evaluation is being made);
2. Uses the ratings of the like-minded users of step 1 to evaluate the considered user.

GriF adopts also another form of CF based on implicit observations of the natural user behavior (as opposed to the artificial behavior imposed by a defined metric). In this case, one observes what a user has done comparing it what all other users (e.g. what programs they have run and what running sequence they have followed). Then, GriF uses that data to evaluate the user behavior or to predict what will be his/her attitude and needs. It is worth mentioning here that CF is at the heart of our collaborative monitoring and evaluation procedure for the purpose of assigning Credits through the Grid Credit System (named GCreS) that will be illustrated in chapter 6 of this

ISSN: 2279-8773

Thesis. To this end, however, new higher level additional requirements are also considered in order to allow users to tailor their requests of a specific Grid application using characteristics (like security, reliability and/or performance) rather than machine parameters (like memory, cpu and storage capabilities). This has the advantage of allowing a classification of the users in a way that permits the implicit selection of specific policies of execution (based on the user class level, as described in chapter 6) and the award of Credits. As a result, this approach makes the use of the Grid similar to that of a black box pushing the G-C to a higher level of transparency (as in the case of U-C or C-C).

Indeed, this strong collaborative connotation is one of the most strategic assets of a VRC because it stimulates a great deal of research and a continuous improvement of the software patrimony. It prompts, in fact, not only the validation of stable and easy-to-use versions of the programs but also the development of the necessary support, documentation and maintenance procedures (including the proper handling of trademark and commercial obligations for commercial packages). This has also shown to be a crucial cross point of the COMPCHEM VO mission. In this respect, it also clearly motivates the members of a VRC to structure the programs as Grid Services and stimulates the introduction of some forms of Grid Economy that finds its actualization in the already mentioned GCreS tool that puts at the center

ISSN: 2279-8773

of the action the Sustainability of the VRCs. In other words, to the extent in which the collaboration of the VRC users can increase the Quality of the work done (and of the results obtained), the Sustainability is fostered through a systematic evaluation.

4.2 SOA, Web and Grid Services.

4.2.1 Service Oriented Architectures (SOA)s.

SOA is emerging as the premier integration and architecture framework in today's complex and heterogeneous computing environment for building software applications by making use of services available in a network (such as the Web). In this respect, a service is an implementation of a well-defined functionality that can be consumed by clients in different applications or processes. Therefore, SOA allows the reuse of existing assets when new services are to be created from an existing IT infrastructure. As a matter of fact, applications in SOA are built on services and SOA helps organizing streamline processes by promoting loose coupling among software components to allow an easy reuse. This enabling of software as a service allows also to do business more efficiently and in a fashion adaptable to the changing needs and competition. From a business point of view, SOA enables users to leverage on

ISSN: 2279-8773

past investments by allowing them to reuse existing applications and offering interoperability between heterogeneous applications and technologies. This means that SOA provides the users with an unprecedented level of flexibility relying on the fact that:

- Services are software components with well-defined implementation-independent interfaces. An important aspect of SOA is the separation of the service interface (the what) from its implementation (the how). Such services are consumed by clients (now called Consumers) which are not concerned with the way these services will execute their requests;
- Services are self-contained (perform predetermined tasks) and loosely coupled (for independence);
- Services can be dynamically discovered;
- Composite services (e.g. Workflows) can be built from aggregation of other services.

SOA uses the *find-bind-execute* paradigm [70] as shown in Fig. 4.1. In this paradigm, service Providers register their services in a public registry. This registry is consulted by Consumers to find services matching certain criteria. If the registry has such a service, SOA provides the Consumer with a contract

ISSN: 2279-8773

(SLA) and an endpoint address for that service.

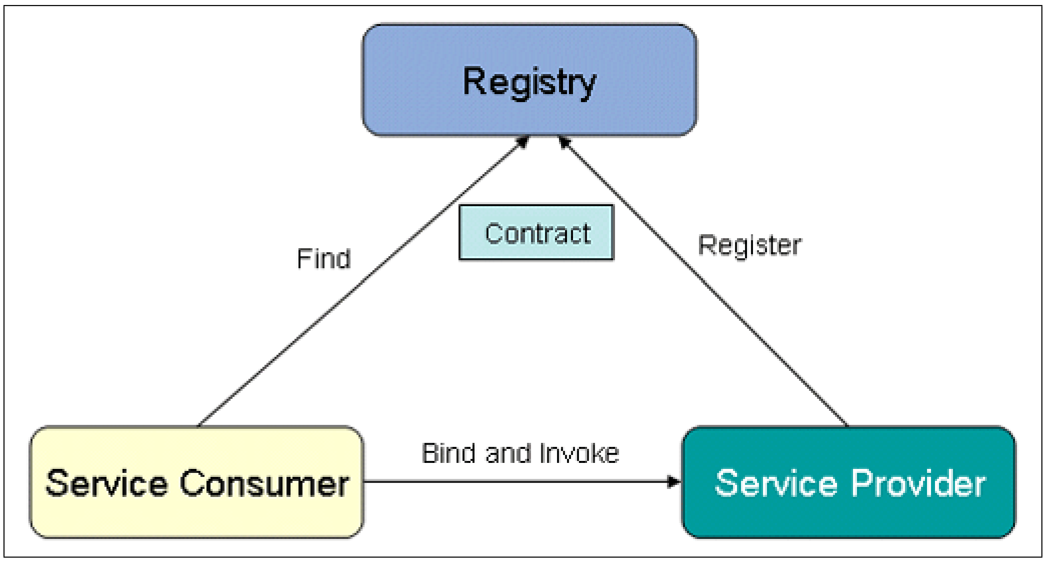


Figure 4.1: **The *Find-Bind-Execute* Paradigm of SOA.**

SOA-based applications are distributed multi-tier applications articulated into *Presentation*, *Business Logic* and *Persistence* layers. This is, indeed, important because although services are the building blocks of SOA applications and any functionality can be rendered as a service, the real challenge is to define service interfaces bearing the right level of abstraction (in other words, services should provide coarse-grained functionalities).

As expected, interoperability is a key feature of SOAs. A simple way to ensuring interoperability is the use of Web Services which run on a variety of software platforms and hardware architectures. Therefore, although SOA cannot be confused with Web Services, it is usually based on them, as

ISSN: 2279-8773

detailed in the next section.

4.2.2 Web Services.

Web Services are software systems designed to support interoperable machine-to-machine interaction over the network. This type of interoperability is ensured through a set of XML-based open standard protocols, such as WSDL, SOAP and Universal Description, Discovery and Integration (UDDI) [71], as illustrated in Fig. 4.2.

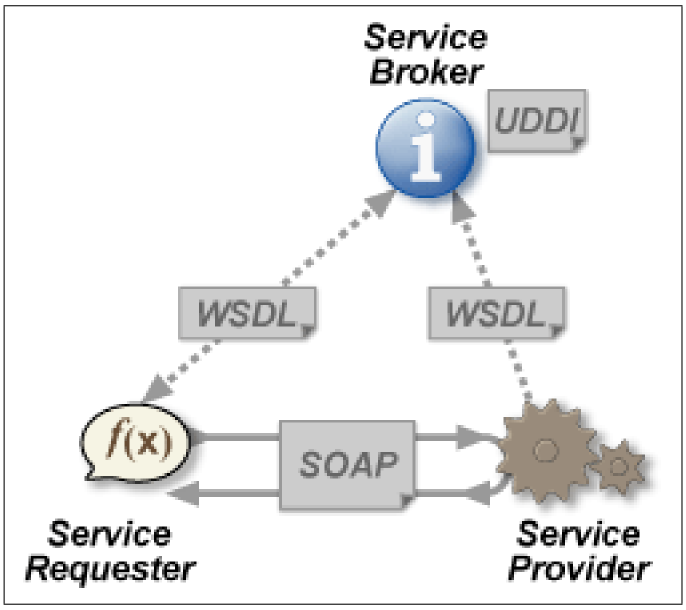


Figure 4.2: **The SOA Web Services implementation.**

These standards provide a common approach to defining, publishing and using Web Services. Once a Web Service is discovered, the Consumer makes a request to it. The selected Web Service is then responsible for processing the

ISSN: 2279-8773

request and sending the response back to the Service Requester. All the internal details occurring between the request and the response are transparent (e.g. one deals only with typical Java programming language semantics, such as Java method calls and Java data types, without worrying about mapping Java to XML and vice-versa or constructing SOAP messages) to the developers allowing them to focus on the high-level issues. In other words, once a Web Service is discovered, the Consumer has just to either invoke remote procedure calls on the methods offered by the Web Service, or send an XML document to the Web Service in order to process it.

In the present Thesis work AXIS (Apache eXtensible Interaction System) [72] has been used as SOAP engine. AXIS is, in fact, a framework for constructing SOAP processors (such as clients, servers and gateways) which plugs into servlet engines (such as Apache Tomcat [73]) and supports Java Web Services (JWS)s including extensive support for the WSDL to generating Java classes out of it. When a Web Service has to be invoked by a Consumer, three methods can be used [74]:

- Using generated *Stubs* from Service WSDL description: this method makes use of a platform-specific Stub created before runtime during the WSDL to Java mapping stage. Because the Stub is created before runtime, it is sometimes called *Static Stub*. The advantage of this

ISSN: 2279-8773

method is its simplicity. Basically, only a few lines of code are required to access and invoke a Web Service's operation. However, a limitation of the method is the fact that one needs to know the WSDL Uniform Resource Locator (URL) at development-time and to run the WSDL to Java mapping tool. Moreover, Stubs are not portable because they depend on implementation classes and should not be packaged as part of an application;

- Using *Dynamic Proxy*: the advantage of using this method is that one can write portable and vendor-independent codes. However, also in this case one needs to know the WSDL URL at development-time. Moreover, if the WSDL Uniform Resource Identifier (URI) is likely to change, this method cannot be used;

- Using *Dynamic Invocation Interface (DII)*: making DII calls through a Call object is programmatically more complex than using a Stub or Dynamic Proxy. However, the advantage of using a DII Call interface is that a Consumer can call a remote procedure without knowing at development-time the WSDL URI or the Web Service operation's signatures. This makes the code easy to modify if the Web Service details change. Moreover, with DII clients runtime classes generated by the WSDL to Java mapping tools are not required (as instead is needed for

ISSN: 2279-8773

Dynamic Proxy or Static stub cases).

An alternative approach to the design and implementation of Web Services, based on the architectural style of the Web itself, is provided by the REpresentational State Transfer (REST) [75] method. REST was first introduced by Roy Fielding (one of the principal authors of the HTTP specification [77] and a co-founder of the Apache HTTP Server project [76]) in the year 2000 as part of his doctoral Thesis dissertation. For the past several years, the merits of REST with respect to the SOAP architectural style for Web Services have been largely debated. At present, both approaches are useful for implementing SOA components. For simple applications, REST is an easy way to get started. Its services (called RESTful services) adhere to a set of constraints and architectural principles which include the following:

- RESTful services are stateless: each request from client to server contains all the information necessary to understand the request and cannot take advantage of any context stored on the server;

- RESTful services have an uniform interface: the only allowed operations are the HTTP ones like GET, POST, PUT and DELETE;

- REST-based architectures are built out of resources (pieces of information) that are uniquely identified by URIs;

ISSN: 2279-8773

- REST components manipulate resources by exchanging representations of the resources (e.g. by an XML document).

Fielding also says that "REST-based architectures communicate primarily through the transfer of representations of resources". This is fundamentally different from the Remote Procedure Call (RPC) approach that encapsulates the notion of invoking a procedure on the remote server. Therefore, RPC messages typically contain information about the procedure to be invoked or the action to be taken. This information is referred to as a 'verb' in a Web Service request. In the REST model, the only verbs allowed are HTTP operations. Moreover, in the RPC approach typically many operations are invoked at the same URI. On the contrary, with the REST approach there is a unique URI for each resource.

More significant deviations from Fielding's definition of REST involve getting around the "uniform interface" constraint by embedding verbs and parameters inside URLs. Some REST systems, for example, include verbs in query strings and don't have unique URIs for each resource. Systems like this, although labeled as RESTful, are indeed starting to look very much like RPC ones (using XML over HTTP without SOAP).

For complex applications (as for example the Grid Framework GriF developed by us) SOAP is preferred because it is a well-known protocol (and not

ISSN: 2279-8773

an approach as REST) with the following additional characteristics:

1. Transport agnostic (not only the HTTP transport model is supported as in REST);
2. Handling of distributed computing environments;
3. Better support from other standards like WSDL and those related to Web Services;
4. Built-in error handling.

4.2.3 Grid Services.

As mentioned before, Web Services are the technology of choice for Internet-based applications with loosely coupled clients and servers. That makes them an obvious choice as a tool for building the next generation of Grid-based applications.

Starting from plain Web Services (as currently specified by the W3C [78]), formally Grid Services are Web Services with improved characteristics and other services. This was introduced in Open Grid Services Infrastructure (OGSI) that was the first recommendation meant to provide a layer for the Open Grid Services Architecture (OGSA) describing an architecture for a service-oriented Grid computing environment for business and scientific use.

ISSN: 2279-8773

The description was derived from [79] and developed within the Global Grid Forum (GGF) [80]. OGSI has taken into account the statelessness issues (along with others [81]) by essentially extending Web Services to accommodate Grid computing resources that are both transient and stateful. At present, the Web Services Resource Framework (WSRF) [82] has superseded OGSI providing a set of Web Service specifications developed by the Organization for the Advancement of Structured Information Standards (OASIS) [83] that is a global consortium driving the development, the convergence and the adoption of e-business and Web Service standards. Accordingly, WSRF defines a generic and open model to develop and access stateful resources using Web Services. This includes mechanisms to describe views on the state, to support the management of the state through properties associated with the Web Service and to illustrate how these mechanisms are extensible to groups of Web Services. In other words, WSRF provides a set of operations that Web Services may implement to become stateful. As a matter of fact, when Consumers talk to the Web Service, they include the identifier of the specific resource that should be used inside the request, encapsulated within the endpoint reference. This can be a simple URI address rather than a complex XML content that helps in identifying or even in fully describing the specific resource in question.

It is worth mentioning here that in the present Thesis work we consider a

ISSN: 2279-8773

Grid Service as a set of collaborative Web Services implementing a particular high-level distributed function on the Grid middleware.

4.3 A New Collaborative Grid Framework: GriF.

4.3.1 The Architecture.

The vision of G-C is to provide ubiquitous and secure access to high-end computing for scientists, researchers and industry. This enables new classes of applications to be developed in communities as diverse as HEP, CC and LS. These applications typically require significant compute resources and access to large data sets, often integrated from disparate locations. Advances in middleware technology and the growing prevalence of high-speed networks are bringing us closer to this vision, while placing greater demands on the realtime collaboration tools, security infrastructure, middleware and transport protocols needed to support these VRCs.

Support needed by such communities often includes the use of standardized collaboration tools, such as those provided by GriF [84]. The main features of GriF are the adoption of common standards, friendliness and ability in efficiently tracking user activities. The collected information provide useful

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

indications on the behavior of the user, the paths he/she preferentially follows and semantic inferences out of the job run (derived for example from program names). More objective information are added to further specify the user profile and levels of trust and reputation. Moreover, subjective information (e.g. user feedbacks) are also supported.

GriF is a Java-based SOA Grid Framework aimed at running on the EGI Grid (supporting the gLite middleware) multi-purpose applications. The basic goal of GriF is to provide the users with a user friendly tool allowing them to exploit the innovative features of Grid computing with no need for mastering the low-level Grid environment. This means that there is no need for using specific Grid operating system dependent commands, as for example to establish links to the Grid Proxies (and/or to the Grid Certificates (GC)s) and to manage all the other operations (as, for example, running Grid jobs, checking their status and retrieving related results from the Grid middleware) as well. In other words, GriF makes Grid applications black-box like pushing the G-C to a higher level of transparency as in the case of C-C. This makes GriF a tool of extreme importance for enhancing the VRC activities. Its utilization, in fact, leads to better memory usage, reduced cpu and wall times consumption as well as to an optimized distribution of tasks over the Grid platform. Moreover, GriF leads to a more efficient exploitation of the innovative features of the Grid when building applications of higher level of

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

complexity. Thanks to its SOA Framework nature, in fact, it can support collaboration among researchers bearing complementary expertise. Because of this, GriF is enabled to articulate the computational application as a set of sequential, concurrent or alternative Grid Services by exploiting the features of SOA.

The SOA organization of GriF consists essentially of two Java servers and one Java client, as sketched in Fig. 4.3.

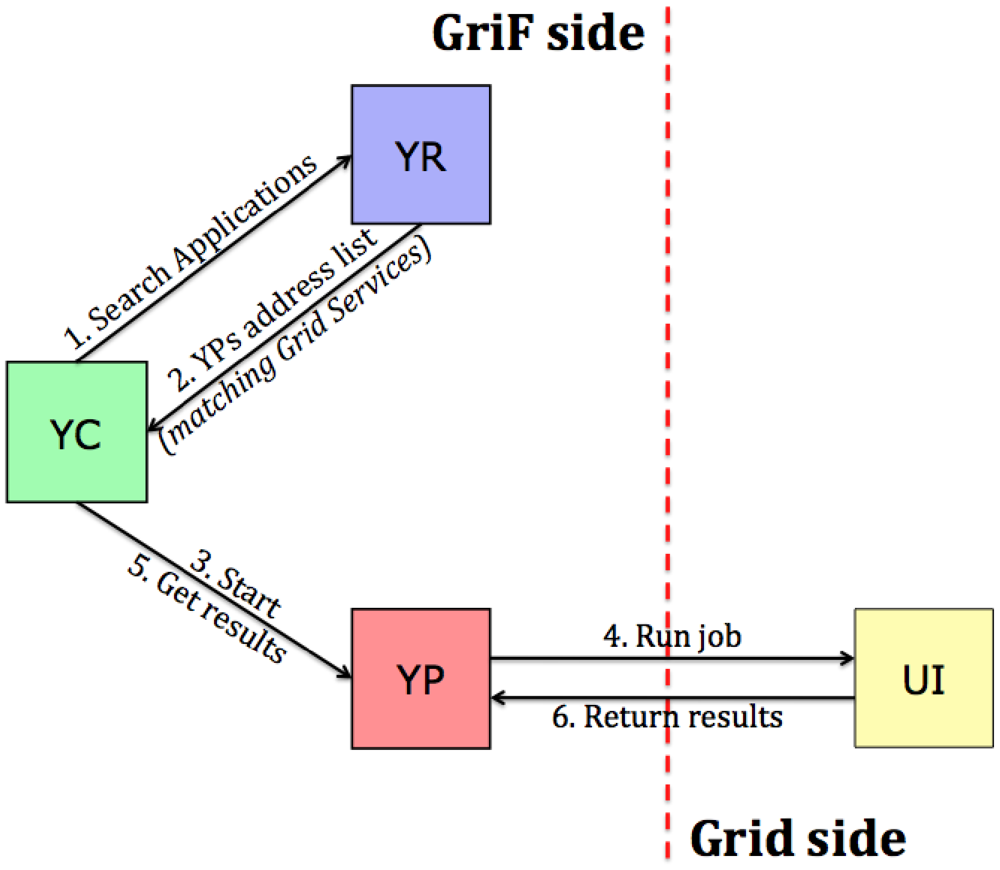


Figure 4.3: **The GriF Architecture and the Grid (*taken with modifications from ref. [2]*).**

ISSN: 2279-8773

Virt&l–Comm.3.2012.22

The first Java server is YR (Yet a Registry) that is based on the UDDI protocol and acts as a directory listing to carry out registry operations. In this respect, VRC users inspect YR to the end of finding and invoking the appropriate Grid Service. The second Java server is YP (Yet a Provider) that makes use of SOAP which is the XML-based messaging format established as transmission framework for inter-service communications (via HTTP or HTTPS). Making use of the HTTP protocol (and of the associated well-known port 8080) GriF is ready to be distributed with no particular issues regarding network security aspects (e.g. no need, in most cases, to open specific IP addresses and/or ports on perimetric firewalls).

YP holds the Web Services of the VRC and handles quality parameters. Both YR and YP make use of WSDL to describe the services provided and to reference self-describing interfaces that are published in platform-independent XML documents. The Java client is YC (Yet a Consumer) that is the entry point for operating with GriF within the Grid. Accordingly, YC (that is entirely based on the DII mentioned above) needs not the issuing of GCs. Moreover, Security is granted by the fact that only VRC users can access GriF (from a technical point of view, every Web Service is authenticated transparently each time it is used against unauthorized accesses and Man-In-The-Middle (MITM) attacks as well) and they only need to specify on YC the related username and password at the beginning of operations. The

ISSN: 2279-8773

selected YP takes care of running the jobs on the associated User Interface (UI), of managing their status and of notifying the users upon completion. YC is weakly coupled in respect to the Grid Middleware and implements all the already mentioned extensions and protocols to correctly interface the Web Services offered by GriF. As a matter of fact, VRC users wishing to run an application on the Grid can utilize YC to perform the various actions devoted to the management of the basic Grid operations, to run existing programs, to upload new applications and to compile as well new executables (e.g. a different version of the same application or a new one) on the selected YP (to this end efforts were devoted to the assemblage of the server programs needed to transparently manage the Grid Proxies by using a Robot Certificates (RC)s [85] strategy). Moreover, YC is also used to search applications on YR, to send messages and feedbacks to GriF team, to monitor the status of Grid jobs and to retrieve the related results (some Web Services wrapping Grid middleware commands and managing their execution have been implemented to this end). In addition, YC has been transformed into the YA (Yet an Applet) Java applet in order to allow the use of GriF also on a client machine with no YC.

Particular attention was paid to use a univocal nomenclature. For this purpose, reference is always made to the program's name, the user's name and the starting operations (date and time expressed in seconds) when identi-

ISSN: 2279–8773

fying the files involved in the process (e.g. including the `.txt` results file). This avoids the typical problems of multi-user (in which different users can execute the same program at the same time) and job-intensive (in which the same user can execute the same program several times) environments.

Typical fragment pseudo-codes used respectively in YP (to wrap Grid applications into Web Services) and in YC (to invoke the Web Services exposed by the YP), are listed in Fig. 4.4. In the YC section of Fig. 4.4, the variable `YP_NAME` takes the value of the YP name selected by the user during the Service Discovery phase performed using YR (see Fig. 4.3). Then a Web Service is invoked on YP waiting for the results (see last line of the YC section). In the YP section, after a method called `run` has respectively received the date, the user name and the specific operation input data as input, a shell script called `rungrid.sh` is invoked in order to perform the real low-level Grid operation. Accordingly, related results are returned by `rungrid.sh` to the Web Service and then to the YC.

As a result, Software Providers (see item 2 of Table 3.1) can expose their services in an open, standard and secure way suited to serve all kinds of users including those having little familiarity with the wrapped applications and the Grid platform. In this way, the applications gain a high level of friendliness and portability while the Grid system reaches an high level of expertise. As already mentioned, in fact, GriF does not require the user to handle bi-

ISSN: 2279-8773

YP Web Service code:

```
public class RunGrid {
   public String run(String input, String user, String date) {
      String cmd = ''rungrid.sh'' + input + user + date + ''>'' + outputfile;
      String[ ] command = {''sh'', ''-c'', cmd};
      Process p = null;
      Runtime r = Runtime.getRuntime();
      p = r.exec(command);
      p.waitFor();
      FileReader file = new FileReader(outputfile);
      BufferedReader reader = new BufferedReader(file);
      while ((line = reader.readLine()) != null) output += line;
      return output;
   }
}
```

YC Client code:

```
public class YC {
   public static void main(String [ ] args) {
      String endpoint = ''http://'' + YP_NAME + ''/RunGrid.jws'';
      Service service = new Service();
      Call call = (Call) service.createCall();
      call.setTargetEndpointAddress(new java.net.URL(endpoint));
      call.setOperationName(''run'');
      call.addParameter(''input'', XMLType.XSD_STRING, ParameterMode.IN);
      call.addParameter(''user'', XMLType.XSD_STRING, ParameterMode.IN);
      call.addParameter(''date'', XMLType.XSD_STRING, ParameterMode.IN);
      call.setReturnType(XMLType.XSD_STRING);
      String result = (String) call.invoke(new Object [ ] {input, user, date});
   }
}
```

Figure 4.4: **Pseudo-code of wrapping and invoking Web Services in GriF.**

ISSN: 2279–8773

nary programs, to choose among the Grid resources the suitable Computing Elements (CE)s, Storage Elements (SE)s and UIs, and to know as well operating system specific commands which usually discourage normal users from exploiting the Grid power. GriF users can, instead, carry out most of their operations using mainly a natural-like language (for example, when searching for an application of interest, the VO name, the program name, as well as some keywords matching the desired application description and functions are a sufficient means). Moreover, additional qualitative procedures have been implemented to carry out various Framework-side operations like those related to the Grid resources match-making for the specific applications to be run.

As described in Fig. 4.4, from the GriF point of view a ”service” is the wrapping of a binary program into a (Java) Web Service (then assembled in a Grid Service). Accordingly, GriF offers the possibility of running an existing application, of loading new binary programs and of compiling as well (although in an experimental mode) a new or a modified source program. Once the application has been identified, the GriF flux will follow the three main steps given below:

1. The input file is loaded by the user (sometimes it can be generated by a user-driven system procedure and also validated);

ISSN: 2279-8773

2. The job is distributed on the Grid and run;

3. Once the job is completed, the results are returned back to the user (e.g. in ASCII format) and/or forwarded to another visualization (three Dimensional, or 3D) system.

After these steps, GriF ends up by having collected several new information (typically not included in the existing gLite middleware sensors) about the user activity.

4.3.2 Yet a Registry (YR).

YR is the server (Unix-like systems compliant) that implements the SOA Registry of Fig. 4.1. To this end, YR makes use of Apache jUDDI [86] and UDDI4J [87]. The former is an open source Java implementation of the UDDIv3 specification. UDDIv3 is a cross-industry effort driven by major platform and software providers. It is aimed at developing the building blocks required for Web Services which describe a standard and interoperable platform enabling communities and applications to quickly, easily and dynamically find and use them over Internet. It also allows operational registries to be maintained for different purposes and different contexts for Web Services. The latter is a Java class library providing Application Programming Interfaces (API)s to interact (via YC) with a UDDI Registry.

ISSN: 2279-8773

YR has been tested on Scientific Linux 5.2 [88] and it consists essentially of a MySQL [89] Database and of a Web application integrated in the already mentioned Apache Tomcat (5.5 version). Tomcat is an Open Source Web container implementing the Java Servlet and the JavaServer Pages (JSP) technologies adopted in order to build both YR and YP. Within a YR, one can manage more VRCs and maintain several Grid Services for each of them. Accordingly, information on each Grid Service access points (in other words the YPs addresses hosting the Grid Services then invoked by YCs) are published by the `publish` URL of YR (see Fig. 4.5).

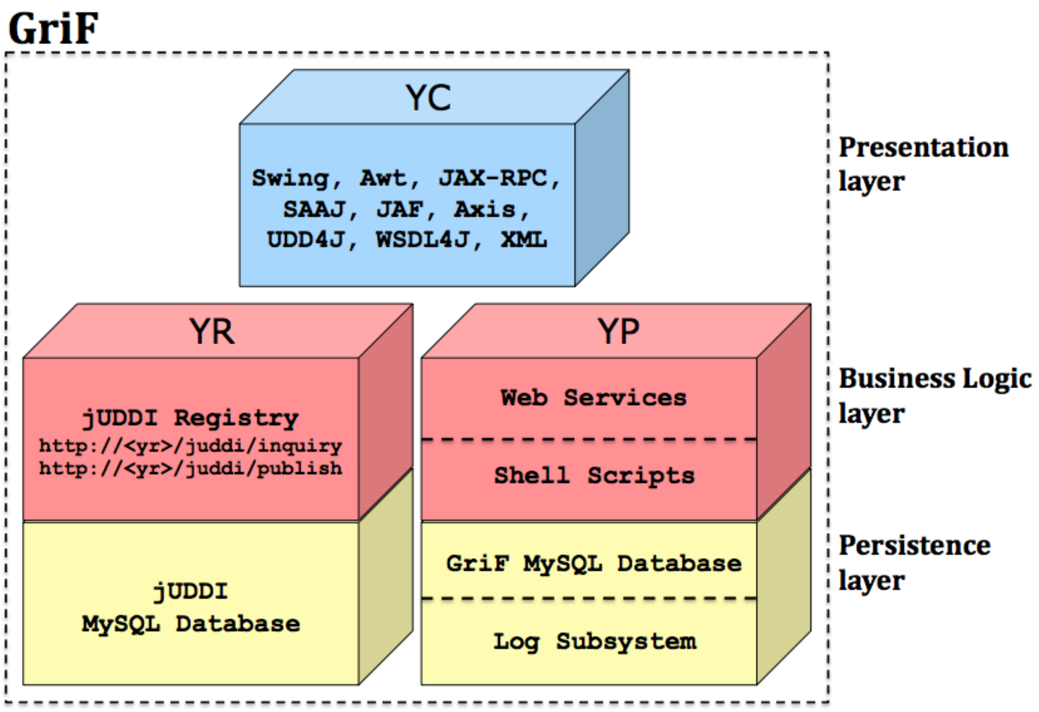


Figure 4.5: **GriF SOA Layers Mapping.**

ISSN: 2279-8773

To this end, YC is designed in a way allowing to:

- Always use the same YP (in a case in which just one YP is expected, its connection information are built-in into the YC);
- Choose between different YPs (in cases in which two or more YPs exist, the related connection information are recorded in a file in the client machine named by GriF `compchem.prop`).

In other words, a YR (expected to be unique within the Grid infrastructure) maintains the associations between a VRC, its Grid Services and the related YPs. For example, in our case study, we have considered as a community the COMPCHEM VO offering two distinct Grid Services (the already mentioned ABC [63] atom diatom TI quantum reactive scattering application and the General Purpose (GP) one and aimed at running on the Grid generic scientific applications) which have been used as testbeds in the present Thesis work and are both hosted by the same YP (see Appendix B in which part of the code illustrating either the database information (item B1) or the interaction between a YC and a YR (item B2)).
As a result, using YC and YR a GriF user is able to discover all the Grid Services (and the related access points, or YPs) belonging to a given VO/VRC and then to choose the desired one by selecting the corresponding YP.

ISSN: 2279-8773

4.3.3 Yet a Provider (YP).

YP is the server (Unix-like systems compliant) that implements the SOA Service Provider (see Fig. 4.1) and can be considered the heart of GriF. More than one YP can be deployed and typically at least one should exist for each UI of the Grid infrastructure. Accordingly, each YP (based on Apache Tomcat 5.5 and AXIS 1.4 and tested on Scientific Linux 5.2 as well, as in the case of YR) is responsible in bridging its GriF users (using YCs) and the Grid middleware (through the UI). Moreover, a YP can host different Grid Services at the same time and it is structured along three GriF layers: the Web Services and the Scripts (corresponding to the Business Logic layer of a SOA, as sketched in Fig. 4.5), and the Persistence layer.

The Web Services layer.

The Web Services layer of a YP consists of different sets of collaborative Web Services in which each set shares a common goal (as mentioned above, in GriF we consider a set of this type as a Grid Service, implementing each of them on different directories starting from the directory `$TOMCAT_HOME/webapps/axis`) exposing to YCs all the needed features in order to run on the Grid and to manage all the subsequent operations as well. Accordingly, each Web Service of a Grid Service waits for a YC request and always replies with a response.

ISSN: 2279-8773

Moreover, depending on the nature of the YC call, each Web Service can reply immediately or invoke an associated shell script waiting for its output. This is the case in which an operation to/from the (gLite) Grid middleware is required. To this end, a blocking system call will take place and then the operation will be performed on the Grid. Once finished, the operation results come back from the shell script to the associated Web Service (in other words from the UI on the Grid to YP) and then from the Web Service to YC. As a result, considering the GP Grid Service, the main Web Services (and the associated shell scripts when needed) implementing it are the following:

- `Login.jws`: authenticates VRC users on YP;
- `Welcome.jws`: identifies VRC users on YP;
- `Charge.jws`: loads the already available applications from YP to YC;
- `Compile.jws`: enables the compilation of applications (this feature is still experimental) on YP from sources (through the associated `compile.sh` shell script);
- `GetBinary.jws` and `DeleteBinary.jws`: get user-compiled binaries deleting them from YP;
- `Upload.jws`: uploads applications from YC to YP (through the associated `upload.sh` shell script);

ISSN: 2279-8773

- `RunGP.jws`: submits applications (and the related inputs) on the Grid middleware (through the `rungp.sh` shell script);
- `Check.jws`: checks jobs status on the Grid middleware (through the associated `check.sh` shell script);
- `Result.jws`: retrieves job results from the Grid middleware (through the associated `result.sh` shell script);
- `ClearGriF.jws`: clears the GriF environment after a successful retrieving of results (through the associated `cleargrif.sh` shell script);
- `DB.jws`: manages all the GriF database operations;
- `GriFStatus.jws`: checks the health of GriF and of its subsystems (through the associated `grifstatus.sh` shell script);
- `Feedback.jws`: sends feedbacks provided by VRC users to GriF.

The general programming structure of each of them is that of Fig. 4.4. Particular mention has to be made for the Web Service called `RunGP.jws` (fully reported in Appendix C, item 5), which is the Web Service actually responsible for the running on the Grid of an application. After receiving all the necessary parameters (as for example the input and the name of the application to be run), at the beginning `RunGP.jws` performs all the required

ISSN: 2279-8773

security checks by using the received username, the password and the server key parameters. Then, after receiving the provided input by a `DataHandler` strategy (described in detail later on), the implicit type of method that a user has determined with his/her choice in running jobs is checked. Accordingly, the first type is called `STANDARD` (meaning that an already available application has to be run) while the second type is called `CUSTOM` (meaning that a new (just uploaded) application has to be considered). Therefore, in both cases the associated `rungp.sh` shell script is invoked (fully reported in Appendix D, item 1). It performs the following nine macro-functions:

1. Manage all the necessary variables setting up the right environments;
2. Determine the CE queue to be used for the run depending on the type of Ranking (described in detail later on) chosen by the user;
3. Copy the application to be run to the UI (when the `STANDARD` method has been adopted);
4. Manage the input (a single plain file or a compressed archive including multiple input files) provided by the user properly uploading it to the UI;
5. Create a proper Job Description Language (JDL) [90] file at runtime uploading it to the UI;

ISSN: 2279-8773

6. Create a proper shell script file at runtime uploading it to the UI. This file is responsible for wrapping the application driving its execution on the Grid. Furthermore, it is also responsible for registering wall time, cpu time (in seconds) and memory values (in KB) consumed as well as for uploading the final results to an appropriate SE (determined by the configuration of the LCG File Catalog (LFC) [91] subsystem adopted by GriF). The use of SEs as output repositories (instead of the default `OutputSandbox` available in the JDL that is limited to 100 MB) enables the user to deal with very large (even GBs) results file. Moreover, when the user does not choose to use GriF Ranking, this shell script is also responsible for specifying the minimal requirements needed to provide a sufficient reliability to the Grid job;

7. Submit the job on the Grid (then retrieving the resulting CE queue used by the Grid when GriF Ranking has not been selected by the user);

8. Return back to the `RunGP.jws` Web Service the HTTPS URL (the job identifier that is unique on the Grid) and the CE queue name related to the submitted job;

9. Clear the UI environment.

ISSN: 2279-8773

Once it has finished, the `RunGP.jws` Web Service takes again control of the flux and gives back to YC all the needed job information on the basis of which, among the other things, the URI where YC will fetch the final results is determined.

Other parts of the implemented Web Services code (in particular, as mentioned above, the common part shared by all the Web Services providing Security and the three main database operations (select, insert, delete[1]) performed by the Web Service called `DB.jws`) are reported in Appendix C (items 1–4).

The Scripts layer.

As mentioned before, the Scripts layer (consisting in a set of several BASH [92] shell scripts implemented in a different directory for each Grid Service starting from the directory called `$TOMCAT_HOME/programs/`) of a YP is firstly needed in order to bridge (some) Web Services on YP with low-level Grid commands on the UI. Accordingly, the main implemented shell scripts and their macro-functions (in addition to the already described `rungp.sh` shell

[1]Instead of a true deletion of a job from the database, the 'delete' operation changes its status into 'cancelled'. Accordingly, a dedicated off-line procedure (called `cleargrid.sh` and described in the next section) performs the real deletion for all the Grid jobs of this type.

ISSN: 2279-8773

Virt&l–Comm.3.2012.22

script that is responsible for the Grid jobs submission) are the following:

- the `upload.sh` shell script (fully reported in Appendix D, item 2), which given an application, uploads it on the section of the UI related to his/her owner user;
- the `check.sh` shell script (fully reported in Appendix D, item 3), which given a Grid job identifier (the HTTPS URL), checks and then returns to YC the related status of the Grid job specified by the user;
- the `result.sh` shell script (fully reported in Appendix D, item 4), which given a storage location (the SE URI), gets and then returns to YC the related results file of the Grid (”Done”) job specified by the user.

Secondly, this layer is of fundamental importance in order to implement all the complex procedures (stored in a dedicated directory) required by GriF to be run (e.g. by a `cron` daemon [93] in our case) in off-line mode, as for example:

- `GP-proxy.sh`: which checks (e.g. every 15 minutes) and (if needed) automatically renews the RC used as a Proxy for all the VRC users. It makes use of `voms-proxy-info`, `voms-proxy-init`, `myproxy-info` and `myproxy-init` gLite commands;

ISSN: 2279-8773

- `state.sh`: which checks (e.g. every 5 minutes) and then updates Grid jobs status. It makes use of `glite-job-status` and `lfc-ls` gLite commands;

- `cleargrid.sh`: which clears (e.g. once a day at 4:30) already retrieved job results and cancelled jobs (by VRC users) from the Grid middleware. It makes use of `glite-wms-job-cancel` and `lcg-del` gLite commands;

- `clearcompiled.sh`: which clears (e.g. once a day at 5:45) user-compiled binaries from YP after `<X>` days even if they were not retrieved by VRC users;

- `errors.sh`: which tries to determine (e.g. once a day at 2:30) the reasons for errors aborting Grid jobs. It makes use of `glite-job-status` and `glite-wms-job-output` gLite commands;

- `extra_values.sh`: which tries to retrieve (e.g. once a day at 3:30) extra values for Grid jobs that have terminated, for example wall time, cpu time and memory amount consumed for each "Done" job. It makes use of the same gLite commands as for the previous shell script `errors.sh`;

- `mail.sh`: which notifies (e.g. once a day at 6:00) VRC users for "Done" jobs;

ISSN: 2279-8773

- `queues.sh` and `rank.sh`: which implement the Ranking feature of GriF described in detail later on in a dedicated section. They make use of `glite-wms-job-list-match` and `lcg-infosites` gLite commands, respectively.

