AGH University of Science and Technology
Faculty of Physics and Applied Computer Science

Master thesis

Marek Pomocka

major: **applied computer science**

specialisation: **computer techniques in science and technology**

Data source registration in the Virtual Laboratory

Supervisor: **Marian Bubak, Ph.D.**

Consultants: **Piotr Nowakowski, M.Sc.**
Daniel Harężlak, M.Sc.

Cracow, September 2009

Aware of criminal liability for making untrue statements I decleare that the following thesis was written personally by myself and that I did not use any sources but the ones mentioned in the dissertation itself.

Cracow, September 2009

The subject of the master thesis and the internship by Marek Pomocka, student of 5th year major in Applied Computer Science, specialisation in computer techniques in science and technology

The subject of the Master Thesis: **Data source registration in the Virtual Laboratory**

Supervisor:	Marian Bubak, Ph.D.
Reviewer:	Piotr Gronek, Ph.D.
A place of the internship:	Academic Computer Centre Cyfronet AGH, Cracow

Programme of the Master Thesis and the Internship

1. Discussion with the supervisor and consultants on realization of the thesis.
2. Collecting and studying the references relevant to the thesis topic.
3. The internship:
   - getting to know the environment of Virtual Laboratory and the problem to be solved
   - learning the necessary programming languages
   - identifying project requirements and possible implementation technologies
   - drafting the design
   - discussion with the supervisor on the proposed design
   - preparation of the Internship report.
4. Specifying detailed software requirements.
5. Prototyping possible solutions.
6. Making decisions regarding the implementation.
7. Creating complete design plan.
8. Implementing the solution.
9. Correctness tests, measuring performance and software limits.
10. Final analysis of the problem and to what extend the created software solves it, conclusions – discussion and final approval by the thesis supervisor.
11. Typesetting the thesis.

Dean's office delivery deadline: 30 September 2009

Acknowledgements

I would like to express my thanks to Marian Bubak and Piotr Nowakowski for their invaluable help, guidance, advice and thoughtfulness. Furthermore, I would like to thank David and Gillian Crowther for their language help. I dedicate this thesis to my mother who was always with me.

Contents

List of Tables

List of Figures

1 Definitions, acronyms and abbreviations

Note: If you have not found the term you are looking for, please check one of these glossaries: [63, 116–119, 234], the *Abbreviations and acronyms* chapter of [150] or the glossary chapter of [47].

1.1 Acronyms and abbreviations

Below, the table of acronyms used throughout the thesis is presented. Some definitions can be found in the subsequent section.

Table 1: Acronyms and abbreviations

Acronym	***Meaning***
BDII	Berkeley Database Information Index
DAC	Data Access Client
DAC2	Data Access Client 2
DAS	VL Data Access Services
DSR	Data Source Registry
DSS	Decision Support System
EGEE	Enabling Grids for E-sciencE
EPE	Experiment Planning Environment
ExpRepo	Experiment Repository
GREDIA	GRid enabled access to rich mEDIA content
GScript	GridSpace Script
GSEC	GSEngine Client
GSEngine	GridSpace Engine
GSES	GSEngine Server
GSI	Grid Security Infrastructure
GUID	Grid Unique Identifier
HLA	High Level Architecture
LCG	LHC Computing Grid
LFC	LCG File Catalog
LFCDS	LFC Data Source
LHC	Large Hadron Collider
OGSA	Open Grid Services Architecture
OGSA–DAI	Open Grid Services Architecture Data Access and Integration
PKI	Public Key Infrastructure
RFIO	Remote File Input/Output
SRS	Software Requirements Specification

Table 1: Acronyms and abbreviations (continued)

Acronym	*Meaning*
SURL	Storage URL
TURL	Transport URL
URL	Uniform Resource Locator
VDT	Virtual Data Toolkit
ViroLab	"ViroLab" Virtual Laboratory project
VL	Virtual Laboratory
VO	Virtual Organization
WLCG	Worldwide LHC Computing Grid

1.2 Definitions

Table 2: Definitions

Term	*Abbr.*	*Definition or explanation*
Berkeley Database Information Index	*BDII*	Metadata service used in EGEE. It is an equivalent to Globus Metadata Directory Service (MDS) [85]. The BDII service is based on catalogue service using LDAP [235] protocol and a database backend. The structure of the BDII is hierarchical. At the lowest level, information providers deliver service-related data which then is consolidated into a site BDII service. The site BDII service is queried by Top Level BDIIs (TL BDIIs) to create a complete view of the whole infrastructure. Each TL BDII exposes information about entire Grid. [22, 83]
ChemPo		"The ChemPo project develops a computational chemistry portal which facilitates the use of numerous packages (e.g. Gaussian or NAMD) deployed on the Grid infrastructure." from [61]
Clinician (in ViroLab terminology)		A healthcare professional who executes a ViroLab experiment or uses the DSS in order to decide how to treat a particular patient. [177, section 2.4]

Table 2: Definitions (continued)

Term	***Abbr.***	***Definition or explanation***
Data Access Client	*DAC*	First generation of data access component for the GSEngine. At the time of writing this document, the DAC component is being upgraded to a version that takes advantage of Data Source Registry [18, 20, 108].
Data Access Client 2	*DAC2*	"A complete rebuild of the Data Access Client, taking into account the capabilities provided by the Data Source Registry." [60]
Data Source Registry	*DSR*	Registry of data sources used by GSEngine DAC2. Information stored in the registry include type of the data source, its technology (e.g. DAS, MySQL [227], WebDAV [75] or PostgreSQL [229]), the URL, credentials and user access rights.
DSR plug-in		EPE plug-in that enables the developer to manage data sources registered in the DSR.
Enabling Grids for E-sciencE	*EGEE*	A series of projects (EGEE-I, EGEE-II and EGEE-III) funded by European Commission whose purpose is to construct production Grid infrastructure for researchers of many scientific disciplines along with a lightweight Grid middleware (gLite) for this infrastructure. [13, 98]
Experiment (in ViroLab terminology)		Experiment or in-silico experiment is a process that combines data and computations in order to obtain results [63]. In other words a *dynamic scenario* (See [150, section 1.1.2])
Experiment developer (in ViroLab terminology)		A computer science professional who creates experiment plans - often with the help of domain scientists. [177, section 2.4], [63]
Experiment Planning Environment	*EPE*	The ViroLab EPE is an Eclipse based tool for managing development process of experiment plans. It is on of the two main components of ViroLab presentation layer - the second one is the ViroLab portal. [96, 97]
gLite		gLite is a Grid middleware produced by EGEE project. It integrates several distributions, including LCG and VDT. Currently, it can be installed on Scientific Linux 3, 4 and 5. [47, 138, 140]

Table 2: Definitions (continued)

Term	*Abbr.*	*Definition or explanation*
Globus Toolkit	*GT*	Globus Toolkit is an open source software toolkit developed by Globus Alliance. It is intended for building Grid systems and applications. [88, 89]
Grid		A few definitions of the Grid are recognized [150, sec. 1.2.1], i.e. two definitions produced by Foster and Kesselman: "A Grid is a system that coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service." [87] "A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities." [91] and IBM's definition: "Grid computing enables the virtualization of distributed computing and data resources such as processing, network bandwidth and storage capacity to create a single system image, granting users and applications seamless access to vast IT capabilities." [150, sec. 1.2.1]
Grid enabled access to rich media content	*GREDIA*	A project funded by EC whose objective is to create a Grid application development platform with support to design, implementation and deployment of secure Grid business applications. Its two prototype applications are in the field of banking and journalism. [14, 15, 31, 44, 45, 133, 136, 137, 212]
GridFTP		GridFTP is a protocol based on the FTP protocol, developed by the Globus Alliance. It is GSI enabled and optimized for usage in the Grid environment. [4]
GridSpace Engine	*GSEngine*	GridSpace Engine is the main component of the ViroLab Virtual Laboratory. It is responsible for executing experiments and resource orchestration. It is the backend of Virtual Laboratory. [58, 107]
GridSpace Script	*GScript*	Script executed by GSEngine written in JRuby language [86, sec. 1.2.1]. In ViroLab a GScript is the main part of an experiment plan. [96, 153, 154]

Table 2: Definitions (continued)

Term	*Abbr.*	*Definition or explanation*
GSIFTP		Former name for GridFTP. [215]
LCG File Catalog	*LFC*	File catalog that maintains mappings between LFN(s), GUID and SURL(s). [1, 205], [47, chapter 7.4]
LFC Data Source	*LFCDS*	Software developed as part of this thesis.
LHC Computing Grid	*LCG*	LCG is a middleware system whose original purpose was to allow scientists involved in Large Hadron Collider experiments to efficiently run their programs in a distributed environment. It is a complete set of software for creating Grid systems. [38, 139]
Open Grid Services Architecture	*OGSA-DAI*	An architecture build on concepts and technologies from the Grid and Web services communities. It defines a uniform exposed service semantics – a Grid service; defines standard mechanisms for creating, naming, and discovering transient Grid service instances. OGSA also defines, in terms of WSDL interfaces, mechanisms required for creating and composing sophisticated distributed systems, including lifetime management, change management, authorization, and notification. [92–94, 208]
Open Grid Services Architecture Data Access and Integration	*OGSA-DAI*	Globus Alliance project that produces a web services framework for accessing and integrating data resources. The OGSA-DAI web services can be deployed within a Grid environment. [10, 129]
Proxy Certificate		From the RFC: “The term Proxy Certificate is used to describe a certificate that is derived from, and signed by, a normal X.509 Public Key End Entity Certificate or by another Proxy Certificate for the purpose of providing restricted proxying and delegation within a PKI based authentication system.” [220]
Remote File Input/Output	*RFIO*	Protocol used to access CASTOR Mass Storage System. [47, sec. 7.2.1]

Table 2: Definitions (continued)

Term	***Abbr.***	***Definition or explanation***
ViroLab Virtual Laboratory	*VL, ViroLab*	The thesis author found two definitions: ViroLab is a Grid-based decision-support system for infectious diseases. It is intended for individualized drug ranking in human immunodeficiency virus (HIV) diseases. [196] "The virtual laboratory is a set of integrated components that, used together, form a distributed and collaborative space for science. Multiple, geographically-dispersed laboratories and institutes use the virtual laboratory to plan, and perform experiments as well as share their results. The term experiment in this context means a so-called in-silico experiment - that is, a process that combines data and computations in order to obtain new knowledge on the subject of an experiment." [213]
Virtual Data Toolkit	*VDT*	VDT is a collection of Grid software (Condor-G, Globus, VOMS) along with its dependencies. It also includes Tomcat, MySQL and Apache plus many other software components. [104]
VL Data Access Services	*DAS*	ViroLab specific type of data source. It is an aggregation of hospital data accessed using OGSA-DAI. [17, 18, 20]
VLRuntime		Former name of GSEngine.