As well as to the `rank.sh` and `queues.sh` shell scripts devoted to the Ranking, particular mention can be made to the `state.sh` shell script (fully reported in Appendix D, item 5) which is the responsible in maintaining the Grid job status in the GriF database updated. Its main activities (also recorded in a Log file called `state.log`) are devoted to:

- Determine the unfinished jobs in the GriF Database;

- Determine, for each of them, the associated current Grid status. In this respect, the following options are considered:

  1) The returned Grid status is `Done` or `Cleared`: in both cases the `state.sh` script determines if useful results have been produced or not. In the former case the status of 'DONE' while in the latter case the status of 'FAILED' will be assigned for the related Grid job;

  2) The returned Grid status is `Running`: also in this case the `state.sh` script determines if useful results have been produced or not. In the former case the status of 'DONE' will be assigned while in the latter case the status of 'RUNNING' will be maintained for the related Grid

ISSN: 2279-8773

job;

3) The returned Grid status is `Scheduled`, `Ready`, `Waiting`, `Submitted` or `Cancelled`: in this case the Grid job status will be maintained as it is. Moreover, if more than a given period (e.g. a week) has been spent starting from the submission up to the current date of the Grid job, it will be considered as 'FAILED' and then a specific error code will be assigned to its status;

4) In all the other cases the status of 'FAILED' will be assigned for the related Grid job.

It is worth noticing here that the strategy adopted for handling `Running` jobs allows GriF, with respect to the use of the facilities offered by the Grid, to understand in advance (about 30 minutes) if a Grid job is going to reach the status FAILED or the status DONE (with success). Accordingly, using GriF a VRC user saves time in re-submitting (and/or in modifying) his/her work rather than in retrieving successful results, respectively.

The Persistence layer.

The Persistence (also called Back-end) layer of a YP is formed by two parts: the Database and the Log Subsystem (see Fig. 4.5). The former is a MySQL database named `grif-db`. It consists of nine tables (a full dump of their structure is reported in Appendix E) called, respectively, `applications`

ISSN: 2279-8773

(storing information on each application made available by GriF to its users, such as the name and description, the associated YPs list, the type of license, the area of interest and subject covered as well as the author and maintainer), `compilations` (storing information on each compilation made by users, such as the date, the YP used to compile, the source file name as well as the compiled program name), `feedbacks` (storing information on the user feedbacks provided), `jobs` (storing information on each job run on the Grid by GriF, such as the owner, the YP name used to run, the related URL and URI, the corresponding application's name and description, the related input file name, the current status, the final exit status, the submission date, the job type, the assigned CE queue, the wall and the cpu times consumed as well as the total memory used), `queues` (storing information on the available CE queues, such as their name and their ranking), `queues_tmp` (storing temporary data on CE queues, such as their performance and latency, useful to implement the Ranking used by the `queues` table and also described in detail in the next section), `services` (storing the mapping between hosted Grid Services, YPs and VOs as well as the *Costs* for each Grid Service in production), `vos` (storing the VOs list belonging, for example, to a given VRC and their related administrator) and `vousers` (storing information on each user, such as username, password and email, the type of user, the VO membership as well as the related available *Credits*).

ISSN: 2279-8773

The latter is based on a quite simple recording of the various GriF activities on dedicated YP log files. In particular, two different sources save their information on the Log Subsystem. The first source are the Web Services. They record all the (SOAP) requests and responses on a log file called `axis.log`. As illustrated in chapter 6, this log file turns out to be very useful when the tracking of VRC users and services will be needed. The second source are the shell scripts implementing the off-line procedures of GriF described in the previous section. Accordingly, each of them records its information on a different log file, like for example the `state.log` file (mentioned above), the `extra_values.log` and the `cleargrid.log` files respectively for the `state.sh`, the `extra_values.sh` and the `cleargrid.sh` shell scripts. These log files (still stored in a different directory for each Grid Service) can be turn very useful to follow what happens and also to control how GriF evolves.

The Ranking.

In this section one of the most complex and important feature of GriF is described. With the term 'Ranking' we define the ability of GriF to evaluate, by making use of adaptive algorithms developed by us, the Quality of a CE queue (considering several different variables, like for example the performance, the latency and the Grid Ranking) of a VRC (or a VO) running

ISSN: 2279-8773

Grid jobs.

When a VRC user chooses to utilize the GriF Ranking feature[2], YP automatically selects (querying the GriF database) the CE queue that has to be used for the current job run. In this respect, each YP provides a shell script called `rank.sh` (fully reported in Appendix D, item 6) that is responsible for calculating the ordered list of CE queues (currently in the GriF database) at regular intervals (e.g. 15 minutes in our testbed). Moreover, a second shell script (which can be run, for example, once a week (say on Sunday at 5:30)) called `queues.sh` (fully reported in Appendix D, item 7) is provided in order to update the CE queues list (in other words, it just ensures that any new CE queue belonging to a given VRC or VO will be considered by the Ranking feature) on the GriF database (tables `queues` and `queues_tmp` of `grif-db`). Both shell scripts save their Log information on the `rank.log` file of the Log Subsystem. Therefore, the Ranking of a CE queue is given by the following formula:

$$R(q) = R_{GRID} + R_{GRIF} \qquad ; \qquad R_{GRID} > 0 \qquad (4.1)$$

[2]YC allows VRC users not to use GriF Ranking. In this case, a minimum level of reliability will be ensured (choosing to privilege the amount of available memory rather than the maximum allowed wall time) counting on the available features of the Grid middleware based on the Requirements and the Rank variables of JDL supported by GriF.

ISSN: 2279-8773

where R_{GRID} is a positive integer number representing the positional order of the related CE queue resulting from the Grid Ranking (see item 2 below). This means that, in order to be considered by GriF Ranking, a CE queue has to be first returned by Grid Ranking (in fact, in Eq. 4.1, $R_{GRID} > 0$ is the necessary and sufficient condition for a CE queue to own a $R(q)$ value and then to be considered active). In other words, Grid Ranking selects which CE queues have to be considered; then GriF Ranking determines their position in a quality-ordered list.

In Eq. 4.1 R_{GRIF} can be defined as follows:

$$R_{GRIF} = K_{PERF} * \frac{n_f}{n_t} - \left(B[\mathbf{Q}] + \frac{1}{K_{LAT} * \frac{\sum_{i=1}^{n_d}(wt_i - ct_i)}{n_d}} \right) \tag{4.2}$$

where:

(a) K_{PERF} is a constant weighing the ratio between the failed (n_f) and the total (n_t) number of jobs run by a CE queue (that we consider here as *Performance* (P) of a CE queue);

(b) K_{LAT} is a constant weighing the *Average Latency* (AL) in the GriF Ranking defined as the ratio between the sum over the n_d "Done" jobs of the differences between wall time (wt_i) and cpu time (ct_i) of each i "Done" job and n_d;

(c) B is a constant (called `bonus` in the `rank.sh` shell script) depending by

ISSN: 2279-8773

the AL of each CE queue which can assume the values quoted in the vector $\mathbf{Q}(q_1, q_2, q_3, q_4)$ according to four CE queue classes called '*fast*', '*normal*', '*slow*' and '*not available*', respectively. These classes are bound by the upper limit values (in minutes) given in the vector $\mathbf{L}(l_1, l_2, l_3)$. Both **Q** and **L** values are based on the GriF experience and can be also optimized by making use of adaptive learning mechanisms enabled to properly update any previous settings in real-time.[3]

Therefore, the `rank.sh` shell script performs the following five macro-activities in order to:

1. Determine the (bad) CE queues which will never be considered by GriF. They correspond to a failure of 100 percent in running Grid jobs;

[3]In the running version of GriF, we have chosen $\mathbf{Q} = (10, 3, 0, -10)$ and $\mathbf{L} = (5, 60, 1440)$. Accordingly, $B = 10$ when AL $\leq$ 5 minutes (*fast* CE queue class), $B = 3$ when $5 <$ AL ≤ 60 minutes (*normal* CE queue class), $B = 0$ when $1 <$ AL ≤ 24 hours (*slow* CE queue class) and $B =$ -10 otherwise (*not available* CE queue class).
We have also considered and experimented for the implementation on the next versions of GriF a function of continue values of the type:

$$B = a + b \log_{10}(AL) \tag{4.3}$$

where a and b are coefficients to be properly tuned in time (e.g. by using adaptive algorithms and/or cut-off strategies). For example, the values of a and b best fitting the above (discontinuous) dependence of B from **Q** and **L** are $a = 11.82$ and $b = -3.98$.

ISSN: 2279-8773

2. Find and order the remaining CE queues according to the Grid middleware facilities (which return only available CE queues) as mentioned above. Accordingly, a fake Grid job will make use of the GLUE Schema [94] (in particular of both the attributes `Requirements` and `Rank`) to carry out five attempts (creating at runtime a proper JDL file specifying each time different requirements and related order, from highest to lowest values) in order to find, respectively:

   (a) Those CE queues (ordered from lowest to highest cpu time) having the (maximum − current) number of running jobs > 0, the number of free cpus > 0, the number of waiting jobs equal to 0 and ensuring a wall time of 2 days at least. *Or (if no CE queue matches this criterion)*:

      (b) Those CE queues (ordered as in the previous case) having the number of free cpus > 0, the number of waiting jobs equal to 0 and ensuring a wall time of 2 days at least. *Or (if no CE queue matches this criterion)*:

         (c) Those CE queues (ordered as in the previous case) having the number of waiting jobs equal to 0 and ensuring a wall time of 2 days at least. *Or (if no CE queue matches this criterion)*:

ISSN: 2279-8773

(d) Those CE queues (ordered from the lowest to the highest number of waiting jobs) having the number of free cpus > 0 and ensuring a wall time of 2 days at least. *Or (if no CE queue matches this criterion)*:

(e) Those CE queues (ordered as in the previous case) ensuring a wall time of 2 days at least. *Or (if no CE queue matches this criterion)*:

(f) A new R is not calculated and the previous Ranking is adopted;

3. Check (when the CE queues set resulting from the previous step is not empty) if it contains (in the order returned by Grid Ranking) a new CE queue (that has never been used by GriF). In this case, a network connection test will take place for the first one and, if positive, it will be used for the next job run made by a VRC user. Otherwise the `rank.sh` shell script will assign the R_{GRID} value to each CE queue in the resulted set as described above;

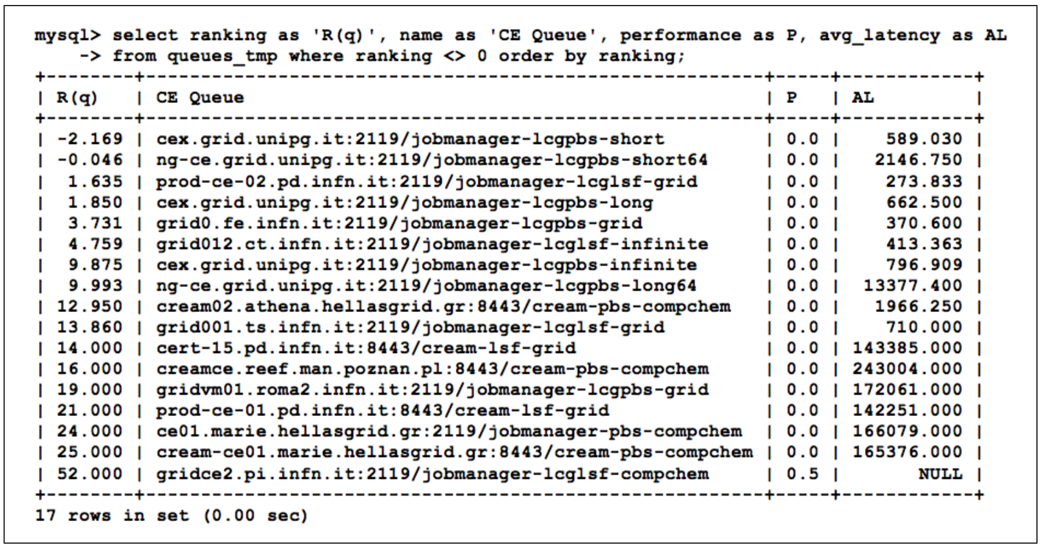
4. Calculate (when the CE queues set resulting from the step 3 is not empty and a new CE queues has not been selected in the previous step) R_{GRIF} for each CE queue in the resulted set as defined in Eq. 4.2.

ISSN: 2279-8773

Accordingly, R will be determined for each CE queue (see Eq. 4.1) and, after a further network connection test for each of them (as in the case of a new CE queue), the GriF database will be updated properly;

5. Clear YP and UI environments.

In our testbed, we have considered K_{PERF} = 100 and K_{LAT} = 1 / K_{PERF} in order to privilege the evaluation of P making use of AL only to refine the resulting quality-ordered CE queues list (useful when in presence of very similar CE queues in respect of P). As a matter of fact, in Fig. 4.6 the resulting $R(q)$ (applied to all the CE queues of the COMPCHEM VO that at present are 77) related to an interval of 15 minutes, is illustrated.



```
mysql> select ranking as 'R(q)', name as 'CE Queue', performance as P, avg_latency as AL
    -> from queues_tmp where ranking <> 0 order by ranking;
+--------+--------------------------------------------------------+-----+------------+
| R(q)   | CE Queue                                               | P   | AL         |
+--------+--------------------------------------------------------+-----+------------+
| -2.169 | cex.grid.unipg.it:2119/jobmanager-lcgpbs-short         | 0.0 |    589.030 |
| -0.046 | ng-ce.grid.unipg.it:2119/jobmanager-lcgpbs-short64     | 0.0 |   2146.750 |
|  1.635 | prod-ce-02.pd.infn.it:2119/jobmanager-lcglsf-grid      | 0.0 |    273.833 |
|  1.850 | cex.grid.unipg.it:2119/jobmanager-lcgpbs-long          | 0.0 |    662.500 |
|  3.731 | grid0.fe.infn.it:2119/jobmanager-lcgpbs-grid           | 0.0 |    370.600 |
|  4.759 | grid012.ct.infn.it:2119/jobmanager-lcglsf-infinite     | 0.0 |    413.363 |
|  9.875 | cex.grid.unipg.it:2119/jobmanager-lcgpbs-infinite      | 0.0 |    796.909 |
|  9.993 | ng-ce.grid.unipg.it:2119/jobmanager-lcgpbs-long64      | 0.0 |  13377.400 |
| 12.950 | cream02.athena.hellasgrid.gr:8443/cream-pbs-compchem   | 0.0 |   1966.250 |
| 13.860 | grid001.ts.infn.it:2119/jobmanager-lcglsf-grid         | 0.0 |    710.000 |
| 14.000 | cert-15.pd.infn.it:8443/cream-lsf-grid                 | 0.0 | 143385.000 |
| 16.000 | creamce.reef.man.poznan.pl:8443/cream-pbs-compchem     | 0.0 | 243004.000 |
| 19.000 | gridvm01.roma2.infn.it:2119/jobmanager-lcgpbs-grid     | 0.0 | 172061.000 |
| 21.000 | prod-ce-01.pd.infn.it:8443/cream-lsf-grid              | 0.0 | 142251.000 |
| 24.000 | ce01.marie.hellasgrid.gr:2119/jobmanager-pbs-compchem  | 0.0 | 166079.000 |
| 25.000 | cream-ce01.marie.hellasgrid.gr:8443/cream-pbs-compchem | 0.0 | 165376.000 |
| 52.000 | gridce2.pi.infn.it:2119/jobmanager-lcglsf-compchem     | 0.5 |       NULL |
+--------+--------------------------------------------------------+-----+------------+
17 rows in set (0.00 sec)
```

Figure 4.6: **The GriF Ranking at work.**

In Fig. 4.6 only 17 CE queues (second column) have been ranked (in other words, only these CE queues have satisfied the requirements of the Grid Ranking, as described above in item 2). Accordingly, in the first column the calculated Ranking (please note that lower values of $R(q)$ correspond to higher evaluations) is shown. It is worth noticing here that only the last CE queue in the resulting set has P $\neq 0$ (third column). In particular, the reported value of 0.5 means a 50 percent of failures. For this reason, the related CE queue was returned in the last position with a $R(q)$ value of 52. On the contrary, the best CE queue has a $R(q)$ value of -2.169 thanks to which it is elected by GriF as the active CE queue for the next 15 minutes. In this respect, it is also worth noticing here how a YP interacts with GriF Ranking in time: the fact that the active CE queue is used for the current Grid job could introduce some changes in the evaluation of the CE queue itself (without recalculating a new $R(q)$ value). As a matter of fact, to the end of avoiding the use of the same CE queue for subsequent Grid job runs (submitted by VRC users) within the same time interval, a Round Robin (RR) scheduling algorithm [95] has been implemented in GriF. In particular, a special field (of the `queues` and `queues_tmp` tables of the GriF database, see Appendix E) called `rr` is incremented by 1 at each use of the active CE queue. Accordingly, when subsequent job runs between two consecutive executions of the `rank.sh` shell script (`rr` values are, therefore, reset to 0 at each `rank.sh` run) occur, the CE queue with the lowest `rr` value (then

ISSN: 2279-8773

following the order resulting from $R(q)$ when it is not unique) is activated, used and then rotated.

4.3.4 Yet a Consumer (YC).

YC is the multi-platform Java-based client (tested on Mac OS X 10.6, Scientific Linux 5.2 and Windows XP platforms) that implements the SOA Service Consumer (that is the Presentation layer) that of Figs. 4.1 and 4.5. It supports SOAP 1.1 and XML 1.0 offering to VRC users all the interfaces to the use of all the GriF facilities. In order to use GriF, one needs to have the '`java`' command in the system path (if it isn't, the PATH environment variable has to be checked) and a Java Runtime Environment (JRE) ≥ 1.6. From a network point of view, an Internet connection with outbound connections to the HTTP protocol and port 8080 active, is needed. Please note that a degraded Internet connection can introduce considerable delays during all the interactions between GriF and the Grid. Moreover, in the client machine an utility to access the Grid job results which will be returned in a compressed TAR archive format (`.tar.gz`) is required. After downloading for free the GriF package choosing between [84, 96, 97], YC can be started with the command:

```
# java -jar GriF.jar
```

ISSN: 2279-8773

Accordingly, the authentication phase will take place under the responsibility of a specific Web Service (see Appendix F, item 1, reporting part of the YC code illustrating this step). After YC is loaded (authentication successful), four different panels called respectively 'Run Applications', 'Manage Jobs', 'Set Providers' and 'Contacts' are displayed. Please note that multiple instances of YC are allowed.

Within the 'Run Applications' panel, a desired binary application (or a shell script) on the EGI Grid (after uploading it by pressing the 'Upload' button) can be run. One has also the option of choosing among those already offered by the VRC to its users. Next, the related input needs to be provided either as a single plain text file or as compressed TAR or ZIP format file for multiple input (in this case, respectively, the `.tar.gz` or the `.zip` extension is required). Before running, one can choose between using the GriF Ranking described above or not. In the latter case, reliability is not guaranteed; yet one can choose between three running days or three GB RAM ensured for the Grid job. Accordingly, by pressing the 'Start' button, the job is started on the Grid (see Appendix F, item 2). In this respect, some delays during this task (up to 30–35 seconds) can occur (especially when one chooses not to use GriF Ranking). At the same time, some errors apparently resulting from the use of YC can be related to the real status of the Grid middleware and of the network connection used. Therefore, sometimes it might happen that one

ISSN: 2279-8773

can experience troubles (e.g. on retrieving "Done" results or in running Grid jobs) not depending on GriF. Moreover, please note that the application is allowed to produce more than one file of results (returned under the form of compressed TAR archive).

Within the 'Manage Jobs' panel, the management of pending Grid jobs takes place. In the main window one has his/her pending Grid jobs list that can be refreshed by pressing the button 'Refresh'. Just by selecting one Grid job one can see its description on the left. Please note that "Done" jobs are returned on top of the list. The remaining Grid jobs are ordered by status and then by submission date. The effective status of the Grid jobs is updated by YP at regular intervals, as mentioned above. When a Grid job is cancelled (by pressing the 'Delete' button), or its results are retrieved, it will disappear from the list. One can retrieve Grid job results only for "Done" jobs by pressing the 'Get Results' button (see Appendix F, item 3). After retrieving, the related results will be automatically purged from the Grid. Accordingly, remember that they have to be saved in a safe way on the client side.

Within the 'Set Providers' panel, the search & setting of different YPs (making use of the client part of the UDDI4J library mentioned above and of the `inquiry` URL of YR, see Fig. 4.5) is allowed by a simple copy & paste. For example, this function can be useful when more than one YP is returned (for

ISSN: 2279-8773

the same Grid Service) and one wants to change it (e.g. after previous YP server errors).

Within the 'Contacts' panel, one can send messages and feedbacks (but also questions and errors) to the GriF people. Moreover, other minor functions of the YC (also documented by Javadoc in the '`docs/`' directory shipped within the GriF package) are the following:

- Checking of the health of the whole system (Database, Web Services, UI connection, LFC subsystem and CE queues availability) by pressing the 'System Status' button of the 'Manage Jobs' panel (e.g. when one changes type and/or address of network connection);
- Checking of the last job run on the Grid by pressing the 'Check the status on the Grid in real-time' button of the 'Running Applications' Panel (e.g. to immediately verify that the Grid has correctly taken in charge the submitted Grid job);
- Saving of YC messages by pressing the 'Save' button of the 'Running Applications', 'Manage Jobs' and 'Set Providers' panels (e.g. when one desires to send to the GriF team log information or locally store useful data returned by YC);
- Clearing of YC messages by pressing the 'Clear' button of the 'Running Applications', 'Manage Jobs' and 'Set Providers' panels;

ISSN: 2279-8773

- Exiting from YC by pressing the 'Logout' button.

4.3.5 SOAP with Attachments in Java.

SOAP with Attachments (SwA) or Multipurpose Internet Mail Extensions (MIME) for Web Services refers to the method of using Web Services to send and receive files using a combination of SOAP and MIME, primarily over HTTP [98]. As an XML-based messaging protocol, SOAP messages require considerable processing power and memory. All parts of a SOAP message must conform to XML rules for allowed characters and character sequences preventing binary data from being included directly. Furthermore, SOAP implementations typically parse the entire SOAP message before deciding what to do with the contents, so that large data fields could easily exceed available memory. For all these reasons, it was recognized that SOAP requires some mechanism for carrying large payloads and binary data as an attachment rather than inside the SOAP message.

Attachments are available to Java developers through both JAX-RPC (the Java API for XML-based RPC) and SAAJ (SOAP with Attachments API for Java). The difference between them is the level of abstraction.J AX-RPC is a high-level API that is more abstract than SAAJ. Accordingly, JAX-RPC hides most of the protocol-oriented aspects of SOAP behind a Remote

ISSN: 2279-8773

Method Invocation (RMI) layer (java developers works on Java objects and the pre-processor turns them into SOAP nodes) while SAAJ is closer to the protocol. As a matter of fact, SAAJ can model the structure of SOAP messages, in particular the SwA (java developers can use SAAJ to create, read, or modify SOAP messages) and their APIs include classes and interfaces modeling SOAP elements, XML namespaces, attributes, text nodes and MIME attachments[4].

Moreover, SAAJ relies on the Java Activation Framework (JAF) to add attachments and to handle the conversion of objects into data streams and viceversa. JAF provides a framework for dynamically discovering visual widgets to handle any kind of data described by MIME headers. While JAF is focused on the GUI side of things, as a framework for dynamically discovering objects that can manipulate specific MIME types it is useful for non-visual system like SAAJ as well. In particular, JAF can map Java types to special handlers that seamlessly convert them to streams of data. This mechanisms is important in SOAP messaging because it allows SAAJ to automatically convert Java objets into raw data contained by SwA MIME parts. Central to JAF is the `javax.activation.DataHandler` class which is also the central figure in SAAJ facilities for creating and adding attachments to SOAP mes-

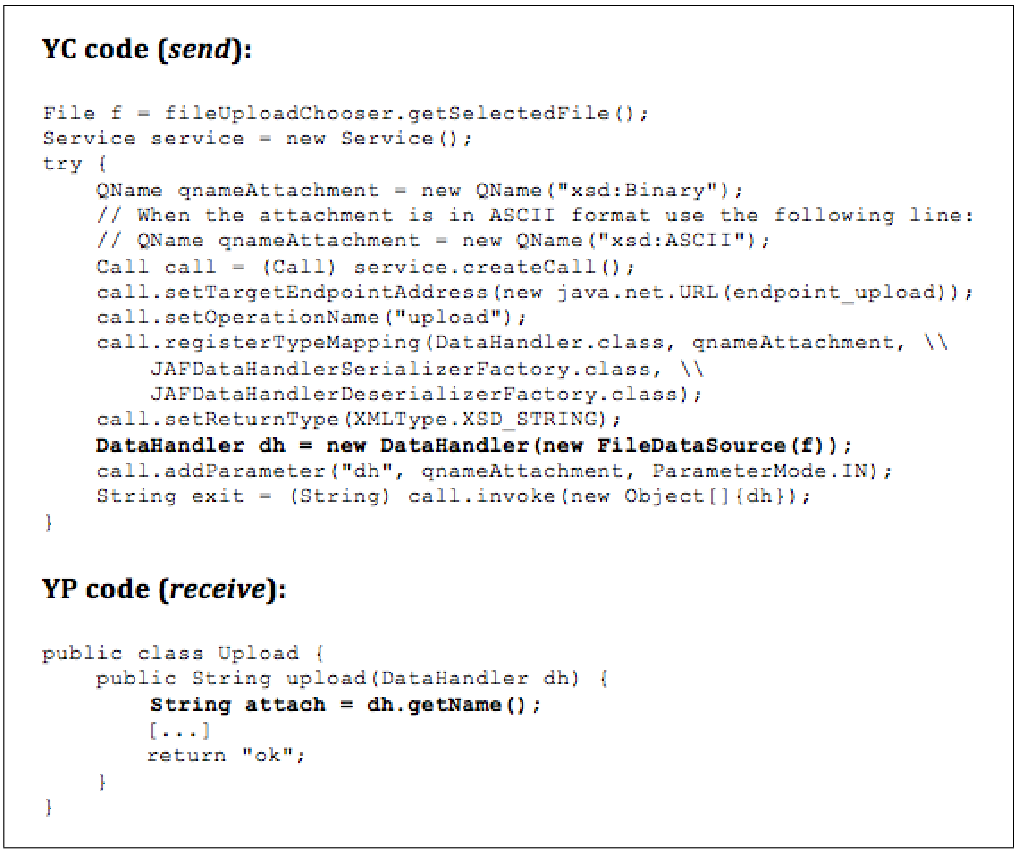
[4]Starting from version 1.3, SAAJ conforms also to the SOAP 1.2 specification (in the present Thesis work SAAJ 1.2 has been used since SOAP 1.1 has been adopted).

ISSN: 2279-8773

Virt&I-Comm.3.2012.22

sages. Whenever an attachment is added to a SAAJ message, the attached object is invariably embedded in a `DataHandler` object (extensively used in GriF) that provides methods for reading and writing data streams, accessing the MIME type of the data and creating a Java object able to represent the data in a stream [99].

As an example, part of the (simplified) code implementing the upload of an application from YC to YP and the retrieving of the results from YP to YC are illustrated in Fig. 4.7 and 4.8, respectively.



YC code (*send*):

```
File f = fileUploadChooser.getSelectedFile();
Service service = new Service();
try {
    QName qnameAttachment = new QName("xsd:Binary");
    // When the attachment is in ASCII format use the following line:
    // QName qnameAttachment = new QName("xsd:ASCII");
    Call call = (Call) service.createCall();
    call.setTargetEndpointAddress(new java.net.URL(endpoint_upload));
    call.setOperationName("upload");
    call.registerTypeMapping(DataHandler.class, qnameAttachment, \\
        JAFDataHandlerSerializerFactory.class, \\
        JAFDataHandlerDeserializerFactory.class);
    call.setReturnType(XMLType.XSD_STRING);
    DataHandler dh = new DataHandler(new FileDataSource(f));
    call.addParameter("dh", qnameAttachment, ParameterMode.IN);
    String exit = (String) call.invoke(new Object[]{dh});
}
```

YP code (*receive*):

```
public class Upload {
    public String upload(DataHandler dh) {
        String attach = dh.getName();
        [...]
        return "ok";
    }
}
```

Figure 4.7: **Part of the code implementing SwA (from YC to YP) in Java.**

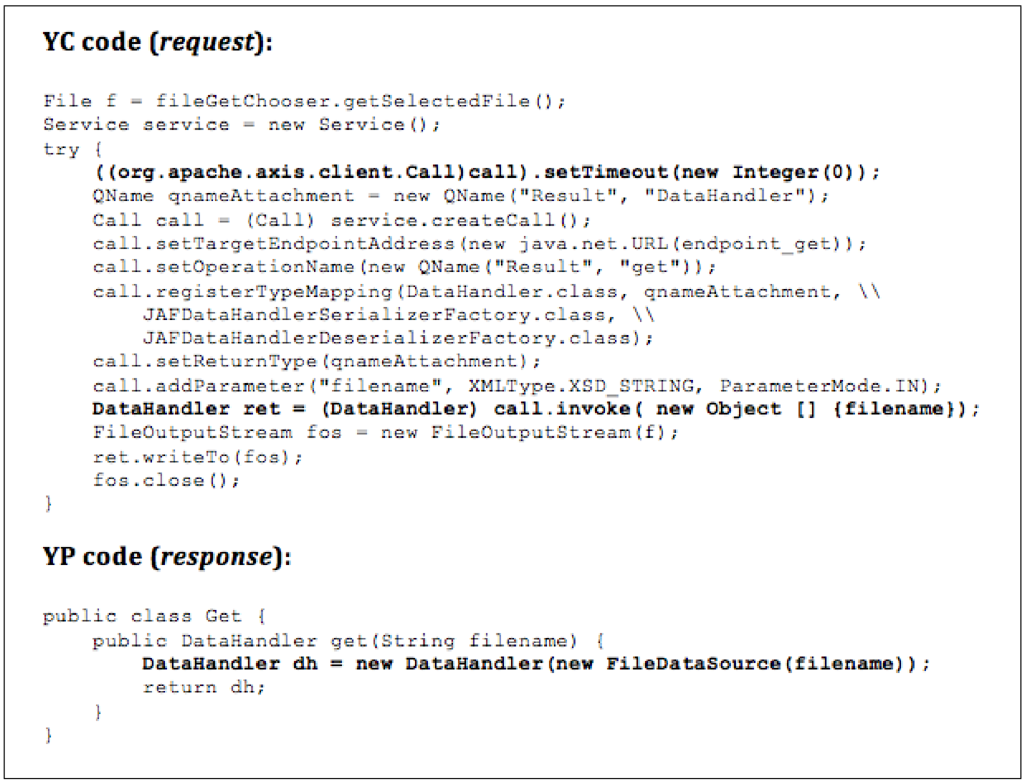
ISSN: 2279-8773

Accordingly, in Fig. 4.7 are highlighted (in bold) the `DataHandler` reference to the application file to be send from YC (line 14 of the YC code part) as well the receiving of the related file by YP (line 3 of the YP code part) while in Fig. 4.8 are highlighted the definition of the timeout (unlimited due to the possibility of having very large files) when YC gets a results file (line 4 of the YC code part), the request to YP for a results file (called `filename`) making use of the `DataHandler` type (line 12 of the YC code part) as well the response of the YP giving back the requested file making use of the `FileDataSource` class (line 3 of the YP code part) as already done by YC (see the ”upload” case of Fig. 4.7).

Moreover, Fig. 4.9 (where two typical SOAP messages respectively formed by the Envelope and the related Body with at least an element, are shown) illustrates the one-to-one correlation between the elements in a SOAP 1.1 message (where the ’In message’ part is the SOAP request and the ’Out message’ part is the SOAP response) and the variables and values used by YC and YP. It is worth noticing here (line 12) the use made by SAAJ of the `cid` reference (according to the ”upload” case of Fig. 4.7). As a matter of fact, the attachment is associated to the `dh` parameter by adding an `href` attribute and then referred to through a CID (Content-ID, the scheme providing identifiers for messages and their body parts [100]) URL.

As already mentioned, in a SOAP request and response one can find the

ISSN: 2279-8773



YC code (*request*):

```
File f = fileGetChooser.getSelectedFile();
Service service = new Service();
try {
    ((org.apache.axis.client.Call)call).setTimeout(new Integer(0));
    QName qnameAttachment = new QName("Result", "DataHandler");
    Call call = (Call) service.createCall();
    call.setTargetEndpointAddress(new java.net.URL(endpoint_get));
    call.setOperationName(new QName("Result", "get"));
    call.registerTypeMapping(DataHandler.class, qnameAttachment, \\
        JAFDataHandlerSerializerFactory.class, \\
        JAFDataHandlerDeserializerFactory.class);
    call.setReturnType(qnameAttachment);
    call.addParameter("filename", XMLType.XSD_STRING, ParameterMode.IN);
    DataHandler ret = (DataHandler) call.invoke( new Object [] {filename});
    FileOutputStream fos = new FileOutputStream(f);
    ret.writeTo(fos);
    fos.close();
}
```

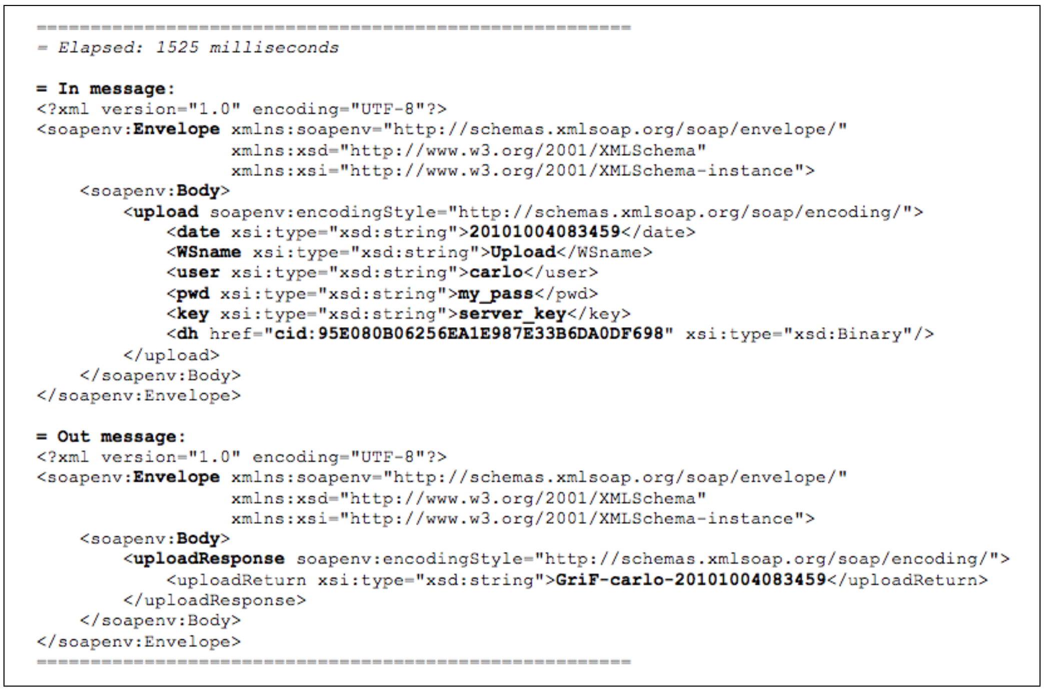
YP code (*response*):

```
public class Get {
    public DataHandler get(String filename) {
        DataHandler dh = new DataHandler(new FileDataSource(filename));
        return dh;
    }
}
```

Figure 4.8: **Part of the code implementing SwA (from YP to YC) in Java.**

mapping between all the variables and the values that are used in any Web Service communication of GriF. As a matter of fact, in the SOAP request of Fig. 4.9 the date (`date` is equal to ’`20101004083459`’), the name of the GriF Web Service invoked (`WSName` is equal to ’`Upload`’), the username (`user` is equal to ’`carlo`’) and his password (`pwd` is equal to ’`my_pass`’) as well as the server key against the MITM (`key` is equal to ’`server_key`’) are highlighted for the `upload` method. Accordingly, in the SOAP response of Fig. 4.9 the related method called `uploadResponse` returns the string value

ISSN: 2279–8773



```
==================================================
= Elapsed: 1525 milliseconds

= In message:
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <upload soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <date xsi:type="xsd:string">20101004083459</date>
            <WSname xsi:type="xsd:string">Upload</WSname>
            <user xsi:type="xsd:string">carlo</user>
            <pwd xsi:type="xsd:string">my_pass</pwd>
            <key xsi:type="xsd:string">server_key</key>
            <dh href="cid:95E080B06256EA1E987E33B6DA0DF698" xsi:type="xsd:Binary"/>
        </upload>
    </soapenv:Body>
</soapenv:Envelope>

= Out message:
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <uploadResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <uploadReturn xsi:type="xsd:string">GriF-carlo-20101004083459</uploadReturn>
        </uploadResponse>
    </soapenv:Body>
</soapenv:Envelope>
==================================================
```

Figure 4.9: **A typical SOAP message in GriF.**

of `GriF-carlo-20101004083459` corresponding to a real upload in which a unique application name is determined by YP and then returned back to YC.

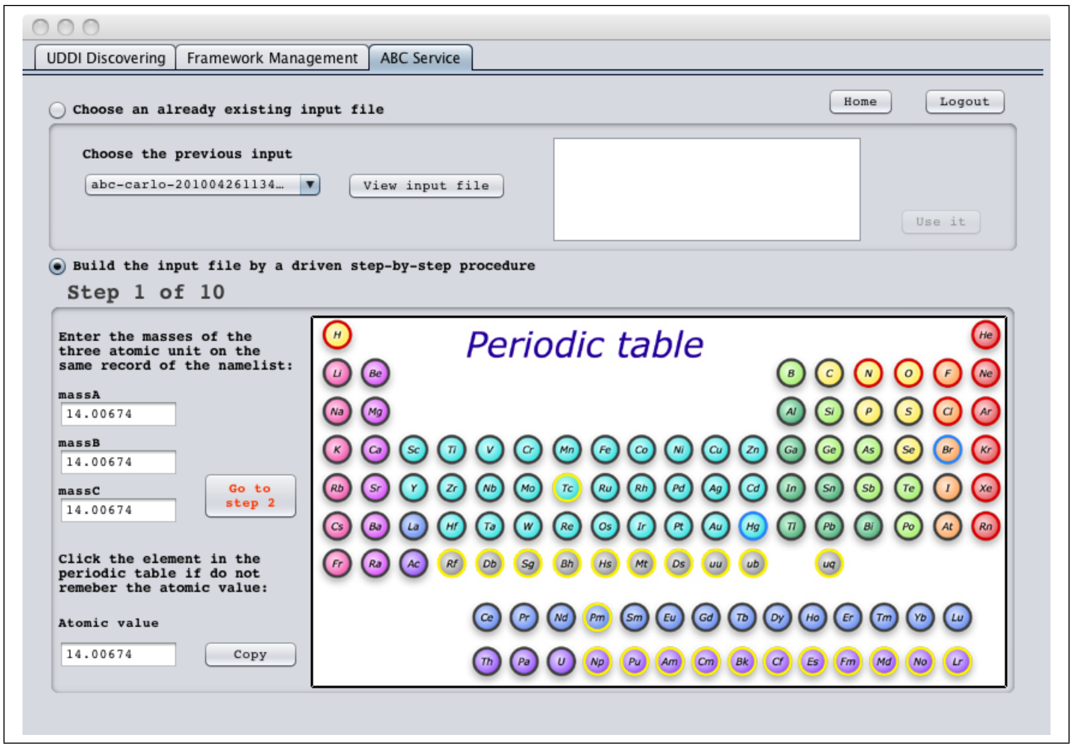
4.3.6 Version 1.0: the Origin of GriF.

The development of GriF has been started by the end of the year 2008. One year later (December 2009), the first (beta) version of GriF (`1.0`) was released. This version (successfully tested on the EGI Grid middleware) was fully devoted to the implementation as Grid Service of the already mentioned ABC program in which VRC users can select various options (like modifying part of the code or accepting to be driven in building the input) before enter-

ISSN: 2279–8773

Virt&l–Comm.3.2012.22

ing the running phase. Particular attention was paid to the implementation of the so called 'Wizard' modality, in which a minimal workflow driving the user from the beginning (the generation of the input file, as illustrated in Fig. 4.10) to the end (the retrieval of the results from the Grid) of the process was developed.


Choose an already existing input file

Choose the previous input: abc-carlo-201004261134…

Build the input file by a driven step-by-step procedure

Step 1 of 10

Enter the masses of the three atomic unit on the same record of the namelist:

massA: 14.00674

massB: 14.00674

massC: 14.00674

Click the element in the periodic table if do not remeber the atomic value:

Atomic value: 14.00674

Figure 4.10: **The first step of the Wizard process of GriF 1.0 in building the input file.**

For example, before running, the generated input file is checked and validated using a dedicated Web Service to spot syntactic (like improper typing and/or wrong number/name of parameters specified) and semantic (like

ISSN: 2279-8773

`kmax > jtot`, `nnrg = 0` or `enrg > emax`, see Fig. 4.19) errors.

In the developed procedure the user preliminarily chooses between a Classical Mechanics (Quasi-Classical-Trajectory (QCT)) and a Quantum Mechanics (QM) treatment and then selects the chemical system to be investigated. The way these two choices are dealt is different. In fact, while the choice between a QCT and a QM treatment implies the adoption of a different application and of the related set of executables, the choice of a different system can be confined to the modification of the values of some parameters of the input file (like masses and energies) and the adoption of a different PES. The application workflow was, therefore, designed to search on the Web for the availability of an appropriate PES and related parameters before starting the integration of the scattering equations. If a proper PES is not available, GriF 1.0 was conceived to search on the Web for the availability of a sufficiently large set of ab initio potential energy values for the system considered. If such values are available they are fitted to a proper functional form using an appropriate fitting procedure (see for example [60]). If the ab initio values are not available, one has to produce them by running ab initio electronic structure calculations using again an ad hoc procedure (see for example [56]). Moreover, this version was confined to the case in which various PESs were already available.

In this case, when using GriF 1.0, users access a User Driven graphical in-

ISSN: 2279-8773

terface on which they can start the execution of the ABC application just by pressing the button Start (this is the training-purpose case in which the default H + H_2 system, the default PES (called `lsth.f` that implements the celebrated LSTH [101, 102] PES) and the related default input file are used). Users can, instead, choose to prepare their own input file still for the same default PES (for example when isotopic effects or the dependence of the reactive probability on the collision energy has to be investigated). Alternatively, users can also choose to adopt a different PES by selecting it among those available in the ad hoc panel (for which a corresponding executable is already available) or to adopt a new one (in this case, before getting started, VRC users have to produce the new executable by using the ad hoc interface enabling the compilation of the ABC source code).

Finally, the Grid job is submitted for the distributed execution on the Grid using an adaptive algorithm making use of the Parametric Job option [103]. GriF 1.0 was structured so as to allow the carrying out of computational campaigns (this is one of main original goals of the COMPCHEM VO) made of several distributed jobs in which only one parameter (e.g. total energy) varies (parameter study) with no direct human intervention. To do that, before the program is executed, GriF 1.0 splits the Grid job into N subjobs to be distributed on the Grid depending both on the total number of energies to be calculated and on the Distribution Grid Constant (DGC). DGC is an

ISSN: 2279-8773

integer number tentatively set by GriF to indicate how many energies will be run on each worker node (e.g. if the number of energies is set equal to 10, *DGC* is equal to 1 and therefore *N* is equal to 10). The value of *DGC* is chosen on the basis of the experience of the Grid middleware and of the Web Service requested. For example, a typical ABC calculation consisting of a set of 10 energies run on the section of the Grid at that time available to COMPCHEM for about 2-3 days. Accordingly, when a set of more than 10 energies need to be considered, GriF 1.0 splits the Grid job in *N* subjobs with *N* being large enough to keep *DGC* below the limiting value of 10. To this end, GriF 1.0 invokes the mentioned Parametric Jobs option using `enrg`, `dnrg` and `nnrg` as job starting (*ParamStart*), stepping (*ParamStep*) and ending (*ParamStop* that is set equal to `enrg` + `dnrg` · `nnrg`, see Fig. 4.19), respectively.

As a result, in the test run 10 subjobs with *DGC*=1 were distributed (at the same time) over 10 cpus of the Grid (using only nodes of the same type) for the LEPS [104] and L3 [105] PESs with the input file shown in Fig. 4.19. The resulting average execution time for the two distributed jobs was 329 (for LEPS) and 459 (for L3) minutes while that of the sequential ones (obtained by forcing the Grid to use only one of the above mentioned nodes) was 552 (for LEPS) 1773 (for L3) minutes, respectively. Accordingly, when considering also some expected Grid delays (like those related to the waiting

ISSN: 2279-8773

and/or scheduled jobs and to the Grid latency as well), distributing in parallel on the Grid the corresponding 10 subjobs by the use of the Parametric Jobs execution modality (implemented by GriF 1.0 as mentioned above), the resulting saved wall clock time came out to be, respectively, 40 and 74 percent.

Further performance details were that in the parametrized versions the original sequential input split using blocks of 1 energy for each subjob the time difference between the shortest and the longest run was 278 and 358 minutes, respectively. When one uses a large number of energies (as in this case in which the number of energies, DGC and N were chosen to be 153, 10 and 16, respectively) the total saved time underwent a further significant increase (up to 93 percent). This is not surprising because the first part of the ABC code (that is common to all runs) is not repeated for each single energy calculation (as in the case illustrated before) but it is performed once in each subjob made of 10 energies. Accordingly, the theoretical limit associated with the minimum execution time achievable in absence of Grid latencies, errors and repeated common parts of the code is equal to the execution time of a no jobs-splitting run divided by N (in the case in which the number of energies is multiple of DGC) or by $N - 1$ (in all the other cases).

With respect to the Quality evaluation, using GriF 1.0 it is possible to collect information on the type of PESs preferentially utilized by the users, on

ISSN: 2279-8773

how complex is the job assembled for running on the Grid (e.g. considering the parameter values and, when is the case, the syntactic and semantic errors made), on which are the critical parameters for Grid job execution (e.g. date, time, program name and if it is a new compiled version) and on results retrieval activities.

4.3.7 Version 2.0: Details of a Test Run.

As already mentioned, quite recently we have transformed GriF into a more general user-friendly tool for running Grid jobs (GriF version `2.0`, July 2010) for which a detailed step-by-step example with screenshots is illustrated here in detail. In the example, we make use of the GP Grid Service within its experimental feature for the compilation of Grid program sources. Moreover, a generic application (named `YATTEC`) instead of the ABC program (or even shell scripts) is considered. After entering in GriF (specifying a proper username and password, as shown in Fig. 4.11), the main window of YC will appear.

Within the first panel called "Compile Programs" and by pressing the button "Load and Compile" a compressed (in a `.tar.gz` format) package containing the sources of the YATTEC application is loaded on YP as shown in Fig. 4.12. After its successful compilation (performed by GriF on the UI),

ISSN: 2279-8773

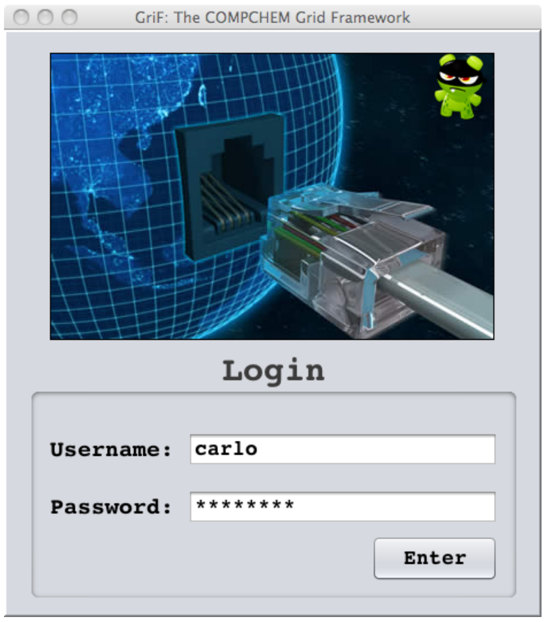


Figure 4.11: **GriF 2.0: Login.**

the corresponding binary file (called `yattec.x` in this example) is retrieved, as shown in Fig. 4.13.

Within the second panel called "Run Applications" and by pressing the button "Upload your Application" the YATTEC executable can be uploaded on YP, as shown in Fig. 4.14. Moreover, starting from this panel, the upload of any kind of already compiled application (and/or shell scripts) as well as the choice between those offered by the GP Grid Service (available by the dropdown menu of Fig. 4.15), is allowed. Accordingly, in Fig. 4.15, after uploading a plain-text file (the input), filling the "Short Description" textarea and then selecting the use of "GriF Ranking" from the available "Options", by pressing the button "Start" the YATTEC application is run on the Grid

ISSN: 2279-8773

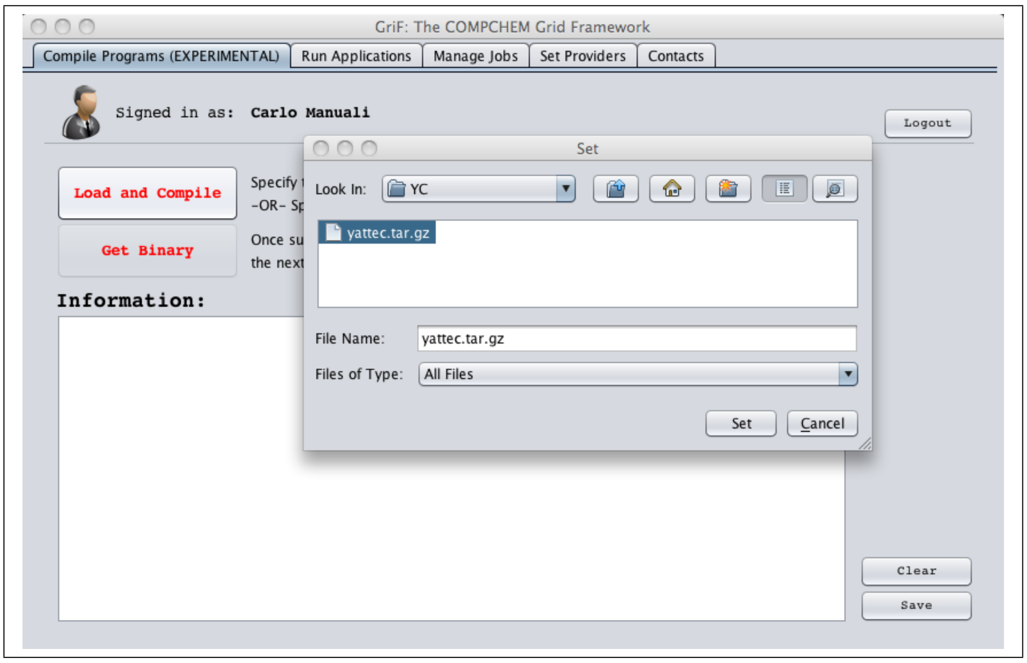
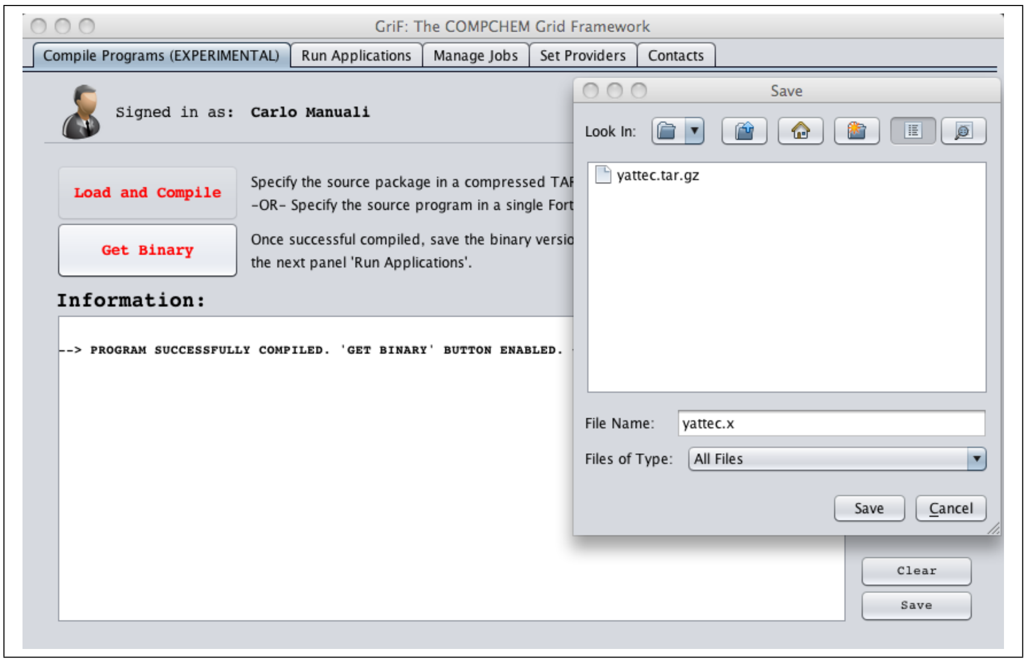


Figure 4.12: **GP Grid Service: Loading and Compiling Program Sources.**

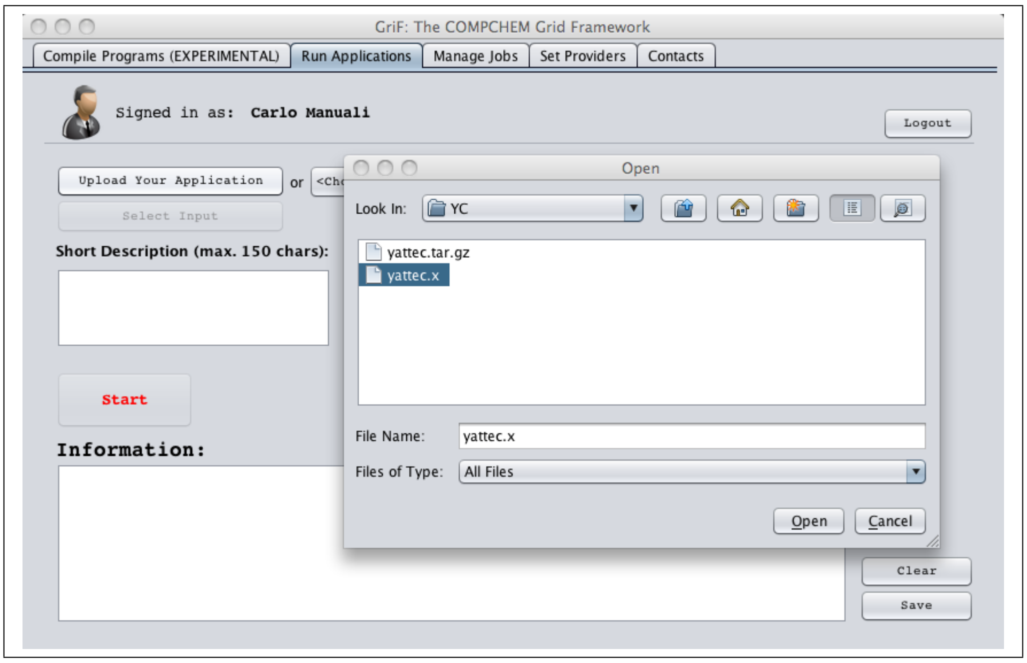


Specify the source package in a compressed TAR
-OR- Specify the source program in a single Fort
Once successful compiled, save the binary versio
the next panel 'Run Applications'.

```
--> PROGRAM SUCCESSFULLY COMPILED. 'GET BINARY' BUTTON ENABLED.
```

Figure 4.13: **GP Grid Service: Retrieving Compiled Binaries.**

and the related `jobid` as well as the CE queue used (corresponding to the first CE queue of Fig. 4.6) by GriF, are returned on the "Information" area. In addition, by pressing the button "Check the status on the Grid in real-time" a preliminary check on the status of the Grid job (`Ready` in our example of Fig. 4.15) is performed.



Signed in as: Carlo Manuali

Short Description (max. 150 chars):

Information:

Look In: YC

yattec.tar.gz
yattec.x

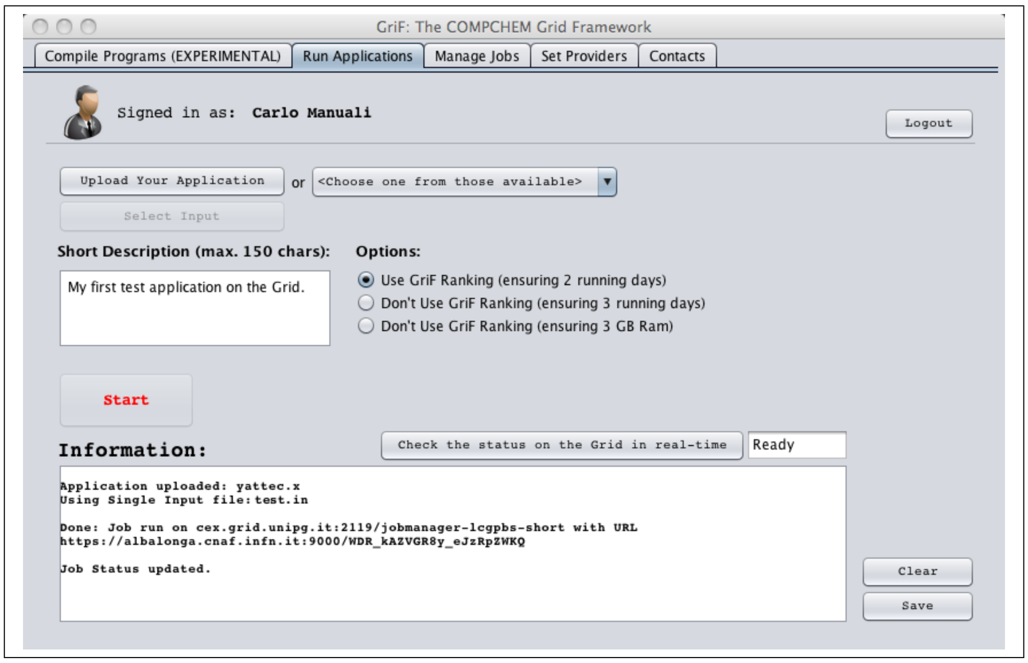
File Name: yattec.x

Files of Type: All Files

Figure 4.14: **GP Grid Service: Loading Applications to GriF.**

In the third panel called "Manage jobs", after reaching the final status associated with a successful completion of a Grid job (*"Scheduled"*, *"Running"* and *"Done (Success)"*), the final status of "Done" is assigned to the considered Grid job by GriF (by the `state.sh` shell script), as resulting from the last part of the line highlighted in blue of Fig. 4.16. Consequently, users can

ISSN: 2279–8773


GriF: The COMPCHEM Grid Framework
Compile Programs (EXPERIMENTAL)
Run Applications
Manage Jobs
Set Providers
Contacts
Logout
Upload Your Application
or
<Choose one from those available>
Select Input
Start
Check the status on the Grid in real-time
Clear
Save

Signed in as: Carlo Manuali

Short Description (max. 150 chars): My first test application on the Grid.

Options:
- [✓ Use GriF Ranking (ensuring 2 running days)]
- Don't Use GriF Ranking (ensuring 3 running days)
- Don't Use GriF Ranking (ensuring 3 GB Ram)

Ready

Information:

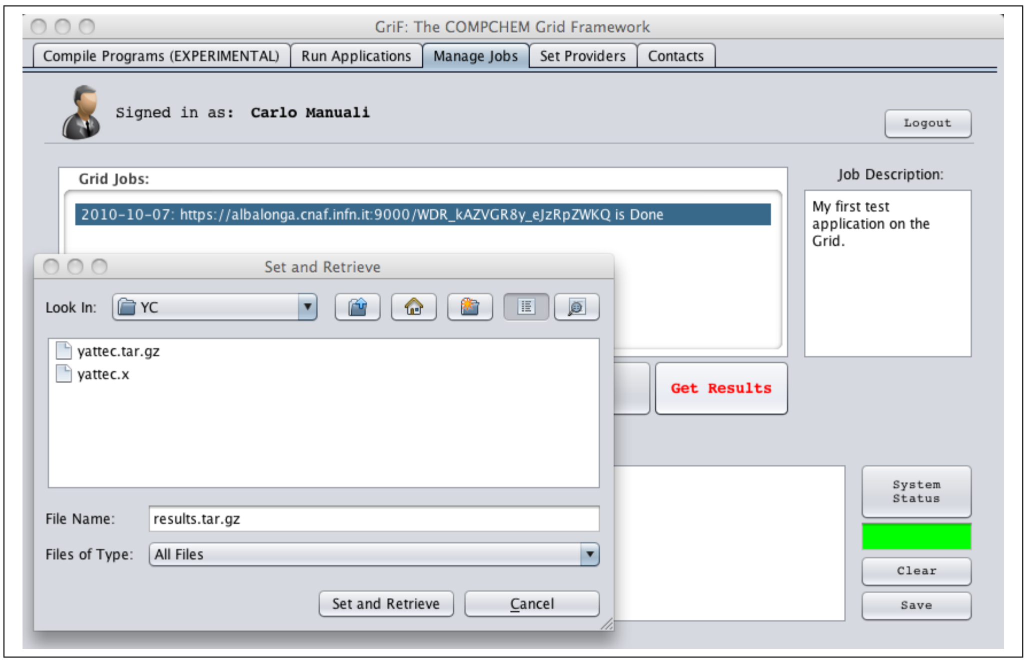
```
Application uploaded: yattec.x
Using Single Input file:test.in

Done: Job run on cex.grid.unipg.it:2119/jobmanager-lcgpbs-short with URL
https://albalonga.cnaf.infn.it:9000/WDR_kAZVGR8y_eJzRpZWKQ

Job Status updated.
```

Figure 4.15: **GP Grid Service: Running Applications on the Grid.**

access the final results for the Grid job by clicking the button ”Get Results” and then choosing a proper results file name (according to a compressed `.tar.gz` format), as shown in Fig. 4.16. Finally, after the end of the downloading process, the retrieved results file will be available on YC for further use.

Within the fourth panel called ”Set Providers” one can search (and then set) different YPs hosting a Grid Service of GriF, as shown in Fig. 4.17. Accordingly, in the case considered, only one YP offering the GP Grid Service (called `gw2-hpc.chm.unipg.it` on port 8080), has been configured and activated.

ISSN: 2279-8773



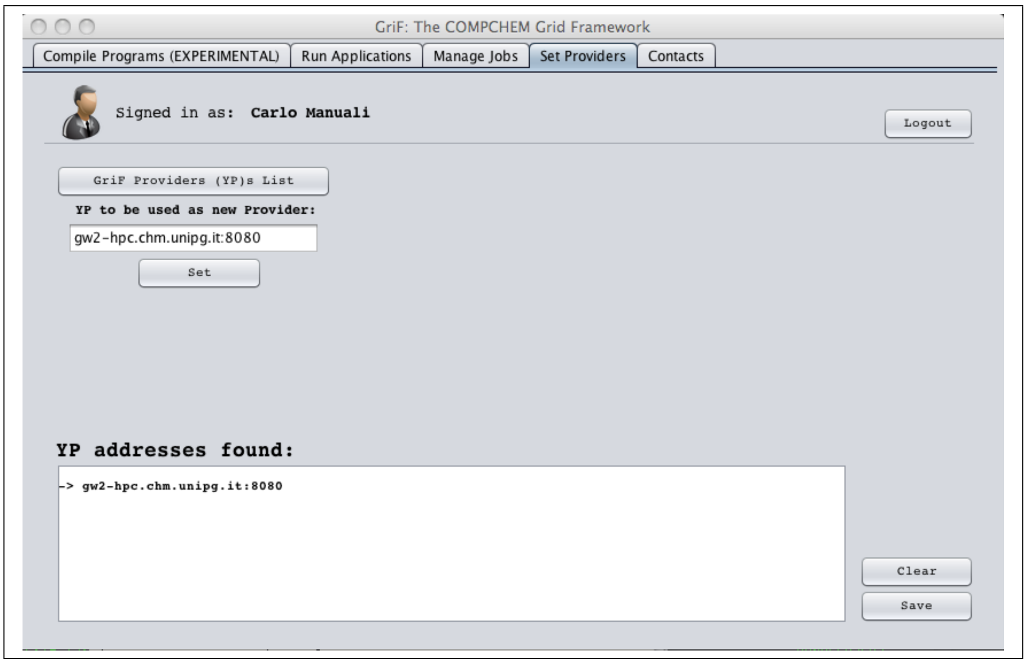
2010-10-07: https://albalonga.cnaf.infn.it:9000/WDR_kAZVGR8y_eJzRpZWKQ is Done

My first test application on the Grid.

yattec.tar.gz
yattec.x

File Name: results.tar.gz

Figure 4.16: **GP Grid Service: Retrieving Results from the Grid.**


GriF: The COMPCHEM Grid Framework
Compile Programs (EXPERIMENTAL)
Run Applications
Manage Jobs
Set Providers
Contacts
Signed in as: Carlo Manuali
Logout
GriF Providers (YP)s List
YP to be used as new Provider:
Set
YP addresses found:
Clear
Save

gw2-hpc.chm.unipg.it:8080

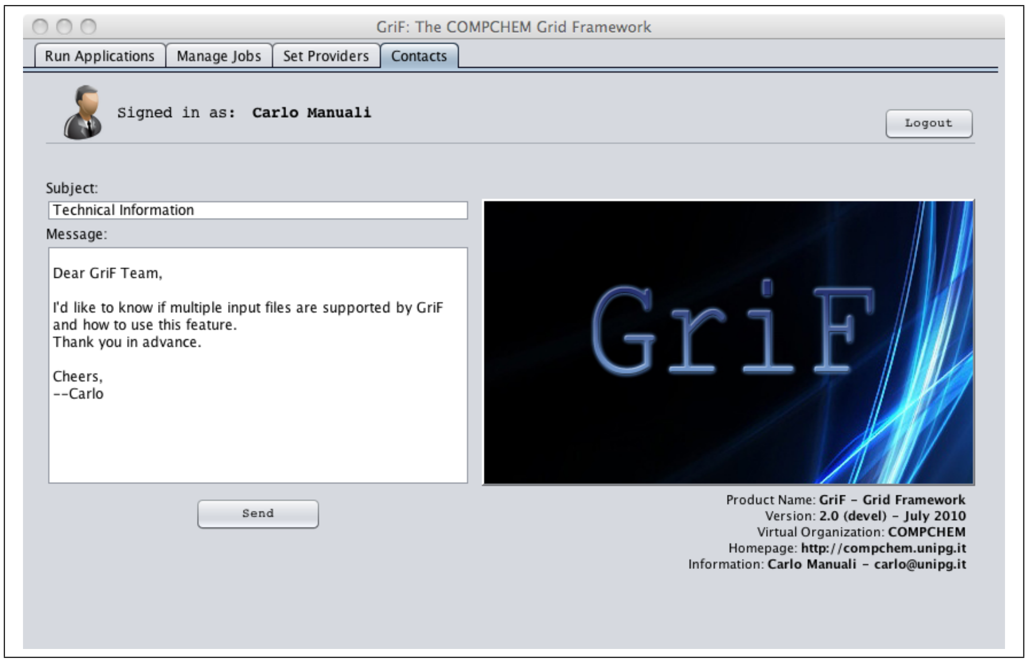
-> gw2-hpc.chm.unipg.it:8080

Figure 4.17: **GriF 2.0: Choosing YPs for Grid Services.**

ISSN: 2279-8773

Virt&l–Comm.3.2012.22

Within the fifth panel called ”Contacts”, one can send messages (and/or feedbacks) to the GriF team, as is shown in Fig. 4.18 where, for example, technical information are requested to the GriF people (and for which the related answer has been already given in the section of this chapter reporting the YC description).



Signed in as: Carlo Manuali

Subject:
Technical Information

Message:

Dear GriF Team,

I'd like to know if multiple input files are supported by GriF and how to use this feature.
Thank you in advance.

Cheers,
--Carlo

Product Name: GriF - Grid Framework
Version: 2.0 (devel) - July 2010
Virtual Organization: COMPCHEM
Homepage: http://compchem.unipg.it
Information: Carlo Manuali - carlo@unipg.it

Figure 4.18: **GriF 2.0: Sending Feedbacks to GriF People.**

An enhanced (alternative) GriF version (`2.0-en`) has also been developed essentially in order to facilitate the retrieving of the results from the Grid especially when using very large files. In fact, within this version of GriF the results for ”Done” jobs have not to be retrieved from the Grid in real-time but they are made available to users through the YPs. This increases sig-

ISSN: 2279-8773

nificantly their retrieving speed (up to 7–8 times, depending on the network bandwidth between YC, YP and the UI). By going into more detail, minor changes had to be made to YC in order to implement version `2.0-en`. With respect to YP, main changes were confined to the fact that the shell script `result.sh` ended up to be unnecessary because the associated Web Service is now able to return the requested results file with no need to bridge the UI and the Grid. Moreover, the off-line shell script `state.sh` was replaced by two other ones in which the former not only monitors the status of Grid jobs (as before) but also invokes in background the latter (when Grid job-related results are available), called `en.sh`, using the command:

```
nohup ./en.sh $uri 2>&1 >/dev/null & 2>&1 >/dev/null
```

in which the `uri` variable contains the location where to fetch the results. Accordingly, `en.sh` will download the desired results file while the former script continues its work asynchronously.

However, at present, although this version introduces significant improvements in terms of performance, was decided not to bring it in operation because of the storage (space and reliability) heavy requirements introduced for YPs. Therefore, the use of the available standard Grid components (like SEs) is still to be preferred as results file repositories.

ISSN: 2279-8773

4.3.8 Relevant ABC Results.

Atom diatom reactive scattering quantum studies have become nowadays routine computational applications when needing either to validate the related PES or to estimate accurately the efficiency of a reactive system on an available PES [106, 107]. The corresponding computational effort, though, when wishing to perform an exact full 3D calculations of the reactive efficiency, even for the simplest atom diatom systems, are usually out of reach due to the scarce availability of sufficiently powerful machines for an adequately long period of time. The key reason for this is the fact that, in order to construct quantum evaluations of key CMB experimental observables (say state specific cross sections) related calculations need to be repeated for increasing values of the total angular momentum quantum number J and increasing dimensions of the expansion basis set. This makes full dimensional exact quantum calculations unviable unless the reduced mass of the atom diatom system is very light (like in hydrogen exchange reactions) and its total energy is small (usually smaller than 1 eV). To make the calculations viable for heavier systems one has to inevitably resort into the concurrent computing offered either by supercomputers or from the Grid. In our case we have considered as a study case the electronically adiabatic three dimensional reactive scattering $N + N_2$ by calculating the energy dependence of the re-

ISSN: 2279-8773

lated scattering matrix (**S**) elements. Our calculations have been performed on the Grid using the already described ABC program (see chapter 3) and the already mentioned PESs: the LEPS given as a FORTRAN routine called `leps.f` and the L3 one given as a FORTRAN routine called `lagrobo3.f`. The most important details of the ABC FORTRAN code used for the calculations are described in [63] in which both the theoretical approach and the hyperspherical formalism adopted are illustrated. After accessing GriF, the already mentioned sequence of steps:

1. The Input file was selected;
2. The ABC executable (related to the PES adopted) was submitted for running on the Grid;
3. The Results were gathered from the Grid to be returned to the user.

was undertaken.

In step 1, the input file (called `nn2.d`) shown in Fig. 4.19 is prepared for multiple energy runs. In the figure the (equal) values of the mass (`mass`) of the three Nitrogen atoms are quoted on the first line while the values of the total angular momentum quantum number J (`jtot`), the triatomic parity eigenvalue `ipar` (1 in our case) and the diatomic parity eigenvalue `jpar` (also 1 in our case) are shown in the second, third and fourth line, respectively. Among the remaining parameters, `emax` indicates in line 5 the

ISSN: 2279-8773

maximum quantity of energy (in eV) allocatable as internal in any channel while `jmax` (line 6) is the maximum rotational quantum number. Further down in the list are `kmax=0`, the truncation parameter related to the helicity projection `k` of the total angular momentum **J** on the quantization axis (necessarily zero for **J**=0 total angular momentum), `rmax=12`, the maximum value of the propagation coordinate (in bohr), and `mtr=150`, the number of the log derivative propagation sectors used to carry out the integration of the scattering equations along the hyperradius. The quantities `enrg`, `dnrg` and `nnrg` given in the subsequent three lines indicate the initial value, the increment and the number of values considered for the total energy E in eV (in our case we start from an initial value of 1.2 that is incremented 10 times in steps of 0.1). The last two parameters `nout` and `jout` state the maximum rotational and vibrational states (lines 13 and 14, respectively) to be provided in the output (only $v' = 0$ and $j' = 0$ in our case with the priming labeling as usual the product states).

Step 2 is devoted to the choosing and executing on the Grid platform of the ABC executable. In particular, in the test runs two Grid jobs have been distributed (at the same time) for the LEPS and L3 PESs with the input file shown in Fig. 4.19.

Finally, in step 3, the results file is assembled and returned. As already mentioned, at the end of the whole process both the GriF and the Grid en-

ISSN: 2279-8773

nn2.d:

```
&input
   mass=14.00674,14.00674,14.00674
   jtot=0
   ipar=1
   jpar=1
   emax=3.5
   jmax=75
   kmax=0
   rmax=12
   mtr=150
   enrg=1.2
   dnrg=0.01
   nnrg=10
   nout=0
   jout=0
&end
```

Figure 4.19: **Input file used by GriF to run ABC on the Grid for both LEPS and L3 PESs to carry out the reactive scattering calculations for the $\mathbf{N + N_2}$ system.**

vironments are automatically cleared (even in the presence of user and/or system errors) and the user is notified by mail.

The key outcome of these ABC test runs (typical of non massive computational campaigns) is given in Table 4.1 where the Real (Re(**S**)), the Imaginary (Im(**S**)) and the square modulus of the **S** matrix are given for the LEPS (the three left hand side columns) and the L3 (the three right hand side columns) PES, respectively.

Table 4.1 clearly shows that detail of the calculations allows to figure out

ISSN: 2279-8773

$N + N_2$	LEPS			L3		
E/eV	Re(S)	Im(S)	$\|S\|^2$	Re(S)	Im(S)	$\|S\|^2$
1.2	-0.0000	0.0000	0.0000	-0.0000	-0.0000	0.0000
1.3	0.0000	0.0000	0.0000	-0.0000	0.0000	0.0000
1.4	-0.0000	-0.0000	0.0000	0.0000	0.0000	0.0000
1.5	-0.0003	0.0001	0.0000	0.0000	0.0001	0.0000
1.6	-0.0019	0.0014	0.0000	-0.0001	0.0005	0.0000
1.7	-0.0025	-0.0007	0.0000	-0.0035	0.0014	0.0000
1.8	-0.0046	0.0016	0.0000	-0.0098	-0.0152	0.0003
1.9	-0.0015	0.0066	0.0000	0.0567	-0.0245	0.0038
2.0	0.0040	0.0008	0.0000	-0.0019	0.1486	0.0221
2.1	-0.0071	0.0021	0.0001	-0.1932	-0.1606	0.0631

Table 4.1: **The S matrix elements calculated for the $N + N_2$ system on the LEPS and L3 PESs when $a = 1$, $v = 0$, $j = 0$, $k = 0$, $a' = 2$, $v' = 0$, $j' = 0$ and $k' = 0$ (with a and a' being arrangement labels of the reactants and the products, respectively).**

that the probability (the square modulus) of the adiabatic ($v = 0, j = 0 \rightarrow v' = 0, j' = 0$) reactive process rises earlier for L3 (that has a lower barrier to reaction) than for the LEPS. However, it shows also that while the difference in height of the barrier is only 0.15 eV the difference in threshold values of the energy is definitely larger. This is a clear advantage of having been able to carry out such a fine energy scanning of the threshold region. The spotted variation of the Real and Imaginary components of **S** in that range has allowed the singling out of the occurrence of strong quantum effects (unexpected for such a heavy system) leading to clearly resonant structures of the probability.

ISSN: 2279-8773

To rationalize the just mentioned results a computationally heavier campaign exploiting the already discussed possibility of distributing the various computational tasks on the Grid was performed (also described in [108]). To this end, the transparency and the user-friendliness (in other words the automation of the Grid implementation performed using GriF) have been decisive to enable the massive calculations needed to evaluate the $J = 0$ N + N_2 reactive probabilities for E values ranging from 1.2 to 3.1 in steps of 0.0125 on both the LEPS and the L3 PESs. These two calculations amount in total to the launching of 32 Parametric jobs in parallel (16 for each PES, respectively) each consisting in a block of 10 energies (except for the last block of each calculation that consists of only 3 energies because the total number of energies to be calculated for each PES is equal to 153) and, as mentioned before, requiring an average execution time of about 329 and 459 minutes for the LEPS and the L3 PES, respectively.

The transformation of the code into a Grid Service has, in fact, reduced both the preparatory work and the training time for the acquisition of the specific programming skills of the COMPCHEM users. As a result, this has made it possible for a starting master student having just a sufficient command of quantum reactive scattering (and not expertise in G-C) to carry out a detailed comparison of the fine structure of the atom diatom reactive probability curves as that shown in Fig. 4.20.

ISSN: 2279-8773

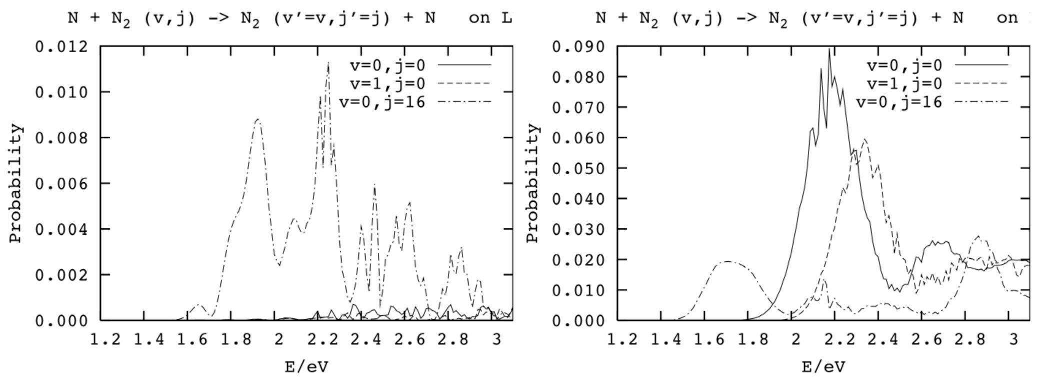


Figure 4.20: **State (v, j) to state (v', j') adiabatic reactive probabilities for the N + N_2 reaction calculated on the LEPS (left hand side panel) and L3 (right hand side panel) PESs and plotted as a function of the total energy.**

In particular, Fig. 4.20 shows the plots of the probabilities of the vibrotationally adiabatic transitions of the diatom ground state ($v = 0$, $j = 0 \rightarrow v' = 0$, $j' = 0$, dotted line), of the first excited vibrational state ($v = 1$, $j = 0 \rightarrow v' = 1$, $j' = 0$, dashed line) and of the (equivalent in terms of internal energy) excited rotational state ($v = 0$, $j = 16 \rightarrow v' = 0$, $j' = 16$, dashed dotted line) computed over the same interval of energy on the already mentioned LEPS (left hand side panel) and L3 (right hand side panel) PESs.