Science is what we understand well enough to explain to a computer.
Art is everything else we do.

Donald Knuth

When we had no computers, we had no programming problem either.
When we had a few computers, we had a mild programming problem.
Confronted with machines a million times as powerful,
we are faced with a gigantic programming problem.

Edsger W. Dijkstra

2 Introduction

2.1 Motivation

The work of a contemporary scientist no longer resembles the work of a scientist of the beginning of the twentieth century. Mathematicians very rarely use pen and paper to solve mathematical equations, tending to use programs like Mathematica [228], Mapple, Matlab (though, it is more oriented towards engineers) or their open source counterparts like Octave, Maxima[1]. Furthermore, proving mathematical theorems is not a pure intellectual work. An example would be the *four color theorem* [11, 12] which was proved in 1976 using a computer program that checked all special cases of 1936 maps. An increasing number of both general purpose and dedicated programs are applied in researcher's everyday work. I gave the example of mathematics, but this trend applies to almost all fields of science and technology with physicists seldom analyzing data on paper, preferring to utilize data analysis software such as ROOT. Engineers rarely crash cars, to check their safety; usually the simulation is more than satisfactory. Moreover, it can sometimes provide more detailed information than the actual crash test, together with considerations such as visualization, computer stress analysis, computational fluid dynamics (CFD), computer aided design (CAD) or more general computer aided engineering (CAE).

From the perspective of telecommunication, the work with the research tools at a distance is becoming increasingly widespread. The thesis author recalls his personal experience during the first beam day at CERN, where he had the pleasure of being in this remarkable place. He was in a large conference room where employees not directly related to the main event could observe its progress on a large screen. Many observers were watching remote consoles on their laptops to see the results from research facilities, while the team in the CERN Control Center (CCC) was conducting the first beam trial. A significant example of remote usage of scientific apparatus is the use of satellites as indicated by Hey and Trefethen [113] stating that European

[1]My former mathematics professor was very fond of Maxima. I suppose he used some kind of GUI, like WxMaxima, as it is very hard to use from the command line, in my opinion.

Space Agency (ESA) satellites generate 100 Gigabytes of data per day. However, the document cited is relatively old, so that figure may be even larger. The Hubble Space Telescope can also be mentioned here, because none of the research facilities used until now gave so much insight into our universe from the astrophysics point of view. Let us mention the Hubble Ultra Deep Field photograph just one of its breakthrough results, but probably even more can be expected from the Webb Space Telescope which is planned to be more advanced.

As industry and research centers have advanced, computer technique has stabilized. Nowadays, it is difficult to imagine that integrated circuits were designed by using large masks. However, today, hardware description languages, like Verilog and VHDL are used for this purpose and additionally analog electronics are often checked using programs like SPICE before building. Even historians whose discipline may seem a very humanistic, use of statistical tools (quantitative history) and employ computer technology for collaboration and sharing of documents. An example can be the project *Codex Sinaitucus* [211], which includes the oldest preserved complete copy of New Testament – handwritten 1600 years ago, which has been published collaboratively on the Internet by The British Library, National Library of Russia, St. Catherines Monastery and Leipzig University Library.

This phenomenon is called *'application pull'* [196]: the computer technology becomes ubiquitous in the world of science and scientists strive to solve more and more problems with the help of these technologies. If we take into account an experimental discipline, such as physics, we can note that the simulation, in addition to theory and experiment, has become a third way to practice science. On the other hand, in medicine, a predominantly empirical discipline, which is such because of the extreme complexity of systems it deals with, next to the terms *in vivo* and *in vitro*, yet another term, appears: *in silico* [196, 230]. The practice of science through computing is the essence of today's buzz word: *e-Science*. The interest in computer technology among researchers from different disciplines is a natural consequence of the possibility of process automation and rapid processing of large amounts of data, with a possibility of reaching goals that could not have been achieved using the available technology. With the increasing computerization of equipment and the large rise in accuracy, it follows, that the amount of data to be processed by computers will grow dramatically [113]. The existing classical model of computing is not able to meet these tasks. Very few supercomputers in the world are able to process data of such a huge size as human genome, though greater sizes may be required to be handled if there are more dimensions of data. The increasing efficiency of computers in accordance with Moore's law, which pleases everyone, is not able to provide the CPU resources, memory, disk and bandwidth required for processing an escalating amount of research data due to the volume growing much faster [113].

Fortunately, many researchers have anticipated this problem and have developed middleware that facilitates virtualization of resources in spite of administrative barriers, allowing collaborative use of processing and disk resources belonging to various institutions in different countries

and continents. These technologies have been named *'Grid technologies'* from electrical grid, where by plugging a plug into an outlet we have access to electricity without worrying where it comes from and who provides it. Similarly, 'Grid technologies' aim to provide a researcher computing power and storage resources, services, data from sensors, research results and knowledge. A scientist does not need to worry who delivers them[2]; his concern is the importance of the service provided. Thanks to virtualization of resources, 'Grid technologies' have enabled the use of the infrastructure of many different institutions and individuals (desktop Grids), to solve some problems of enormous complexity [115]. Usefulness of 'Grid technologies' has been confirmed by a number of applications from various fields of science and technology. Some examples are presented in table 3.

Table 3: Examples of Grid computing applications

Application	***Projects***
AEC[3]	InteliGrid [69, 70], Conflet Framework [176]
Air polution simulation	int.eu.grid[4] [195], LSAPM[5] [210]
Astrophysics simulations	MUSE[6] [183], G-HLAM [115]
Bioinformatics	myGrid [90, 203, 204, 230], LITBIO[7] [142], GADU[8] [186], SigWin-detector [120], The Virtual Instrument [52], HIPCAL and HUGOREP [39], Taverna [167], EUChinaGrid [148, 149, 179, 180]
Climate modeling	The Earth System Grid (ESG) [37]
Creating computer films	Big Buck Bunny[9] [157], VirtualRenderer[10] [182]
Design and optimization of casting processes	PartnerGrid [30]
Design of drugs, biopolymers, biomaterials and pesticides	CancerGrid [81], OpenMolGRID [193]

[2]Although it may not be completely true for research results and knowledge as we need to know their provenance.

[3]Architecture, engineering and construction

[4]Interactive European Grid

[5]Large Scale Air Pollution Model

[6]Multiscale Multiphysics Scientific Environment

[7]Laboratory for Interdisciplinary Technologies in Bioinformatics

[8]Genome Analysis and Database Update system

[9]The "Big Buck Bunny" was rendered using network.com, Sun Grid compute utility service. However, Foster [87] does not qualify Sun Grid Engine as a Grid due to its centralized control of the hosts it manages. See the *Grid* defintion in the table 2

[10]Grid renderer based on SunFlow [84, section 5], MOCCA [147] and Java Media Framework (JMF). The software was created by the thesis author for the Students' Scientific Association Session; section Applied Computer Science, in 2008. Do not confuse with other software with the same name [219].

Table 3: Examples of Grid computing applications (continued)

Application	***Projects***
Data mining	GridMiner [40–42], DataMiningGrid [200], DMGA [207], ESSE[11] [239]
Earth sciences	DEGREE [218]
FEM analysis	ParallelNuscaS [170, 171]
Flood forecasting	CROSSGRID [155]
Forest fire simulation	Medigrid [175]
General technical computing	GBPM[12] [126]
Heat Transfer Simulation	Grid Approach to Heat Transfer Simulation in Atomistic-continuum Model [2]
HEP[13]	ATLAS[14] [74, 100, 178], int.eu.grid [76], RMOST[15] [143]
Life and medical sciences	VL-e[16] [169, 226], MediGRID [79], Interactive Grid-Access for Ultrasound CT [111], G-HLAM[17] [189]
N-body simulation	G-HLAM [188]
Neural simulation	System of Parallel and Biologically Realistic Neural Simulation [187], Liquid State Machines and Large Simulations of Mammalian Visual System [145]
Parameter study	Saleve [77], P-GRADE [128], AppLeS [51]
Predictive maintenance	DAME[18] [121]
Searching large data sets	DAME [23], Ant-Home [125]
Videoconferencing	GlobalMMCS[19] [222], DiProNN [185]
Visualization	GVK[20] [135], River Soca Project [221], Medigrid [175], Multimodal Grid Visualization Framework [225], GVid [181], UniGrids[21] [36]

Grid infrastructure available today is impressive with many having been established. These include EGEE, DEISA, Grid'5000, TeraGrid, Open Science Grid, National Grid Service, D-

[11] Environmental Scenario Search Engine
[12] GRID Based Parallel MATLAB
[13] High Energy Physics
[14] A Toroidal LHC ApparatuS
[15] Remote Monitoring and Online Steering Tool
[16] Virtual Laboratory for e-Science
[17] Grid HLA Management System
[18] Distributed Aircraft Maintenance Environment
[19] Global Multimedia Collaboration System
[20] Grid Visualization Kernel
[21] Uniform Interface to Grid Services

Grid, NAREGI, China Grid [150, sec. 1.2.2]. In addition to traditional Grids there are *desktop Grids*, e.g. BOINC[22] [7], XtremWeb [82], SZTAKI Desktop Grid [127], DG-ADAJ[23] [172, 173] and Entropia [55]. Some of them have attracted a large community of volunteers who share their computer resources, particularly BOINC – 330,000 hosts [8] and SZTAKI DG – 12000 users donating more than 23,000 desktop machines [24]. Applications running on these machines have an impact on equally important disciplines of science as the traditional grids, with some examples being the search for cancer drugs [80], climate prediction [199] or research in digital signal processing [209] etc. The progress in setting up the infrastructure for e-Science, Grid software and hardware has been named the *'technology push'*. This advancement in computer technology resulted in the possibility, that today's infrastructure, at least in theory, will allow to meet some of the greatest challenges of science. But to dream of solving the problems of the scale, "from biological cells made of thousands of molecules, the immune systems built from billions of cells, to our society of more than 6 billion individuals interacting" [196] or simulating complex systems such as a galaxy made up of hundreds of billions of the stars [115], there is a need for integration of scientific applications and databases with the Grid infrastructure. This is a huge integration problem. Sloot et al. Sloot et al. [196] argue that a system-level approach is needed. The authors say that the bottom-up approach, i.e. creating applications that are independent and non-compatible with each other, and then integrating them, is definitely a wrong path. They justify their opinion by the fact, that in the latter case, even if we succeed integrating the applications, the problem of collaboration and interaction will remain. For the purpose of bridging the gaps between *'application push'* and *'technology pull'*, i.e. to utilize the great prospects of Grid technology, the *ViroLab Virtual Laboratory* was created which is a joint effort of several universities, hospitals, research institutes and companies (for more information, see [213]).

Its pilot application is a collaborative decision support system (DSS) for the treatment of infectious diseases, with an emphasis on HIV infections. The DSS system is already in a production stage and will soon be implemented in hospitals. A vision of this system has been widely presented in [196], while the results are contained in [198].

To effectively manage the data stored in heterogeneous EGEE / WLCG grid resources, the following data catalogs have been developed in recent years: European Data Grid Replica Location Service (RLS EDG) [35, 160], File Replica Manager (FiReMan) [163] and LCG File Catalog (LFC) [35]. Experimental data challenges show limitations and performance problems in EDG RLS, which was the motivation to create the latter two catalogues and withdrawal of RLS. Creators of FiReMan, and the LFC, as target users, took into account the HEP and biomedical community. Kunszt et al. [138] admitted: "Most importantly, the initial two application groups to work with gLite are the High Energy Physics and Biomedical communities,

[22]Berkeley Open Infrastructure for Network Computing

[23]Desktop GRID – Adaptive Distributed Application in Java

for whom data are stored mostly in files."[24]

An example of efforts made to adapt the Grid storage to the requirements of grid medical users is the introduction of Encrypted Data Storage (EDS) [1, 95]. Its design can be summarized as follows: ARDA Metadata Catalogue (AMGA) is used to store relational data of medical images, along with patient information. HYDRA library encodes and decodes files and is also responsible for producing and storing security keys. A special extended version of Storage Resource Manager (SRM) interface has been developed – SRM DICOM, which is compatible both with the EGEE / WLCG grid and with the DICOM[25] protocol. The EDS allows safely storing and transferring medical DICOM images retrieved from computer tomography (CT) or nuclear magnetic resonance (NMR) machines [26].

However, these solutions do not solve the "difficulty of use" problem that affects gLite storage services. FiReMan provides web-services interface, which cannot be said for the LCG File Catalog. LFC interfaces of the highest abstraction level are: the LCG-utils Command Line Interface (CLI), Python and Perl GFAL[27] and LCG-utils bindings along with related C application interfaces. No service-oriented API is available at the highest abstraction level in the case of LFC. Web-services APIs are available only at Storage Resource Manager (SRM) interface level[28]. Abadie et al. [1] argue that "Regardless of whether a grid user is a physicist, physician or an engineer, they should all be able to use the client utilities to access the gLite services and in particular the storage system". Surprisingly, there are scientific disciplines not normally related to computer science which have the most enormous storage and processing demands when it comes to computational research. These include computational chemistry and biology. *Computational scientists* as opposed to computer scientists do not necessarily have a broad information technology background, especially in the field of grid computing. They are experts in their discipline, e.g. physics, human physiology, pharmacy, biology, chemistry or environmental sciences. Nevertheless, these experts would benefit most from grid technology. Therefore, it is essential to help them employ grid resources in their fieldwork for the benefit of science and humanity.

Nonetheless, scientific users encounter many obstacles in accessing Grid services, which in the first instance is trying to obtain a Grid certificate. It is an intricate and error prone procedure which requires both patience (the certificate will not arrive immediately) and some

[24]The authors probably thought of DICOM[26] images stored in files. Experiences with ViroLab project showed, that biomedical information stored in relational databases is equally pervasive [18].

[25]Digital Image and Communication in Medicine

[26]TeleDICOM [48] project is worth mentioning here. It has been developed by students and alumni of the AGH University "Grupa.NET" scientific circle. TeleDICOM, although not a Grid project, shares some of the Grid ideas. It is a distributed system, allowing for interactive and collaborative work on medical documentation in the form of image files.

[27]Grid File Access Library

[28]An LFC SOAP API called Data Location Interface (DLI) is available. Still, it does not include authentication, is read only and not intended for end-users, but for Workload Management service.

technical skills, e.g. generation of `PKCS#12` certificates to be used in a browser requires knowledge of `openssl` command line parameters. A second complication is the management of grid certificates, generation of proxy certificates and keeping user credentials secure. Finally, the data handling through the command line interface is somewhat cumbersome, requiring remote login to an UI[29], sending files to storage elements (SE), publishing them in the LFC catalog and downloading files to the UI in order to be able to perform operations on these files. The mentioned operations incorporate unnecessary burdens. gLite data services are difficult to use for non-computer scientists.

The purpose of this thesis project is to relieve some strain from medical and scientific users by providing service-oriented API for the LFC catalog, managing user grid certificates and integrating the created API with the Virtual Laboratory, which is a comfortable grid environment that was designed especially for them.

2.2 Objectives

These four constituents can abridge the ambitions of the dissertation and the related project:

Adding support for data sources available through LFC catalogue. This will involve creating an API for experiment developers, that will allow effortless manipulation of these data sources, in particular reading and writing data, browsing directories, deleting files and directories and retrieving some of the document attributes – specifically their sizes[30]. This is the main aspiration of the thesis entailing several accompanying goals being enumerated in the ensuing items.

Reorganization of Data Source Registry (DSR) so that it will be possible to store all requisite information about data sources of the new type along with apposite user credentials.

Extending the DSR EPE plug-in, to enable browsing of data sources with the support of new data source type and to allow registering further data sources accompanied by relevant user credentials.

Integration with GridSpace Engine, in whose context, the DAC2 data access layer operates.

[29] Computer from which the Grid can be accessed.

[30] Dr. Maciej Malawski proposed the retrieving of these file properties.

2.3 Organization of the thesis

Chapter 3. Background In chapter 3 I will outline what has been done by the ViroLab team within this project and other endeavours that employ GSEngine: GREDIA and ChemPo. The section "The GridSpace Engine" will discuss the architecture of the GSEngine – an engine, on which the Virtual Laboratory experiments are performed, revealing what led the system designers to the selection of particular computer language for the expression of experiments, elucidating the techniques GSEngine brings to bear for the execution and optimization of remote operations on the Grid, together with the strategy it uses to conceal specifics of implementation technologies. The section 3.3 – "The Virtual Laboratory" delineates the conceptual vision of Virtual Laboratory and identifies modules directly related to its operation being Provenance Tracking System (PROToS), Query Translation Tool (QUaTRO), Experiment Management Interface (EMI) and Experiment Planning Environment (EPE). The section 3.5 portrays the GREDIA and ChemPo, i.e. further undertakings making use of the GridSpace Engine, while section 3.4 comments on the ViroLab data access layer, including VL Data Access Services (DAS), Data Resource Registry and Data Access Client 2 (DAC2).

Chapter 4. Needs to be addressed / Problems to be solved Chapter 4 presents the challenges that must be tackled by the thesis author together with their perspective. Nevertheless, section 4.1 portrays the organization of data access in gLite, taking into account LFC catalogue with 4.2, demonstrating various alternatives to provision access to LFC and to files published in it. The clause 4.3 illustrates difficulties with the management of users' grid certificates, their protection and usage, with an accompanying discussion on feasible resolutions of these problems. Finally, section 4.4 will demonstrate the current shape of EPE DSR plug-in and new requirements it needs to fulfil. In brief, chapter 4 sketches project requirements as an informal discussion. Formalized description will be delivered in chapters 6 and 7.

Chapter 5. Related work Chapter 5 alludes to miscellaneous projects, which touch upon comparable substance mooted in the dissertation. Section 5.1 refers to other Virtual Laboratories, such as myExperiment, Triana, Kepler and more low-level gLite, whereas 5.2 talks talks about undertakings that strive for making Grid more service-oriented, for instance Open Grid Services Architecture (OGSA) and Semantic OGSA (S-OGSA). Thereupon the clause 5.3 will shed light on how diverse Grid projects read and write data, which is noteworthy in the analysis of scientific literature[31], which was carried out by the thesis author. An overwhelming majority of projects still store data in relational, XML or occasionally, object databases located outside of Grid. However, it is of no interest from the thesis point of view and therefore will not be discussed. Nevertheless, projects of interest in the thesis are those which store and read on Grid and several such projects will be discussed. Besides these projects sundry grid file systems

[31] Cracow Grid Workshop 2004 – 2007 (CGW'04 – CGW'07)

will be identified alongside cloud computing file systems, as cloud computing is an area to a certain extent linked to grid computing. Eventually, section 5.4 will elucidate diverse libraries providing access to gLite storage resources such as, LFC C/C++ API, Grid File Access Library (GFAL), some low-level application interfaces and wrappers in assorted programming languages.

Chapters 6. General software requirements 7. Detailed requirements present requirements to be met by software developed within the thesis.

Chapter 8. Design description – this illuminates the chosen architecture of LFC Data Source, highlights decomposition into design entities and illustrates dependencies between the entities together with their internal structure and interaction patterns. Furthermore, it communicates component interfaces: everything designers, programmers and testers need to know which will allow correct use of the functions delivered by the entities.

Chapter 9. Verification and validation Chapter 9 describes the testing approach for functional and performance tests. Both types of tests are divided into those that assess LFC DS connector and those that test LFC DS client library.

Chapter 10. Conclusions Section 10.1 summarizes achievements of the thesis project and how they were achieved, while section 10.2 is an analysis of potential extensions of LFC DS presenting possible improvements, such as 'fine grained role-based security'. An additional important aspiration would be to provide superior performance and scalability. An element that could also be taken into consideration when envisioning further enhancement, is a more generic API and accessible from languages other than Java. Ancillary refinements are also deliberated.

Computer science is no more about computers
than astronomy is about telescopes.

Edsger W. Dijkstra

3 Background

I gave an overview of grid computing and motivation to develop virtual laboratories in the *Motivation* section of the former chapter. This chapter will focus on our Virtual Laboratory and software developed by ViroLab consortium, especially three of its members: ACC Cyfronet[32] (GSEngine, EPE, EMI, GRR, DSR, AppRepo, GrAppO, MOCCA, security components), GridwiseTech (ViroLab Portal, VO management, security components) and HLRS[33] (VL Data Access Services - DAS).

3.1 The GridSpace platform

GridSpace Engine [58, 107]**,** abbreviated GSEngine, is a runtime environment for the Virtual Laboratory. Indeed, it was formerly termed the Virtual Laboratory Runtime (VLRuntime). At the release of version 0.2.6 its name was changed to GridSpace Engine, to reflect generality of this software, i.e. that it can be used in a broader spectrum of problems than those related to Virtual Laboratory.