The extremely fine energy grid used for the calculations thanks to massive computing capabilities of the Grid platform (as can be seen from the detail of the plotted probability values) has allowed us to work out a quantitative estimate of threshold effects. As a matter of fact, on both PESs a rotational excitation of the reactants lowers the threshold to reaction with that on the

L3 PES being, as expected, lower than that on the LEPS PES. Moreover, while the rotational ground state provides the largest contribution to the adiabatic reactivity of the L3 PES, this becomes negligible for the LEPS PES when compared to that of the rotationally excited states.

The most interesting result, however, of such a fine energy grid calculation, that has been made possible by the simplicity of the use of the computing Grid through GriF, is the possibility of quantitatively singling out the structure of the reactive probability plots of the N + N_2 reaction that was quite unexpected for such a heavy system.

ISSN: 2279-8773

Chapter 5

New Grid Sensors

5.1 An Overview on gLite Middleware: the Workload Manager Service (WMS).

The purpose of the Workload Manager Service (WMS), the main underlying package of the gLite middleware (3.2 version) on which the EGI Grid is at present based, is to accept requests for job submission and management coming from its clients and take the appropriate actions to satisfy them. The complexity of managing applications and resources on the Grid is hidden by WMS to the users. The user interaction with WMS, is indeed, limited to the description of the characteristics and requirements of the request via a high-level, user-oriented specification language, the JDL and to its submission through the provided interfaces. It is then responsibility of WMS to translate these abstract resource requirements into the provision of a set of actual resources taken from the overall Grid resource pool (provided that the user has an adequate access permission). The WMS provides a set of client tools (that will be referred to as WMS-UI hereinafter) allowing the user to access the main services (job management services) made available by the WMS itself. These client tools encompass a command line interface, a graphical interface and an API, providing both C++ and Java bindings, which allow the requests to be submitted and managed programmatically. The main operations made possible by the WMS-UI are:

- Find the list of resources suitable to run a specific job;
- Submit a job/DAG for execution on a remote CE;
- Check the status of a submitted job/DAG;
- Cancel one or more submitted jobs/DAGs;
- Retrieve the output files of a completed job/DAG (from output sandbox);
- Retrieve and display bookkeeping information related to submitted jobs/DAGs;

ISSN: 2279-8773

- Retrieve and display logging information related to the submitted jobs/DAGs;
- Retrieve checkpoint states of a submitted checkpointable job;
- Start a local listener for an interactive job.

Once submitted from the WMS-UI, the request passes through several other components of the WMS before it completes its execution. In such a process the request goes through many states that can be represented as a state machine. The main WMS components (shown in Fig. 5.1 that summarizes the internal architecture of the WMS through a package diagram described in detail later on) handling a Grid job are:

- WMProxy/Network Server;
- Workload Manager;
- Resource Broker;
- Job Controller (JC);
- CondorC/ DAG Manager (DAGMan);
- Logging and Bookkeeping;
- Log Monitor.

ISSN: 2279-8773

The Network Server (NS) is a generic network daemon that provides support for the job control functionality. It is responsible for accepting incoming requests from the WMS-UI (e.g. job submission and removal), which, if valid, are then passed to the Workload Manager (WM).
WMProxy is a service providing access to WMS functionality through a Web Services-based interface. Besides being the natural replacement of the NS in the passage to the SOA approach for the WMS architecture, it provides additional features such as bulk submission and the support for shared and compressed sandboxes for compound jobs. It can be directly accessed by `http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wmproxy` (the published WSDL URL) or by the provided client tools.
WM is the core component of WMS. Given a valid request, it has to take the appropriate actions to satisfy it. To do so, it may need support from other components, which are specific of the different requests. For a computational job there are two main types of request: submission and cancellation. In particular the meaning of the submission request is to pass the responsibility of the job to the WM. The WM will then pass the job to an appropriate CE for execution, taking into account the requirements and the preferences expressed in the job description. The decision on which resource should be used is the outcome of a matchmaking process between the submission requests and the available resources. The availability of resources for a particular

ISSN: 2279-8773

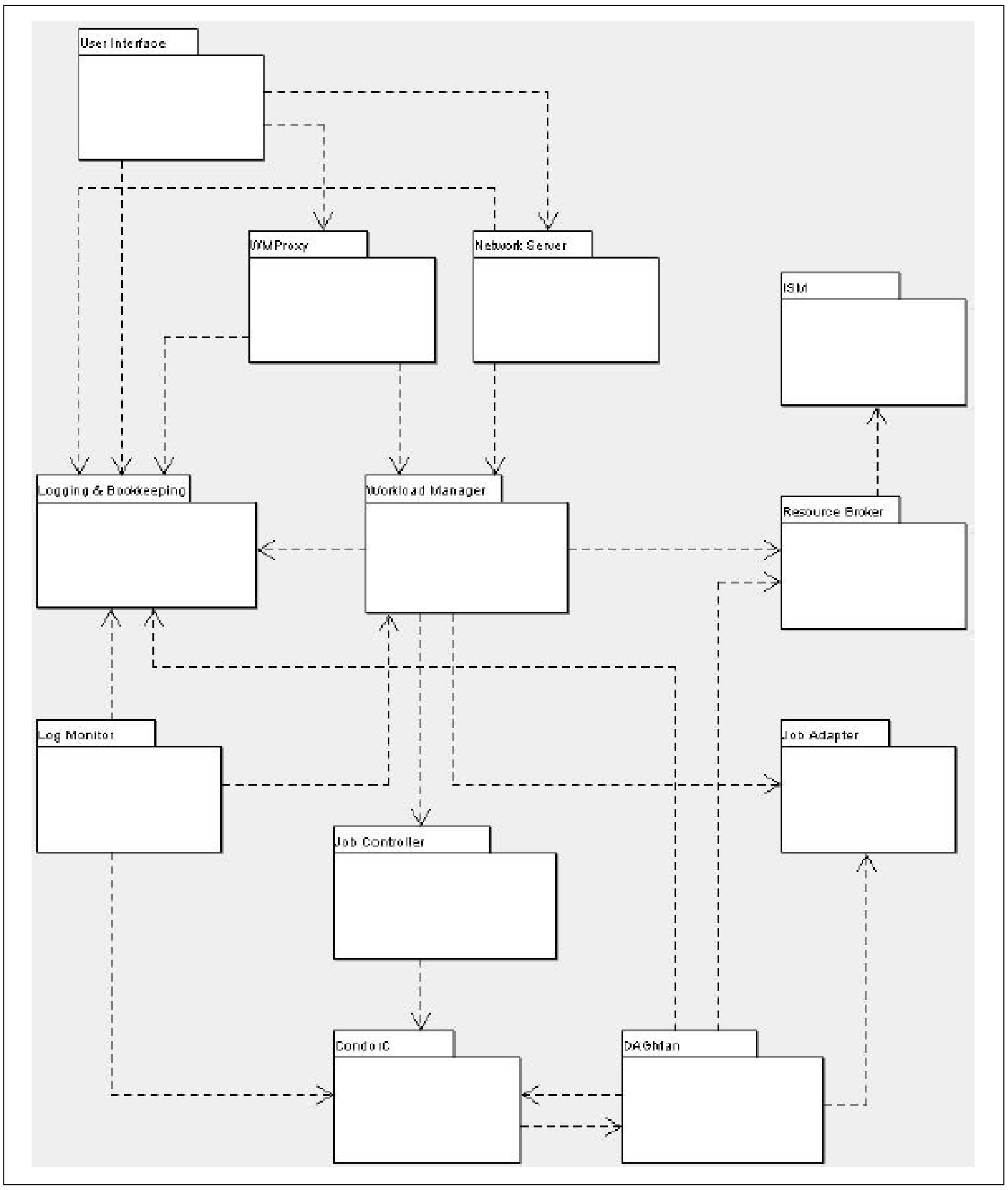


Figure 5.1: **The WMS Architecture.**

ISSN: 2279–8773

Virt&l–Comm.3.2012.22

task depends not only on the state of the resources, but also on the utilization policies that the resource administrators and/or the administrator of the VO/VRC the user belongs to have put in place.

The Resource Broker (RB) or Matchmaker is one of the "helper classes" offering support to the WM in taking the above mentioned decision. It provides a matchmaking service: given a JDL expression (e.g. for a job submission), it finds the resources that best match the request. A WM can adopt different policy in order to schedule a job. At one extreme a job is matched to a resource as soon as possible and, once the decision is taken, the job is passed to the selected resource for execution. At the other extreme the jobs are held by the WM until a resource becomes available (eager scheduling), at which point that resource is matched against the submitted jobs and the job that fits best is passed to the resource for immediate execution (lazy scheduling policy). The mechanism that allows the flexible application of different policies is the decoupling between the collection of information concerning resources and its use.

The Information Super Market (ISM) is the component that implements this mechanism and basically consists of a repository of resource information that is available in read only mode to the matchmaking engine and whose update is the result of either the arrival of notifications or active polling of resources or some arbitrary combination of both. Moreover the ISM can be configured

ISSN: 2279-8773

so that certain notifications can trigger the matchmaking engine. These functionalities besides improving the modularity of the software also support the implementation of lazy scheduling policies.

The other fundamental component of the WM internal design is the Task Queue (TQ). TQ gives the possibility to keep a submission request for a while if no resources matching the job requirements are immediately available. This technique is used, among others, by the AliEn [109] and Condor [110] systems. Non-matching requests will be retried either periodically (in an eager scheduling approach) or as soon as notifications of available resources appear in the ISM (in a lazy scheduling approach). Alternatively such situations could only lead to an immediate abort of the job for lack of a matching resource.

Continuing on the WMS components handling the job during its lifetime, one has to consider the Job Adapter (JA) that is responsible for making the final touches to the JDL expression for a job, before it is passed to CondorC for the actual submission. So, besides preparing the CondorC submission file, this module is also responsible for creating the job wrapper script that creates the appropriate execution environment in the CE worker node (this includes the transfer of the input and of the output sandboxes). CondorC is the module responsible for performing the actual job management operations (e.g. job submission and removal), issued on request of the WM.

ISSN: 2279-8773

DAGMan is a meta-scheduler whose main purpose is to navigate the graph (e.g. the DAG request), determine which nodes are free of dependencies, and follow the execution of the corresponding jobs. A DAGMan instance is spawned by CondorC for each handled DAG. The Log Monitor (LM) is responsible for watching the CondorC log file, intercepting interesting events concerning active jobs, that is events affecting the job state machine (e.g. job done or job canceled as showed in Fig. 5.2), and therefore triggering appropriate actions.

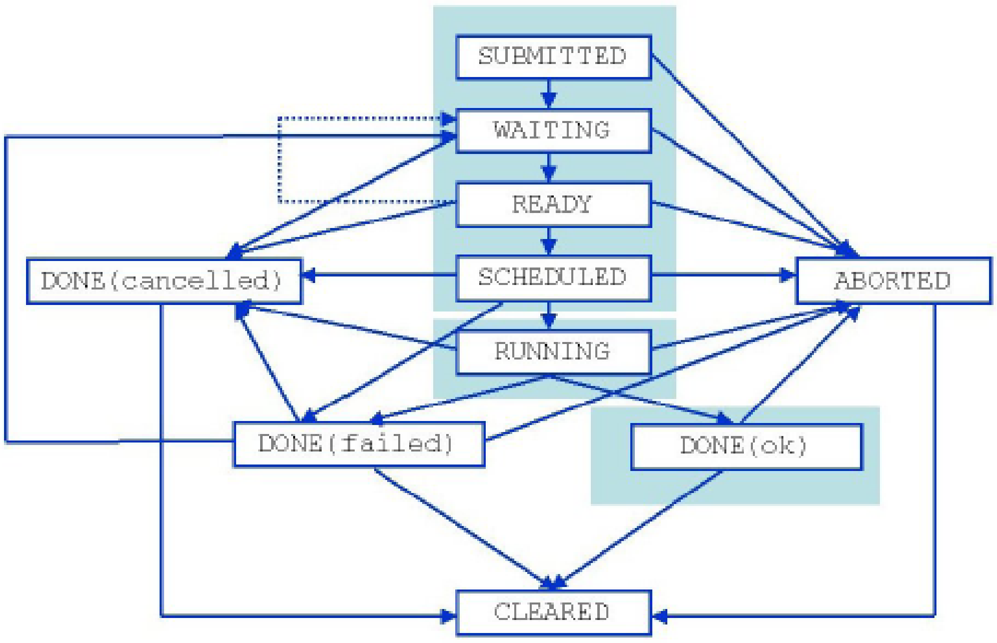


Figure 5.2: **The Job State Machine.**

In the followings a brief description of the meaning of each possible state on

ISSN: 2279-8773

which a job/DAG can enter is provided:

- *Submitted*: job is entered by the user to the UI but not yet transferred to NS for processing;
- *Waiting*: job has been accepted by NS and is waiting for WM processing or is being processed by WM Helper modules (e.g. WM is busy, no appropriate CE (cluster) has been found yet, required dataset is not available, job is waiting for resource allocation);
- *Ready*: job has been processed by WM and its Helper modules (especially, appropriate CE has been found) but not yet transferred to the CE (local batch system queue) via JC and CondorC. This state does not exist for a DAG as it is not subjected to matchmaking (the nodes are) but passed directly to DAGMan;
- *Scheduled*: job is waiting in the queue on the CE. This state also does not exist for a DAG as it is not directly sent to a CE;
- *Running*: job is running. For a DAG this means that DAGMan has started processing it;
- *Done*: job exited or is considered to be in a terminal state by CondorC (e.g. submission to CE has failed in an unrecoverable way);

ISSN: 2279-8773

- *Aborted*: job processing was aborted by WMS (e.g. waiting in the WM queue or CE for too long, over-use of quotas as well as user credentials expiration);
- *Cancelled*: job has been successfully cancelled on user request;
- *Cleared*: output sandbox was transferred to the user or removed due to the timeout.

Moreover, a Proxy Renewal Service (relying on the MyProxy service in order to renew credentials associated to the request) is available to ensure that, for all the lifetime of a job, a valid user proxy exists within the WMS.

The Logging & Bookkeeping (LB) service provides support for the job monitoring functionality: it stores logging and bookkeeping information concerning events generated by the various components of the WMS. Using this information, the LB service keeps a state machine view of each job. The user can learn in which state its jobs are by querying the LB service (using the appropriate command provided by the WMS-UI). Besides querying for the job state actively, the user may also register for receiving notifications on particular job state changes (e.g. when a job terminates). The notifications are delivered using an appropriate infrastructure.

All the WMS components described above interact with the LB for logging information about the jobs they are handling and for querying information

ISSN: 2279-8773

about them when needed. Moreover, during the matchmaking phase RB interacts with the Data Management Catalogs through the StorageIndex interface to resolve the locations of existing file names specified in the JDL Input data list, so that the submitted job would run on worker node, close to the used files. The WMS service also interacts with the Virtual Organization Membership Service (VOMS) [123] indirectly as it reads information about VO, groups and capabilities (Fully Qualified Attribute Names, or FQAN) from the users proxy credentials issued by VOMS to enforce user authentication and authorization. The MyProxy service is contacted by the Proxy Renewal component of the WMS to renew automatically the credentials associated to long lived jobs. Finally, the WMS interacts with the CEs for submitting jobs (this goes through CondorC subsystem) and for receiving synchronous/asynchronous notifications about resource status and characteristics. There could be configurations in which the WMS also interacts with the BDII to collect information about the whole resource pool available for a given VO.

5.2 The official EGI Accounting Portal.

Accounting is a powerful tool for users, VRCs and VOs to obtain information on Grid resources usage. Within the ended EGEE project, the Supercomput-

ISSN: 2279-8773

ing Center of Galicia (CESGA) [111], that is the center for high-performance computing, communications and advanced services used by the Scientific Community of Galicia, the University System of Galicia and the Spanish National Research Council (CSIC) providing horizontal services in HPC, HTC and Networking Services to scientific, technological and industrial users, has been responsible for European Grid Support, Operation and Management, in particular for the management of the official Accounting Portal [112]. In this respect, we have concentrated our attention on the kind of data used by it which are obtained from three external and distinct sources:

- *Grid Operations Centre (GOC) Accounting*: proper Accounting data can be directly accessed from the main GOC Accounting host (at present it is `GOC-accounting.Grid-support.ac.uk`);
- *GOCDB*: all the necessary information concerning the registered Grid sites (e.g. websites, nodes and administrators) can be obtained from this database;
- *Core Infrastructure Center (CIC) Operations Portal*: all the information on the registered VOs (e.g. name, discipline, managers and members) and the data challenge of each of them can be retrieved from the CIC Operations Portal.

ISSN: 2279-8773

GOC is part of the Regional Operations Center (ROC) that has a key role in the process of building and operating the Grid infrastructure. Accordingly, ROC has the expertise to assist the growing number of resource centers in joining the Grid, through the deployment of middleware releases and the development of procedures and capabilities to operate those resources. As a matter of fact, GOC is responsible for coordinating the overall operation of the Grid acting as a central point of operational information such as configuration information and contact details. Moreover, GOC has responsibility for monitoring the operation of the Grid Infrastructure as a whole, devising and managing mechanisms and procedures which encourage optimal operation of the Grid, and working as well with local support groups to assist them in providing the best possible service while their equipment is connected to the Grid. In this respect, GOCDB (hosted at the Rutherford Appleton Laboratory (RAL) [113] in the UK) stores and gives access to information about regions, countries, resources (sites, nodes and services), users, roles and contacts.

CIC Operations Portal has been a management and operations tool that has acted as an entry point for all the related Grid actors for their operational needs. It has been also devoted to the management of the available information about VOs recording day-to-day operations on Grid resources and services. Moreover, each Grid actor has been able to enter or have access to

ISSN: 2279-8773

information from an operational point of view according to its role.

Accordingly, the EGI Accounting Portal stores the data in three separate databases: the first one (named `acctUsers`) contains users information (aggregated Accounting data summarized by local shell scripts), the second one (named `gocdb3`) contains the information obtained from GOCDB mentioned above and the third one (named `accounting`) contains raw Accounting data, VOs information and their data challenge. In particular, the `UserCPU2` table of the `acctUsers` database which contains all the aggregated data per user belonging to a specific VO, has been analyzed. This table is derived from the `Group_VO`, `UserCPU` and `DNAssociations` tables. In this respect, the `Group_VO` table (`accounting` database), locally derived, associates each published VO with its real name. The `UserCPU` table (`acctUsers` database), derived from GOC Accounting, contains all the Accounting information grouped by site, user, VO, month and year. The `DNAssociations` table (`acctUsers` database), locally derived, stores associations between the user Distinguished Name (DN)s and the one currently in use. Moreover, several scripts are used in order to maintain and updated all the mentioned databases. The main fields of the `UserCPU2` table considered by us are the following:

- `PrimaryGroup` (according to the new Grid sensors proposed, it will map

ISSN: 2279-8773

the name of the application run);

- `UserDN` (according to the new Grid sensors proposed, it will map the real name of the VRC user when a RC strategy is adopted);
- `Njobs` (the number of jobs run per user);
- `SumCPU` (the CPU time consumed per user);
- `NormSumCPU` (the normalized CPU time consumed per user);
- `SumWCT` (the WALL time consumed per user);
- `NormSumWCT` (the normalized WALL time consumed per user);
- `ExecutingSite` (the corresponding Grid site name registered on the Grid).

Furthermore, the EGI Accounting Portal also contains, for example, all the Accounting data coming from the Distributed Grid Accounting System (or DGAS, developed by INFN [114] and used in the Italian Grid sites) [115] since its database is populated (via MySQL import/export procedures) by the global Relational Grid Monitoring Architecture (R-GMA) database system. As a matter of fact, R-GMA (consisting of 'Producers' which publish information and of 'Consumers' which subscribe [116]) provides a service for information, monitoring and logging in a distributed computing environment

ISSN: 2279-8773

making all the information appear like one large relational database that may be queried to find the information required.

To this end, the log files of batch systems and those of the CEs (usually each Grid site is composed by one or more CE and batch system, as well as by SEs and UI) are managed by the Accounting Processor for Event Logs (APEL) [117] matching the DN (registered within a CE) with the local user running on the different batch systems (e.g. the Sun Grid Engine (SGE) [118] and Torque/Maui [119]). Accordingly, the APEL system (composed by two Java scripts known as 'Parser' and 'Publisher') is responsible for initiating the process of uploading and then publishing every new data (typical relevant information for us will be the job exit status and the real/virtual memory consumed by a job) to R-GMA. Therefore, once the R-GMA database has been populated by APEL, it can properly export its data (under the form of summarized table depending by the view to be accomplished) to the EGI Accounting Portal one.

5.3 New Parameters.

As mentioned before, the current Accounting system is oriented to gather job-level information from batch systems merging it with VO and user-level information. Accordingly, Grid users can access the EGI Accounting Portal

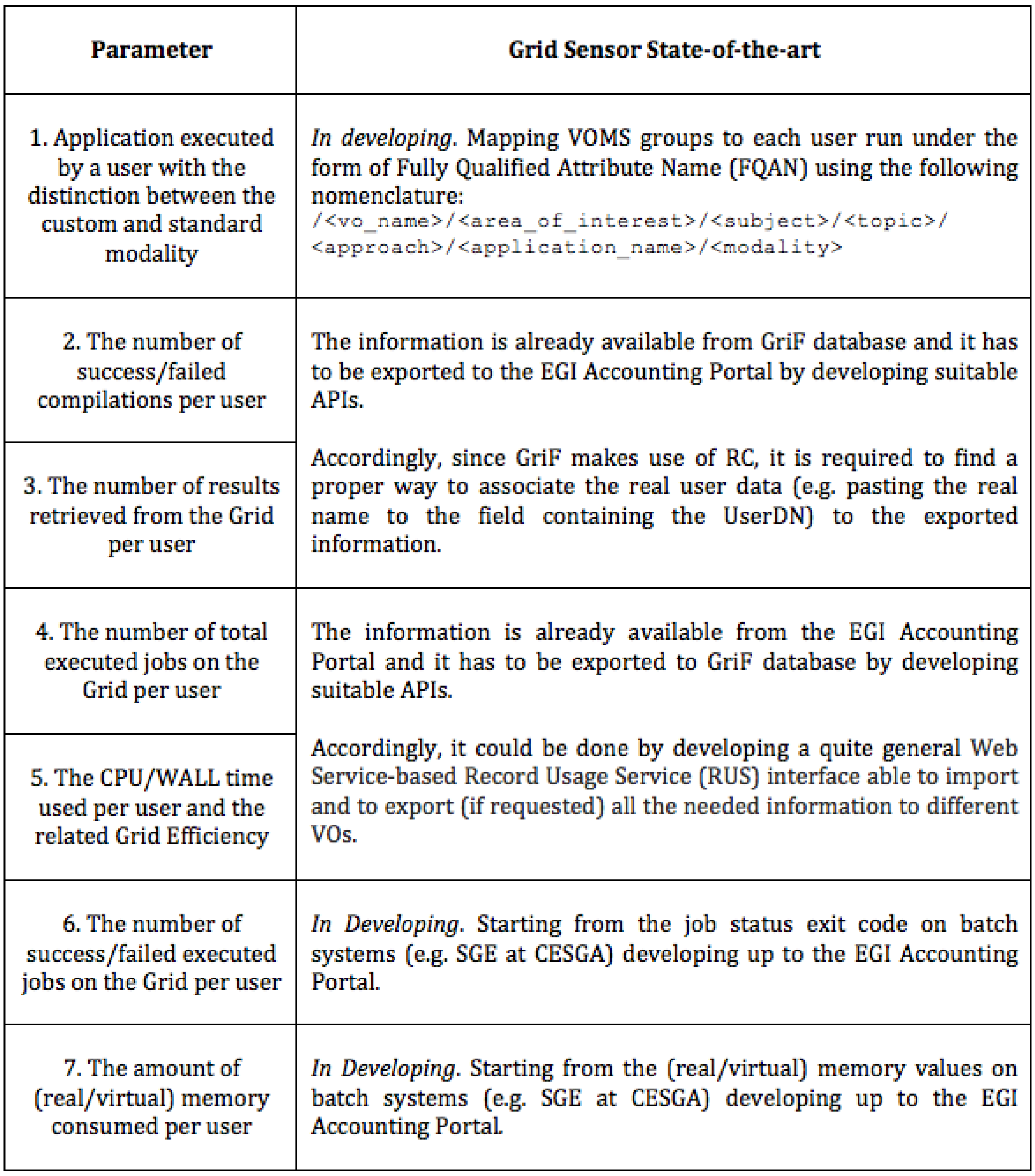
ISSN: 2279–8773

to display information on the executed jobs. However, this kind of information is not detailed enough (for example, at present, VOs have no information whatsoever on the software utilized and on the results gathered by the users). Therefore, they prompt further improvements to enhance the present Accounting system to foster collaborative endeavors.

Our attention has focused on the Application Domain Accounting in order to implement QoS and QoU evaluations. To this end, COMPCHEM and CESGA have suggested to introduce new Grid sensors able to gather additional information (e.g. on the most executed programs, on the amount of retrieved results and successful jobs, and on the used resources) [120] from the Grid. Accordingly, in order to satisfy this new application-level requirements, gLite middleware may need to be improved for example by adopting new fields and values to be agreed and validated by Open Grid Forum (OGF) [121] to become part of the Usage Record (UR) [122] standard (and then to be introduced on the database schema). In this respect, the development of application-level accounting tools will require the joint effort of different development teams along the next years. According to this research line, COMPCHEM and CESGA have also started to design a preliminary draft which contains the specific parameters about users and applications to be considered. Accordingly, Table 5.1 summarizes the state-of-the-art in implementing these parameters by new Grid Sensors.

ISSN: 2279-8773

Virt&l–Comm.3.2012.22



Parameter	Grid Sensor State-of-the-art
1. Application executed by a user with the distinction between the custom and standard modality	*In developing*. Mapping VOMS groups to each user run under the form of Fully Qualified Attribute Name (FQAN) using the following nomenclature: `/<vo_name>/<area_of_interest>/<subject>/<topic>/<approach>/<application_name>/<modality>`
2. The number of success/failed compilations per user	The information is already available from GriF database and it has to be exported to the EGI Accounting Portal by developing suitable APIs. Accordingly, since GriF makes use of RC, it is required to find a proper way to associate the real user data (e.g. pasting the real name to the field containing the UserDN) to the exported information.
3. The number of results retrieved from the Grid per user	
4. The number of total executed jobs on the Grid per user	The information is already available from the EGI Accounting Portal and it has to be exported to GriF database by developing suitable APIs. Accordingly, it could be done by developing a quite general Web Service-based Record Usage Service (RUS) interface able to import and to export (if requested) all the needed information to different VOs.
5. The CPU/WALL time used per user and the related Grid Efficiency	
6. The number of success/failed executed jobs on the Grid per user	*In Developing*. Starting from the job status exit code on batch systems (e.g. SGE at CESGA) developing up to the EGI Accounting Portal.
7. The amount of (real/virtual) memory consumed per user	*In Developing*. Starting from the (real/virtual) memory values on batch systems (e.g. SGE at CESGA) developing up to the EGI Accounting Portal.

Table 5.1: **The new evaluation parameters agreed by COMPCHEM VO and CESGA.**

ISSN: 2279–8773

Due to the simplicity in understanding how to implement the parameters 2–5 of the Table (broadly speaking it will be sufficient to design and to develop an appropriate set of standard APIs to upload/download the corresponding MySQL information), we have taken into account, in particular, how to pave the way to the development of the other parameters (1, 6 and 7) for which, respectively, new configurations have to be introduced on the gLite middleware components (e.g. on the WMS), the implementation of new fields have to be officially requested to the APEL team (e.g. for using the values of job exit status codes) as well some open (at present) issues have to be resolved (e.g. memory information on the SGE are not correctly selected and recorded by APEL even if related official fields already exist).

With respect to parameter 1, the mapping of the relationship (for each user run) between Grid jobs and real applications (in both STANDARD and CUSTOM, mentioned in the previous chapter, modalities) making use of the `PrimaryGroup` field (of the `UserCPU2` table of the `acctUsers` database) handled by VOMS, will be enabled. To this end, a hierarchical tree (also reflected by the information recorded on some tables of the GriF database, see Appendix E) mapping applications and their associated information (such as the VO, the area, subject and topic of interest as well as the run modality) can be adopted, as for example:

ISSN: 2279-8773

```
/[compchem|cesga]/molecular_science/chemistry/quantum_chemistry
/time_independent/[abc|rwavepr]/[standard|custom]
```

In Fig. 5.3, the example of the two applications abc and rwavepr (highlighted by an oval) resulting from some user runs (5 runs for abc, 2 runs for rwavepr) after using a simplified version of the strategy mentioned above for the CESGA VO, is shown. As a matter of fact, after creating the VOMS Proxy specifying the related VO and the application to be used in the --voms option[1] (and also requested to be replicated in the following order option) as follows:

```
# voms-proxy-init --voms cesga:/cesga/abc/ --order /cesga/abc/
```

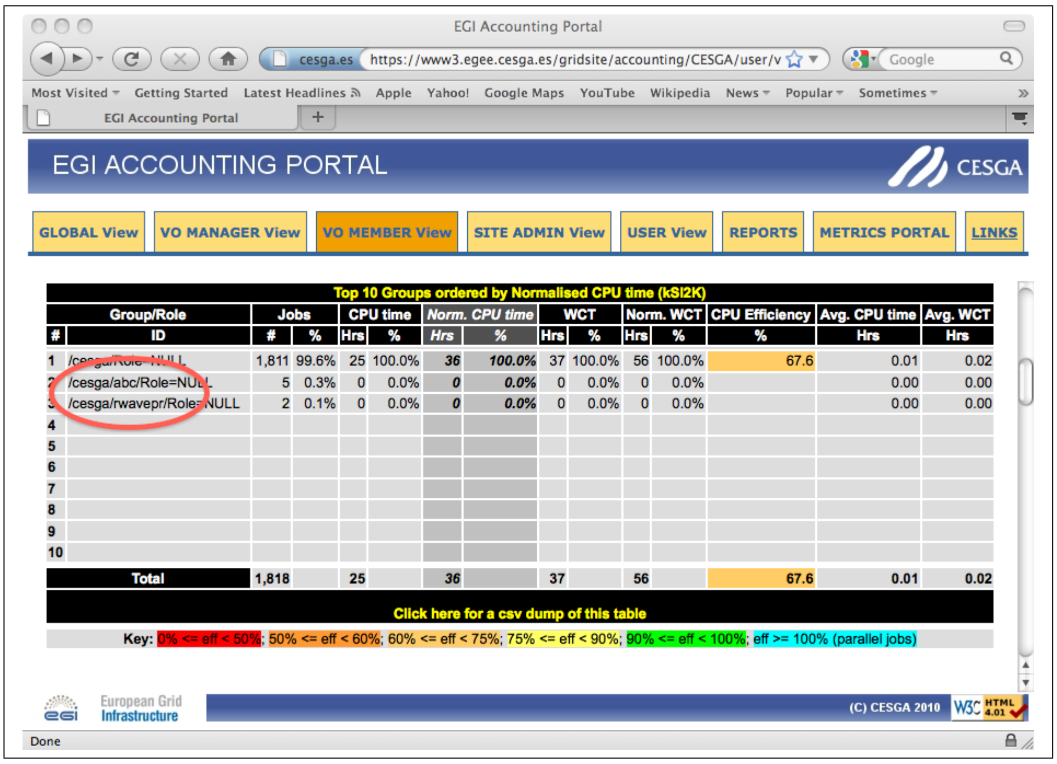
the related information (in this example no further information are associated with abc, the name of the application used) will be propagated on the Grid firstly in its running components (starting from the VOMS up to the selected CE), then to the Accounting ones (gLite MONitoring system collector server (MON), R-GMA, Log Subsystem and databases) and finally

[1]In addition to the VO name (cesga in the example), VOMS allows to specify additional information (separating them by a colon) that will be recorded in the PrimaryGroup field mentioned above.

ISSN: 2279-8773

Virt&l–Comm.3.2012.22

published to the production Website.



EGI ACCOUNTING PORTAL

Top 10 Groups ordered by Normalised CPU time (kSI2K)													
	Group/Role	Jobs		CPU time		Norm. CPU time		WCT		Norm. WCT		CPU Efficiency	Avg. CPU time	Avg. WCT
#	ID	#	%	Hrs	%	Hrs	%	Hrs	%	Hrs	%	%	Hrs	Hrs
1	/cesga/Role=NULL	1,811	99.6%	25	100.0%	36	100.0%	37	100.0%	56	100.0%	67.6	0.01	0.02
2	/cesga/abc/Role=NULL	5	0.3%	0	0.0%	0	0.0%	0	0.0%	0	0.0%		0.00	0.00
3	/cesga/rwavepr/Role=NULL	2	0.1%	0	0.0%	0	0.0%	0	0.0%	0	0.0%		0.00	0.00
4														
5														
6														
7														
8														
9														
10														
Total		1,818		25		36		37		56		67.6	0.01	0.02

Click here for a csv dump of this table

Key: 0% <= eff < 50%; 50% <= eff < 60%; 60% <= eff < 75%; 75% <= eff < 90%; 90% <= eff < 100%; eff >= 100% (parallel jobs)

(C) CESGA 2010

Figure 5.3: **A result of the new strategy in mapping jobs and applications on the Grid.**

With respect to parameter 6 (also useful to understand when a job run fails and for what 'Grid' reason), the needed information (jobs status exit codes) is already available on many batch systems (e.g. SGE). To this end, the official EGI databases Accounting schema (fully adherent to the UR Format Recommendation [124], that is the OGF standard providing information to the Grid community in the area of UR and Accounting) has to be modified introducing proper fields as well. As a matter of fact, the UR Format Recom-

ISSN: 2279-8773

mendation already expects a suitable field of this type (naming it 'Status', see section 3.9). Accordingly, a formal request in order to add the mentioned field (with proper values[2], see the simplified example of Table 5.2) to the official body (the RAL mentioned above), can be accomplished. Moreover, also the APEL system must be adapted in order to deal correctly with this new type of information.

With respect to parameter 7, the databases Accounting schema is already enabled to accept (even if at moment the corresponding fields are not properly filled) both the values related to the real and virtual memory consumption. Accordingly, they have to be properly retrieved from batch systems (when they are present, as in the case of SGE) and then made available by the APEL Parser.

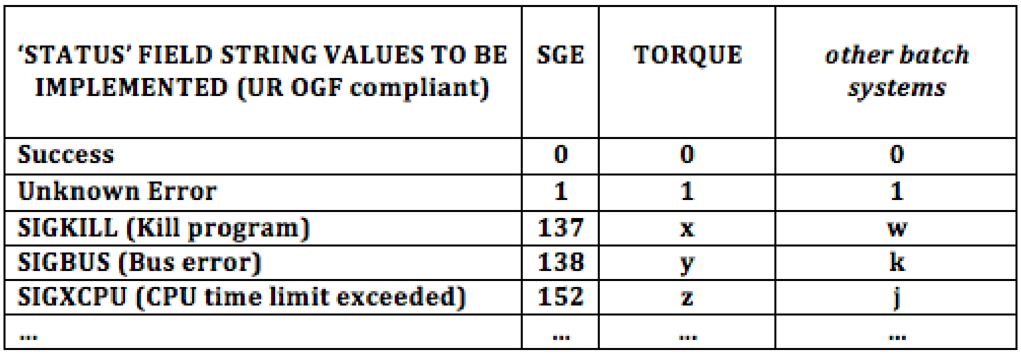
[2]On SGE (the batch system considered in our collaboration with CESGA), the value of a Grid job exit code higher than 100 means that no errors are caused by the SGE itself (otherwise a specific indication is provided). Consequently, the actual operating system error occurred (corresponding to a Unix Signal [125]) can be determined by the following formula:

$$Error = ExitCode - 128 \tag{5.1}$$

where *ExitCode* is the Grid job exit code returned by SGE.

ISSN: 2279-8773

Virt&l-Comm.3.2012.22



'STATUS' FIELD STRING VALUES TO BE IMPLEMENTED (UR OGF compliant)	SGE	TORQUE	*other batch systems*
Success	0	0	0
Unknown Error	1	1	1
SIGKILL (Kill program)	137	x	w
SIGBUS (Bus error)	138	y	k
SIGXCPU (CPU time limit exceeded)	152	z	j
...	...	...	...

Table 5.2: **An example of some string values for the new proposed Status Field (OGF compliant).**

5.4 Planned Work.

Many activities have been already planned for being carried out to test the new evaluation parameters described above and the Grid sensors to be correspondingly implemented.

As to parameter 1 of Table 5.1, a new Accounting Portal view (oriented to the applications) has been planned to be developed by CESGA in order to offer the possibility of accessing to the Grid applications usage data paving the way to a fully Application Domain Accounting system. At the same time, in this case help is needed to drive the VO/VRC users behavior when running Grid jobs (e.g. when they have to initialize their VOMS Proxy using the '--voms' option as described above). To this end, due to its user-friendly nature, the promotion of the use of GriF (also because no support can be given in the UI unless a dedicated wrapper is developed to assist VO/VRC

ISSN: 2279-8773

users in the management of this function) is of strategic importance. Accordingly, direct support to the use of User Certificates (UC)s has to be provided by GriF. However, another option is based on the fact that the information related to parameter 1 are, in any case, already available within the GriF database and then they could be properly exported to the official Accounting Portal (as it is expected for parameters 2 and 3).

As mentioned above, for parameters 2 and 3 of Table 5.1, suitable APIs were planned for development by the COMPCHEM VO in order to export the related information to the official Accounting Portal. Moreover, since GriF is based, at present, on the use of RC (please note that this issue is present in all the applications making use of them, as for example in the case of Pilot Jobs of the Distributed Infrastructure with Remote Agent Control (DIRAC) project [126] owned by the LHCb VO [127]), the development of a real-time mechanism mapping for each Grid job run the real VO/VRC user information (stored and already available into the GriF database) to the EGI Accounting System that is, of course, UserDN-based (in other words, for each Grid job run based on RC in GriF a corresponding real VO/VRC user is requested by the Grid), is required. To this end, a request to the JDL people (and therefore also to the WMS people in order to implement the needed matching on the job wrapper) to add a new attribute (that we call "Real User" here), can be made. In this respect, after a RC-based Grid job run the WMS can

ISSN: 2279-8773

append the ”Real Name” of the VO/VRC user (that is the value contained in the field ”Real User” mentioned above) to the current Robot DN separating them, for example, by a semicolon. Another option is to develop a specific utility (to be installed on each CE) devoted to perform a similar work manipulating the CEs log file (e.g. `/opt/edg/var/gatekeeper/<last_log_file>`) properly. A last option (less elegant) is to develop an off-line procedure that, at regular interval, updates the official Accounting database with the proper VO/VRC user data available in the GriF one for all the Grid jobs. In addition, all the three options mentioned above have been already tested on the EGI Grid platform.

As to parameters 4 and 5 of Table 5.1, a quite general Web Service-based set of RUS APIs in order to import and export the related information to different VOs and/or VRCs have planned to be developed by CESGA. Moreover, it is also worth noticing here that a new variable derived from parameter 5 (taken into account in the next chapter) called Grid Efficiency (GE) can be introduced. GE can be defined in a quite general way as follows:

$$GE = \frac{\sum ct}{\sum wt} \tag{5.2}$$

where ct and wt are the elapsed cpu and wall time, respectively. Accordingly, GE can be used in evaluating VO/VRC users (in this case ct and wt will be related to their Grid jobs, as will be discussed in the next chapter),

ISSN: 2279-8773

CE queues (in this case *ct* and *wt* will be related to the Grid jobs run on each of them) as well as to the Grid middleware itself (in this case all the jobs run on the Grid will be considered).

As to parameter 6 of Table 5.1, the support of a proposal to the RAL for modifying the EGI databases Accounting schema in order to include, as mentioned above, the new field called 'Status' also contemplated by the already mentioned UR Format Recommendation, have been planned by both COMPCHEM VO and CESGA. Moreover, in order to accomplish this action, it will be necessary to define a complete list of proper 'Status' string values integrating that of Table 5.2.

Finally, for parameter 7 of Table 5.1, an official request to the APEL team for improving their Parser in order to work correctly with (real/virtual) memory values coming from SGE Accounting, will be issued by CESGA. Accordingly, more extensive tests for those Grid Sites and CEs making use of different batch systems as for example Torque/Maui, LSF [128] and Condor, will be performed by the COMPCHEM VO.

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

Chapter 6

GCreS: A Grid Credit System for Virtual Research Communities Sustainability.

6.1 Quality Evaluations.

To reward the engagement of the various groups in the VRC collaborative endeavour, we decided to develop a Grid Economy Model based on Credits offered in return for the efforts spent on behalf of the organization. Fundamental to this approach is the possibility of formulating a fair evaluation and recognition for the work done. This implies the development of new structured methods to operate on the Grid by evaluating both the QoS achieved

ISSN: 2279-8773

in a Grid Services-based producing modality and the QoU associated with the VRC users running on the Grid.

As described in chapters 4 and 5, an important advantage of using GriF (and therefore its database) and the new Grid sensors as architectural pillars in the modeling of GCreS is their suitability to support the introduction of QoS and QoU evaluations. GriF also offers to Grid developers new interesting perspectives like those associated with Selection [129] (rather than pure Discovery) of Grid Services and, as already mentioned, the evaluation of the activities carried out on the Grid. More than that, GriF is open to an user-side usage allowing so far the management of a domain-specific operation logic. This has a clear value for the development of new Workflow Design and Service Orchestration advanced features and the establishing of collaborative operational modalities in which VRC users and providers collaborate and their activities are rewarded in terms of *Credits*.

Basically, in GCreS users are rewarded for the work done on behalf of an organization by being assigned a certain amount of Credits to be redeemed via a preferential utilization of the resources (including the financial ones) of a VO (or a VRC). Such development, in addition to leveraging on collaboration, stimulates also a certain extent of competition among the members to produce innovative Grid Services and improve the existing ones as well. GCreS is also a means to stimulate the COMPCHEM VO members to further

ISSN: 2279-8773

step up their membership and to contribute to the infrastructure development (Hardware Provider) by conferring to the VO some of their computing resources (Passive) and taking care of their deployment (Active) on the Grid (see item 3 of Table 3.1). At later stages of the VO evolution a higher level of membership (Stakeholder) is also foreseen (see item 4 of Table 3.1) for members strongly committed to take care of its global management. After producing QoS and QoU evaluations, new higher level ways of managing Grid Services can be adopted:

1. by *VRC users*, which will be able to ask for Grid Services by specifying as keywords high-performance capabilities rather than memory size, cpu/wall time and storage capacity;

2. by *GriF*, which will be able to automatically select the most appropriate low-level capabilities related to the current Grid job (in other words, when a Grid job has to be run, GriF can make use of different system requirements in terms of memory size, cpu/wall time and storage capacity according to the class level of the related VRC user owning the Grid job).

Moreover, as discussed in more detail later on in this chapter, GCreS is designed to transform the obtained QoS and QoU values respectively into *Costs* and *Credits* (both already supported by GriF by making use, respectively,

ISSN: 2279-8773

of dedicated fields of the `services` and `vousers` table of its database, see Appendix E). In particular:

1. *Costs*, to be paid by VRC users for the use of Grid Services on the basis of their QoS;

2. *Credits*, to be awarded to the VRC users on the basis of their QoU.

In case 1 above, to assign *Costs* to Grid Services, the concept of Service Discovery (in which when VRC users search for Grid Services they receive back an unranked list of matching) is better replaced by that of Service Selection (in which the ranking is provided by QoS). In case 2, to award *Credits* to VRC users, the new concept of QoU is applied.

6.2 Quality of Services (QoS).

QoS has become a significant factor in ranking the success of Grid Service Providers and it plays a crucial role in Grid by estimating the usability and the utility of a Grid Service both of which influence its popularity. Implementing QoS evaluation on the Grid is a real challenge because of its implementing dynamics and unpredictable nature. Applications with very different characteristics and requirements compete for heterogeneous Grid resources. At the same time, with standards like SOAP, UDDI and WSDL

ISSN: 2279-8773

being adopted by all major Web Service players, a whole range of Web Services are being currently deployed for various purposes. Moreover, as most of the Web Services are increasingly required to comply with standards, QoS will become a necessary label and an important differentiation parameter for them. QoS relies on a whole range of criteria matching the needs of Grid users with those of the Service Providers in a competition for available Grid resources. For QoS, we refer to non-functional properties of Web Services such as for example Performance, Reliability, Availability and Security [130]. As a matter of fact when defining QoS for a VRC we specifically concentrated on:

- *Availability* (S_{ava}): whether the Web Service is present or ready for immediate use. A useful parameter associated with Availability is the Time-To-Repair (TTR). TTR represents the time Web Service Providers takes to repair a Web Service that has failed. If there are no errors TTR can be considered equal to 0;

- *Accessibility* (S_{acc}): the capability of satisfying a Web Service request. There could be situations in which a Web Service is available but unaccessible. A low value of the sum of TTRs associated with each error divided the total number of successful operations represents a good estimate of accessibility. High-accessibility of Web Services can be achieved

ISSN: 2279-8773

by building highly scalable systems. Scalability refers to the ability to consistently serve the requests despite variations in their volume;

- *Integrity* (S_{int}): how much the Web Service preserves the correctness of the interaction with respect to the source. Proper execution of Web Service transactions provides the correctness of interaction. A transaction refers to a sequence of activities to be treated as a single unit of work. When a transaction does not complete, a rollback procedure is required to cancel all the partial operations already performed;

- *Performance* (S_{per}): a combination of throughput and latency. Throughput represents the number of Web Service requests served at a given time period. Latency is the round-trip time between sending a request and receiving the response;

- *Reliability* (S_{rel}): the extent of preservation of the Web Service and its Quality. The complement to the number of failures can suitably represent a measure of the reliability of a Web Service;

- *Regulatory* (S_{reg}): the scalability of the Web Service with regard to rules and laws, its compliance with Official Standards (OS)s (e.g. SOAP, UDDI and WSDL), established SLAs and documentation. Strict adherence to OSs and SLAs by Web Service Providers is of fundamental

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

importance for a proper use of Web Services by VRC users;

- *Security* (S_{sec}): the extent of confidentiality and non-repudiation by authenticating the parties involved, encrypting messages, managing access control and authorization.

As already mentioned in chapter 4, we consider as a Grid Service a set of collaborative Web Services running on the Grid by sharing a common distributed goal. Therefore, we shall apply to them all the QoS properties mentioned above. Yet, in addition we shall also apply the *Innovation* parameter (S_{inn}). Innovation does meet, in fact, one of the central goal of the COMPCHEM VO and is aimed at specifically rewarding *Active Software Providers* (see item 2 of Table 3.1). S_{inn} can be defined as a function of the following seven variables:

- *Age (A)*: number of days (N_d) elapsed starting from the publishing date of the Grid Service considered;
- *Consolidation (C)*: a number (N_c) indicating how many previous consolidated Grid Services (starting from the input of the former up to the output of the latter) have been wrapped into a new (workflow-enabled) one;
- *Diffusion (D)*: the ratio between the number of times (N_t) that the

ISSN: 2279-8773

Grid Service has been used and the number of VRC users (N_u) in a considered time interval (N_t / N_u). D implicitly comprises also the friendliness of use;

- *Efficiency (E)*: the ratio between the number of results produced (N_p) by the Grid Service and the number of VRC users in a considered time interval (N_p / N_u);

- *Production (P)*: the ratio between a value (ranging between 0 and 1) indicating the level of new functions (e.g. with respect also to already available Grid Services recognized as standards by the scientific community of the specific domain of application considered) offered by the Grid Service (N_l, that is based on both the number and kind of them) and their related developing costs in terms of time and money also expressed by a value (N_v) ranging between 0 and 1;

- *"Social" aspect (S)*: the weighed sum of three values N_{s_1}, N_{s_2} and N_{s_3} (each of them ranging between 0 and 1) representing, respectively, the level of ethics (e.g. the promotion of universal human values like peace), of fairness (e.g. the promotion of universal availability like open source software) and of social impact (e.g. the promotion of a universal welfare) introduced by the Grid Service;

ISSN: 2279-8773

Virt&l–Comm.3.2012.22

- *"Green" aspect (G)*: the amount of natural harmlessness including Energy saving (in Watt), Ecocompatibility of materials (energy saved for their materials recycling) and Space saving (in Rack Unit, or RU) of the Grid computing systems of a Hardware Provider (the Grid site).

Particular care has been put in defining the parameters S and G. The steps required in order to build a "Green" Provider have been sketched in Table 6.1 together with its evaluation.

Steps	**Objects**
1a. *Design Aspects*: Collapsing	servers -> cpus -> cores -> hyper-threading technology
	memory modules, hard disks, network cards, fans.
1b. *Design Aspects*: Virtualizing	services.
2. *Basic Hardware Aspects*: Buying	efficient hardware components (also by rack size).
3. *Basic Software Aspects*: Enabling	server processor power-saving features.
4. *Advanced Core Aspects*: Developing	user/system on-demand application-oriented policies to scale CPU frequencies.
5. *Human Aspects*: Saving	switching off monitors and unused peripherals.

Table 6.1: **How to build a "Green" Provider.**

This particular attention to the parameter G is due to the fact that Green Computing is an emerging field of the last years [131] and can be considered another fundamental aspect of Sustainability. Green Computing (or Green IT), refers to environmentally sustainable computing or IT and can be defined as the study and practice of designing, manufacturing, using and disposing of computers, servers and all the associated subsystems (such as monitors, print-

ISSN: 2279–8773

ers, storage devices, and networking and communications systems) efficiently and effectively with minimal or no impact on the environment [132]. The goals of Green Computing are similar to those of the more celebrated Green Chemistry: reduce the use of hazardous materials, maximize energy efficiency during the product's lifetime and promote the recyclability or biodegradability of defunct products and factory waste. Research continues into key areas such as making the use of computers as energy-efficient as possible and the design of algorithms and systems for efficiency-related computer technologies. Accordingly, in order to determine standard evaluations for G, we can refer to the SWaP (Space (S), Watt (W) and Performance (P)) formula proposed by Sun Microsystems:

$$SWaP = P\frac{S}{W} \tag{6.1}$$

in which one can determine P (usually measured in Giga FLOPS[1]) using standard benchmarks (as for example [133]), S measuring the size of the Grid site in RU and W calculating its power consumption (using a professional power meter recording the total power consumed during a test run). As a matter of fact, in order to avoid inaccurate measurements, the GREEN500 [134] tu-

[1]In computing, FLOPS (or flops or flop/s) is an acronym meaning FLoating point Operations Per Second. The FLOPS is a measure of a computer's performance, especially in fields of scientific calculations that make heavy use of floating point calculations, similar to the older, simpler, instructions per second.

ISSN: 2279-8773

torial can be adopted [135] (if one has not a suitable power meter, the data available from existing benchmark runs or vendor site planning guides can be used).

6.3 Quality of Users (QoU).

6.3.1 Active and Passive Filtering.

As to QoU, we have already mentioned the necessity of collecting and filtering different implicit and explicit information provided by VO/VRC users. In this field, Active Filtering (AF) is the method that in recent years has become increasingly popular due to an ever growing base of information available on the Web. The AF method differs from the non active CF one mentioned in chapter 4. AF is based on the fact that users want to share information with other peers while non active CF does not need that the users explicitly intend share opinions and feedbacks. In our case, VO/VRC users will utilize GKMS to send their feedbacks (e.g. users satisfaction) over the Grid where others can access it and use the ratings of the Grid Services to make their own decisions. There are various advantages in using AF. One of them is the fact that the rating is given to something of interest to the user who has actually used the considered Grid Service. This corroborates the credibility

ISSN: 2279-8773

of this type of ranking. Another advantage of using AF is the fact that the users explicitly want to (and ultimately do) provide information regarding the matter dealt. There are some disadvantages in using AF, though. One is that the opinions expressed might be biased. Another is that fewer feedbacks are obtained than when using passive approaches (see below) because the act of providing feedbacks requires explicit action by the user. In addition, small amounts of highly favorable opinions might result in a bloated evaluation for a Grid Service that makes it rank unrealistically with respect to other Grid Services with longer lasting evaluations. A type of non active CF methods that has a great potential is Passive Filtering (PF). PF collects information implicitly without involving the direct input of opinion by the user whose evaluation is instead deducted by his/her actions. This reduces the variability of the opinions and the pressure exerted on the users leading to more natural outcomes. These implicit filters are, therefore, aimed at determining which are the real wishes of the user and the applications of potential interest for him/her because they rely on all the actions undertaken by VO/VRC users. An important feature of PF is the use of time to determine when a VO/VRC user is running a program and the evaluation of semantic aspects to single out high-value execution paths (*Execution Path Similarity*) as well as the ”goodness” of the experiments. The major strength of PF is that it does not include the analysis of certain variables that would normally be considered

ISSN: 2279-8773

in AF. For example, in AF only certain types of VO/VRC users will take the time to rank a Grid Service while in PF anyone accessing the system automatically does it.

To get an optimal evaluation of the QoU, GCreS is designed to combine the classical AF and PF CF methods with the VO/VRC users profile. In other words, CF can be used to generate some users profile information at an initial stage then utilized as well to validate the received feedbacks in the subsequent fully-operational state of GCreS. Moreover, a user feedback provides useful information on the user him/her self (as for example when a program is already well-evaluated by the community and gets a different evaluation by a VO/VRC user).

6.3.2 An Example of Passive Filtering.

PF indicators can be used to help GCreS to evaluate VO/VRC user activities. As for example, in the case of Passive Users (see item 1 of Table 3.1), it could be useful to determine the ratio of the number of compilations over the number of runs, the ratio of the number of compilations over the number of successful results and the ratio of the number of successful results over the number of runs as well as the correct utilization of the Grid middleware itself.

ISSN: 2279-8773

In this respect, a preliminary users classification based on the different characteristics of their compiling and running activities (performed on the section of the Grid available to the COMPCHEM VO) has been firstly carried out in GriF. It offers, in fact, the possibility of collecting at different levels (from the upload of the input file to the collection of the results) a certain amount of information (stored in the `grif-db` database mentioned in chapter 4) that will turn out to be particularly useful to optimize the management of a VO/VRC by defining QoS parameters and by profiling VO/VRC users evaluating their QoU.

To this end, we have analyzed the working habits of a sample of COMPCHEM users (the group is made of 10 users) in terms of their ability and/or interest to run and/or modify the codes used. The results of this analysis are illustrated in Fig. 6.1 where the histogram of the frequency of the value of the daily ratio between the number of Grid program compilations (`Nc`) and the number of related executions (`Nx`) measured during the month considered is shown.

As apparent from the figure, the measurements are characterized by a clear bimodal structure made by a sharper peak placed on the extreme left hand side region of the plot and a smoother one placed on the opposite side. The first peak is centered on the interval 0.05 – 0.10 (with a bin width of 0.05). It clearly indicates that the largest number of Grid jobs run using an already

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

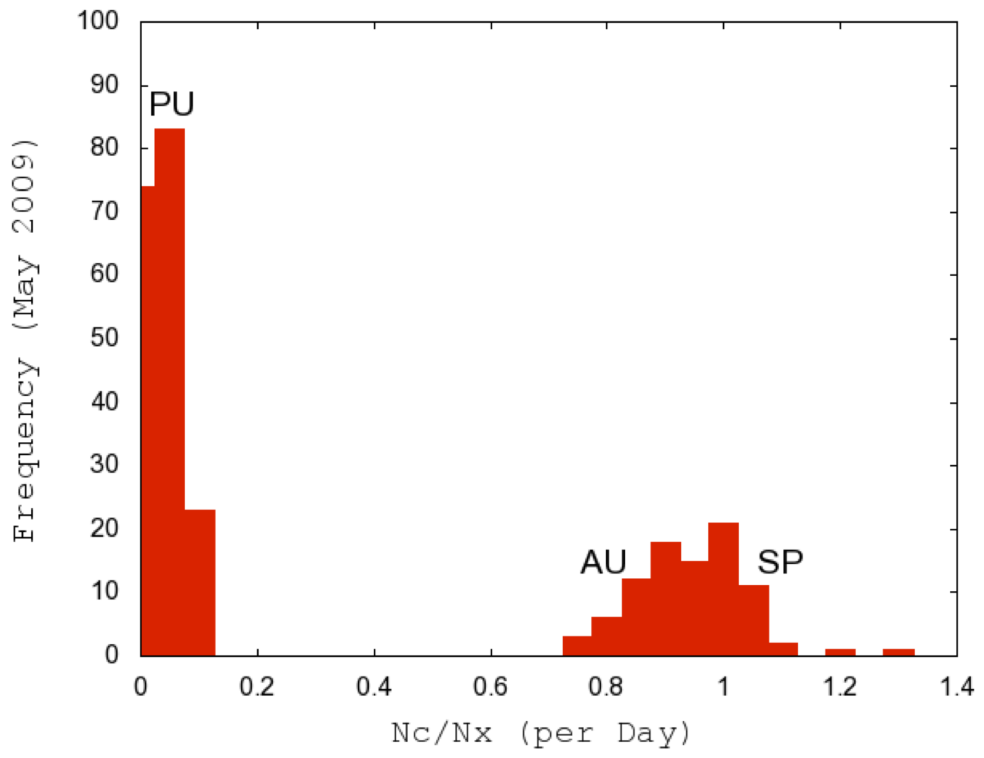


Figure 6.1: **The three COMPCHEM user profiles: the Passive User (PU), the Active User (AU) and the Service Provider (SP).**

compiled executable and that the changes are confined to the data of the input file. On the contrary, the second peak tells us that for the second largest number of Grid jobs (though smaller than the first one) the compilation is performed almost every time the Grid job is executed. It is worth noticing, however, that the broad shape of this maximum implies that there is an appreciable extent of variability as far as compilation before execution is concerned. This means that the application is sometimes not recompiled before being executed whereas other times it is compiled more than once.

ISSN: 2279-8773

6.3.3 Passive and Active Users and Service Providers.

The appearance of such a bimodal structure suggests the existence of two main types of VO users. The first type (associable with the left hand side peak) can be assimilated to the *Passive User* (PU) of Table 3.1 (see item 1, upper right hand side). PUs typically run on the Grid platform a stable version of an application and, therefore, do not need to compile the application every time a related Grid job is sent for execution. This type of VO user is more a "production scientist" who builds his/her scientific investigations on a massive production of numerical experiments carried out using well-established numerical procedures. In our study this is the case, indeed, of the scientists carrying out extensive computational campaigns.

The second type of VO user (that can be associated with the right hand side maximum) can be assimilated to the *Active User* (AU) of Table 3.1 (see item 1, bottom right hand side). AUs typically develop on the Grid platform new applications or new versions of an application and, therefore, need to compile about each time the application is sent for execution. This type of user is a "chemistry expert" that, in the case of the sample of users considered by us, develops new applications of the quantum mechanics codes dealing with chemical processes. As a matter of fact, in the testbed presented in chapter 4, the main developments have been concerned with the implementation of a

ISSN: 2279-8773

set of new executables based on different PESs for different chemical systems (this has widened the offer of the ABC program by the extensive use of both ABC and GP Grid Services).

Another type of VO user bears similar (though not identical) characteristics as those of the AUs. This is the *Service Provider* (SP), who carries out the work of wrapping computer applications into Grid Services (and testing them as well) of GriF. This means that in the right hand side structure of Fig. 6.1 the broad structured maximum could be split into an AU and an SP slightly displaced distributions centered one just above and the other just below the `Nc/Nx = 1` value. As a matter of fact, AU and SP users in the sample investigated by us do not have the same nature (as also pinpointed in a forthcoming paper [136]). In particular, while the AUs are mainly ”expert chemists” using GriF more or less transparently to build their own or shared applications, SPs are mainly ”computer scientists lent to Chemistry” for the purpose of developing Grid Services on behalf of the generic user (PUs in most cases). In this respect, SPs can be considered as a technical specialization of Software Providers of Table 3.1 (see item 2) due also to the fact that they use GriF more wisely adapting it even for the interaction among existing applications.

ISSN: 2279-8773

6.3.4 Software Developers.

Our study, however, has singled out a more extreme case of SPs that we call Software Developers (SD)s. This is clearly indicated by the structure shown in Fig. 6.2 in which, as in Fig. 6.1, the frequency of the value of the daily `Nc/Nx` ratio is plotted as a histogram (though using larger bins of width 1) whose abscissa extends to 20. Fig. 6.2 tells us that in our case SD users bear statistical importance. SDs have little interest in running existing computational scientific applications (as in the case of GriF and/or Grid Services developers). Yet they are interested in building new applications and exploit for example new theoretical approaches to a given problem. For this reason, they compile much more frequently than running. In this respect, SDs are mainly "framework developer" (elsewhere defined "software architect" in Computer Science) aimed at developing new parts of the GriF structure and of its functionalities (as for example the Web Services implementing the compilation feature) and can be considered as well as an abstraction of Hardware Providers of Table 3.1 (see item 3). Consequently, for them the value of the daily `Nc/Nx` ratio is abundantly larger than 1 (in our case we found a peak at `Nc/Nx = 11`).

A caveat, however, has to be issued at this point. In fact, while the PU characteristics (daily `Nc/Nx` values close to 0) are extremely localized, `Nc/Nx`

ISSN: 2279-8773

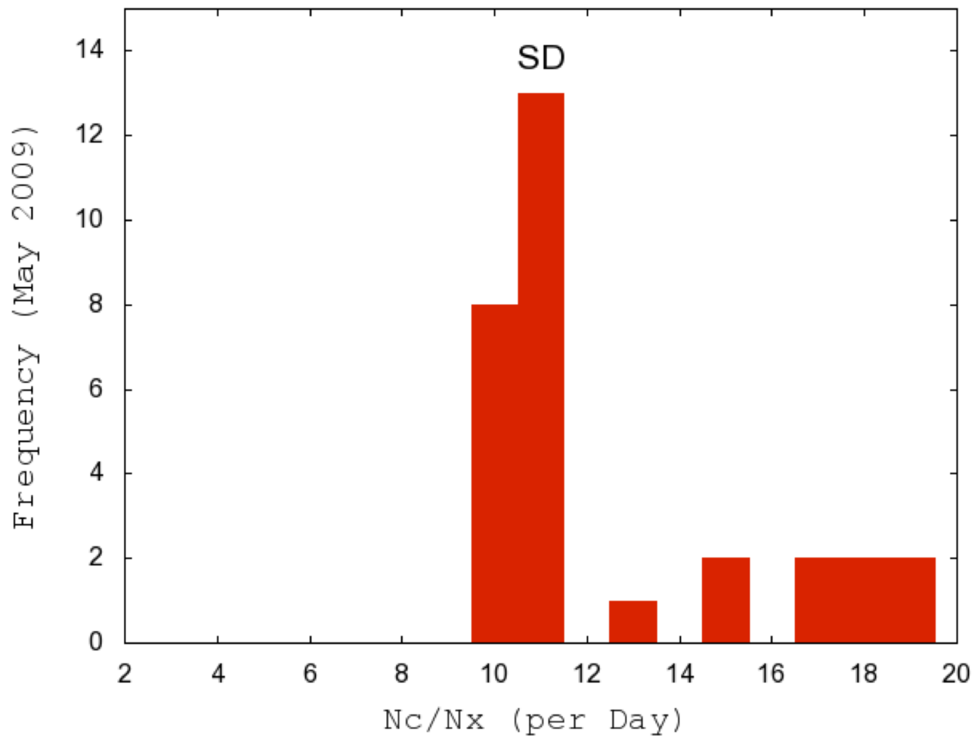


Figure 6.2: **The fourth COMPCHEM user profile: the Software Developer (SD).**

values ranging from about 1 to larger values are much more scattered. Moreover, it would not come as a surprise a further smoothing of the results for larger statistical samples that would make the separation of the various kinds of COMPCHEM users more difficult to establish. In some cases, in fact, the same user may play different roles at the same time. This makes a more articulated definition of a user of a VO (or even of a VRC) appropriate and the implementation of new Grid sensors (as those introduced in chapter 5) appears to be strongly necessary.

ISSN: 2279-8773

6.4 GCreS: A Prototype Credit System Model.

6.4.1 The Architecture of GCreS.

The above mentioned concepts find a useful application in fostering the Sustainability of a VRC by equipping it with mechanisms suited to better evaluate the commitment of VRC users to their organization and to reward them appropriately for the associated work. To this end, we have designed GCreS, a prototype tool based on GriF operating in a standard Grid scenario aimed at rewarding VRC users by assigning *Credits* to them and at evaluating Grid Services by assigning *Costs* to them as well.

GCreS consists essentially of a 3-Tier Architecture [137] based on a Back-end, on a Business Logic and on a Presentation layer (see left hand side of Fig. 6.3), respectively. Accordingly, both GCreS and GriF will establish the Grid Knowledge Management System proposed by the present Thesis work, as sketched in Fig. 6.3.

The Back-end layer is devoted to collect, initially in a MySQL Database (called `gcres-db`), all the data related to the available information on VRC users and Grid Services. Therefore, as apparent from the figure, GKMS will provide a single Grid Database integrating both the local `gcres-db` database of GCreS and the local `grif-db` database of GriF. Accordingly, by applying a well-known Data Warehousing strategy [138] in which different kind of

ISSN: 2279-8773

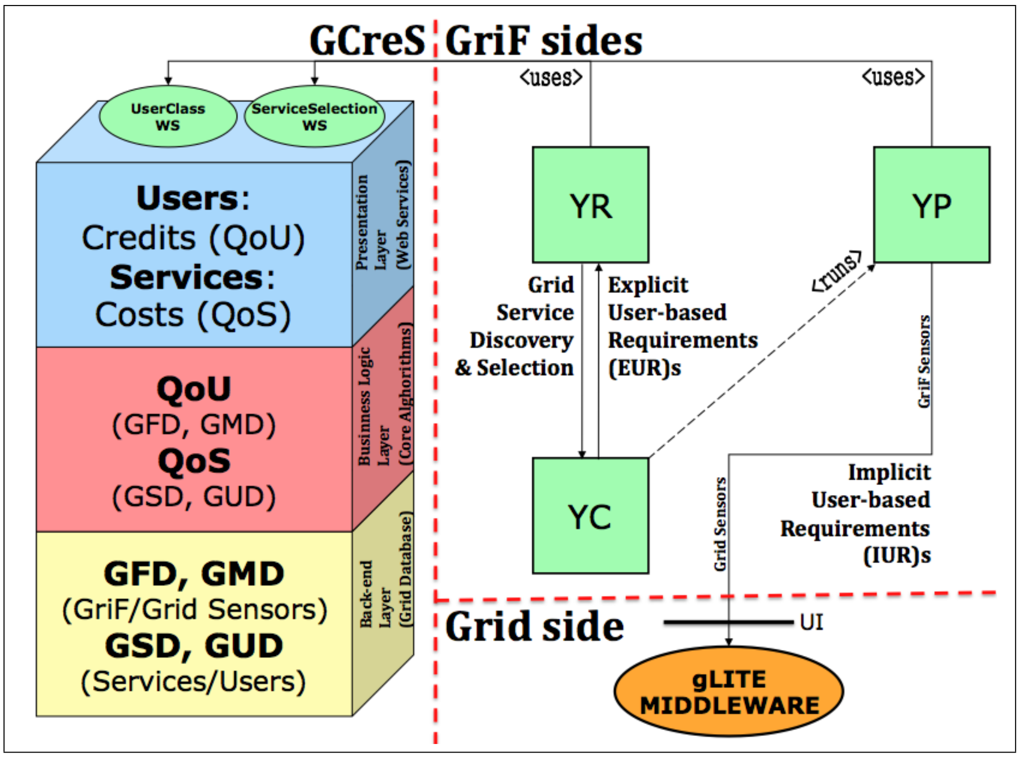


Figure 6.3: **Grid Knowledge Management System (GKMS) Communications.**

structured and non-structured large amount of data are gathered together, GCreS collects four types of data:

- *Grid Framework Data* (GFD), in which all the information saved in GriF (in both the Database and the Log Subsystem, as for example the number of run and the success/failure ratio, the name of the related application and its run modality ('STANDARD' or 'CUSTOM'), the number of compilations performed as well as information related to the operations time (aimed at identifying execution path similarities discussed above for PF in order to produce semantic inferences on the VRC users behavior)), can be recorded and analyzed per user;

- *Grid Middleware Data* (GMD), in which accounting information related to the operations time, to the amount of memory consumed (separating the real from the virtual), to the cpu time elapsed, to the CE queue used and to the number of jobs run registered by the Grid Middleware, can be recorded (in order to integrate that of GFD) per VO/VRC and/or per user;
- *Grid Service Data* (GSD), in which QoS parameters can be calculated and saved per Grid Service (and YP name);
- *Grid User Data* (GUD), in which users feedbacks and profiles can be organized.

To collect GFD, use can be made of various off-line scripting procedures which will be responsible for the handling of the information from both GriF Log Subsystem and database to GCreS. Moreover, GCreS will be also able to access other different Web Services-based GFD sources (for example the retrieving of proper information from the Grid Enabled Molecular Science Through Online Networked Environments (GEMSTONE) [139] which is an integrated SOA framework for accessing computing resources). To collect

ISSN: 2279-8773

GMD, use can be made of various sources (like for example the already mentioned DGAS system and/or the official EGI Accounting, as well as the new Grid Sensors proposed in the previous chapter). To collect GSD, a separate application producing detailed measurements of the QoS evaluation parameters for each Grid Service offered by a VRC, has to be implemented. To collect GUD, at present the quite crude method of exporting the user profiles and feedbacks already recorded by GriF into the GCreS database could be used (although a dedicated procedure (or a set of them) should be better implemented in order to improve the already existing collaborative part of GriF).

The Business Logic layer (that will be fully-implemented in the final version of GCreS) paves the way to the definition of the Credits/Costs schema (then consumed by the Presentation upper layer) by implementing the core algorithms producing the QoS (for which a first formulation will be given in the next section essentially as a function of GSD and of GUD) and the QoU (for which a first formulation will be given later on essentially as a function of GFD and of GMD though, sometimes, also GUD could be used especially when an implicit self-evaluating point of view is considered).

The Presentation layer, as typical of Business-to-Business (B2B) relationships, in addition to reward VRC users in terms of *Credits* on the basis of their calculated QoU and to produce *Costs* for Grid Services on the basis

ISSN: 2279-8773

of their calculated QoS as well, is intended to interact with GriF (see upper right hand side of Fig. 6.3) exposing two Web Services (called `UserClass` and `ServiceSelection`, respectively) that can be consumed (in the next `3.0` version of GriF), respectively, by:

- *YPs* (the GriF Providers), which before running a job on the Grid can automatically add to their running policy specific requirements derived from the related user class level provided by the `UserClass` Web Service (on the basis of the QoU of the VRC user owning the Grid job);

- *YRs* (the GriF Registries), which before returning the YPs list matching an applications request (made by VRC users) can access each related Grid Service QoS (provided by the `ServiceSelection` Web Service) to return a ranked list of the hosting YPs (as is typical of the above mentioned Service Selection concept) instead of the unranked one (as is typical of Service Discovery).

Accordingly, we define, respectively, *Implicit User-based Requirements* (IUR)s the low-level requirements automatically generated by YPs during their GriF running phase (from this point onwards IURs can be adopted transparently for VRC users allowing YPs to use them in order to run jobs on the Grid with different priority) and *Explicit User-based Requirements* (EUR)s the high-level requirements (e.g. reliability and/or performance rather than ma-

ISSN: 2279-8773

chine parameters as already mentioned in the first section of chapter 4) which can be requested by VRC users to YR during each (Grid) Service Discovery and/or Selection phase.

In addition, GCreS can be considered in theory as a single component to be plugged (or not) into the Grid middleware. Accordingly, in the former case servers hosting GCreS must be initialized as Grid machines while, in the latter case, they have to be bridged to UI (as in the case of YPs of GriF) in order to access the needed Grid information (in particular for the GMD data).

6.4.2 The proposed Formulation of QoS.

In the first formulation of QoS we have considered as applicable to the Grid Services of a VRC adopting GKMS the following QoS parameters (already described above): *Accessibility*, *Integrity*, *Reliability*, *Security*, *Performance* and *Innovation*. This means that each QoS parameter indicates the strength of the Grid Service in one of the related areas. At the same time, their sum provides us with an overall **QoS** evaluation according to the following expression:

$$QoS = w_0 S_{acc} + w_1 S_{int} + w_2 S_{rel} + w_3 S_{sec} + w_4 S_{per} + w_5 S_{inn} \tag{6.2}$$

ISSN: 2279-8773

where $w_{(i=0..5)}$ are the weights that a Quality Manager (QuM) or a Quality Board (QuB) of a VRC chooses for each QoS parameter which we have respectively defined as follows:

$$S_{acc} = 1 - \frac{\sum_{i=1}^{N_e} TTR_i}{N_f} \qquad ; \qquad N_f \neq 0 \tag{6.3}$$

where N_f is the number of functions (f)s (each f corresponds to a stable Web Service call or a set of them) invoked by a Grid Service, TTR_i is the Time-To-Repair associated with each error occurred with N_e being the number of errors occurred in the time interval considered. To determine N_e, a dedicated test program checking the accessibility of each function of the Grid Service (for example the submission of a Grid job, called 'RunGP' later on) every X minutes (TTR starts from 0 and is incremented by 1 for every failure reported by the checking procedure related to the same error), can be adopted. Moreover, the value X can be chosen on the basis of the SLA of the Grid Service (e.g. low values, for example, $X < 5'$ and high values, for example, $X > 60'$ (1 hour) can be adopted, respectively, for Grid applications requiring high availability and for applications accepting a reasonable amount of downtime);

$$S_{int} = k_{int} \tag{6.4}$$

where k_{int} is a constant indicating rollback absent (the value is 0), partially implemented (not fully-tested in production, value 0.5) or fully-operating

ISSN: 2279-8773

(value 1), respectively;

$$S_{rel} = 1 - \frac{N_e}{N_f} \qquad ; \qquad N_f \neq 0 \tag{6.5}$$

$$S_{sec} = (k_{en} + 2k_{ae} + \frac{4k_{ao}}{2^{k_{ae}}})/5 \tag{6.6}$$

where k_{en}, k_{ae} and k_{ao} are three constants (value either 0 or 1) indicating if encryption, authentication and authorization are supported or not, respectively. Accordingly, $k_{ao} = 1 \Rightarrow k_{ae} = 1$. Moreover, k_{ae} has to be supported for each Grid Service (otherwise, when only encryption is enabled, the Grid Service is assumed as un-secure and it should not be released to the public);

$$S_{per} = \frac{TR - \alpha}{LT - \beta} \qquad ; \qquad LT \neq \beta \tag{6.7}$$

where the Throughput (TR) can be assumed as the number of times that a Grid Service has been consumed in a time interval while the Latency (LT) evaluates the way it is conveyed. LT is most often quantified as the delay (the difference between the expected and the actual time of arrival) of the data (as in chapter 4 for AL). It has to be considered, however, that a YP with high average TR and LT can be worse for some applications (and their Grid Services) than the one with low average TR and LT. For this reason, the α and β coefficients are used by the QuM to favor one aspect or the another by properly shifting the related scales. In Eq. 6.7:

$$TR = N_t \qquad ; \qquad N_t \in \Delta_t \tag{6.8}$$

ISSN: 2279-8773

where Δ_t is the time interval considered, and:

$$LT = \frac{\sum_{i=1}^{N_f}(t_i - k_i)}{N_f} \qquad ; \qquad N_f \neq 0 \in \Delta_t \tag{6.9}$$

where t_i is the time (say in seconds) elapsed by each f (e.g. the retrieve of the results, called 'Result' later on) and k_i is the associated time constant indicating the optimal time (in the same unit as t_i) for it (also depending, for some fs, by the length of the files involved). For example we have identified (by making use of the special variable `WSName` already mentioned in chapter 4 distinguishing every f and also reporting the related elapsed time, as shown in Fig. 4.9) 10 different fs for GP called, respectively, 'Login', 'Upload', 'Charge', 'RunGP', 'Check', 'DB:Refresh', 'DB:Delete', 'Result', 'GriFStatus' and 'Send Feedback' for which the corresponding k are[2]: 0.006, 1.1 (for each MB uploaded), 0.005, 6 + 1.1 for each MB input used (or 31 + 1.1 when GriF Ranking has not been selected by a VRC user), 0.81, 0.008, 0.007, 1.3 (for each MB downloaded), 2.8 and 0.385 seconds. It is worth noticing here that in order to obtain a realistic evaluation of S_{per} (and also of the QoS) for a Grid Service it is necessary to apply it to a real production environment (for example more than one Grid Service hosted for each YP, several VRC users accessing GriF and large amounts of Grid job runs) and to develop as

[2]The values of `k` have been calculated by averaging the elapsed time by each `f` during six months of activities.

ISSN: 2279-8773

well a dedicated program manipulating properly all the information saved in the `axis.log` file (otherwise S_{per} is *Not Applicable* and is considered equal to 0);

$$S_{inn} = (\gamma N_d + \delta N_c + \epsilon \frac{N_t}{N_u} + \sigma \frac{N_p}{N_u} + \rho \frac{N_l}{N_v} + \lambda \sum_{i=1}^{3} \tau_i N_{s_i})/6 \qquad (6.10)$$

where $N_u > 0$, $N_v > 0$ and γ, δ, ϵ, σ, ρ, λ and τ are coefficients used by QuM to weight the different contributions. Moreover, the G (Green) aspect has not been considered here because its nature is better suited to evaluate Hardware Providers and not Grid Services (as it will be addressed in the last section of the present Thesis work). In Eq. 6.10, we also assume that N_d is equal to 1, 0.5 and 0 when, respectively, the number of days elapsed starting from the publishing date of the considered Grid Service is smaller than 182 (six months) days, larger than 183 days and smaller than 365 (one year) days, and larger than one year.

After calculating the **QoS** of a given Grid Service, a *Cost* (and then a corresponding position in the ranked list of available Grid Services) is determined using the `ServiceSelection` Web Service. Moreover, when GUD will be fully-available, the global QoS as well as each (objective) QoS evaluation parameter will be improved and refined with the (subjective) information provided by VRC users (e.g. their feedbacks) also considering their reliability (e.g. their profiles).