The aim of the GSEngine is to enable access to computing and storage resources and to coordinate the execution of experiments written in GScript language, i.e. JRuby extended with capabilities provided by specialized GSEngine components. Thanks to dedicated libraries, GSEngine facilitates interactive execution and monitoring of dynamic execution scenarios, otherwise called experiments. There are different methods of providing the source of an experiment to the GSEngine (see figure 1):

- Executing the experiment code line by line using a dedicated API.
- Passing the whole source code using the API.
- Using a command line client to pass the experiment code.
- Finally, one can load an experiment script from the experiment repository [109], which is a software component based on Subversion (see figure 2). It is the most common way of executing experiments when they reach production stage.

[32] Academic Computer Centre Cyfronet AGH
[33] High Performance Computing Center Stuttgart

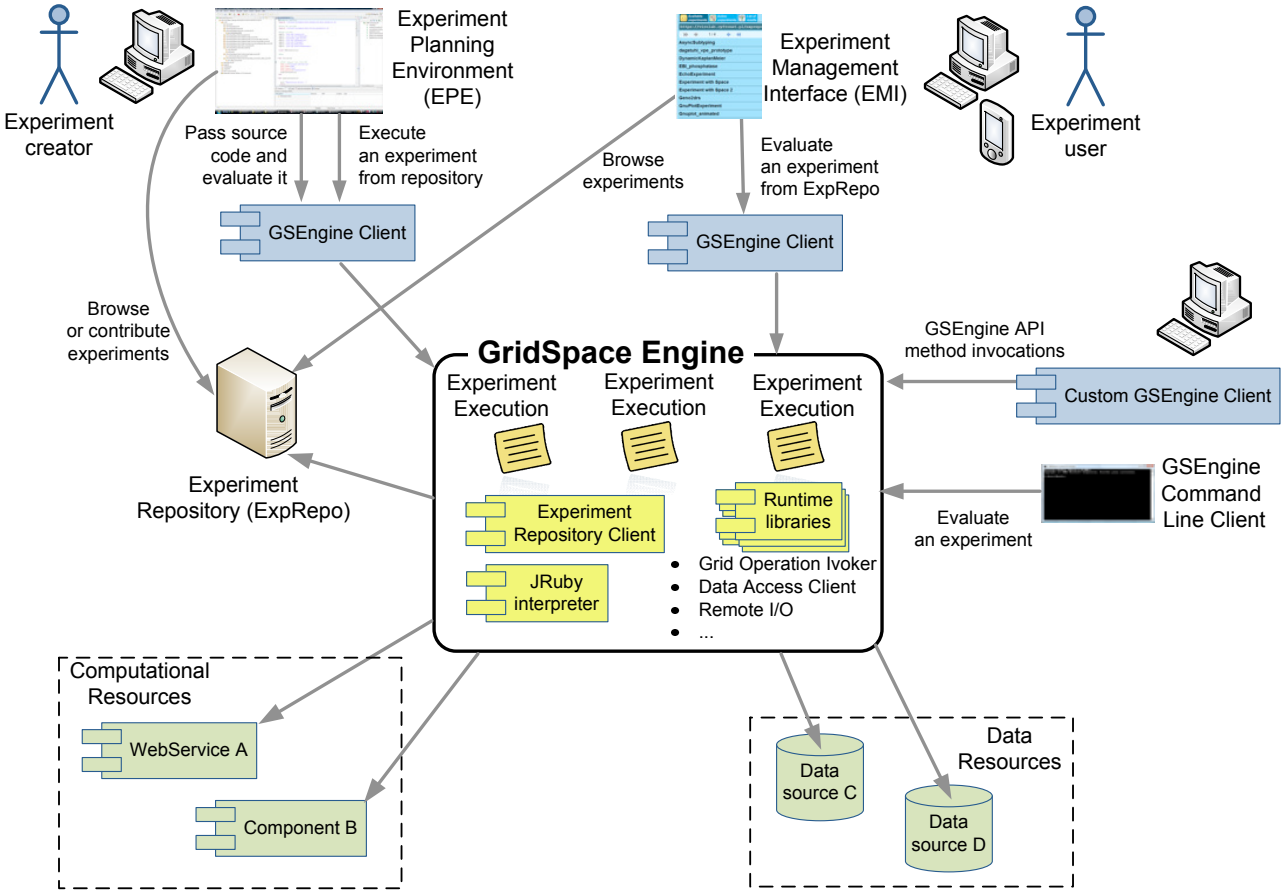


Figure 1: GridSpace Engine in Virtual Laboratory environment. The figure illustrates the role of GSEngine Server, which orchestrates access to data and computational resources. In addition, GSE various client tools are portrayed, cf. figure 1 in [58].

As Ciepiela et al. [58] indicate, the main goal of creating GSEngine was to separate the client programs that assist in planning and executing experiments, from the engine that actually effectuates them. It allows the GSEngine to be shared independent of the users' machines, empowering it to conduct long-running experiments on user's behalf, taking advantage of grid resources. Such an approach to performance of 'in silico' experiments gives the opportunity to carry out calculation-intensive experiments to the dispersed groups of users, probably connecting to the GSEngine from mobile devices.

Projects, such as Triana, Kepler, myGrid, made workflows available to the users, as a means to specify the experiment execution plan. An alternative approach would be to use scripting language for that purpose which was a choice for projects Athena[34] [100], where as a 'glue' language Python is used, and Geodise, which employs Matlab and Jython scripts.

The Virtual Laboratory authors, by contrast, chose JRuby language. There are several reasons that led them to this decision:

- The JRuby project is distributed under CPL/GPL/LGPL licenses, which makes it suitable

[34] ATLAS software framework

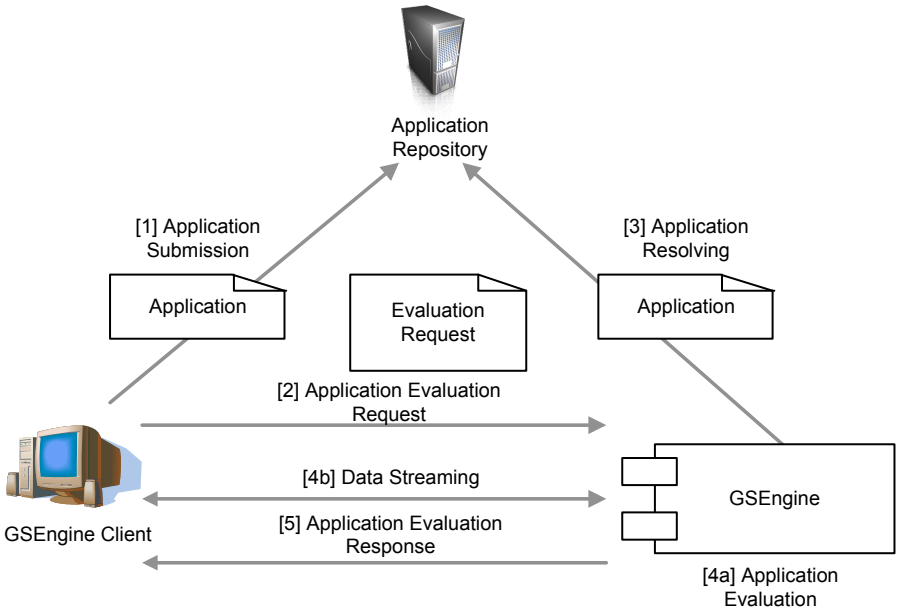


Figure 2: A process of executing an experiment from Experiment Repository. Application Repository, in ViroLab terminology termed 'Experiment Repository' or 'ExpRepo', is used to share subsequent versions of experiments. After experiment submission by an experiment developer (1), the experiment becomes available to experiment users and other developers. When they pass an experiment execution request to GSEngine (2), the experiment code is downloaded (3), evaluated (4a) and the results are streamed to the client tool during execution (4b). Eventually, the experiment ends and GSEngine sends its status and response to the client (5) [58].

for the GSEngine being issued under the GPL licence. Bubak et al. [43] emphasised that because of project research character, they preferred FLOSS software[35].

- Numerous libraries written for Java platform are accessible from JRuby language.
- JRuby is a very expressive and purely object-oriented programming language allowing for articulation of any logic complexity with additions of new functionalities being simplified by developed metaprogramming [86].

GSEngine, as previously mentioned, contains modules providing access to grid resources with Grid Operation Invoker (GOI) facilitating execution of remote operations on Grid and Data Access Client a façade for access to typical data resources, for instance MySQL and PostgreSQL relational databases, unstructured data sources, e.g. WebDAV and atypical, specialized resources, e.g. Data Access Service (DAS) aggregations [18]. Apart from the GOI and DAC libraries, there is a component making possible run parameter requests during the script execution, for instance, a request for patient ID. From the developer's point of view it allows for dynamically creating forms from the script code, which is a very convenient feature. In addition, libraries for streaming results to the client tools exist.

[35]Free Libre/Open Source Software

Outside of GSEngine, in the context of Virtual Laboratory, client tools have been developed with Experiment Planning Environment (EPE) helping design the experimental plans, Experiment Management Interface (EMI) serving the purpose of performing and managing experiments by end users. These are present only in the case of Virtual Laboratory. Other projects, that employ the GSEngine, provide disparate tools, e.g. in the GREDIA project the role of EPE is occupied by Application Execution Planning Tool – abbreviated AEPT or the Developer GUI.

Among other responsibilities, an important GSEngine task is to manage user sessions, which allow a Single Sign On (SSO) access to computational and data resources. Apart from these features, GSEngine monitors access to data and execution of grid operations, collects log messages and status of the performed experiments, so as to convey this information to the monitoring tools and client programs.

Grid Operation Invoker After this short introduction to the GSEngine I will present the Grid Operation Invoker [32, 34, 154]. DAC will be discussed in section 3.4.

The goal that VL team members endeavoured to achieve when envisioning GOI was raising grid operations to a similar high level of abstraction as found in ordinary JRuby methods [33], which is a complicated matter due to the diversity of grid middlewares. Bubak et al. [43] admit, that apart from the support of divergent types of users and heterogeneity of resources it was one of the biggest challenges to be unravelled. Despite the difficulties, the creators of GOI succeeded and delivered experiment developer, a high-level object-oriented API leveraging the following technologies:

1. WebServices based
    - Stateless based on SOAP and WSDL purely
    - Stateful extension of WebServices: Web Services Resource Framework (WSRF)
2. Component technologies: MOCCA [151], ProActive [50]
3. Job-oriented systems: EGEE and DEISA

The GOI authors tackled the assortment of grid technologies by introducing 3 levels of abstraction (see figure 3). Every grid object is an abstract entity, which can perform a set of operations[36] which are invocated from the GScript, but executed on remote machines located somewhere on or outside Grid. Every Grid Object can have a number of implementations in a variety of technologies, with each implementation representing the same function. Similarly, each implementation may have an assortment of instances running on grid resources. Machine load, class of equipment, as well as speed of network connection may be dissimilar; consequently,

[36]In object-oriented programming 'an operation' is sometimes described as an act of sending a message to an object. Ruby also supports such a mean of operation invocation using the 'send' method semantics.

discrete instances of the given Grid Object possibly will work with disparate performance. To relieve the user from the dilemma of deciding which instance to choose, the Grid Application Optimizer (GrAppO) selects the best instance for executing operations, with the user needing to know only the characteristics of a Grid Object that they use, i.e.:

1. Whether it is stateful or stateless.
2. If the method invocations are synchronous or asynchronous
3. If the objects are shared by other users or solely by the user.

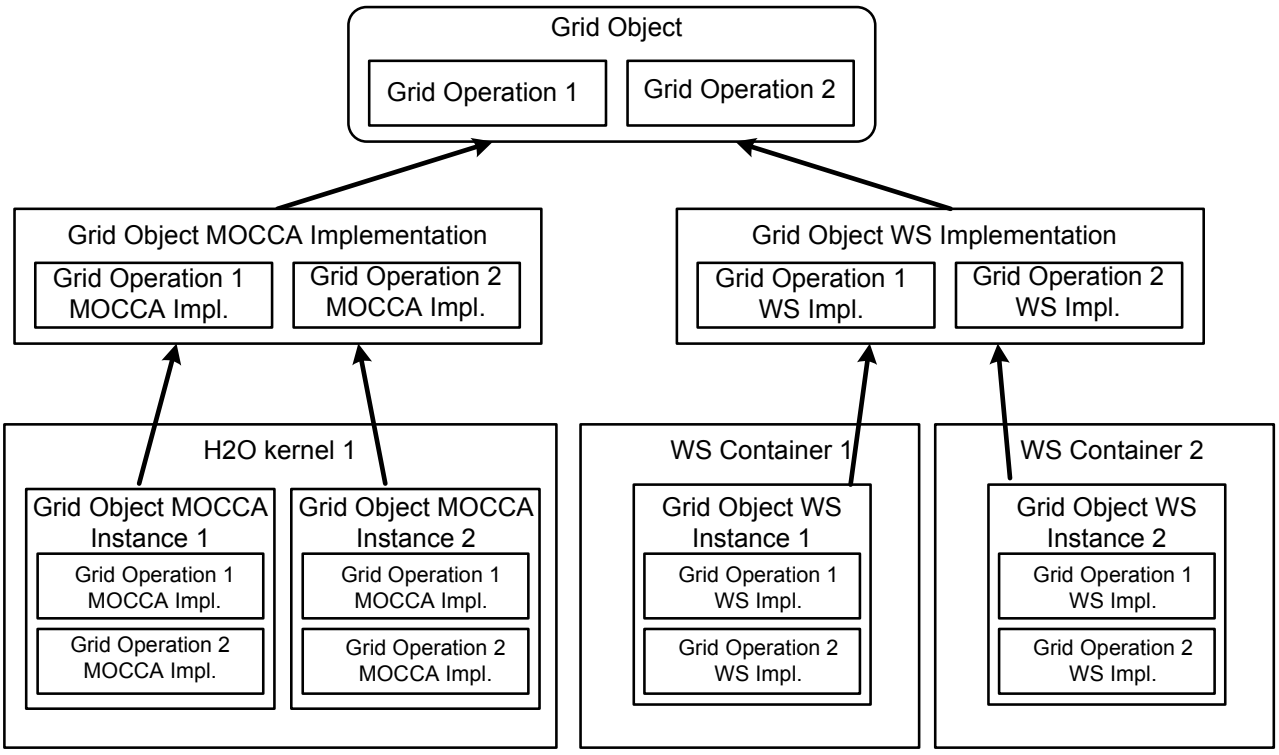


Figure 3: Three levels of Grid Operation Invoker abstraction [33].

GOI is a light library creating Grid Object proxies that in turn maintain remote method invocations in appropriate technologies. The GOI adapters are written in JRuby language and call relevant Java libraries for specialized operations. Analogous approach has been chosen in DAC with another similarity being the usage of external Data Source Registry, which contains information about data sources and user credentials. GOI, on the other hand, uses Grid Resource Registry (GRR) that provide Grid Object technology particulars (figure 4). The role of GRR and DSR can be likened to the role of Enterprise Service Bus of business applications developed in conformity with Service Oriented Architecture model. Apart from high-level APIs to Grid Objects, experiment developers have the possibility to use lower-level application interfaces. They can bypass the GrAppO by passing an instance ID or choosing a technology adapter without the help of GrAppO, which in the case of higher-level API is selected automatically.

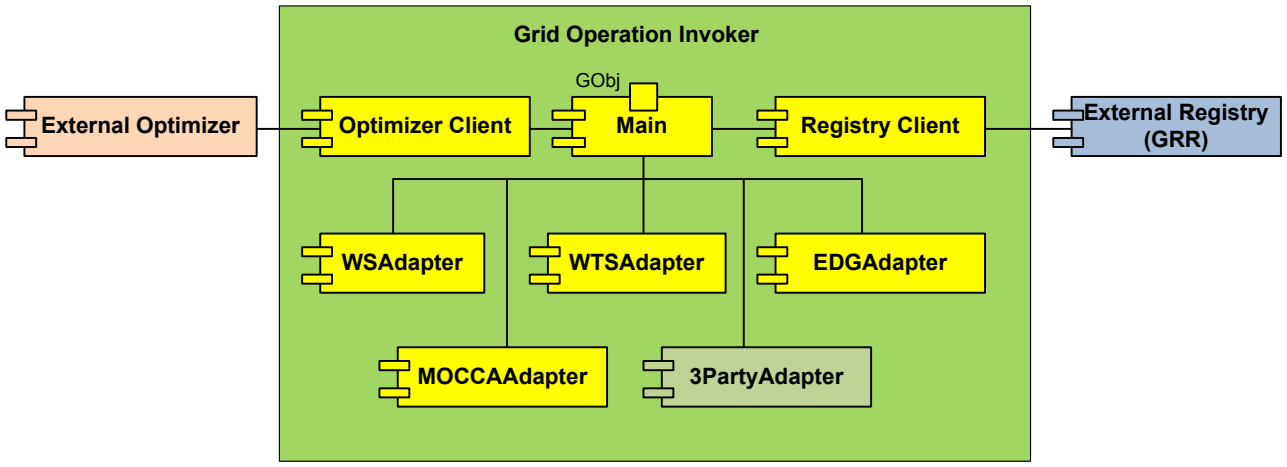


Figure 4: Grid Operation Invoker architecture and external components, with which it communicates [33].

Grid Application Optimizer [132, 152] or GrAppO is an optimization engine for the GridSpace Engine responsible for making most effective use of grid resources. GrAppO is underpinned by systems for monitoring [26, 57] and collecting provenance data, with its decisions being taken on the basis of information retrieved from Grid Resource Registry (GRR), agiLe MONitoring ADherence Environment (leMonAdE) and Provenance Tracking System (PROToS) [223]. GrAppO offers 3 modes of optimization: short-, medium-, and far-sighted (figure 5). leMonAdE is illustrated in figure 6. Data Access Client, which is also one of the core

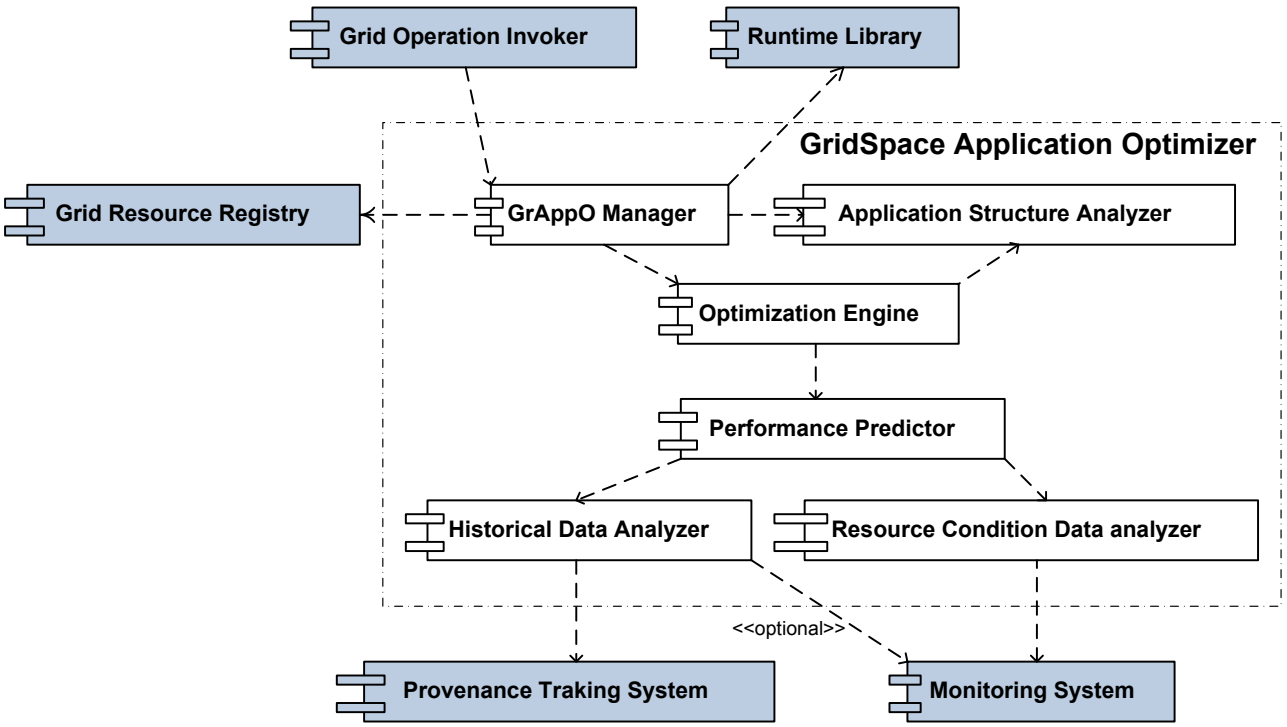


Figure 5: GrAppO architecture [152].