ISSN: 2279-8773

6.4.3 The proposed Formulation of QoU.

In the first formulation of QoU for a VRC adopting GKMS, we have produced some custom indicators useful for all type of users and applications run by a specific Grid Service (as for example the GP one). In particular, some of them can also be useful to corroborate the singling out of the four user profiles PU, AU, SP and SD discussed in the previous subsections. For example, when we consider the percentage $P_{c,x}$ of runs executed as 'CUSTOM' (e.g. applications uploaded for running by VRC users) rather than as 'STANDARD' ($1 - P_{c,x}$, e.g. applications made already available by the Grid Service considered), it is likely that a $P_{c,x}$ is larger for SDs, SPs and AUs, respectively, than for PUs (which typically run on the Grid platform a stable version of an application and, therefore, do not need to compile every time they execute) because, in general, 'to compile' tends to coincide with 'to deal with custom forms'. For this reason, a $P_{c,x}$ value close to 1 should be considered more likely for PUs than for the other three profiles. Accordingly, an accurate study of this relationship (when the number of the available applications offered under the form of Grid Services to users will be large enough to be considered statistically meaningful) can confirm the four profiles already identified by the accurate investigation of the COMPCHEM user behaviors described above. At the same time, the sum of the QoU parameters described below provides

ISSN: 2279-8773

an appropriate overall **QoU** evaluation that was formulated as:

$$QoU = p_0 w_6 U_{cx} + p_1 w_7 U_{cu} + p_2 w_8 U_{cm} + w_9 U_{ge} + w_{10} U_{fb} \tag{6.11}$$

where $p_{(i=0..2)}$ are three coefficients (with possible values of 0.1, 1 and 10)[3] weighing the contribution of the different user profiles and valued as shown in Table 6.2. In the same equation $w_{(i=6..10)}$ are weights chosen by either the QuM or QuB for each QoU parameter (in addition to U_{cx} corresponding to the ratio `Nc/Nx` already described above and mainly exploited in order to identify the various user profiles), respectively, as follows:

User Profiles	p_0	p_1	p_2
Passive User (PU)	10	0.1	0.1
Active User (AU)	1	0.1	0.1
Service Provider (SP)	0.1	1	1
Software Developer (SD)	0.1	10	10

Table 6.2: **Coefficients of the U_{cx}, U_{cu} and U_{cm} parameters for the various COMPCHEM user profiles.**

$$U_{cu} = P_{c,x}(N_b * R_{r,N_b} + (N_x - N_b) * R_{r,N_{(x-b)}}) \tag{6.12}$$

[3]In this respect, it is worth noticing here that the resulting values for U_{cx}, U_{cu} and U_{cm} have different meanings depending by the profile of the user considered while those for U_{ge} and U_{fb} can be assumed to be independent of it.

where N_b is the number of runs not derived from compilation, R_{r,N_b} is the ratio between the number of results obtained (related to N_b) and N_b, $R_{r,N_{(x-b)}}$ is the ratio between the number of results obtained (related to runs derived from compilation) and the difference between N_x (corresponding the the `Nx` studied above) and N_b. Accordingly, U_{cu} gives additional qualitative information on the quantities `Nx` and `Nc` already studied (N_b is, in fact, inversely proportional to `Nc`). Moreover, U_{cu} favors the 'CUSTOM' running modality mentioned above since the $P_{c,x}$ is applied;

$$U_{cm} = N_r * (\psi\overline{C} + \omega\overline{M}) \tag{6.13}$$

where N_r is the number of results retrieved from the Grid by a VRC user, ψ and ω are normalization coefficients, $\overline{C}$ and $\overline{M}$ are, respectively, the average cpu time (in hours) and the average virtual memory amount (in GB) used per "Done" job by a VRC user (in this respect, in order to integrate the quantitative evaluation of GFD one can also take into account GMD);

$$U_{ge} = GE_{user} \tag{6.14}$$

where GE_{user} is the Grid Efficiency (see the previous chapter) applied to a specific VRC *user* in order to refine the QoU formulation defined in Eq. 5.2;

$$U_{fb} = N_m \tag{6.15}$$

ISSN: 2279-8773

where N_m is the number of messages (e.g. feedbacks) produced by a VRC user.

After calculating the total **QoU** (obtained by adding up the QoU related to each Grid Service used) for a VRC user, *Credits* corresponding to the class level of the type *Low*, *Medium* or *High* are determined using the `UserClass` Web Service. Moreover, as mentioned in the case of the QoS formulation, also QoU will be refined when GUD will be fully-available on the basis of VRC users self-evaluations.

6.4.4 A first trial Simulation of VRC Sustainability.

A first trial simulation of VRC Sustainability based on Quality Evaluation (applied to a VO in this case) has been carried out by considering different periods of GriF activities (one month for the QoS example and three months for the QoU one) when offering the already mentioned ABC and GP Grid Services to the COMPCHEM VO users. Accordingly, we have collected GFD, GMD and GSD for both ABC and GP as well as for the COMPCHEM users. At that time, partial GUD (only related to some aspects of QoU) were available. Our goal here is to illustrate the initial evaluation of the QoS for the GP Grid Service and of the QoU for a COMPCHEM PU named *sylvain* (who typically runs applications dealing with chemical processes in

ISSN: 2279–8773

order to carry out realistic a priori simulations) with respect to both ABC and GP.

The QoS of a sample Grid Service.

In this case, we have measured the above formulated QoS parameters for GP during the month of September 2010 (in production state) without considering the management activity of the related SP user (that is expert in the handling of GP) in order to attempt an evaluation of the meaning of the reported values.

In this way we have obtained $N_d = 1$, $N_c = 0$, $N_t = 17$ (assuming $N_t = N_x$ in this example), $N_u = 9$, $N_p = 8$, $N_l = 0.5$ (an average value over all the functions introduced by the GP since some functions have been already implemented also by other Grid softwares while others are quite innovative), $N_v = 0.5$ (a long time was needed to develop GP while no money costs was provided), $N_{s_1} = 0$, $N_{s_2} = 1$ (GP is available for free and it has been developed only by making use of open source software), $N_{s_3} = 0$, $N_f = 194$ and $N_e = 3$ (for which $TTR_{(i=1..3)} = (2, 0, 3)$, respectively). Moreover, by applying Eq. 6.2–6.10 (and choosing $w_{(i=1..5)} = 1$ for all the QoS parameters, $\alpha = \beta = 0$ as well as $\gamma = \delta = \epsilon = \sigma = \rho = \lambda = \tau_1 = \tau_2 = \tau_3 = 1$) to the GSD for the GP Grid Service, a QoS value of 4.3 was obtained by the QuM (see Table 6.3 for details).

ISSN: 2279-8773

	Grid Service: *GP*
S_{acc}	0.9742
S_{int}	1
S_{rel}	0.9845
S_{sec}	0.4
S_{per}	*Not Applicable*
S_{inn}	0.9630
QoS	**4.3217**

Table 6.3: **An example of QoS evaluation for a Grid Service.**

It has to be commented here, however, that only after comparing the resulting QoS value with those of other Grid Services of the same type one will be able to evaluate a truly reliable *Cost* for the Grid Service *GP* considered in this example (although a high-ranking value can be guessed for it as suggested by the upper limiting value 1 for S_{acc}, S_{int}, S_{rel} and S_{sec}).

The QoU of a sample User.

In this case we have measured the QoU parameters formulated above for the COMPCHEM PU *sylvain* during the considered three months of his activity in Grid (from August 2010 to October 2010) carried out using both ABC and GP Grid Services (summing up the related values).

ISSN: 2279-8773

To this end, by choosing $p_0 = 10$ (indeed, low values of U_{cx} are more likely for PUs because they compile less than other users), $p_1 = 0.1$ and $p_2 = 0.1$ (indeed, high values of both U_{cu} and U_{cm} are more likely for PUs and AUs because they run on the Grid more than SPs and SDs) according to Table 6.2, we have obtained, in total for both the Grid Services illustrated in the present Thesis work, $\texttt{Nc} = 10$, $\texttt{Nx} = 167$, $P_{c,x} = 88.24\%$, $N_b = 162$, R_{r,N_b} $= 0.9815$ (having 3 not-retrieved r), $R_{r,N_{(x-b)}} = 0.8$ (having 1 not-retrieved r), $N_r = 163$, $\overline{C} = 1.2995$, $\overline{M} = 0.2873$, $GE_{sylvain} = 0.8449$ and $N_m = 12$. Moreover, by applying Eq. 6.11–6.15 (and choosing $w_6 = 0.5$ for U_{cx} because one (the GP) of the two Grid Service has the compilation function not yet completely stable resulting in a number of compilations performed by VO users theoretically lower, $w_{(i=7..10)} = 1$ for all the other QoU parameters as well as $\psi = \omega = 1$) to both the GFD and GMD (and also to a reduced set of the GUD) for the VO user mentioned above, a QoU value of 53.4 was obtained by the QuM (see Table 6.4 for details).

Finally, after comparing the obtained QoU value with those of other COMPCHEM users of the same type (even if in the presence of an inadequacy of the GUD estimate), we have been able to assign a *High* class level (see the three class levels already mentioned above in the formulation of QoU) to the *sylvain* PU considered in this example for the corresponding *Credits* award.

ISSN: 2279-8773

	PU user: *sylvain*
U_{cx}	0.0598
U_{cu}	143.8338
U_{cm}	258.6484
U_{ge}	0.8449
U_{fb}	12
QoU	**53.3921**

Table 6.4: **An example of QoU evaluation for a Passive User.**

The Quality Evaluation of a VRC.

With respect to a global evaluation of VRCs, it is worth mentioning that some of the concepts and outcomes illustrated by the present Thesis work (e.g. the Ranking and "Green" Providers) allow the introduction of new quality evaluations called by us Quality of Computing (QoC) and Quality of Provider (QoP), respectively.

QoC consists of an evaluation of computing-related objects of the Grid Middleware belonging to a VRC (e.g. UIs, SEs, CEs and batch systems). Accordingly, for example the concept of Ranking (that encompasses P an LT) described in chapter 4 as well as Eqs. 4.1 and 4.2 can be applied to a gLite CE queue (or a set of them). Moreover, also the GE (introduced in chapter 5)

ISSN: 2279-8773

can be used.

QoP consists of an averaged evaluation of Hardware Providers (the YPs) offering Grid Services belonging to a VRC. To this end, for example, YPs hardware and network characteristics should be considered[4]. Moreover, also the concept of "Green" Provider (related to the variable 'G' and not included in the present formulation of the S_{inn} parameter of the QoS) as well as two of the general QoS parameters (described before in this chapter) respectively called *Accessibility* (S_{ava}) and *Regulatory* (S_{reg}), can be applied. Accordingly, S_{ava} and S_{reg} can be formulated as follows:

$$S_{ava} = \frac{1}{N_e + 1} \tag{6.16}$$

$$S_{reg} = (k_a + k_b + k_c + k_d)/4 \tag{6.17}$$

where k_a, k_b, k_c and k_d are four constants (with possible values of 0, 0.5 and 1) indicating, if the SOAP engine (e.g. AXIS), the Web container (e.g.

[4]An example of Hardware Provider offering the GP Grid Service is the new CE of the COMPCHEM VO Grid site assembled by us early in 2010 (that is the YP of our testbed) made by 40 2.13 GHz Intel Xeon quad-core 4 MB cache 64-bit nodes (occupying 40 RU in space) equipped with 160 GB of RAM (4 GB for each Grid node), 10 TB of SATA Storage (250 GB for each Grid node as well) and a firewalled Gigabit-based Ethernet Network supporting NAT [140] and Channel Bonding [141] for which the resulted P (even knowing Instructions Per Cycle (IPC) equal to 4 for it) is linear and equal to 0.91 Tflops (with a consequent constant efficiency of 79 per cent) making use of the High Performance Computing Linpack Benchmark (HPL) [142] and of the Intel® MPI Library [143].

ISSN: 2279-8773

Apache Tomcat), the UDDI implementation (e.g. Apache jUDDI) as well as the JRE are either obsolete (no support is given anymore), still supported (although an upgrade is needed) or updated, respectively.

Accordingly, the overall **Quality-to-Community** (*Q2C*) referred to a VRC in the present Thesis work (that could be also applied to the HUCs mentioned in chapter 2 as well as to a Grid Community in general) is proposed here to be formulated as follows:

$$Q2C = QoC \cup QoP \cup QoS \cup QoU \tag{6.18}$$

where QoS and QoU are, in this case, the sum of the **QoS** for each Grid Service belonging to the VRC considered and the sum of each VRC user **QoU**, respectively.

ISSN: 2279-8773

Chapter 7

Conclusions and Future Work

Grid empowered calculations have become an important advanced tool indispensable for scientific advances as confirmed from the funding of several european projects by European Commission in these last years in the field of Grid Computing, as in the case of EGEE and then of EGI. In this respect, Collaboration is the new angular stone on which Virtual Research Communities and Heavy User Communities rest their efforts to build their Sustainability. Moreover, the advent of SOA, Web Services and Grid Services has offered an enormous potential for lower integration costs and greater flexibility. A crucial aspect of SOA has been the independence of the service interface from its implementation. In fact, such services are consumed by VRC users that are not concerned with how these services will execute their requests. Accordingly, Web and Grid Services will be the pillars of the next step in the Web's evolution, since they promise the infrastructure and tools for automation of academic (e.g. in the field of T&L and research) and commercial relationships (e.g. Business-to-Consumer (B2C) as well as B2B between GriF and GCreS as already described in chapter 6) over Internet and the EGI Grid.

In the present Thesis work the development of GriF, a Collaborative Java SOA Grid Framework structuring computational applications and scientific programs under the form of a set of collaborative Web Services (that are Grid Services) that can be run transparently on the Grid (black-box like) even without explicit knowledge of Grid technicalities and great familiarity with Computer Science, has been developed. GriF is open to be driven by a user-side control allowing the management of a domain-specific operation logic in which the user of a VO (or an HUC/VRC) can select various options before entering the Grid running phase facilitating a user friendly adoption of collaborative and alternative computational schemes. In particular, several efforts have been undertaken within the CheMIST community (and in particular within the COMPCHEM VO which is deeply involved in several other Grid activities as for example the proposal of a new VO in T&L named TELCHEM, the development of GEMS and the gridification of the EChemTest libraries) to make reactive scattering applications truly user friendly and composable for the assemblage of more complex computational

ISSN: 2279-8773

procedures. As an example, this has allowed a systematic quantum investigation of the reactive properties of fairly heavy atom diatom systems on a fine energy Grid played with different PESs and the investigation of the N + N_2 reaction leading to the singling out of an interesting rich structure for the energy dependence of the state-to-state reactive probability (also in the case of heavy system) that have been discussed.

As a result, not only it has been possible to carry out on the Grid massive computational campaigns by spending a minimum effort and achieving maximum throughput but it has been also possible to profile some types of users. This feature has been of particular interest because the profiling of the users is the basis on which an evaluation of the work carried out in a HUC/VRC can be performed and, therefore, it will be the ground on which the CheMIST community will pinpoint one of its main research activities. Yet, the indications on the VO users classification developed in this Thesis work aimed at fostering an accurate evaluation of the work carried out by UHC/VRC users have also led to a new structuring of the COMPCHEM VO into four new levels of collaboration. To this end GCreS, a Grid Credit System architecture model (relying on GriF) aimed at rewarding the contributions of HUCs/VRCs in a new sustainable Grid Economy era, has been designed and developed. GCreS is based on the Quality evaluation concept exploited in two different implementation of QoS and QoU for services and

ISSN: 2279-8773

user activities, respectively. In this respect, a first example of VRC Sustainability based on new formulations of QoS and QoU, has been also provided. Future Work along the direction of further validating the Grid Knowledge Management System presented in this Thesis work and composed by both GriF and GCreS to make it fully suitable to the common endeavours of a HUC/VRC is easy to foresee. Such work will be targeted, in particular, to make the Grid applications truly user friendly and oriented towards the adoption of the new implementations of QoS and QoU in order to empower HUC/VRC Sustainability. This will imply the introduction of new evaluation parameters and innovative Grid sensors (that we have already planned to develop in collaboration with CESGA that is the official monitoring site of the computational activities carried out on the EGI Grid).

On top of all, further work needs to be planned on some strategic decisions to be taken regarding some technical aspects of GriF (as for example the security of the communication to and from the Grid. In particular, YPs and YRs can be located both inside and outside the Grid. In the former case, they will have to be Grid machines. In other words, they will need to be authorized and authenticated as VO (or VRC) hosts). Accordingly, UIs will be no longer required and the GriF user account will be sufficient to run applications on the Grid under the form of Grid Services. In the latter case (as it is at present), a Secure SHell (SSH) tunnel (bridging the

ISSN: 2279-8773

Grid) is required in order to perform the communication between each YP and the related UI when a RC strategy has been implemented to perform all the Grid operations under the responsibility of GriF. For this purpose, several activities have been already planned for the next releases of GriF like the management of encrypted results, the writing of a more general interface extensively enabling the compilation of Grid applications, the recording of distinct values for real and virtual memory to GriF accounting, the development of a self-activating procedure providing a first evaluation for new CE queues preventing their automatic assignment to the running user, the full support for parallel (e.g. MPI) jobs as well as for Collection, Parametric and Directed Acyclic Graph (DAG, or Workflows) Grid job types [144] and also the deployment of the enhanced version of the subsystem responsible in retrieving the results from the Grid as mentioned at the end of section 4.3.7. Finally, the present work paves the way to the development of a fully running version of GKMS. To this end, the operational procedures needed to populate properly its Back-end layer will have to be improved and the GCreS Architecture Model proposed here will have to be fully-implemented on it.

ISSN: 2279-8773

Glossary and page of first occurrence

3D three Dimensional. 92

AF Active Filtering. 181

AL Average Latency. 108

APEL Accounting Processor for Event Logs. 160

API Application Programming Interface. 92

ARC Advanced Resource Connector. 21

ASCII American Standard Code for Information Interchange. 92

AU Active User. 186

AXIS Apache eXtensible Interaction System. 78

ISSN: 2279-8773

Glossary and page of first occurrence

B2B	Business-to-Business. 194
B2C	Business-to-Consumer. 211
BASH	Bourne Again SHell. 100
BEinGRID	Business Experiments in GRID. 13
C-C	Cloud Computing. 10
CC	Computational Chemistry. 25
CE	Computing Element. 89
CERN	Conseil Européen pour la Recherche Nucléaire. 13
CESGA	CEntro de Supercomputación de GAlicia. 156
CF	Collaborative Filtering. 71
CheMIST	Chemistry and Molecular Innovation Science and Technology. 2
CIC	Core Infrastructure Center. 156
CID	Content-IDentifier. 122
CMB	Cosmic Microwave Background. 62
CMST	Chemistry and Molecular Science and Technology. 32

ISSN: 2279–8773

Glossary and page of first occurrence

COMPCHEM COMPutational CHEMistry *(a Virtual Organization of EGI)*. 34

CoP Community of Practices. 43

COST European Cooperation in Science and Technology. 31

CSIC (Consejo Superior de Investigaciones Científicas. 156

DAG Directed Acyclic Graph. 214

DAGMan DAG Manager. 147

DBMS DataBase Management System. 67

DCI Distributed Computing Infrastructure. 25

DGAS Distributed Grid Accounting System. 159

DGC Distribution Grid Constant. 127

DII Dynamic Invocation Interface. 79

DIRAC Distributed Infrastructure with Remote Agent Control. 168

DN Distinguished Name. 158

e-CheMIST e-Chemistry, Molecular and Materials Innovation, Sciences & Technologies. 50

Glossary and page of first occurrence

EC	European Commission. 13
EC2E2N	European Chemistry and Chemical Engineering and Education Network. 43
EChemTest	European Chemistry Test. 51
Ecocompatibility of materials	*(Energy consumption per weight unit recycling)*. 179
ECTNA	European Chemistry Thematic Network Association. 43
EDG	European DataGrid. 13
EGEE	Enabling Grids for E-sciencE. 13
EGI	European Grid Initiative. 1
EGI.org	EGI Organization. 17
ENEA	*(a Virtual Organization of EGI owned by the homonymous Italian National Agency for New Technologies, Energy and Sustainable Economic Development (ENEA))*. 47
ERA	European Research Area. 16
ESFRI	European Strategy Forum on Research Infrastructures. 22
EUR	Explicit User-based Requirement. 195

ISSN: 2279-8773

Glossary and page of first occurrence

eV	electron Volt. 139
FAQ	Frequently Asked Question. 23
FP	Framework Programme. 13
FQAN	Fully Qualified Attribute Names. 155
G-C	Grid Computing. 9
G-DSE	Grid-Data Source Engine. 54
G-LOREP	Grid LOs REPository. 44
GAUSSIAN	*(a Virtual Organization of EGI focused on ab initio electronic structure calculation package)*. 46
GB	GigaByte. 10
GC	Grid Certificate. 85
GCreS	Grid Credit System. 2
GE	Grid Efficiency. 169
GEANT	Gigabit European Academic Networking Technology. 15
GEMS	Grid Empowered Molecular Simulator. 50

ISSN: 2279–8773

GEMSTONE	Grid Enabled Molecular Science Through Online Networked Environments. 193
GFD	Grid Framework Data. 191
GGF	Global Grid Forum. 82
GHz	GigaHertz. 207
GKMS	Grid Knowledge Management System. 1
gLite	Lightweight Middleware for Grid Computing. 21
GMD	Grid Middleware Data. 192
GOC	Grid Operations Centre. 156
GOCDB	GOC DataBase. 156
GP	General Purpose. 94
GriF	Grid Framework. 2
GSD	Grid Service Data. 192
GUD	Grid User Data. 192
GUI	Graphical User Interface. 37
HEP	High Energy Physics. 25

ISSN: 2279-8773

Glossary and page of first occurrence

HPC	High-Performance Computing. 25
HPL	High Performance computing Linpack benchmark. 208
HTC	High-Throughput Computing. 25
HTTP	HyperText Transfer Protocol. 38
HUC	Heavy User Community. 1
IBM	International Business Machines. 7
ICT	Information and Communication Technology. 24
INFN	Instituto Nazionale di Fisica Nucleare. 159
IP	Internet Protocol. 42
IPC	Instructions Per Cycle. 208
ISM	Information Super Market. 150
IT	Information Technology. 19
IUR	Implicit User-based Requirement. 194
JA	Job Adapter. 151
JAF	Java Activation Framework. 119

ISSN: 2279-8773

JAX-RPC	Java Api for Xml-based RPC. 118
JC	Job Controller. 147
JDL	Job Description Language. 98
JSP	JavaServer Pages. 93
JVM	Java Virtual Machine. 64
JWS	Java Web Service. 78
KB	KiloByte. 99
KM	Knowledge Management. 69
LB	Logging and Bookkeeping. 154
LCG	LHC Computing Grid. 13
LFC	LCG File Catalog. 99
LHC	Large Hadron Collider. 14
LM	Log Monitor. 152
LO	Learning Object. 40
LS	Life Sciences. 25

ISSN: 2279-8773

LSF	Load Sharing Facility. 170
LT	LaTency. 197
MB	MegaByte. 99
MC	Management Committee. 37
MCB	Middleware Coordination Board. 21
MIME	Multipurpose Internet Mail Extensions. 118
MITM	Man-In-The-Middle. 87
MON	gLite MONitoring system collector. 164
MoSGrid	Molecular Simulation Grid *(a Virtual Organization of EGI)*. 47
MPI	Message Passing Interface. 28
MPICH	Message Passing Interface CHameleon. 55
NAT	Network Address Translation. 208
NGI	National Grid Initiative. 16
NS	Network Server. 148
OASIS	Organization for the Advancement of Structured Information Standards. 83

ISSN: 2279–8773

Glossary and page of first occurrence

OGF Open Grid Forum. 161

OGSA Open Grid Services Architecture. 82

OGSI Open Grid Services Infrastructure. 82

OS Official Standard. 176

OSG Open Science Grid. 21

P Performance. 108

PB PetaByte. 14

PC Personal Computer. 7

PES Potential Energy Surface. 58

PF Passive Filtering. 182

PU Passive User. 186

Q2C Quality-to-Community. 208

QC Quantum Chemistry. 54

QCT Quasi-Classical-Trajectory. 125

QM Quantum Mechanics. 125

ISSN: 2279–8773

QoC Quality of Computing. 207

QoP Quality of Provider. 207

QoS Quality of Service. 2

QoU Quality of User. 2

QuB Quality Board. 196

QuM Quality Manager. 196

R-GMA Relational Grid Monitoring Architecture. 159

R&D Research and Development. 32

RAL Rutherford Appleton Laboratory. 157

RAM Random Access Memory. 115

RB Resource Broker. 150

RC Robot Certificate. 88

RESPECT Recommended External Software for EGEE CommuniTies. 28

REST REpresentational State Transfer. 80

RI Research Institution. 18

RMI Remote Method Invocation. 119

ISSN: 2279-8773

ROC	Regional Operations Center. 157
RPC	Remote Procedure Call. 81
RR	Round Robin. 113
RT	Research Teams. 18
RU	Rack Unit. 179
RUS	Record Usage Service. 169
SAAJ	Soap with Attachments Api for Java. 118
SaaS	Software-as-a-Service. 10
SATA	Serial Advanced Technology Attachment. 208
SD	Software Developer. 188
SE	Storage Element. 89
SGE	Sun Grid Engine. 160
SIMBEX	Molecular Beam EXperiments whose SImulator. 33
SLA	Service-Level Agreement. 11
SNA	Systems Network Architecture. 7
SOA	Service Oriented Architecture. 2

ISSN: 2279-8773

SOAP Simple Object Access Protocol. 38

SP Service Provider. 187

SSH Secure SHell. 213

SwA SOAP with Attachments. 118

SWaP Space, Watt and Performance. 180

T&L Teaching and Learning. 40

TAR Tape ARchive. 114

TB TeraByte. 10

TCP/IP Transmission Control Protocol / Internet Protocol. 7

TD Time Dependent. 49

TELCHEM Teaching & Learning in CHEMistry *(a Virtual Organization of EGI)*. 40

TI Time Independent. 49

toec terms of exchange credits. 39

TQ Task Queue. 151

TR ThRoughput. 197

ISSN: 2279-8773

TTR Time-To-Repair. 175

U-C Utility Computing. 9

UC User Certificate. 168

UCS User Community Service. 20

UDDI Universal Description, Discovery and Integration. 77

UI User Interface. 87

UMD Unified Middleware Distribution. 21

UNICORE UNiform Interface to COmputing REsources. 21

UR Usage Record. 161

URI Uniform Resource Identifier. 79

URL Uniform Resource Locator. 79

UTF-8 Unicode Transformation Format-8. 64

VDT Virtual Data Toolkit. 21

VO Virtual Organization. 1

VOIP Voice Over IP. 42

ISSN: 2279–8773

Glossary and page of first occurrence

VOMS	Virtual Organization Membership Service. 155
VRC	Virtual Research Community. 1
W3C	World Wide Web Consortium. 82
WM	Workload Manager. 148
WMProxy	Workload Manager Proxy. 148
WMS	Workload Manager Service. 145
WP	Working Package. 25
WSDL	Web Services Description Language. 38
WSRF	Web Services Resource Framework. 83
XML	eXtensible Markup Language. 38
YA	Yet an Applet. 88
YC	Yet a Consumer. 87
YP	Yet a Provider. 87
YR	Yet a Registry. 86

ISSN: 2279–8773

Bibliography

[1] The European Grid Initiative (EGI) Design Study, `http://web.eu-egi.eu/`

[2] C. Manuali, A. Laganà: GriF: A New Collaborative Framework for a Web Service Approach to Grid Empowered Calculations, Future Generation Computer Systems, 27(3), 315-318 (2011).

[3] C. Manuali, A. Laganà: GriF: A New Collaborative Grid Framework for SSCs, Cracow Grid Workshop (CGW'09) Proceedings, ISBN 9788361433019, 188-195 (2010).

[4] ITU-T Technology Watch Report 9: Distributed Computing: Utilities, Grids & Clouds, International Telecommunication Union (2009).

[5] Short History of Study Group 17, `http://www.itu.int/ITU-T/studygroups/com17/history.html`

[6] A. S. Tanembaum, M. Van Steen: Distributed Systems: Principles and Paradigms, Prentice Hall, ISBN 0130888931 (2001).

[7] R. J. Anderson: Security Engineering: A Guide to Building Dependable Distributed Systems, Wiley, ISBN 9780470068526 (2008).

[8] K. A. Delic, M. A. Walker: Emergence of The Academic Computing Cloud, Ubiquity 9 (31) (2008).

[9] P. T. Jaeger, J. Lin, J. M. Grimes: Cloud Computing and Information Policy: Computing in a Policy Cloud?, Journal of Information Technology & Politics, 5 (3), 269-283 (2008).

[10] The Seventh Framework Programme (FP7), `http://cordis.europa.eu/fp7/`

[11] The Business Experiments in Grid (BEinGRID) project, `http://www.beingrid.eu/`

[12] The Enabling Grids for E-sciencE (EGEE) project, `http://www.eu-egee.org/`

[13] The European DataGrid (EDG) project, `http://eu-datagrid.web.cern.ch/`

ISSN: 2279–8773

[14] The Large hadron collider Computing Grid (LCG) project, http://lcg.web.cern.ch/lcg/

[15] The Large Hadron Collider (LHC), http://lhc.web.cern.ch/lhc/

[16] The European Research Area (ERA), http://ec.europa.eu/research/era/

[17] The European Grid Initiative (EGI) Blueprint, http://www.eu-egi.eu/blueprint.pdf

[18] The NorduGrid middleware: Advanced Resource Connector (ARC), http://www.nordugrid.org/middleware/

[19] The gLite middleware, http://glite.web.cern.ch/glite/

[20] The UNICORE (Uniform Interface to Computing Resources) middleware, http://www.unicore.eu/

[21] ESFRI: the European Strategy Forum on Research Infrastructures, http://ec.europa.eu/research/infrastructures/index_en.cfm?pg=esfri

[22] The EGI-InSPIRE project (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe), http://www.egi.eu/projects/egi-inspire/

ISSN: 2279–8773

[23] The Message Passing Interface (MPI) standard, http://www.mcs.anl.gov/research/projects/mpi/

[24] The Experiment Dashboard, http://dashboard.cern.ch/

[25] Ganga: Gaudi/Athena aNd Grid Alliance, http://ganga.web.cern.ch/ganga/

[26] Diane: DIstributed ANalysis Environment, http://it-proj-diane.web.cern.ch/it-proj-diane/

[27] The RESPECT program, http://technical.eu-egee.org/index.php?id=290

[28] HYDRA homepage, https://twiki.cern.ch/twiki/bin/view/EGEE/DMEDS

[29] The Grid Relational Catalog (GRelC) project, http://grelc.unile.it/

[30] SOMA2: Open Source Molecular Modelling Workflow Environment, http://www.csc.fi/english/pages/soma

[31] Taverna Workflow Management System, http://www.taverna.org.uk/

ISSN: 2279-8773

[32] Kepler: A system for scientific workflows, http://www.gridworkflow.org/snips/gridworkflow/space/Kepler

[33] Migrating Desktop: Environment for Grid Interactive Applications, http://desktop.psnc.pl/

[34] The GridWay Metascheduler, http://www.gridway.org/

[35] COST: European Cooperation in Science and Technology, http://www.cost.esf.org/

[36] O. Gervasi, A. Laganà: SIMBEX: a portal for the a priori simulation of crossed beam experiments, Future Generation Computer Systems, 20(5), 703-715(2004).

[37] A. Laganà, A. Riganelli, O. Gervasi: On the Structuring of the Computational Chemistry Virtual Organization COMPCHEM, Lecture Notes in Computer Science, 3980, 665-674 (2006).

[38] O. Gervasi, C. Manuali, A. Laganà, A. Costantini: On the Structuring of a Molecular Simulator as a Grid Service, in Chemistry and Material science applications on Grid Infrastructures, ICTP Lecture Notes, 24, 63-82, ISBN 929500342X (2009).

ISSN: 2279–8773

[39] Web Service Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl/

[40] Simple Object Access Protocol (SOAP) 1.2, http://www.w3.org/TR/soap/

[41] Extensible Markup Language (XML), http://www.w3.org/XML/

[42] A. Laganà, A. Riganelli, C. Manuali, N. Faginas Lago, O. Gervasi, S. Crocchianti, S. Schanze: From Computer Assisted to Grid Empowered Teaching and Learning Activities in Higher Level Chemistry, in Innovative Methods of Teaching and Learning Chemistry in Higher Education, RSC Publishing, ISBN 9781847559586, 153-189 (2009).

[43] European Chemistry and Chemical Engineering and Education Network (EC2E2N), http://www.ec2e2n.net/

[44] European Chemistry Thematic Network Association, http://ectn-assoc.cpe.fr/

[45] S. Pallottelli, S. Tasso, N. Pannacci, A. Costantini, N. Faginas Lago: Distributed and Collaborative Learning Objects Repositories on Grid Networks, Lecture Notes in Computer Science, 6019, 29-40 (2010).

ISSN: 2279-8773

[46] Work Programme 2011 - Part1: Research Infrastructures, `http://cordis.europa.eu/fp7/ict/e-infrastructure/docs/wp2011.pdf`

[47] CIC Operations Portal, `http://cic.gridops.org/index.php?section=home&page=volist`

[48] OGSA-DAI homepage, `http://www.ogsadai.org.uk/`

[49] AMGA: The gLite Grid Metadata Catalogue, `http://amga.web.cern.ch/amga/`

[50] A. Laganà, A. Costantini, O. Gervasi, N. Faginas Lago, C. Manuali, S. Rampino: COMPCHEM: Progress Towards GEMS a Grid Empowered Molecular Simulator and Beyond, Journal of Grid Computing, 8(4), 571-586 (2010).

[51] A. Laganà: Supercomputer Algorithms for Reactivity, Dynamics and Kinetics of Small Molecules, Kluwer Academic Publisher, ISBN 0792302265 (1989).

[52] D. M. Hirst: A Computational Approach to Chemistry, Blackwell Scientific Publications, ISBN 0632024313 (1990).

[53] G. C. Schatz, M. A. Ratner: Quantum Mechanics in Chemistry, Prentice Hall, ISBN 0130750115 (1993).

ISSN: 2279-8773

[54] A. Laganà, A. Riganelli: Lecture Notes in Chemistry, Springer-Verlag, ISBN 3540412026 (2000).

[55] A. Laganà, G. Lendvay: Theory of Chemical Reaction Dynamics, Kluwer, ISBN 1402021656 (2004).

[56] L. Storchi, F. Tarantelli, A. Laganà: Computing Molecular Energy Surfaces on a Grid, Lecture Notes in Computer Science, 3980, 675-683 (2006).

[57] M. Verdicchio, Thesis of the Euromaster in Theoretical Chemistry and Computational Modelling, Perugia (2009); M. S. Gordon, M. W. Schmidt: Advances in electronic structure theory: GAMESS a decade later, in Theory and Applications of Computational Chemistry: the first forty years, 1167-1189, Elsevier (2005).

[58] DALTON: a Molecular Electronic Structure Program, `http://daltonprogram.org/`

[59] Molpro quantum chemistry package, `http://www.molpro.net/`

[60] L. Arteconi, A. Laganà, L. Pacifici: A Web Based Application to Fit Potential Energy Functionals to ab Initio Values, Lecture Notes in Computer Science, 3980, 694-700 (2006).

ISSN: 2279-8773

[61] A. Laganà, A. Riganelli: Computational Reaction and Molecular Dynamics: from Simple Systems and Rigorous methods to Complex Systems and Approximate Methods, Lecture Notes in Chemistry, 75, 1-10 (2000).

[62] D. Skouteris, L. Pacifici, A. Laganà: Time dependent wavepacket calculations for the N(^{4}S) + N$_2$($^1\Sigma_g^+$) system on a LEPS surface: inelastic and reactive probabilities, Molecular Physics, 102 (21-22), 2237-2248 (2004).

[63] D. Skouteris, J. F. Castillo, D. E. Manolopulos: ABC: a quantum reactive scattering program, Computer Physics Communications, 133, 128-135 (2000).

[64] P. Casavecchia, N. Balucani, G. G. Volpi: The Chemical Dynamics and Kinetics of Small Radicals, Adv. Ser. in Physical Chemistry, 6 (8), World scientific, 1995, chapter 8.

[65] java.com: Java + You, `http://www.java.com/`

[66] Questionmark homepage, `http://www.questionmark.com/`

[67] UTF-8 and Unicode Standard, `http://www.utf8.com/`

[68] The EChemTest Testing Center of Perugia, `http://www.ectn-assoc.org/echemtest/tc/IT_UPerugia.htm`

ISSN: 2279-8773

[69] SQL Server 2008 Overview, http://www.microsoft.com/sqlserver/2008/en/us/default.aspx

[70] SOA and Web Services, http://www.oracle.com/technetwork/articles/javase/index-142519.html

[71] Universal Description, Discovery and Integration (UDDI) 3.0.2, http://www.oasis-open.org/specs/ (2005)

[72] Axis User's Guide, http://ws.apache.org/axis/java/user-guide.html

[73] Apache Tomcat, http://tomcat.apache.org/

[74] Invoking Web Services with Java clients, http://www.ibm.com/developerworks/webservices/library/ws-javaclient/index.html

[75] From SOA to REST: Designing and Implementing RESTful Services, http://dret.net/netdret/docs/soa-rest-www2009/

[76] The Apache Software Foundation, http://www.apache.org

[77] HTTP: HyperText Transfer Protocol Overview, http://www.w3.org/Protocols/

[78] World Wide Web consortium (W3C), http://www.w3.org/

ISSN: 2279-8773

[79] I. Foster, C. Kesselman, J. M. Nick, S. Tuecke: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Open Grid Service Infrastructure WG, Global Grid Forum (GGF), http://www.globus.org/alliance/publications/papers/ogsa.pdf (2002).

[80] The Global Grid Forum (GGF), http://www.ggf.org/

[81] What is a Grid Service?, http://gdp.globus.org/gt3-tutorial/multiplehtml/ch01s03.html

[82] Web Service Resource Framework (WSRF), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

[83] OASIS: Advancing Open Standards for the Global Information Society, http://www.oasis-open.org/

[84] GriF: The Grid Framework, http://www.hpc.unipg.it/grif/

[85] INFN Certification Authority, http://security.fi.infn.it/CA/docs/

[86] Apache jUDDI, http://ws.apache.org/juddi/

ISSN: 2279-8773

[87] UDDI4J, http://uddi4j.sourceforge.net/

[88] Welcome to Scientific Linux (SL), https://www.scientificlinux.org/

[89] MySQL: The world's most popular open source database, http://www.mysql.com/

[90] Job Description Language (JDL) Reference, http://www-numi.fnal.gov/offline_software/srt_public_context/GridTools/docs/jobs_jdl.html

[91] LCG File Catalog (LFC) administrators' guide, https://twiki.cern.ch/twiki/bin/view/LCG/LfcAdminGuide

[92] BASH - GNU Project - Free Software Foundation (FSF), http://www.gnu.org/software/bash/

[93] Introduction to UNIX cron and at Utilities, http://www.rahul.net/raithel/MyBackPages/crontab.html

[94] Grid: The GLUE Schema, http://www-numi.fnal.gov/offline_software/srt_public_context/GridTools/docs/glue_schema.html

[95] What is Round-Robin Scheduling?, http://www.wisegeek.com/what-is-round-robin-scheduling.htm

ISSN: 2279-8773

[96] SourceForge.net: GriF (Grid Framework), http://sourceforge.net/projects/gridframework/

[97] Freshmeat.net: GriF, http://freshmeat.net/projects/grif

[98] SOAP Messages with Attachments, http://www.w3.org/TR/SOAP-attachments (2000).