GridSpace platform elements, will be discussed in section 3.4.

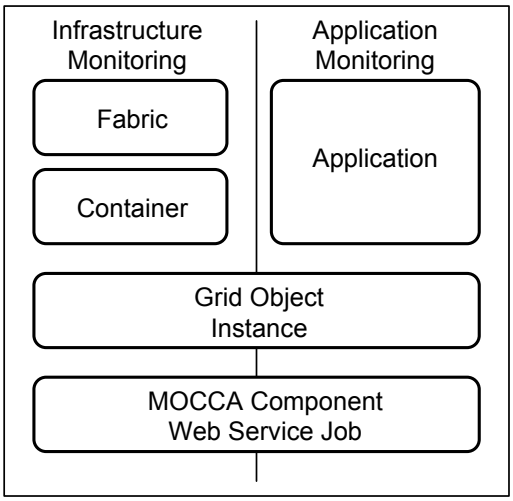


Figure 6: agiLe MONitoring ADherence Environment (leMonAdE) architecture divided into two parts: Infrastructure monitoring and Application Monitoring [152].

3.2 GridSpace Engine deployment

The engine of GridSpace Platform may be incorporated into a user's application, started from command line as a local instance or can be launched as a remote accessible server, which can be contacted by using a dedicated client library or a client command line tool. Each of these possibilities will be discussed in this section.

GSEngine command line tools Shell scripts that fall into this category are

- `gsel` – GSEngine Evaluate Locally
- `gseql` – GSEngine Entity Query Local
- `gsdql` – GSEngine Data Query Local
- `maketrusted`
- `gses` – GSEngine Server
- `gsec` – GSEngine Evaluation Client
- `dotrust`
- `gseqc` – GSEngine Entity Query Client
- `gsdqc` – GSEngine Data Query Client
- `gsep` – GSEngine Proxy

Commands above are available for both Windows and Linux and are contained in three packages ('xxx' indicates version number):

***gsengine-client-vl-xxx*: `gsec`, `gseqc`, `gsdqc`, `dotrust`** – The main component of this bundle is client command line application `gsec` that connects to remote GSEngine Server passing GScript code. Additionally commands (`gseqc`, `gsdqc`) that utilize data access façade of a remote GSEngine Server are provided. Before running the client, it is necessary to add a server certificate to trust store – this is the purpose of `dotrust` script.

***gsengine-vl-xxx*: `gses`, `gsel`, `gseql`, `gsdql`, `maketrusted`** – Package that ships with GSEngine

Server, a local embedded version (`gsel`) and a remote, accessible server (`gses`), together with an utility (`maketrusted`) to generate server key pairs with self-signed certificate and tools to access data access façade of a local GSEngine Server (`gseql` and `gsdql`).

***gsengine-proxy-vl-xxx*: `gsep`, `maketrusted`, `dotrust`** – GSEngine Proxy (`gsep`) is a module that acts like a server from the point of view of client and like a client, enables the passing of messages between actual client and server. Package also contains `maketrusted` and `dotrust` scripts that have an identical purpose that as in bundles above, i.e. before using GSEngine Proxy server key pair together with certificate needs to be generated (using `maketrusted`) for use with actual client. Furthermore, adding actual server certificate (i.e. certificate of a server that is the destination of messages) to trust store using `dotrust` is obligatory in order to enable communication with this server. Execution of these two scripts: `dotrust` and `maketrusted` is required as GSEngine Proxy communicates with both client and server.

Installation of the bundles above first involves extraction of bundle archive into a directory of user's choice, while the subsequent steps depend upon what package user wants to install.

If the user wishes to install a local embedded GSEngine, there is a need to configure Java-style properties file `config/engine.properties` adjusting values such as path to JRuby interpreter, RMI registry port where GSEngine JMX[37] server will be registered, application correlation id (*acid*) prefix and credentials to stores containing results and metadata. A user wanting to utilize their own GridSpace infrastructure, i.e. security providers, data, metadata, result and ontology stores, resource registries, application repositories etc., must modify `config/gridspace.properties.xml` appropriately, usually substituting URLs from this file to those pointing to their own services.

If GSEngine server is to be used remotely, in addition to steps above, a generating key pair with certificate is requisite. `maketrusted` is used for this purpose – the only parameter needed is a name to be used for subject of certificate and file name.

On the other hand, when installing a GSEngine client, apart from extracting the bundle, the only step required to make installation valid is to execute `dotrust` script adding a server certificate to GSEngine Client trust store. No additional configuration is required. GSEngine Client bundle is compact compared with GSEngine Server package with 1.7 MB size compared to 50.4 MB (as of version `0.8.1_5`) – this is because the client ships only with necessary libraries. Therefore, an end-user does not have to install heavyweight software with many configuration options. Moreover, and most importantly, such an installation solves problems with server certification – as it was mentioned, the only mandatory step for client is to add server certificate to trust store using simple `dotrust` command. It is a very modest requirement compared to analogous collaborative virtual laboratory engines, e.g. my-

[37] Java Management Extensions

Experiment [67]. In myExperiment, a user wanting to connect to myExperiment service is equipped with *OAuth* library, whose configuration incorporates many steps and prerequisites. Particularly, a user has to unpack `oauth4myexp.tar` bundle into a web folder of a web server that supports PHP[38]. Then a user has to determine URL of the deployed web application, e.g. `http://<someserver>/<somefilepath>/oauth4myexp/`. Subsequently, user logs into myExperiment server, opens OAuth page and clicks *Register Client Application*, enters its details, specifically *Name*, *Main Application URL*, *Callback URL* and optionally *Support URL* and chooses which API calls the client application, which will be able to perform in *Permissions* section [162]. After completing the form and achieving successful registration of client application, user gets so called *Consumer Key* and *Consumer Secret*. The next step is configuration of recently deployed OAuth PHP application – this is done by going to its URL followed by `config_generator.php` suffix, e.g. `http://<someserver>/<somefilepath>/oauth4myexp/config_generator.php`, pasting *Consumer Key* together with *Consumer Secret*, and clicking *Get Access Token* button, which will redirect back to myExperiment website so as to authorize access token for client application. After accessing myExperiment website for the second time, user checks *Authorize Access* checkbox and clicks *Save Changes*, which redirects him or her back to configuration generator page. On the resulting page, user is presented with `Base64` encoded configuration, which is pasted into configuration file `Config.php` in directory of PHP client application. Afterwards, user loads again the PHP application – on successful connection to myExperiment server, the *Connected to Server* field will be displayed, which indicates that the client PHP application can make API calls. Having in mind this procedure, it is not an exaggeration, to say that GSEngine Client deployment is straightforward compared to other solutions.

As regards installation of GSEngine Proxy, it incorporates generation of server key pair and certificate, which will be added to clients' trust stores connecting to this proxy. Additionally, `dotrust` must be invoked with destination server certificate, so that the proxy will be able connect to it. GSEngine Proxy passes messages back and forth from client to server. Moreover, it manages a set of GSEngine Servers acting as workers.

With regard to executables provided by the aforementioned packages, there are scripts to pass GScript code to GSEngine Server or to invoke GSEngine commands interactively; there are commands to approach data access façade and already discussed scripts to generate key pair and certificates and acceptance of certificates. Commands enabling evaluation of code by GSEngine are `gsel` and `gsec`. The former executes GScript in a local embedded version of GSEngine while `gsec` connects to a remote GSEngine Server. GScript code is provided either using a local file name, by specifying application URI to be downloaded from application repository or passed interactively using system console. On the other hand, `gseql`, `gsdql`, `gseqc` and `gsdqc` are used for querying GSEngine data access façade either locally (`gseql`, `gsdql`) or remotely (`gseqc`, `gsdqc`). Finally, to launch a remotely accessible GSEngine Server instance

[38]Hypertext Preprocessor

`gses` is used. For more information, in particular regarding command line arguments of these scripts, the reader is advised to consider [62].

GSEngine API Another option of GSEngine deployment is to incorporate it into user's application and to use its capabilities programmatically. If the application dependencies are managed using Maven, it is sufficient to add certain Maven artifacts to `pom.xml`, particularly `gsengine-api` and depending on the mode of operation, `gsengine-core` for local embedded version of GSEngine or `gsengine-client` for a GSEngine client connecting to a server. Maven repository location and complete XML code snippets can be found in [62]. Otherwise, if a developer does not use Maven, they can add GSEngine dependencies manually by downloading GSEngine bundles: *gsengine-vl-xxx* in the case of local embedded server or *gsengine-client-vl-xxx* when accessing a remote server, and afterwards, adding the content of lib directories into project *CLASSPATH*.

After adding dependencies, developer has access to GSEngine interpreter using `cyfronet.gridspace.engine.impl.interpreter.EmbeddedInterpreter` class for embedded interpreter and `cyfronet.gridspace.engine.client.RemoteInterpreter` for remote interpreter respectively, while both are subclasses of `cyfronet.gridspace.engine.AbstractInterpreter`, which defines evaluate method. As with command line client, when using a client library that connects to a remote GSEngine server, appropriate server certificates must be present in trust store, which is as an argument to `RemoteInterpreter` constructor or a constructor of remote data access façade, depending on what class developer uses. API for execution of GScript applications accepts similar parameters as its command line counterparts, i.e. among obvious server URL and port, there are user handle, applications URI, global constants, arguments, log-level and several more (although for optimization policy there is no counterpart parameter in command line tool). What differs from command line tools mostly, is the ability to receive evaluation *callbacks*. These include notifications about completion of application, about event of application setting its status, writing data to output or error stream (data is passed to callback method as an argument) and notifications of storing results or about exceptions being raised. Additionally, GSEngine expresses various requests by invoking a callback, e.g. demand for providing input, displaying content, providing a file or additional script. Furthermore, interactive mode of GSEngine operation retrieves GScript source code by means of a callback. Another capability of GSE API is to abort a running GScript application. In order to do this, developer passes application correlation id returned by evaluate method of `cyfronet.gridspace.engine.AbstractInterpreter`. Apart from the ability to invoke script code, developer has access to data access façade using `cyfronet.gridspace.engine.dataaccess.DataAccessFacade` class, which enables queries over data sources and retrieving entities from a data source schema (a table in relational database) – for more information the reader is counselled to take [62] into consideration.

API mentioned above is a Java API. As far as JRuby GSEngine API is concerned, some information, especially about dynamically generating forms and about runtime objects and properties, can be found in [62] while detailed information about GSEngine JRuby API can be found in RDoc documentation.