[99] R. Monson-Haefel: J2EE Web Services, Addison-Wesley, ISBN 9780321146182 (2004).

[100] RFC 2111: Content-ID and Message-ID Uniform Resource Locators, http://www.ietf.org/rfc/rfc2111.txt

[101] B. Liu, P. E. M. Siegbahn: An accurate three-dimensional potential energy surface for H_3, Journal of Chemical Physics, 68, 2457-2465 (1978).

[102] D. G. Truhlar, C. J. Horowitz: *Erratum*: Functional representation of Liu and Siegbahn's accurate ab initio potential energy calculations of $H + H_2$, Journal of Chemical Physics, 71, 1514 (1979).

[103] Parametric Jobs - EGEE SEE ROC Wiki, http://wiki.egee-see.org/index.php/Parametric_Jobs

ISSN: 2279-8773

[104] A. Laganà, E. Garcia, L. Ciccarelli: Deactivation of Vibrationally Excited Nitrogen Molecules by Collision with Nitrogen Atoms, Journal of Chemical Physics, 91(2), 312-214 (1987).

[105] E. Garcia, A. Laganà: The largest angle generalization of the rotating bond order potential: the H + H_2 and N + N_2 reactions, Journal of Chemical Physics, 103(13), 5410-5416 (1995).

[106] E. Garcia, A. Saracibar, S. Gómez-Carrasco, A. Laganà: Modeling the global potential energy surface of the N + N_2 reaction from ab initio data, , 10(18), 2552-2558 (2008).

[107] E. Garcia, A. Saracibar, A. Laganà, D. Skouteris: A detailed comparison of Centrifugal sudden and J shift estimates of the reactive properties of the N + N_2 reaction, Physical Chemistry Chemical Physics, 11, 11456-11462, (2009).

[108] C. Manuali, A. Laganà, S. Rampino: GriF: A Grid Framework for a Web Service Approach to Reactive Scattering, Computer Physics Communications, 181, 1179-1185 (2010).

[109] AliEn: ALICE Environment, `http://alien2.cern.ch/`

[110] The Condor Project Homepage, `http://www.cs.wisc.edu/condor/`

ISSN: 2279-8773

[111] CESGA: Centro de Supercomputación de Galicia, http://www.cesga.es/index.php?lang=en

[112] EGI Accounting Portal, http://www3.egee.cesga.es/

[113] Rutherford Appleton Laboratory (RAL), http://www.stfc.ac.uk/About+STFC/51.aspx

[114] Istituto Nazionale di Fisica Nucleare (INFN), http://www.infn.it/indexen.php

[115] DGAS: Distributed Grid Accounting System, http://www.to.infn.it/dgas/

[116] Relational Grid Monitoring Architecture (R-GMA), http://www.r-gma.org/

[117] Accounting Processor for Event Logs (APEL), http://goc.grid.sinica.edu.tw/gocwiki/ApelHome

[118] Sun Grid Engine (SGE), http://gridengine.sunsource.net/

[119] Cluster Resources (Torque and Maui), http://www.clusterresources.com/

[120] E. Freire, A. Simon, J. Lopez, C. Fernandez, R. Diez, S. Diaz, C. Manuali, A. Laganà: Application Domain Accounting

ISSN: 2279-8773

for EGI, 5th EGEE User Forum, Uppsala (SW), April 12-15, http://egee-uf5.eu-egee.org/ (2010).

[121] The Open Grid Forum (OGF), http://www.gridforum.org/

[122] Usage Record WG (UR-WG), https://forge.gridforum.org/sf/projects/ur-wg

[123] The EGEE/gLite Official Documentation, http://glite.web.cern.ch/glite/documentation/

[124] R. Mach, R. Lepro-Metz, S. Jackson: Usage Record (UR) Format Recommendation, http://www.ogf.org/documents/GFD.98.pdf (2006).

[125] Unix Signals, http://www.cs.pitt.edu/~alanjawi/cs449/code/shell/UnixSignals.htm

[126] The Distributed Infrastructure with Remote Agent Control (DIRAC) project, http://lhcbweb.pic.es/DIRAC/info/general/diracOverview

[127] The Large Hadron Collider beauty experiment (LHCb) Virtual Organization, http://lhcb.web.cern.ch/lhcb/

ISSN: 2279-8773

[128] The Load Sharing Facility (LSF) platform, http://www.platform.com/workload-management/high-performance-computing/

[129] K. Karta: An Investigation on Personalized Collaborative Filtering for Web Service, Honours Programme of the School of Computer Science and Software Engineering, University of Western Australia (2005).

[130] A. Mani, A. Nagarajan: Understanding Quality of Service for Web Services, http://www.ibm.com/developerworks/webservices/library/ws-quality.html (2002).

[131] The First International Green Computing Conference (IGCC'10): Sustainable Computing and Computing for Sustainability, http://www.green-conf.org/

[132] S. Murugesan: Harnessing Green IT: Principles and Practices, IEEE IT Professional, 24-33 (2008).

[133] The Linpack Benchmark, http://www.top500.org/project/linpack

[134] The Green500 List: Environmentally Responsible Supercomputing, http://www.green500.org

ISSN: 2279-8773

[135] R. Ge, X. Feng, H. Pyla, K. Cameron, W. Feng: Power Measurement Tutorial for the Green500 List, `http://www.green500.org/docs/pubs/tutorial.pdf` (2007).

[136] C. Manuali, A. Laganà: GriF: Empowering Scientific Calculations on the Grid, International Workshop on Science Gateways (IWSG 2010) Proceedings, (in press).

[137] T. Erl: Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall, ISBN 0131858580 (2008).

[138] Grids in Data Warehousing, `http://www.tdan.com/view-articles/9378`

[139] Grid Enabled Molecular Science Through Online Networked Environments (GEMSTONE), `http://gemstone.mozdev.org/`

[140] Network Address Translation (NAT), `http://www.webopedia.com/TERM/N/NAT.html`

[141] NIC (or Channel) Bonding, `http://www.webopedia.com/TERM/N/NIC_bonding.html`

[142] HPL: A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers, `http://www.netlib.org/benchmark/hpl/`

ISSN: 2279-8773

[143] The Intel® MPI Library, http://software.intel.com/en-us/intel-mpi-library/

[144] An interview with the GriF Manager, http://download.famouswhy.com/publisher/compchem/

ISSN: 2279-8773

Appendix A

The New Socialism

Bill Gates once derided open source advocates with the worst epithet a capitalist can muster. These folks, he said, were a "new modern-day sort of communists", a malevolent force bent on destroying the monopolistic incentive that helps support the American dream. Gates was wrong: Open source zealots are more likely to be libertarians than commie pinkos. Yet there is some truth to his allegation. The frantic global rush to connect everyone to everyone, all the time, is quietly giving rise to a revised version of socialism. Communal aspects of digital culture run deep and wide. Wikipedia is just one remarkable example of an emerging collectivism and not just Wikipedia but wikiness at large. Similar developments suggest a steady move toward a sort of socialism uniquely tuned for a networked world. We're not talking about your grandfather's socialism. In fact, there is a long list of past movements this new socialism is not. It is not class warfare. It is not anti-American; indeed, digital socialism may be the newest innovation. While old-school socialism was an arm of the state, digital socialism is socialism without the state. This new brand of socialism currently operates in the realm of culture and economics, rather than government (for now). The type of communism with which Gates hoped to tar the creators of Linux was born in an era of enforced borders, centralized communications and top-heavy industrial processes. Those constraints gave rise to a type of collective ownership that replaced the brilliant chaos of a free market with scientific five-year plans devised by an all-powerful politburo. This political operating system failed, to put it mildly. However, unlike those older strains of red-flag socialism, the new socialism runs over a borderless Internet, through a tightly integrated global economy. It is designed to heighten individual autonomy and thwart centralization. It is decentralization extreme Instead of gathering on collective farms, we gather in collective worlds. Instead of state factories, we have desktop factories connected to virtual co-ops. Instead of sharing drill bits, picks and shovels, we share apps, scripts and APIs. Instead of faceless politburos, we have faceless meritocracies, where the only thing that matters is getting things done. Instead of national production, we have peer production. Instead of government rations and subsidies, we have a bounty of free goods. I recognize that the word socialism is bound to make many readers twitch. It carries tremendous cultural baggage, as do the related terms communal, communitarian and collective. I use socialism because technically it is the best word to indicate a range of technologies that rely for their power on social interactions. Broadly, collective action is what Web sites and Net-connected apps generate when they harness input from the global audience. Of course, there's rhetorical danger in lumping so many types of organization under such an inflammatory heading. But there are no unsoiled terms available, so we might as well redeem this one. When masses of people who own the means of production work toward a common goal and share their products in common, when they contribute labor without wages and enjoy the fruits free of charge, it's not unreasonable to call that socialism. In the late '90s, activist, provocateur and aging hippy John Barlow began calling this drift, somewhat tongue in cheek, "dot-communism." He defined it as a "workforce composed entirely of free agents," a decentralized gift or

ISSN: 2279-8773

barter economy where there is no property and where technological architecture defines the political space. He was right on the virtual money. But there is one way in which socialism is the wrong word for what is happening: It is not an ideology. It demands no rigid creed. Rather, it is a spectrum of attitudes, techniques and tools that promote collaboration, sharing, aggregation, coordination, ad hocracy and a host of other newly enabled types of social cooperation. It is a design frontier and a particularly fertile space for innovation. A useful hierarchy for sorting through these new social arrangements (for which, at each step, the amount of coordination increases) can be suggested. Groups of people start off simply **sharing** and then progress to **cooperation**, **collaboration** and finally **collectivism**.

I. SHARING (user information):

The online masses have an incredible willingness to share. The number of personal photos posted on Facebook and MySpace is astronomical, but it's a safe bet that the overwhelming majority of photos taken with a digital camera are shared in some fashion. Then there are status updates, map locations, half-thoughts posted online. Add to this the 6 billion videos served by YouTube each month in the US alone and the millions of fan-created stories deposited on fanfic sites. Sharing is the mildest form of socialism, but it serves as the foundation for higher levels of communal engagement.

II. COOPERATION (data aggregation):

When individuals work together toward a large-scale goal, it produces results that emerge at the group level. Thousands of aggregator sites employ the same social dynamic for threefold benefit. First, the technology aids users directly, letting them tag, bookmark, rank and archive for their own use. Second, other users benefit from an individual's tags,

ISSN: 2279–8773

bookmarks and so on. And this, in turn, often creates additional value that can come only from the group as a whole. In a curious way, this proposition exceeds the socialist promise of ”from each according to his ability, to each according to his needs” because it betters what you contribute and delivers more than you need. Community aggregators can unleash astonishing power. That is the whole point of social institutions: the sum outperforms the parts. Traditional socialism aimed to ramp up this dynamic via the state. Now, decoupled from government and hooked into the global digital matrix, this elusive force operates at a larger scale than ever before.

III. COLLABORATION (development):

Organized collaboration can produce results beyond the achievements of ad hoc cooperation. Just look at any of hundreds of open source software projects. In these endeavours, finely tuned communal tools generate high-quality products from the coordinated work of thousands or tens of thousands of members. In contrast to casual cooperation, collaboration on large, complex projects tends to bring the participants only indirect benefits, since each member of the group interacts with only a small part of the end product. An enthusiast may spend months writing code for a subroutine when the program's full utility is several years away. In fact, the work-reward ratio is so out of kilter from a free-market perspective (the workers do immense amounts of high-market-value work without being paid) that these collaborative efforts make no sense within capitalism. Adding to the economic dissonance, we've become accustomed to enjoying the products of these collaborations free of charge. Instead of money, the peer producers who create the stuff gain credit, status, reputation, enjoyment, satisfaction and experience. Not only is the product free, it can be copied freely and used as the basis for new products. Alternative schemes for managing intellectual property, were invented to ensure these ”frees.” Of course, there's nothing

ISSN: 2279-8773

particularly socialistic about collaboration per se. But the tools of online collaboration support a communal style of production that shuns capitalistic investors and keeps ownership in the hands of the workers and to some extent those of the consuming masses.

IV. COLLECTIVISM (government):

While cooperation can write an encyclopedia, no one is held responsible if the community fails to reach consensus and lack of agreement doesn't endanger the enterprise as a whole. The aim of a collective, however, is to engineer a system where self-directed peers take responsibility for critical processes and where difficult decisions, such as sorting out priorities, are decided by all participants. Throughout history, hundreds of small-scale collectivist groups have tried this operating system. The results have not been encouraging. Indeed, a close examination of the governing kernel of, say, Wikipedia, or even Linux, shows that these efforts are further from the collectivist ideal than appears from the outside. A vast army of contributions is managed by a much smaller group of coordinators. This isn't necessarily a bad thing. Some types of collectives benefit from hierarchy while others are hurt by it. Platforms like the Internet and Facebook, or democracy (which are intended to serve as a substrate for producing goods and delivering services) benefit from being as nonhierarchical as possible, minimizing barriers to entry and distributing rights and responsibilities equally. When powerful actors appear, the entire fabric suffers. On the other hand, organizations built to create products often need strong leaders and hierarchies arranged around time scales: one level focuses on hourly needs, another on the next five years. In the past, constructing an organization that exploited hierarchy yet maximized collectivism was nearly impossible. Now digital networking provides the necessary infrastructure. The Net empowers product-focused organizations to function collectively while keeping the hierarchy from fully taking over. The elite core we find at

ISSN: 2279–8773

the heart of online collectives is actually a sign that stateless socialism can work on a grand scale. Most people in the world, including myself, were indoctrinated with the notion that extending the power of individuals necessarily diminishes the power of the state and vice versa. In practice, though, most polities socialize some resources and individualize others. Most free-market economies have socialized education and even extremely socialized societies allow some private property. Rather than viewing technological socialism as one side of a zero-sum trade-off between free-market individualism and centralized authority, it can be seen as a cultural OS that elevates both the individual and the group at once. The largely unarticulated but intuitively understood goal of communitarian technology is this: to maximize both individual autonomy and the power of people working together.

Thus, **digital socialism** can be viewed as a third way that renders irrelevant the old debates. The new OS is neither the classic communism of centralized planning without private property nor the undiluted chaos of a free market. Instead, it is an emerging design space in which decentralized public coordination can solve problems and create things that neither pure communism nor pure capitalism can. Hybrid systems that blend market and non-market mechanisms are not new. For decades, researchers have studied the decentralized, socialized production methods but only since the arrival of low-cost, instantaneous, ubiquitous collaboration has it been possible to migrate the core of those ideas into diverse new realms, like writing enterprise software or reference books. The number of people who make things for free, share things for free, use things for free, belong to collective software farms, work on projects that require communal decisions, or experience the benefits of decentralized socialism has reached millions and counting. Revolutions have grown out of much smaller numbers. On the face of it, one might expect a lot of political posturing from folks who are constructing an alternative to capitalism and corporatism. But the

ISSN: 2279–8773

coders, hackers and programmers who design sharing tools don't think of themselves as revolutionaries. No new political party is being organized in conference rooms. Indeed, the leaders of the new socialism are extremely pragmatic. The most common motivation was "to learn and develop new skills". That's practical. One academic put it this way (paraphrasing): The major reason for working on free stuff is to improve my own damn software. Basically, overt politics is not practical enough. But the rest of us may not be politically immune to the rising tide of sharing, cooperation, collaboration and collectivism. How close to a non-capitalistic, open source, peer-production society can this movement take us? Every time that question has been asked, the answer has been: closer than we thought. A similar thing happened with free markets over the past century. Every day, someone asked: What can't markets do? We took a long list of problems that seemed to require rational planning or paternal government and instead applied marketplace logic. In most cases, the market solution worked significantly better. Much of the prosperity in recent decades was gained by unleashing market forces on social problems. Now we're trying the same trick with collaborative social technology, applying digital socialism to a growing list of wishes (and occasionally to problems that the free market couldn't solve) to see if it works. So far, the results have been startling.

At nearly every turn, the power of sharing, cooperation, collaboration, openness, free pricing and transparency has proven to be more practical than we capitalists thought possible. Each time we try it, we find that the power of the new socialism is bigger than we imagined. We underestimate the power of our tools to reshape our minds. Did we really believe we could collaboratively build and inhabit virtual worlds all day, every day and not have it affect our perspective? The force of online socialism is growing. Its dynamic is spreading

ISSN: 2279-8773

Virt&l–Comm.3.2012.22

beyond electrons, perhaps into elections.

```
http://www.wired.com/culture/culturereviews/magazine/17-06/nep_newsocialism
```

ISSN: 2279–8773

Appendix B

YR Code

B1. JUDDI Database Data.

```
mysql> select name, descr from SERVICE_NAME, SERVICE_DESCR;
+------------+----------------------------------------------------------+
| name       | descr
+------------+----------------------------------------------------------+
| GriF 1.0   | Running ABC Application (User Driven mode) on the EGI Grid |
| GriF 2.0GP | Running General Purpose (GP) Applications on the EGI Grid  |
+------------+----------------------------------------------------------+

mysql> select name, descr from BUSINESS_NAME, BUSINESS_DESCR;
+------------+----------------------------------------------------------+
| name       | descr
+------------+----------------------------------------------------------+
| COMPCHEM   | The Computational Chemistry Virtual Organization of EGI    |
+------------+----------------------------------------------------------+

mysql> select SERVICE_KEY, ACCESS_POINT_TYPE, ACCESS_POINT_URL from BINDING_TEMPLATE;
+--------------+-------------------+------------------------------+
| SERVICE_KEY  | ACCESS_POINT_TYPE | ACCESS_POINT_URL
+--------------+-------------------+------------------------------+
| <KEY_1>      | http              | http://gw2-hpc.chm.unipg.it:8080/ |
| <KEY_2>      | http              | http://gw2-hpc.chm.unipg.it:8080/ |
+--------------+-------------------+------------------------------+
```

Appendix B

B2. YC Code to query YR in order to retrieve YP addresses offering a given Grid Service.

```
import org.uddi4j.client.UDDIProxy;
import org.uddi4j.datatype.Name;
import org.uddi4j.datatype.binding.AccessPoint;
import org.uddi4j.datatype.binding.BindingTemplate;
import org.uddi4j.response.BindingDetail;
import org.uddi4j.response.ServiceInfo;
import org.uddi4j.response.ServiceList;
import org.uddi4j.util.FindQualifier;
import org.uddi4j.util.FindQualifiers;

[...]

private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        UDDIProxy proxy = new UDDIProxy();
        proxy.setInquiryURL(inquiryURL);
        proxy.setPublishURL(publishURL);
        Vector<Object> names = new Vector<Object>();
        names.add(new Name(keyword));
        FindQualifiers findQualifiers = new FindQualifiers();
        Vector<Object> qualifier = new Vector<Object>();
        qualifier.add(new FindQualifier("caseSensitiveMatch"));
        findQualifiers.setFindQualifierVector(qualifier);
        ServiceList serviceList = proxy.find_service(businessKey, names, null, null, \\
            findQualifiers, 5);
        Vector serviceInfoVector = serviceList.getServiceInfos().getServiceInfoVector();
        for (int i = 0; i < serviceInfoVector.size(); i++) {
            ServiceInfo serviceInfo = (ServiceInfo) serviceInfoVector.elementAt(i);
            serviceKey = serviceInfo.getServiceKey();
            BindingDetail bindingDetail = proxy.find_binding(findQualifiers, \\
                serviceKey, null, 5);
            Vector bindingTemplateVector = bindingDetail.getBindingTemplateVector();
            for (int j = 0; j < bindingTemplateVector.size(); j++) {
                BindingTemplate bindingTemplate = (BindingTemplate) \\
                    bindingTemplateVector.elementAt(j);
                AccessPoint accessPoint = bindingTemplate.getAccessPoint();
                addr = accessPoint.getText();
```

ISSN: 2279-8773

```
                host = addr.split("/");
                logPane.append("\n-> " + host[2] + "\n");
                logPane.setCaretPosition(logPane.getText().length() - 1);
            }
        }
    } catch (Exception e) {};
}
```

ISSN: 2279-8773

Appendix C

YP Web Services Code

C1. GriF Web Services Security.

```
[...]
Class.forName("com.mysql.jdbc.Driver").newInstance();
cn = DriverManager.getConnection("jdbc:mysql://localhost/" + DBname + "?user=" + \\
        DBuser + "&password=" + DBpwd);
String query = "select `key`, passwd from vousers where username = '" + username + "'";
Statement stmt = cn.createStatement();
ResultSet rs = stmt.executeQuery(query);
v = new Vector<String[]>();
ResultSetMetaData rsmd = rs.getMetaData();
columns = rsmd.getColumnCount();while (rs.next()) {
        record = new String[columns];
        for (int i = 0; i < columns; i++) record[i] = rs.getString(i + 1); \\
                v.add(record);
}
if (key.equals(v.elementAt(0)[0]) && pwd.equals(v.elementAt(0)[1])) {
        // START THE REAL WEB SERVICE
}
```

ISSN: 2279-8773

Virt&l-Comm.3.2012.22

Appendix C

C2. DB.jws: The 'Select' operation.

```
[...]
query = "Insert into jobs(username, service, yp, url_job, sub_date, uri, application, \\
        input, exitstatus, retrieved, status, queue, wall_time, job_type, description, \\
        cpu_time, lfc_guid, deleted, cleared, mem) values('"+username+"','"+service+ \\
        "','"+yp+"','"+url_job+"','"+sub_date+"','"+uri+"','"+application+"','"+input+ \\
        "','2','N','Submitted','"+queue+"','-1','"+job_type+"','"+description+ \\
        "','-1','','N','N','-1')";
try {
        stmt = cn.createStatement();
        stmt.executeUpdate(query);
        stmt.close();
}
```

C3. DB.jws: The 'Insert' operation.

```
[...]
query = "Select sub_date, url_job, uri, status, description from jobs where \\
        username='"+username+"' and retrieved='N' and status !='Cancelled' \\
        order by status asc, sub_date desc";
try {
        stmt = cn.createStatement();
        rs = stmt.executeQuery(query);
        v = new Vector<String[]>();
        rsmd = rs.getMetaData();
        columns = rsmd.getColumnCount();
        while (rs.next()) {
                record = new String[columns];
                for (int i = 0; i < columns; i++) record[i] = rs.getString(i + 1);
                v.add(record);
        }
        rs.close();
        stmt.close();
}
```

ISSN: 2279-8773

Appendix C

C4. DB.jws: The 'Delete' operation.

```
[...]
query = "Update jobs set status='Cancelled' where username = '"+username+"' and \\
        url_job = '"+url_job+"'";
try {
        stmt = cn.createStatement();
        stmt.executeUpdate(query);
        stmt.close();
}
```

C5. The RunGP.jws Web Service of GriF.

```
import java.io.*;
import java.lang.*;
import java.sql.*;
import java.util.Vector;
import javax.activation.*;

public class RunGP { public String run(String username, String pwd, String applname, \\
        DataHandler dh, String date, String key, String WSname, String paradigm, \\
        String userinputname, String method) {
        String res = "";
        Connection cn = null;
        Vector<String[]> v = null;
        String DBname = "grif-db";
        String DBuser = "<db_user>";
        String DBpwd = "<db_pwd>";
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            cn = DriverManager.getConnection("jdbc:mysql://localhost/" + DBname + \\
                    "?user=" + DBuser + "&password=" + DBpwd);
            String[] record;
            String query = "select `key`, passwd from vousers where username = '" + \\
                    username + "'";
            int columns = 0;
            Statement stmt = cn.createStatement();
            ResultSet rs = stmt.executeQuery(query);
```

ISSN: 2279-8773

```
v = new Vector<String[]>();
ResultSetMetaData rsmd = rs.getMetaData();
columns = rsmd.getColumnCount();
while (rs.next()) {
    record = new String[columns];
    for (int i = 0; i < columns; i++) record[i] = rs.getString(i + 1);
    v.add(record);
}
rs.close();
stmt.close();
if (key.equals(v.elementAt(0)[0]) && pwd.equals(v.elementAt(0)[1])) {
    if (method.equals("STANDARD")) {
        query = "select filename from applications where name = '" + \\
                applname + "'";
        columns = 0;
        stmt = cn.createStatement();
        rs = stmt.executeQuery(query);
        v = new Vector<String[]>();
        rsmd = rs.getMetaData();
        columns = rsmd.getColumnCount();
        while (rs.next()) {
            record = new String[columns];
            for (int i = 0; i < columns; i++) record[i] = \\
                    rs.getString(i + 1);
            v.add(record);
        }
        rs.close();
        stmt.close();
        applname=v.elementAt(0)[0];
    }
    String input = dh.getName();
    String output = "/tmp/GRID/GP-" + username + "-" + date + ".log";
    String cmd = "/var/lib/tomcat5/programs/GP/rungp.sh " + username + \\
            " " + input + " " + applname + " " + date + " " + paradigm + \\
            " " + userinputname + " " + method + " > " + output + \\
            " 2>/dev/null";
    String[] command = {"sh", "-c", cmd};
    Process p = null;
    try {
        Runtime r = Runtime.getRuntime();
        p = r.exec(command);
        p.waitFor();
```

ISSN: 2279-8773

```
                }
                catch (Exception e) {
                    System.err.println(e.toString());
                }
                try {
                    String line = "";
                    FileReader file = new FileReader(output);
                    BufferedReader reader = new BufferedReader(file);
                    while ((line = reader.readLine()) != null) {
                        res += line + "\n";
                    }
                } catch (FileNotFoundException e) { throw \\
                        new RuntimeException("File not found."); }
                catch (IOException e) { throw \\
                        new RuntimeException("IO Error occurred."); }
                File deloutput = new File(output);
                deloutput.delete();
            } else res = "NOT OK";
        } catch(Exception e) { System.err.println("Exception: " + e.getMessage()); }
        finally {
            try {
                if (cn != null) cn.close();
            } catch(SQLException e) {}
        }
        return res;
    }
}
```

ISSN: 2279-8773

Appendix D

YP Scripts Code

D1. The rungp.sh Shell Script of GriF.

```
#!/bin/bash

# Configuration
USER="$1"
INPUT="$2"
APPLNAME="$3"
DATE="$4"
PARADIGM="$5"
USERINPUTNAME="$6"
METHOD="$7"
SSH="/usr/bin/ssh"
SCP="/usr/bin/scp"
UI="ui.grid.unipg.it"
DBNAME="grif-db"
DBUSER="<db_user>"
DBPWD="<db_pwd>"
MYSQL="mysql -u $DBUSER --password=$DBPWD -D $DBNAME -e"
TMP_DIR="/tmp/GRID"
APPS_DIR="/var/lib/tomcat5/programs/GP/APPLICATIONS"
GP_JDL="GP-$USER-$DATE.jdl"
GP_SH="GP-$USER-$DATE.sh"
GRID_RUN_PORT="9000"
```

Appendix D

```
GRID_BIN="/opt/glite/bin"
UIHOME="/home"
TMP_DIR="/tmp/GRID"
VO="compchem"
LFCUSER="carlo"
ROBOTUSER="carlo"

# Full Ranking
if [[ "$PARADIGM" == "1" ]]; then
        QUEUE=`$MYSQL "SELECT name from queues where ranking IS NOT NULL \\
                order by rr, ranking limit 1;"`
        QUEUE=`echo $QUEUE |awk '{print $2}'`
        REQUIREMENTS="RegExp(\"$QUEUE\", other.GlueCEUniqueId);"
        $MYSQL "UPDATE queues set rr=rr+1 where name='$QUEUE';"
fi

# On-line Ranking (ensuring 3 running days)
if [[ "$PARADIGM" == "2" ]]; then

        # Determine BAD CE queues (all jobs run on these CE queues have been failed)
        allbad=`$MYSQL "SELECT distinct queue from jobs where exitstatus ='1';"`
        allbad=`echo $allbad |awk '{for (i=2; i<=NF; i++) print $i}'`
        for i in $allbad; do
                njobs=`$MYSQL "SELECT count(id) from jobs where queue='$i';"`
                njobs=`echo $njobs |awk '{print $2}'`
                nerr=`$MYSQL "SELECT count(id) from jobs where exitstatus='1' and \\
                        queue='$i';"`
                nerr=`echo $nerr |awk '{print $2}'`
                if [[ $njobs -eq $nerr ]]; then
                        BAD=$BAD"(!(RegExp(\"$i\", other.GlueCEUniqueId))) && "
                fi
        done
        REQUIREMENTS=$BAD"(other.GlueCEPolicyMaxWallClockTime > 4319) && \\
                (other.GlueCEStateFreeCPUs > 0);"
        RANK="(-other.GlueCEStateWaitingJobs);"
fi

# On-line Ranking (ensuring 3 GB Ram)
if [[ "$PARADIGM" == "3" ]]; then

        # Determine BAD CE queues (all jobs run on these CE queues have been failed)
        allbad=`$MYSQL "SELECT distinct queue from jobs where exitstatus ='1';"`
```

ISSN: 2279-8773

```
        allbad=`echo $allbad |awk '{for (i=2; i<=NF; i++) print $i}'`
        for i in $allbad; do
                njobs=`$MYSQL "SELECT count(id) from jobs where queue='$i';"`
                njobs=`echo $njobs |awk '{print $2}'`
                nerr=`$MYSQL "SELECT count(id) from jobs where exitstatus='1' and \\
                        queue='$i';"`
                nerr=`echo $nerr |awk '{print $2}'`
                if [[ $njobs -eq $nerr ]]; then
                        BAD=$BAD"(!(RegExp(\"$i\", other.GlueCEUniqueId))) && "
                fi
        done
        REQUIREMENTS=$BAD"(other.GlueHostMainMemoryRAMSize > 4095) && \\
                (other.GlueCEStateFreeCPUs > 0);"
        RANK="(-other.GlueCEStateWaitingJobs);"
fi

# Main Program

$SSH $ROBOTUSER@$UI "mkdir $USER 2>/dev/null"

# Copy to UI the already available application selected by the user
if [[ "$METHOD" == "STANDARD" ]]; then
        $SCP $APPS_DIR/$APPLNAME $ROBOTUSER@$UI:$USER/GriF-$USER-$DATE
        APPLNAME="GriF-"$USER"-"$DATE
fi

# Managing Input
if [[ "$USERINPUTNAME" == ".TGZ" ]]; then
        # Multiple (.tar.gz)
        mkdir $TMP_DIR/$USER-M_INPUT/
        cp -f $INPUT $TMP_DIR/$USER-M_INPUT/
        cd $TMP_DIR/$USER-M_INPUT/
        list=`tar -zxvf $INPUT |awk -F"/" '{print $NF}'`
        dir=`tar -ztf $INPUT |awk -F"/" '{for (i=1; i<NF; i++) print $i}' |uniq`
        cd $TMP_DIR/$USER-M_INPUT/$dir
        ISB="{\"$UIHOME/$ROBOTUSER/$USER/$GP_SH\",\"$UIHOME/$ROBOTUSER/$USER/$APPLNAME\""
        for i in $list; do
                $SCP $i $ROBOTUSER@$UI:$USER/
                ISB=$ISB", \"$UIHOME/$ROBOTUSER/$USER/$i\""
        done
        ISB=$ISB"};"
        rm -rf $TMP_DIR/$USER-M_INPUT/
```

ISSN: 2279-8773

```
elif [[ "$USERINPUTNAME" == ".ZIP" ]]; then
        # Multiple (.zip)
        mkdir $TMP_DIR/$USER-M_INPUT/
        cp -f $INPUT $TMP_DIR/$USER-M_INPUT/
        cd $TMP_DIR/$USER-M_INPUT/
        list=`unzip $INPUT |grep -v Archive |awk '{print $2}' |awk -F"/" '{print $NF}'`
        dir=`unzip -lv $INPUT |grep % |awk '{print $8}' |awk -F"/" \\
                '{for (i=1; i<NF; i++) print $i}' |uniq`
        cd $TMP_DIR/$USER-M_INPUT/$dir
        ISB="{\"$UIHOME/$ROBOTUSER/$USER/$GP_SH\",\"$UIHOME/$ROBOTUSER/$USER/$APPLNAME\""
        for i in $list; do
                $SCP $i $ROBOTUSER@$UI:$USER/
                ISB=$ISB", \"$UIHOME/$ROBOTUSER/$USER/$i\""
        done
        ISB=$ISB"};"
        rm -rf $TMP_DIR/$USER-M_INPUT/
else
        # Single
        $SCP $INPUT $ROBOTUSER@$UI:$USER/$APPLNAME.input
fi

# Create JDL file at Runtime
echo "Type = \"Job\";" > $TMP_DIR/$GP_JDL
echo "JobType = \"Normal\";" >> $TMP_DIR/$GP_JDL
echo "MyProxyServer = \"myproxy.cnaf.infn.it\";" >> $TMP_DIR/$GP_JDL
echo "Executable = \"$GP_SH\";" >> $TMP_DIR/$GP_JDL
echo "StdOutput = \"gp.log\";" >> $TMP_DIR/$GP_JDL
echo "StdError = \"gp.err\";" >> $TMP_DIR/$GP_JDL
if [[ "$USERINPUTNAME" == ".TGZ" || "$USERINPUTNAME" == ".ZIP" ]]; then
        echo "InputSandbox = $ISB" >> $TMP_DIR/$GP_JDL
else
        echo "InputSandbox = {\"$UIHOME/$ROBOTUSER/$USER/$GP_SH\", \\
                \"$UIHOME/$ROBOTUSER/$USER/$APPLNAME\", \\
                \"$UIHOME/$ROBOTUSER/$USER/$APPLNAME.input\"};" >> $TMP_DIR/$GP_JDL
fi
echo "OutputSandbox  = {\"gp.log\", \"gp.err\"};" >> $TMP_DIR/$GP_JDL
echo "Requirements = $REQUIREMENTS" >> $TMP_DIR/$GP_JDL
if [[ "$PARADIGM" == "2" || "$PARADIGM" == "3" ]]; then echo "Rank = $RANK" \\
        >> $TMP_DIR/$GP_JDL; fi
echo "PerusalFileEnable = true;" >> $TMP_DIR/$GP_JDL
echo "RetryCount = 0;" >> $TMP_DIR/$GP_JDL
echo "ShallowRetryCount = 3;" >> $TMP_DIR/$GP_JDL
```

ISSN: 2279-8773

Appendix D

```
# Upload JDL to UI and delete from YP
$SCP $TMP_DIR/$GP_JDL $ROBOTUSER@$UI:$USER/$GP_JDL
rm -f $TMP_DIR/$GP_JDL

# Create SH file at Runtime
echo "#!/bin/sh" > $TMP_DIR/$GP_SH
echo "echo Host is \`hostname\`" >> $TMP_DIR/$GP_SH
echo "echo Time is \`date\`" >> $TMP_DIR/$GP_SH
echo "start=\`date +%s\`" >> $TMP_DIR/$GP_SH
echo "mkdir RESULT-$APPLNAME/ 2>/dev/null" >> $TMP_DIR/$GP_SH
echo "mv -f $APPLNAME RESULT-$APPLNAME/" >> $TMP_DIR/$GP_SH
if [[ "$USERINPUTNAME" == ".TGZ" || "$USERINPUTNAME" == ".ZIP" ]]; then
        # Multiple
        for i in $list; do
                echo "mv -f $i RESULT-$APPLNAME/" >> $TMP_DIR/$GP_SH
        done
else
        # Single
        echo "mv -f $APPLNAME.input RESULT-$APPLNAME/$USERINPUTNAME" \\
                >> $TMP_DIR/$GP_SH
fi
echo "cd RESULT-$APPLNAME/" >> $TMP_DIR/$GP_SH
echo "chmod u+x $APPLNAME" >> $TMP_DIR/$GP_SH
if [[ "$USERINPUTNAME" == ".TGZ" || "$USERINPUTNAME" == ".ZIP" ]]; then
        echo "./$APPLNAME "$list" > $APPLNAME.output &" >> $TMP_DIR/$GP_SH
else
        echo "./$APPLNAME $USERINPUTNAME > $APPLNAME.output &" >> $TMP_DIR/$GP_SH
fi
echo "pid=\$!" >> $TMP_DIR/$GP_SH
echo "mem=0" >> $TMP_DIR/$GP_SH
echo "while :; do" >> $TMP_DIR/$GP_SH
echo "ps ax |grep \$pid |grep -v grep 2>&1 >/dev/null" >> $TMP_DIR/$GP_SH
echo "ret=\$?" >> $TMP_DIR/$GP_SH
echo "if [ \"\$ret\" == \"0\" ]; then" >> $TMP_DIR/$GP_SH
echo "now=\`pmap \$pid |grep total |awk \\
        '{printf substr(\$NF,1,length(\$NF)-1)}' 2>/dev/null\`" >> $TMP_DIR/$GP_SH
echo "if [[ \"\$now\" -gt \"\$mem\" ]]; then mem=\$now; fi" >> $TMP_DIR/$GP_SH
echo "sleep 1" >> $TMP_DIR/$GP_SH
echo "else" >> $TMP_DIR/$GP_SH
echo "echo Execution done." >> $TMP_DIR/$GP_SH
echo "echo Time is \`date\`" >> $TMP_DIR/$GP_SH
```

ISSN: 2279-8773

```
echo "stop=\`date +%s\`" >> $TMP_DIR/$GP_SH
echo "cpu_time=\`expr \$stop - \$start\`" >> $TMP_DIR/$GP_SH
echo "echo CPU_TIME:\$cpu_time" >> $TMP_DIR/$GP_SH
echo "echo MEM:\$mem" >> $TMP_DIR/$GP_SH
echo "rm -f $APPLNAME" >> $TMP_DIR/$GP_SH
echo "cd .." >> $TMP_DIR/$GP_SH
echo "tar -zcvf $APPLNAME.tar.gz RESULT-$APPLNAME/ 2>&1 > /dev/null" >> $TMP_DIR/$GP_SH
echo "echo END_TIME:\`date +%s\`" >> $TMP_DIR/$GP_SH
echo "export LCG_GFAL_INFOSYS=lcg-bdii.cern.ch:2170" >> $TMP_DIR/$GP_SH
echo "export LCG_CATALOG_TYPE=lfc" >> $TMP_DIR/$GP_SH
echo "export LFC_HOST=lfcserver.cnaf.infn.it" >> $TMP_DIR/$GP_SH
echo "export LFC_HOME=/grid/compchem/" >> $TMP_DIR/$GP_SH
echo "echo -n LFC_GUID:" >> $TMP_DIR/$GP_SH
echo "lcg-cr --vo $VO -l lfn:$LFCUSER/$APPLNAME.tar.gz \\
        file://\`pwd\`/$APPLNAME.tar.gz" >> $TMP_DIR/$GP_SH
echo "if test \$? -ne 0; then" >> $TMP_DIR/$GP_SH
echo "echo LFC upload failed. Have a look at gp.err." >> $TMP_DIR/$GP_SH
echo "exit 1" >> $TMP_DIR/$GP_SH
echo "fi" >> $TMP_DIR/$GP_SH
echo "echo Results have been archived, compressed and stored on LFC as \\
        $APPLNAME.tar.gz." >> $TMP_DIR/$GP_SH
echo "echo Time is \`date\`" >> $TMP_DIR/$GP_SH
echo "exit 0" >> $TMP_DIR/$GP_SH
echo "fi" >> $TMP_DIR/$GP_SH
echo "done" >> $TMP_DIR/$GP_SH

# Copy SH to UI and delete from YP
$SCP $TMP_DIR/$GP_SH $ROBOTUSER@$UI:$USER/$GP_SH
rm -f $TMP_DIR/$GP_SH

# Run the Job on the Grid
RESULT=`$SSH $ROBOTUSER@$UI "$GRID_BIN/glite-wms-job-submit -a $USER/$GP_JDL |grep \\
        $GRID_RUN_PORT"`

# Get CE Queue information when using On-line Ranking
if [[ "$PARADIGM" == "2" || "$PARADIGM" == "3" ]]; then
        sleep 25
        INFOQUEUE="$GRID_BIN/glite-job-status $RESULT |grep Destination"
        queue=`$SSH $ROBOTUSER@$UI "$INFOQUEUE" 2>/dev/null`
        QUEUE=`echo $queue |awk '{print $2}'`
fi
```