3.3 The Virtual Laboratory

Virtual Laboratory – advancing treatment and research on HIV[39] One of the reasons, why the HIV-1 virus is pernicious to humans, is the fact that it kills the T-helper cells (Th), holding CD4 antigen (more than 90% lymphocytes possess CD4 glycoprotein). In the absence of treatment, the disease may lead to diminution in number of Th lymphocytes to a level below 200 cells per μL. As a result, human immune system loses its ability to defend from pathogens, leading to Acquired Immunodeficiency Syndrome – AIDS. "The human immunodeficiency virus (HIV) and other retroviruses show extensive genomic variation, which is primarily due to error-prone replication by the viral reverse transcriptase (RT) enzymes." [64]. This is the root stumbling block in finding drugs and vaccines against HIV virus and other retroviruses[40]. Despite this complication, there have been divers attempts to treat HIV infections:

- Disruption of virus replication process by inhibiting the reverse transcriptase (RT) enzyme activity, which is the principle action of drugs such as AZT[41]. Unfortunately, impervious mutations spawn promptly.

- Taking advantage of extensive genomic variation of retroviruses which has the potential to cause an error catastrophe [78]. This phenomenon of error catastrophe occurs, when the quantity of virus mutations is so enormous, that it loses its genetic identity and effectiveness with KP-1212, a drug that tries to exploit it [112].

- Nowadays, the most successful HIV therapy is HAART – Highly Active Antiretroviral Therapy. It combines at least 3 antiretroviral drugs; the initial stage will usually include "favirenz or a ritonavir-boosted protease inhibitor plus 2 nucleoside reverse transcriptase inhibitors (tenofovir/emtricitabine or abacavir/lamivudine)" [110]. The amalgamation of medication inhibits the creation of drug-resistant virus mutations.

- Attempts to boost Th production by averting physiological[42] involution of thymus in HIV-infected patients by administering growth hormone (GH). Napolitano et al. [161] report, that their therapy caused upsurge in Th production by 30%.

[39] Human Immunodeficiency Virus

[40] Retrovirus is an RNA virus that replicates itself using *reverse transcriptase – RT* creating DNA from its RNA. One of the major known retroviruses is HIV-1.

[41] Cluster of differentiation 4

[42] i.e. being in accord with and characteristic of the normal functioning human organism.

The HAART therapy, though the most effective at the moment, must be matched individually to a person receiving treatment. Wrong choice of medication may cause susceptibility to drugs and immunity to further treatment. Wrong choice of medication may cause reduced susceptibility of HIV to drugs and immunity to further treatment. Moreover, according to Stoica et al. [206] "The development of new antiretroviral therapies for HIV is at an impasse". At this point, ViroLab project overcame the aforementioned difficulties by alleviating the risk of wrongly prescribed drugs, not only in the context of a single person but also in the perspective of complex human interactions.

ViroLab Virtual Laboratory [108, 197] delivers a platform for cooperation between scientists of different disciplines, located in distinct distant places around the world, but of the same scientific ambitions. The main goal in establishing this platform, which brings together efforts of computer scientists, virologists, epidemiologists and experiences physicians, is to help advancing HIV research. The system integrates biomedical information on viruses, i.e. facts on proteins and mutations, patients (what virus mutation are they infected with), treatment (drugs admitted) and literature (interpretations of drug resistance).

ViroLab copes with HIV medicine decision processes on all levels of detail [198]: from molecular [190], through the molecule groups, cell-level and the whole immune system to networks of human interactions (see figure 7). For instance, at the molecular level, there are molecular dynamics (MD) simulations performed concerning how drug compositions behave in the presence of virus proteins with binding affinities being calculated to estimate reactions to drugs and an *'individual transmission parameter'*[43]. Sloot et al. present the successes of their simulations in [198], asserting that they can model all phases of infection: from acute, through chronic, drug treatment to onset of AIDS and that their results correspond to clinical data. Similarly, they reported that their simulations of human interactions by means of complex networks, very accurately recreated the development of HIV infections in United Stated. Without doubts, it is an example of conducting simulations at all scales as it was formulated in [115]. Based upon precision of these simulations, the authors of [198] formulated a supposition that models they elaborated on, will help making advised decisions regarding treatment and impeding the proliferation of HIV virus. They have provided virology scientists with powerful tools to investigate the impact of various avowed strategies and drug therapies.

Virtual experiments One of the central ideas behind Virtual Laboratory is the process of conducting experiments – experiment pipeline (see figure 8). An experiment (or in-silico experiment) is a process that combines data and computation in order to obtain results; in other words a 'dynamic scenario'. In the profession of biologist or chemist, experiment is carried out using available substrates and processes to acquire new knowledge. Likewise, in an in-silico experiment, an experiment creator exploits available data sources and computational resources,

[43]Probability of infection during sexual contact.

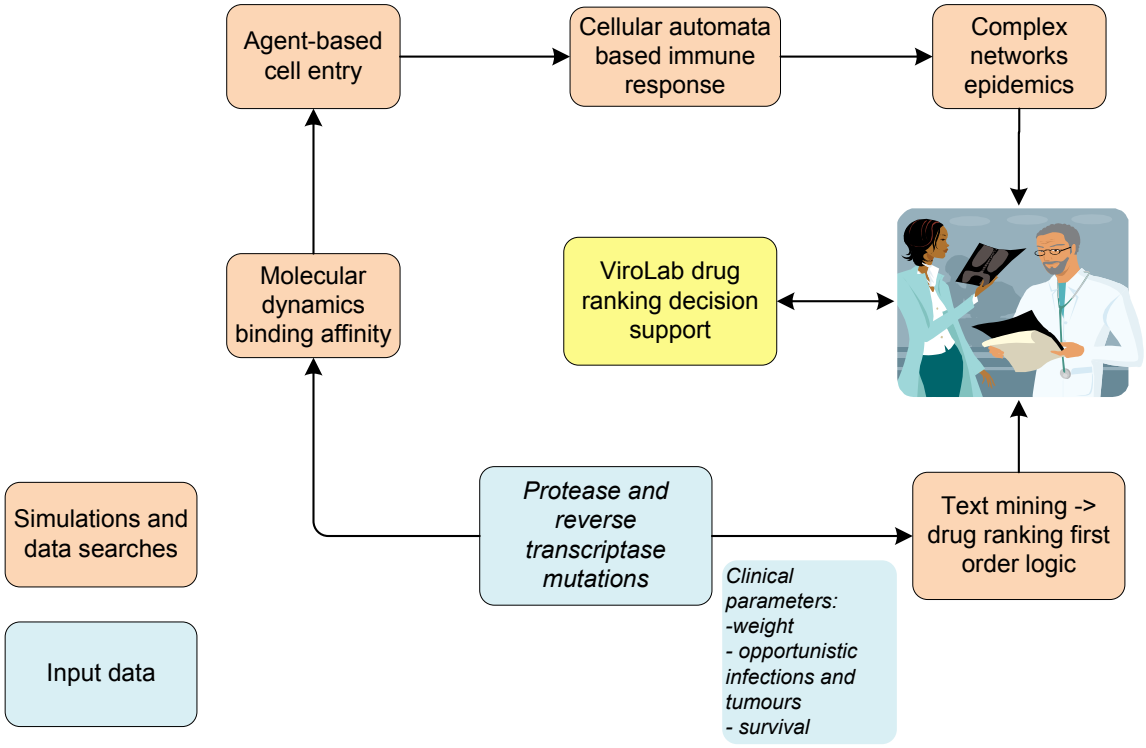


Figure 7: Virtual Laboratory framework conceptual components. They can be used separately or through ViroLab drug ranking decision support system that integrates them into one application [198, 217].

the result of which is new knowledge similar to traditional experiments. In contrast to typical computer programs, such as knowledge gained from experiments, in vivo, in vitro or in silico, we must know where it comes from. In order to apply knowledge in important life threatening matters in question, e.g. which combination of drugs will be able to cease replication of a given virus mutation that a patient is infected with, a clinician must be able to verify origin of information. For that purpose, within ViroLab project Provenance Tracking System (PROToS) has been brought into being, storing provenance data together with QUery TRanslation tOols (QUaTRO), enabling medical users to perform provenance queries on clinical data integrated with ViroLab, along with Semantic Event Aggregator – a component for building ontologies from monitoring data [25–29, 177, 223]. The PROToS architecture has been depicted on figure 9.

Types of users Users, according to Virtual Laboratory concept [46, 156], are divided into:

- Clinicians employing DSS in their clinical practise to better treat HIV-positive patients.
- Scientists, i.e. clinical researchers, virologists and epidemiologists, who are both creators and users of experiments, which analyse federated datasets, to obtain new knowledge which is useful when making recommendations for clinical decisions and to support their research.

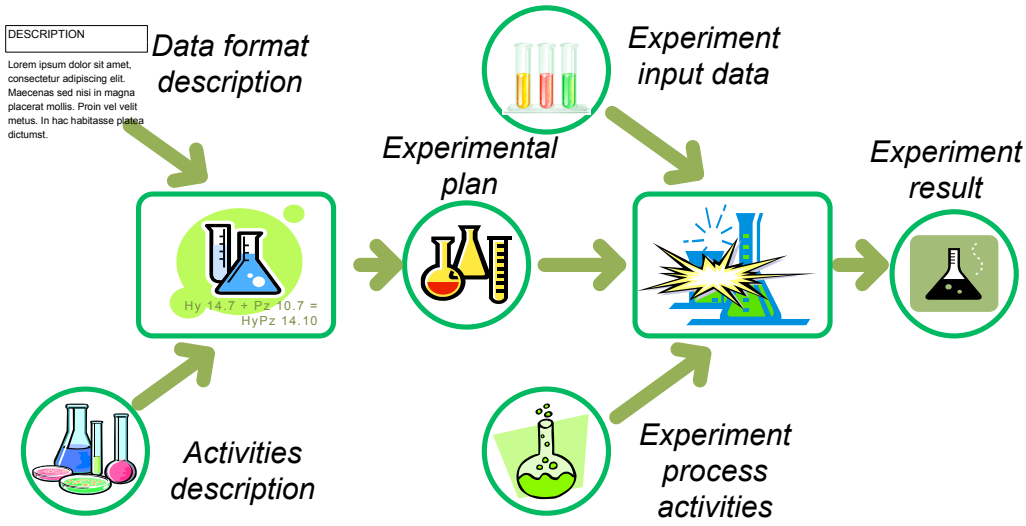


Figure 8: Experiment pipeline – one of the central ideas behind Virtual Laboratory [108].