ISSN: 2279-8773

```
# Return HTTPS URL and CE QUEUE
echo -n $RESULT" "$QUEUE

# Clear UI Environment
if [[ "$USERINPUTNAME" == ".TGZ" || "$USERINPUTNAME" == ".ZIP" ]]; then
        $SSH $ROBOTUSER@$UI "cd $USER; rm -f $GP_SH $GP_JDL $APPLNAME "$list""
else
        $SSH $ROBOTUSER@$UI "cd $USER; rm -f $GP_SH $GP_JDL $APPLNAME $APPLNAME.input"
fi
```

D2. The upload.sh Shell Script of GriF.

```
#!/bin/bash

# Configuration
APPL="$1"
USER="$2"
NAME="$3"
SSH="/usr/bin/ssh"
SCP="/usr/bin/scp"
UI="ui.grid.unipg.it"
ROBOTUSER="carlo"

# Main Program
$SSH $ROBOTUSER@$UI "mkdir $USER 2>/dev/null"
$SSH $ROBOTUSER@$UI "rm -f $USER/*"
$SCP "$APPL" $ROBOTUSER@$UI:$USER/$NAME
```

D3. The check.sh Shell Script of GriF.

```
#!/bin/bash

# Configuration
URL="$1"
USER="$2"
ROBOTUSER="carlo"
```

Appendix D

```
SSH="/usr/bin/ssh"
UI="ui.grid.unipg.it"
GRID_BIN="/opt/glite/bin"
PROGRAM="$GRID_BIN/glite-job-status $URL"
STATUS_STRING="Current"

# Main Program
$SSH $ROBOTUSER@$UI "$PROGRAM |grep $STATUS_STRING |awk '{for (i=3; i<=NF; i++) \\
          printf \$i\" \"}'" 2>/dev/null
```

D4. The result.sh Shell Script of GriF.

```
#!/bin/bash

# Configuration
USER="$1"
URI="$2"
SSH="/usr/bin/ssh"
SCP="/usr/bin/scp"
UI="ui.grid.unipg.it"
LCG_CP="/opt/lcg/bin/lcg-cp"
LFCUSER="carlo"
ROBOTUSER="carlo"
VO="compchem"
UIHOME="/home"
TMP_DIR="/tmp/GRID"

# Main Program
$SSH $ROBOTUSER@$UI "$LCG_CP --vo $VO lfn:$LFCUSER/$URI \\
          file:$UIHOME/$ROBOTUSER/$ROBOTUSER/$URI"
$SCP $ROBOTUSER@$UI:$ROBOTUSER/$URI $TMP_DIR/$URI
echo $TMP_DIR/$URI
```

D5. The state.sh Shell Script of GriF.

```
#!/bin/bash
```

ISSN: 2279-8773

Appendix D

```
# Database configuration
DBNAME="grif-gp"
DBUSER="grif-gp"
DBPWD="ChiCa.76"
MYSQL="mysql -u $DBUSER --password=$DBPWD -D $DBNAME -e"

# Grid configuration
ROBOTUSER="carlo"
SSH="/usr/bin/ssh"
UI="ui.grid.unipg.it"
GLITE_LOCATION="/opt/glite"
PROGRAM="export GLITE_WMS_LOCATION=$GLITE_LOCATION; \\
  $GLITE_LOCATION/bin/glite-job-status "
STATUS_STRING="Current"
LCG_LS="/opt/lcg/bin/lfc-ls -l"
LFCUSER="carlo"

# Users list
users=`$MYSQL "SELECT username from vousers;"`
users=`echo $users |awk '{for (i=2; i<=NF; i++) print $i}'`

function check_lcg() {
  dim=`$SSH $ROBOTUSER@$UI "$LCG_LS $LFCUSER/$uri 2>/dev/null" 2>/dev/null |awk \\
    '{print $5}' 2>/dev/null`
  if [[ -n $dim && $dim != "0" ]]; then
      eval "$2='0'"
  fi
  if [[ -n $dim && $dim == "0" ]]; then
      eval "$2='1'"
  fi
}

for i in $users; do
  ids=`$MYSQL "SELECT id from jobs where status!='Done' and status!='Cancelled' and \\
    status!='Aborted' and status!='FAILED' and username='$i';"`
  if [[ -n $ids ]]; then
    ids=`echo $ids |awk '{for (i=2; i<=NF; i++) print $i}'`
    for j in $ids; do
      url_job=`$MYSQL "SELECT url_job from jobs where id='$j';"`
      url_job=`echo $url_job |awk '{print $2}'`
      uri=`$MYSQL "SELECT uri from jobs where id='$j';"`
```

ISSN: 2279-8773

```
uri=`echo $uri |awk '{print $2}'`
status=`$MYSQL "SELECT status from jobs where id='$j';"`
status=`echo $status |awk '{print $2}'`
now=`$SSH $ROBOTUSER@$UI "$PROGRAM$url_job |grep $STATUS_STRING 2>/dev/null" \\
  2>/dev/null`
now=`echo $now |awk -F": " '{print $2}'`
if [[ $now == 'Done (Success)' ]]; then
  now="Done"
fi
case "$now" in
  'Done'|'Cleared')
    res="2"
    check_lcg $uri res
    if [ $res == "0" ]; then
      $MYSQL "UPDATE jobs set status='Done', exitstatus='0' where \\
        url_job='$url_job' and username='$i';"
      echo `date +%Y-%m-%d" "%T`: Status changed from \'$status\' to \\
        \'Done\' for jobid: $url_job. >> /var/log/GP/state.log
    fi
    if [[ $res == "1" || $res == "2" ]]; then
      $MYSQL "UPDATE jobs set status='FAILED', exitstatus='1' where \\
        url_job='$url_job' and username='$i';"
      echo `date +%Y-%m-%d" "%T`: Status changed from \'$status\' to \\
        \'FAILED\' for jobid: $url_job. >> /var/log/GP/state.log
    fi
  ;;
  'Running')
    res="2"
    check_lcg $uri res
    if [ $res == "0" ]; then
      $MYSQL "UPDATE jobs set status='Done', exitstatus='0' where \\
        url_job='$url_job' and username='$i';"
      echo `date +%Y-%m-%d" "%T`: Status changed from \'$status\' to \\
        \'Done\' for jobid: $url_job. >> /var/log/GP/state.log
    fi
    if [ $res == "1" ]; then
      $MYSQL "UPDATE jobs set status='FAILED', exitstatus='1' where \\
        url_job='$url_job' and username='$i';"
      echo `date +%Y-%m-%d" "%T`: Status changed from \'$status\' to \\
        \'FAILED\' for jobid: $url_job. >> /var/log/GP/state.log
    fi
    if [ $res == "2" ]; then
```

```
      $MYSQL "UPDATE jobs set status='$now' where \\
        url_job='$url_job' and username='$i';"
      if [ $status != $now ]; then
        echo `date +%Y-%m-%d" "%T`: Status changed from \'$status\' to \\
          \'Running\' for jobid: $url_job. >> /var/log/GP/state.log
      fi
    fi
  ;;
  'Scheduled'|'Ready'|'Waiting'|'Submitted'|'Cancelled')
    if [ $status != $now ]; then
      $MYSQL "UPDATE jobs set status='$now' where \\
        url_job='$url_job' and username='$i';"
      echo `date +%Y-%m-%d" "%T`: Status changed from \'$status\' to \\
        \'$now\' for jobid: $url_job. >> /var/log/GP/state.log
    fi
    # Determine Grid jobs not running for over 1 week which will be cancelled
    # with a queue delay error (exitstatus='3').
    sub_date=`$MYSQL "SELECT sub_date from jobs where id='$j'"`
    sub_date=`echo $sub_date |awk '{for (k=2; k<=NF; k++) print $k}'`
    ts=`date -d "$sub_date" +%s`
    now_ts=`date +%s`
    elapsed=`echo "scale=0; $now_ts - $ts" |bc`
    if [ $elapsed -gt "604800" ]; then
      $MYSQL "UPDATE jobs set status='FAILED', exitstatus='3', cleared='Y' where \\
        url_job='$url_job' and username='$i';"
      echo `date +%Y-%m-%d" "%T`: Status changed from \'$status\' to \\
        \'FAILED\' \(job not running for over 1 week\) for jobid: $url_job. >> \\
        /var/log/GP/state.log
    fi
  ;;
  'Done (Exit Code !=0)'|'Aborted')
    $MYSQL "UPDATE jobs set status='FAILED', exitstatus='1' where \\
      url_job='$url_job' and username='$i';"
    echo `date +%Y-%m-%d" "%T`: Status changed from \'$status\' to \\
      \'FAILED\' for jobid: $url_job. >> /var/log/GP/state.log
  ;;
  *)
    $MYSQL "UPDATE jobs set status='FAILED', existstatus='1' where \\
      url_job='$url_job' and username='$i';"
    echo `date +%Y-%m-%d" "%T`: Warning: jobid $url_job has reported an \\
      unknown state \'$now\'. >> /var/log/GP/state.log
esac
```

```
    done
  fi
done
```

D6. The rank.sh Shell Script of GriF.

```
#!/bin/bash

# Database configuration
DBNAME="grif-db"
DBUSER="<db_user>"
DBPWD="<db_pwd>"
MYSQL="mysql -u $DBUSER --password=$DBPWD -D $DBNAME -e"

# Grid configuration
ROBOTUSER="carlo"
UIHOME="/home"
SSH="/usr/bin/ssh"
SCP="/usr/bin/scp"
GRID_BIN="/opt/glite/bin"
UI="ui.grid.unipg.it"

# System configuration
TMP_DIR=/tmp/GRID
NULLGRIFSH="GriF-Rank.sh"
TESTGRIFJDL="GriF-Rank.jdl"
TELNET="/usr/bin/telnet"
CNSTRING="Escape"

# Constant to emphasize the errors reported by a queue
KPERF="100"
# Constant to set the importance of the average latency (bonus)
KLAT="0.01"

# Determine BAD queues (all jobs run on these queues have been failed)
allbad=`$MYSQL "SELECT distinct queue from jobs where exitstatus ='1';"`
allbad=`echo $allbad |awk '{for (i=2; i<=NF; i++) print $i}'`
for i in $allbad; do
  njobs=`$MYSQL "SELECT count(id) from jobs where queue='$i';"`
```

ISSN: 2279-8773

```
  njobs=`echo $njobs |awk '{print $2}'`
  nerr=`$MYSQL "SELECT count(id) from jobs where exitstatus='1' and queue='$i';"`
  nerr=`echo $nerr |awk '{print $2}'`
  if [[ $njobs -eq $nerr ]]; then
    BAD=$BAD"(!(RegExp(\"$i\", other.GlueCEUniqueId))) && "
  fi
done

# maxrunningjobs-runningjobs>0, wt>2days, freecpus>0, waitingjobs=0
REQ="(other.GlueCEPolicyMaxRunningJobs - other.GlueCEStateRunningJobs > 0) && \\
  (other.GlueCEPolicyMaxWallClockTime > 2879) && (other.GlueCEStateFreeCPUs > 0) && \\
  (other.GlueCEStateWaitingJobs < 1);"
# min ct
RANK="(-other.GlueCEPolicyMaxCPUTime);"

try="0"
ces=""
# After this cycle one will MUST have a set of queues (5 different ranking attempts)
while [[ ! -n $ces && $try -le 5 ]]; do

  # Create JDL test file at Runtime
  echo "Type = \"Job\";" > $TMP_DIR/$TESTGRIFJDL
  echo "JobType = \"Normal\";" >> $TMP_DIR/$TESTGRIFJDL
  echo "Executable = \"$NULLGRIFSH\";" >> $TMP_DIR/$TESTGRIFJDL
  echo "StdOutput = \"GriF-Rank.log\";" >> $TMP_DIR/$TESTGRIFJDL
  echo "StdError = \"GriF-Rank.err\";" >> $TMP_DIR/$TESTGRIFJDL
  echo "InputSandbox = {\"$UIHOME/$ROBOTUSER/$ROBOTUSER/$NULLGRIFSH\"};" >> \\
    $TMP_DIR/$TESTGRIFJDL
  echo "OutputSandbox  = {\"GriF-Rank.log\", \"GriF-Rank.err\"};" >> \\
    $TMP_DIR/$TESTGRIFJDL
  if [[ $try != "5" ]]; then
    echo "Requirements = $BAD$REQ" >> $TMP_DIR/$TESTGRIFJDL
    echo "Rank = $RANK" >> $TMP_DIR/$TESTGRIFJDL
  else
    # Exclude BAD queues (in any attempts)
    echo "Requirements = $BAD(other.GlueCEStateStatus == \"Production\");" >> \\
      $TMP_DIR/$TESTGRIFJDL
  fi

  # Create SH test file at Runtime
  touch $TMP_DIR/$NULLGRIFSH
```

ISSN: 2279-8773

Appendix D

```
# Copy to UI
$SCP $TMP_DIR/$TESTGRIFJDL $ROBOTUSER@$UI:$ROBOTUSER/$TESTGRIFJDL 2>/dev/null
$SCP $TMP_DIR/$NULLGRIFSH $ROBOTUSER@$UI:$ROBOTUSER/$NULLGRIFSH 2>/dev/null

# Run Test Job
ces=`$SSH $ROBOTUSER@$UI "$GRID_BIN/glite-wms-job-list-match -a \\
  $ROBOTUSER/$TESTGRIFJDL" 2>/dev/null | grep " - " |awk -F"- " '{print $2}' |uniq \\
  2>/dev/null`

# Count CE queues
count="0"
for i in $ces; do
  count=`expr $count + 1`
done

if [[ ! -n $ces ]]; then
  try=`expr $try + 1`
elif [[ $try == "0" ]]; then
  echo `date +%Y-%m-%d" "%T`: Using Best Ranking between $count queues. >> \\
     /var/log/GP/rank.log
elif [[ $try == "1" ]]; then
  echo `date +%Y-%m-%d" "%T`: Using 2nd attempt Ranking between $count queues. >> \\
    /var/log/GP/rank.log
elif [[ $try == "2" ]]; then
  echo `date +%Y-%m-%d" "%T`: Using 3rd attempt Ranking between $count queues. >> \\
    /var/log/GP/rank.log
elif [[ $try == "3" ]]; then
  echo `date +%Y-%m-%d" "%T`: Using 4th attempt Ranking between $count queues. >> \\
    /var/log/GP/rank.log
elif [[ $try == "4" ]]; then
  echo `date +%Y-%m-%d" "%T`: Using 5th attempt Ranking between $count queues. >> \\
    /var/log/GP/rank.log
elif [[ $try == "5" ]]; then
  echo `date +%Y-%m-%d" "%T`: Using Pure Ranking from the Grid between $count \\
    queues \(after 5 attempts\). >> /var/log/GP/rank.log
fi
if [[ ! -n $ces && $try == "1" ]]; then
  # wt>2days, freecpus>0, waitingjobs=0
  REQ="(other.GlueCEPolicyMaxWallClockTime > 2879) && \\
    (other.GlueCEStateFreeCPUs > 0) && (other.GlueCEStateWaitingJobs < 1);"
  # RANK is as in the previous case
fi
```

ISSN: 2279-8773

```
  if [[ ! -n $ces && $try == "2" ]]; then
    # wt>2days, waitingjobs=0
    REQ="(other.GlueCEPolicyMaxWallClockTime > 2879) && \\
      (other.GlueCEStateWaitingJobs < 1);"
    # RANK is as in the previous case
  fi
  if [[ ! -n $ces && $try == "3" ]]; then
    # wt>2days, freecpus>0
    REQ="(other.GlueCEPolicyMaxWallClockTime > 2879) && \\
      (other.GlueCEStateFreeCPUs > 0);"
    # min waitingjobs
    RANK="(-other.GlueCEStateWaitingJobs);"
  fi
  if [[ ! -n $ces && $try == "4" ]]; then
    # wt>3days
    REQ="(other.GlueCEPolicyMaxWallClockTime > 2879);"
    # RANK is as in the previous case
  fi
done

# If the queues set is empty, use the old ranking
if [[ $count == "0" ]]; then
  echo `date +%Y-%m-%d" "%T`: Using the previous Queues ranking \(Warning: the Grid \\
    has no returned available queues at present\). >> /var/log/GP/rank.log
else
  # Delete old ranking
  $MYSQL "UPDATE queues_tmp set ranking=NULL, performance=NULL, avg_latency=NULL, rr=0;"
  # Choose the best queue between the new ones and rank the old ones with default
  j="1"
  activeisnew="0"
  for i in $ces; do
    state=`$MYSQL "SELECT new from queues_tmp where name='$i';"`
    state=`echo $state |awk '{print $2}'`
    # Choose the same (and the best) new queue until it will be really used
    if [[ $activeisnew == "0" ]]; then
      if [[ $state == "Y" ]]; then
        used=`$MYSQL "SELECT count(id) from jobs where queue='$i';"`
        used=`echo $used |awk '{print $2}'`
        if [[ $used -gt 0 ]]; then
          $MYSQL "UPDATE queues_tmp set new='N', ranking='$j' where name='$i';"
          echo `date +%Y-%m-%d" "%T`: Queue $i appears to be new, instead it has been \\
            skipped because it has been already used $used times in the past. >> \\
            /var/log/GP/rank.log
          j=`expr $j + 1`
        else
          addr=`echo $i |awk -F"/" '{print $1}'`
          host=`echo $addr |awk -F":" '{print $1}'`
          port=`echo $addr |awk -F":" '{print $2}'`
          # Connection test
          cn=`$SSH $ROBOTUSER@$UI "echo \" \" |$TELNET $host $port |grep $CNSTRING \\
            2>/dev/null" 2>/dev/null`
          if [[ -n $cn ]]; then
            # A new queue is chosen (ranking -9999999.000 is the best ranking available)
            $MYSQL "UPDATE queues_tmp set ranking=NULL where ranking=-9999999.000;"
            $MYSQL "UPDATE queues_tmp set ranking=-9999999.000 where name='$i';"
            activeisnew="1"
            active=$i
          fi
        fi
      fi
    fi
    # Ranking the remaining used queues
    if [[ $state == "N" ]]; then
      $MYSQL "UPDATE queues_tmp set ranking='$j' where name='$i';"
      j=`expr $j + 1`
    fi
  done

  # No new queues, Final Ranking is calculated for all the others queues
  if [[ $activeisnew == "0" ]]; then
    for i in $ces; do
      n=`$MYSQL "SELECT count(id) from jobs where queue='$i';"`
      n=`echo $n |awk '{print $2}'`
      if [[ $n != "0" ]]; then

        # Performance
        failed=`$MYSQL "SELECT count(id) from jobs where exitstatus='1' and \\
          queue='$i';"`
        failed=`echo $failed |awk '{print $2}'`
        perf=`echo "scale=3; $KPERF * ($failed / $n)" |bc`
        r=`$MYSQL "SELECT ranking from queues_tmp where ranking IS NOT NULL and \\
          name='$i';"`
        r=`echo $r |awk '{print $2}'`
        if [[ ! -n $r ]]; then
```

ISSN: 2279-8773

```
  r="0"
 fi
$MYSQL "UPDATE queues_tmp set performance=$perf, ranking=$r+$perf where \\
  name='$i';"

# Bonus
num=`$MYSQL "SELECT count(id) from jobs where exitstatus='0' and \\
  wall_time>0 and cpu_time>0 and queue='$i';"`
num=`echo $num |awk '{print $2}'`
if [[ $num != "0" ]]; then
  wt=`$MYSQL "SELECT sum(wall_time) from jobs where exitstatus='0' and \\
    wall_time>0 and cpu_time>0 and queue='$i';"`
  wt=`echo $wt |awk '{print $2}'`
  ct=`$MYSQL "SELECT sum(cpu_time) from jobs where exitstatus='0' and \\
    wall_time>0 and cpu_time>0 and queue='$i';"`
  ct=`echo $ct |awk '{print $2}'`
  avg_lat1=`echo "scale=0; ($wt - $ct) / $num" |bc`
  avg_lat2=`echo "scale=3; ($wt - $ct) / $num" |bc`
  incr=`echo "scale=3; 1 / ($KLAT * $avg_lat2)" |bc`
  if [[ $avg_lat1 -le "0" || ! -n $avg_lat1 ]]; then
    avg_lat2="NULL"
    bonus="0"
  fi
  # 10 points plus $incr to fast queues (avg_lat1 < 5 mins)
  if [[ $avg_lat1 -gt "0" && $avg_lat1 -le "300" ]]; then
    #bonus="10"
    bonus=`echo "scale=3; 10 + $incr" |bc`
  fi
  # 3 points plus $incr to normal queues (avg_lat1 between 5 mins and 1 hour)
  if [[ $avg_lat1 -gt "300" && $avg_lat1 -le "3600" ]]; then
    #bonus="3"
    bonus=`echo "scale=3; 3 + $incr" |bc`
  fi
  # 0 points plus $incr to slow queues (avg_lat1 between 1 hour and 1 days)
  if [[ $avg_lat1 -gt "3600" && $avg_lat1 -le "86400" ]]; then
    bonus=$incr
  fi
  # -10 points plus $incr to very slow queues (avg_lat1 => 1 day)
  if [[ $avg_lat1 -gt "86400" ]]; then
    bonus=`echo "scale=3; -10 + $incr" |bc`
  fi
```

```
        fi
      fi
    done

    # Check the network availability of the ranked queues (telnet on port derived
    # from queue itself)
    ranked=`$MYSQL "SELECT name from queues_tmp where ranking IS NOT NULL order by \\
      ranking;"`
    ranked=`echo $ranked |awk '{for (i=2; i<=NF; i++) print $i}'`
    ok="KO"
    for i in $ranked; do
      addr=`echo $i |awk -F"/" '{print $1}'`
      host=`echo $addr |awk -F":" '{print $1}'`
      port=`echo $addr |awk -F":" '{print $2}'`
      cn=`$SSH $ROBOTUSER@$UI "echo \" \" |$TELNET $host $port |grep $CNSTRING \\
        2>/dev/null" 2>/dev/null`
      if [[ -n $cn ]]; then
        if [[ $ok == "KO" ]]; then active=$i; fi
        ok="OK"
      else
        $MYSQL "UPDATE queues_tmp set ranking=NULL where name='$i';"
      fi
    done
  fi

  # Update Real Ranking (queue table) only if a new active queue has been determined
  # at least
  if [[ $ok == "OK" || $activeisnew == "1" ]]; then
    # Dump Ranking from queues_tmp table to queues table
    $MYSQL "UPDATE queues, queues_tmp set queues.ranking=queues_tmp.ranking, \\
      queues.rr=queues_tmp.rr where queues.id=queues_tmp.id;"
    echo `date +%Y-%m-%d" "%T`: Queues ranking recalculated \(the current active \\
      queue is: $active\). >> /var/log/GP/rank.log
  else
    echo `date +%Y-%m-%d" "%T`: Using the previous Queues ranking \(Warning: \\
      each queue returned by the Grid seems to be down\). >> /var/log/GP/rank.log
  fi
fi

# Clear YP and UI
rm -f $TMP_DIR/$NULLGRIFSH $TMP_DIR/$TESTGRIFJDL
$SSH $ROBOTUSER@$UI "rm -f $ROBOTUSER/$NULLGRIFSH $ROBOTUSER/$TESTGRIFJDL" 2>/dev/null
```

Appendix D

D7. The queues.sh Shell Script of GriF.

```
#!/bin/bash

# Database configuration
DBNAME="grif-db"
DBUSER="<db_user>"
DBPWD="<db_pwd>"
MYSQL="mysql -u $DBUSER --password=$DBPWD -D $DBNAME -e"

# Grid configuration
ROBOTUSER="carlo"
SSH="/usr/bin/ssh"
UI="ui.grid.unipg.it"
VO="compchem"
LCGINFO_CE="/opt/lcg/bin/lcg-infosites ce --vo $VO"

celist=`$SSH $ROBOTUSER@$UI "$LCGINFO_CE |awk '{print \\$6}' |grep -v Waiting" \\
        2>/dev/null`
for i in $celist; do
        $MYSQL "INSERT IGNORE INTO queues_tmp(ranking, name, new, performance, \\
                avg_latency, rr) values (NULL, '$i','Y', NULL, NULL, '0');"
        $MYSQL "INSERT IGNORE INTO queues SELECT id, ranking, name, rr from queues_tmp;"
done
echo `date +%Y-%m-%d" "%T`: CEs List for $VO VO has been updated. \\
        >> /var/log/GP/rank.log
```

ISSN: 2279-8773

Virt&l–Comm.3.2012.22

Appendix E

YP Database Code

E1. The Structure of the nine tables of the GriF Database.

Applications Table

```
+-------------+--------------+------+-----+-------------------+----------------+
| Field       | Type         | Null | Key | Default           | Extra          |
+-------------+--------------+------+-----+-------------------+----------------+
| id          | mediumint(9) | NO   | PRI | NULL              | auto_increment |
| service     | varchar(20)  | NO   | MUL | NULL              |                |
| yp          | varchar(150) | NO   |     | NULL              |                |
| pub_date    | timestamp    | NO   |     | CURRENT_TIMESTAMP |                |
| name        | varchar(100) | NO   | UNI | NULL              |                |
| author      | varchar(100) | NO   |     | NULL              |                |
| maintainer  | varchar(100) | NO   |     | NULL              |                |
| license     | varchar(20)  | NO   |     | NULL              |                |
| description | varchar(200) | NO   |     | NULL              |                |
| filename    | varchar(50)  | NO   | UNI | NULL              |                |
| production  | enum('N','Y')| NO   |     | N                 |                |
| area        | varchar(100) | NO   |     | NULL              |                |
| subject     | varchar(100) | NO   |     | NULL              |                |
| topic       | varchar(100) | NO   |     | NULL              |                |
+-------------+--------------+------+-----+-------------------+----------------+
```

ISSN: 2279-8773

Appendix E

Compilations Table

```
+-----------+---------------+-------+-----+-------------------+----------------+
| Field     | Type          | Null  | Key | Default           | Extra          |
+-----------+---------------+-------+-----+-------------------+----------------+
| id        | mediumint(9)  | NO    | PRI | NULL              | auto_increment |
| username  | varchar(20)   | NO    | MUL | NULL              |                |
| service   | varchar(20)   | NO    | MUL | NULL              |                |
| yp        | varchar(150)  | NO    |     | NULL              |                |
| sub_date  | timestamp     | NO    |     | CURRENT_TIMESTAMP |                |
| source    | varchar(100)  | NO    |     | NULL              |                |
| bin       | varchar(100)  | YES   |     | NULL              |                |
| retrieved | enum('N','Y') | NO    |     | NULL              |                |
+-----------+---------------+-------+-----+-------------------+----------------+
```

Feedbacks Table

```
+----------+--------------+-------+-----+-------------------+----------------+
| Field    | Type         | Null  | Key | Default           | Extra          |
+----------+--------------+-------+-----+-------------------+----------------+
| id       | mediumint(9) | NO    | PRI | NULL              | auto_increment |
| subject  | varchar(100) | NO    |     | NULL              |                |
| text     | text         | NO    |     | NULL              |                |
| sub_date | timestamp    | NO    |     | CURRENT_TIMESTAMP |                |
| username | varchar(20)  | NO    |     | NULL              |                |
+----------+--------------+-------+-----+-------------------+----------------+
```

Jobs Table (partially dumped)

```
+-------------+--------------------------------------------------+-------+-----+
| Field       | Type                                             | Null  | Key |
+-------------+--------------------------------------------------+-------+-----+
| id          | mediumint(9)                                     | NO    | PRI |
| username    | varchar(20)                                      | NO    | MUL |
| service     | varchar(20)                                      | NO    | MUL |
| yp          | varchar(150)                                     | NO    |     |
| url_job     | varchar(200)                                     | NO    | UNI |
| sub_date    | timestamp                                        | NO    |     |
| uri         | varchar(200)                                     | NO    |     |
| application | varchar(40)                                      | NO    |     |
```

ISSN: 2279-8773

Appendix E

```
| input       | varchar(40)                                                  | NO   |      |
| exitstatus  | int(1)                                                       | NO   |      |
| retrieved   | enum('N','Y')                                                | NO   |      |
| status      | enum('Do','Ru','Sc','Re','Wa','Su','Ab','Cl','Ca','F')       | NO   |      |
| queue       | varchar(200)                                                 | NO   |      |
| wall_time   | int(20)                                                      | YES  |      |
| job_type    | enum('Serial','Parametric','DAG','Collection','Other')       | NO   |      |
| description | varchar(150)                                                 | NO   |      |
| cpu_time    | int(20)                                                      | YES  |      |
| lfc_guid    | varchar(60)                                                  | YES  |      |
| deleted     | enum('Y','N')                                                | NO   |      |
| cleared     | enum('Y','N','E')                                            | NO   |      |
| mem         | int(20)                                                      | YES  |      |
+-------------+--------------------------------------------------------------+------+------+
```

Queues Table

```
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| id      | mediumint(9) | NO   | PRI | NULL    | auto_increment |
| ranking | float(10,3)  | YES  |     | NULL    |                |
| name    | varchar(200) | NO   | UNI | NULL    |                |
| rr      | int(10)      | NO   |     | 0       |                |
+---------+--------------+------+-----+---------+----------------+
```

Queues_tmp Table

```
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| id          | mediumint(9)  | NO   | PRI | NULL    | auto_increment |
| ranking     | float(10,3)   | YES  |     | NULL    |                |
| name        | varchar(200)  | NO   | UNI | NULL    |                |
| new         | enum('Y','N') | NO   |     | NULL    |                |
| performance | float(10,3)   | YES  |     | NULL    |                |
| avg_latency | float(10,3)   | YES  |     | NULL    |                |
| rr          | int(10)       | NO   |     | 0       |                |
+-------------+---------------+------+-----+---------+----------------+
```

ISSN: 2279-8773

Appendix E

Services Table

```
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| name        | varchar(20)  | NO   | PRI | NULL    |       |
| yp          | varchar(150) | NO   | PRI | NULL    |       |
| cost        | float(10,2)  | NO   |     | NULL    |       |
| description | varchar(500) | NO   |     | NULL    |       |
| port        | int(5)       | NO   |     | NULL    |       |
| vo          | varchar(25)  | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
```

VOs Table

```
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(20) | NO   | PRI | NULL    |       |
| admin | varchar(20) | YES  | MUL | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

VOusers Table

```
+----------+------------------------------------+------+-----+---------+-------+
| Field    | Type                               | Null | Key | Default | Extra |
+----------+------------------------------------+------+-----+---------+-------+
| username | varchar(20)                        | NO   | PRI | NULL    |       |
| name_vo  | varchar(20)                        | NO   | PRI | NULL    |       |
| passwd   | varchar(50)                        | NO   |     | NULL    |       |
| name     | varchar(30)                        | NO   |     | NULL    |       |
| surname  | varchar(30)                        | NO   |     | NULL    |       |
| address  | varchar(100)                       | NO   |     | NULL    |       |
| phone    | varchar(30)                        | YES  |     | NULL    |       |
| email    | varchar(40)                        | NO   |     | NULL    |       |
| type     | enum('admin','PU','AU','SD','SP')  | YES  |     | NULL    |       |
| key      | varchar(30)                        | NO   |     | NULL    |       |
| notify   | enum('Y','N')                      | YES  |     | N       |       |
| credits  | float(10,2)                        | YES  |     | NULL    |       |
+----------+------------------------------------+------+-----+---------+-------+
```

ISSN: 2279-8773

Appendix F

YC Code

F1. GriF Authentication.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    username = new String(jTextField1.getText());
    pwd = new String(jPasswordField1.getPassword());
    if (username.equals("") || pwd.equals("")) {
        JOptionPane.showMessageDialog(null, "Please fill all fields.", "Error", \\
            JOptionPane.ERROR_MESSAGE);
    } else {
        try {
            this.WSname = "Login";
            Service service = new Service();
            Call call = (Call) service.createCall();
            call.setTargetEndpointAddress(new java.net.URL(WS_Login));
            call.setOperationName("getPwd");
            call.addParameter("username", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("password", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("WSname", XMLType.XSD_STRING, ParameterMode.IN);
            call.setReturnType(XMLType.XSD_STRING);
            res = (String) call.invoke(new Object[]{YattecLogin.username, \\
                YattecLogin.pwd, this.WSname});
            String[] exit = res.split(" ");
            pass = exit[0];
            key = exit[1];
            ext = exit[2];
        } catch (Exception e) {}
        if (!pass.equals("")) {
            if (pwd.equals(pass)) {
                this.setVisible(false);
                YattecFramework fr = new YattecFramework(ext, key, endpoint, \\
                    inquiryURL, publishURL, businessKey, username, pwd, loginhost);
                fr.setVisible(true);
            } else {
                JOptionPane.showMessageDialog(null, "Wrong Password, Access Denied.", \\
                    "Error", JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(null, "Unexisting User, Access Denied.", \\
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

F2. Running Phase: the starting of the Grid job and his insertion on the GriF Database.

```
private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
        this.WSname = "RunGP";
        Service service = new Service();
        try {
            String paradigm = "";
            if (jRadioButton1.isSelected()) paradigm = "1";
            if (jRadioButton2.isSelected()) paradigm = "2";
            if (jRadioButton3.isSelected()) paradigm = "3";
            if (this.idh.getName().endsWith(".tar.gz")) userinputname =".TGZ"; \\
                else if (this.idh.getName().endsWith(".zip")) userinputname =".ZIP"; \\
                else userinputname =this.idh.getName();
            Call call = (Call) service.createCall();
            QName qnameAttachment = new QName("xsd:ASCII");
            call.registerTypeMapping(DataHandler.class, qnameAttachment, \\
                JAFDataHandlerSerializerFactory.class, \\
                JAFDataHandlerDeserializerFactory.class);
            call.setTargetEndpointAddress(new java.net.URL(endpoint_runGP));
            call.setOperationName("run");
            DataHandler dh = new DataHandler(new FileDataSource(this.idh));
            call.addParameter("username", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("pwd", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("applname", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("dh", qnameAttachment, ParameterMode.IN);
            call.addParameter("date", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("key", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("WSname", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("paradigm", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("userinputname", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("method", XMLType.XSD_STRING, ParameterMode.IN);
            call.setReturnType(XMLType.XSD_STRING);
            String res = (String) call.invoke(new Object[]{this.username, this.pwd, \\
                this.applname, dh, mydate, this.key, this.WSname, paradigm, \\
                userinputname, this.method});
            String[] exit = res.split(" ");
            this.url = exit[0];
            this.queue = exit[1].trim();
            if (this.url.equals("") || this.queue.equals("")) {
                this.run=false;
                logPane.append("\nError: The Grid is unavailable.");
                logPane.setCaretPosition(logPane.getText().length() - 1);
                btnUpload.setEnabled(true);
                cmbBinary.setSelectedIndex(0);
                jTextArea1.setText("");
                setCursor(null);
            }
            else if (res.equals("NOT OK")) {
                this.run=false;
                logPane.setText("\nMITM. You have been logged.");
                logPane.setCaretPosition(logPane.getText().length() - 1);
            }
            else {
                this.run=true;
                logPane.append("\n\nDone: Job run on " + this.queue + " with URL " + \\
                    this.url + "\n");
                logPane.setCaretPosition(logPane.getText().length() - 1);
                btnStatus.setEnabled(true);
            }
        } catch (Exception e) {
```

ISSN: 2279–8773

```
        logPane.append("\nNetwork Error.");
        logPane.setCaretPosition(logPane.getText().length() - 1);
        btnUpload.setEnabled(true);
        cmbBinary.setSelectedIndex(0);
        jTextArea1.setText("");
        setCursor(null);
        this.checkbin = false;
    }
    btnStart.setEnabled(false);
    String myuri = "";
    if (this.method.equals("CUSTOM")) {
        myuri = this.applname + ext;
    } else {
        myuri = "GriF-" + this.username + "-" + mydate + ext;
    }
    if (this.run) {
        try {
            String descr = this.jTextArea1.getText().trim();
            if (descr.contains("'")) descr = descr.replaceAll("'", " ");
            if (descr.contains("\n")) descr = descr.replaceAll("\n", " ");
            if (descr.length() > 150) descr = descr.substring(0, 149);
            String[] yp = this.endpoint.split(":");
            Service DBservice = new Service();
            Call call = (Call) DBservice.createCall();
            call.setTargetEndpointAddress(new java.net.URL(endpoint_DB));
            call.setOperationName("insert");
            call.addParameter("username", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("pwd", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("key", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("service", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("yp", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("url_job", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("sub_date", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("uri", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("surface", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("input", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("queue", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("job_type", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("description", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("WSname", XMLType.XSD_STRING, ParameterMode.IN);
            call.setReturnType(XMLType.XSD_STRING);
            String exit = (String) call.invoke(new Object[]{this.username, \\
```

ISSN: 2279-8773

```
                    this.pwd, this.key, "gp", yp[0], this.url, mydate, myuri, \\
                    this.binname, this.idh.getName(), this.queue, "Serial", descr, \\
                    this.WSname});
                if (exit.equals("NOT OK")) {
                    logPane.setText("\nMITM. You have been logged.");
                    logPane.setCaretPosition(logPane.getText().length() - 1);
                }
            } catch (Exception e) {}
        }
}
```

F3. Getting Results in GriF: the Web Service call.

```
[...]

try {
    Service service = new Service();
    Call call = (Call) service.createCall();
    ((org.apache.axis.client.Call)call).setTimeout(new Integer(0));
    call.setTargetEndpointAddress(new java.net.URL(endpoint_result));
    call.setOperationName(new QName("Result", "get"));
    QName qnameAttachment = new QName("Result", "DataHandler");
    call.registerTypeMapping(DataHandler.class, qnameAttachment, \\
        JAFDataHandlerSerializerFactory.class, JAFDataHandlerDeserializerFactory.class);
    call.setReturnType(qnameAttachment);
    call.addParameter("username", XMLType.XSD_STRING, ParameterMode.IN);
    call.addParameter("pwd", XMLType.XSD_STRING, ParameterMode.IN);
    call.addParameter("key", XMLType.XSD_STRING, ParameterMode.IN);
    call.addParameter("WSname", XMLType.XSD_STRING, ParameterMode.IN);
    call.addParameter("uri", XMLType.XSD_STRING, ParameterMode.IN);
    call.addParameter("date", XMLType.XSD_STRING, ParameterMode.IN);
    DataHandler ret = (DataHandler) call.invoke( new Object [] {this.username, \\
        this.pwd, this.key, this.WSname, myuri, date});
    try {
        FileOutputStream fos = new FileOutputStream(f);
        ret.writeTo(fos);
        fos.close();
        if (f.length() > 0) {
            logPane.append("\n\nThe results of the job " + \\
```

ISSN: 2279-8773

```
                this.url[jList1.getSelectedIndex()] + " have been saved in " + \\
                f.getName() + ".\n");
        }
    }
}
```

ISSN: 2279–8773