- Experiment developers – computer scientists, whose role is to support research scientists in implementing experimental plans, produce new computational services and integrate multifarious computational and data services into ViroLab infrastructure which create new tools that take advantage of this infrastructure.

Groups of users in a company with software used have been shown in figure 10.

Architecture The VL structure is presented by figure 11. Some of the components shown in diagram will be described, beginning with Experiment Planning Environment (EPE). EPE was created as an aid for experimental plans. It is based on Eclipse Rich Client Platform (Eclipse RCP) [96, 97] and combines the following components: Domain Ontology Browser, Grid Resource Registry Plug-in, Data Source Registry Plug-in and GScript Editor. Domain Ontology Browser assists in searching an appropriate grid service that can fulfil particular user need. GRR plug-in enables browsing for accessible Grid Objects which can be used in experimental code; in addition it is capable of generating a code snipped that accesses the Grid Object selected by user. DSR Plug-in, on the other hand, enables browsing, modifying and adding new data sources together with credentials, utilized to access them. Lastly, GScript Editor [96] provides syntax highlighting and code completion with support of specific ViroLab features, such as support for Grid Objects. ViroLab portal, also termed the Patient Treatment Support tool [46] – PTS, is based on GridSphere portal, providing the below mentioned user interfaces: Experiment Management Interface (EMI) [97], Database Browser [21] (see section 3.4), Grid Resource Registry (GRR) Browser, VO management portlet, Drug Ranking System, Literature Mining, QUery TRanslation tOols (QUaTRO) [25, 28] and Binding Affinity Calculator (BAC) [190], some of which I will explain. Firstly, using EMI user can load an experiment, execute it and download results. As clause 3.1 indicated, Experiment Repository (ExpRepo) supports storing and sharing subsequent versions of experiment. An example of

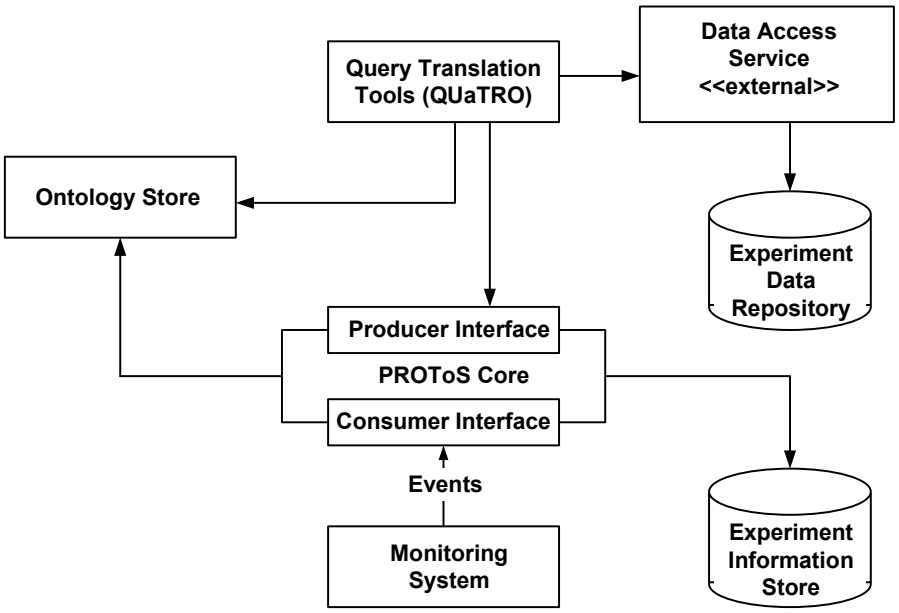


Figure 9: PROToS architecture [27].

this experiment has been shown in listing 1 – the experimental code is from [46] with figure 13 illustrating interactions between ViroLab components during its execution. Access to data is implemented using an older version of Data Access Client. Newer, currently used notation that exploits Data Source Registry (DSR) features will be reviewed in clause 3.4. The cooperation model [109] between experiment creators and users of these experiments has been shown in figure 12 with precise investigation of collaboration aspect given by Tirado-Ramos [216]. Drug Ranking System (DRS), in some documents [46] termed the Decision Support Ranking Service, provides algorithms and databases to study HIV drug resistance, such as Retrogram, REGA, Stanford HIV DB and ANRS, which allow for predicting drug interactions within specific regions of virus: reverse transcriptase or protease. Patient data is drawn from DAS component (see section 3.4).

Security With regard to security, ViroLab provides Single Sign On mechanism. In the case of ViroLab, it is based on Shibboleth framework with suitable extensions developed by ViroLab team to support non-web applications. Implementation of security mechanisms in ViroLab has been covered extensively by Jan Meizner et al. [156]. The author of this paper asserts that security of valuable ViroLab resources must be protected from theft or devastation. This will include medical databases, trust stores with user credentials, source codes of experiments together with their results (also intellectual property, probably obtained after months of research and simulations), as well as computational power and network bandwidth.

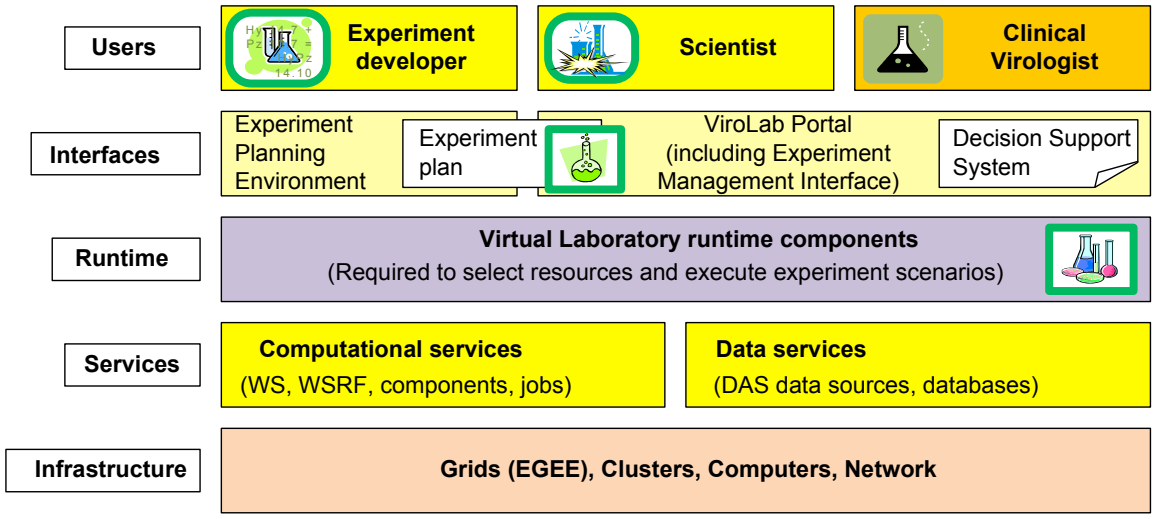


Figure 10: Layered view onto ViroLab architecture. On top there are three kinds of users: experiment developers, scientists and clinical virologists using dedicated interfaces that, in turn, communicate with runtime components that manage computational and data resources located in Grid, clusters or individual computers [198].

3.4 Data access in ViroLab

Data access in ViroLab is possible using varied means, both from ViroLab portal and from GScript, which is used for expressing experimental code. The ensuing components, implemented as portlets, allow for data access from ViroLab portal:

- QUaTRO [25, 28], provides means for executing queries to data repositories and provenance collection systems, using terms from virology domain. It can be used to express queries in respect of PROToS (Provenance Tracking System) and Virtual Laboratory Data Access Services (DAS).
- Database Browser [21], to aid users browsing databases of patients, drugs and virus

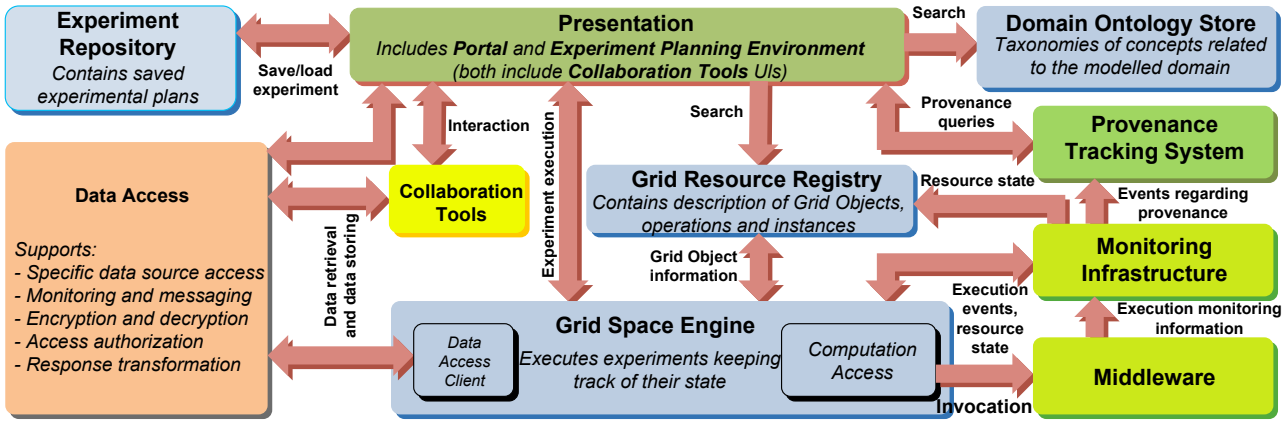


Figure 11: A more technical view of the ViroLab structure with all main constituents illustrated [108].

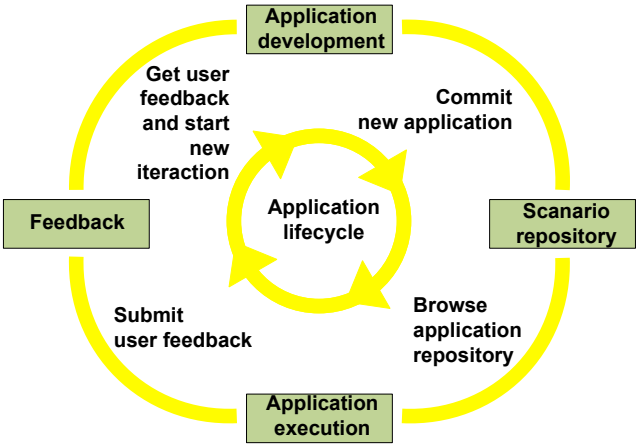


Figure 12: Cooperation model between experiment (application) creators and users of these experiments [46, 109].

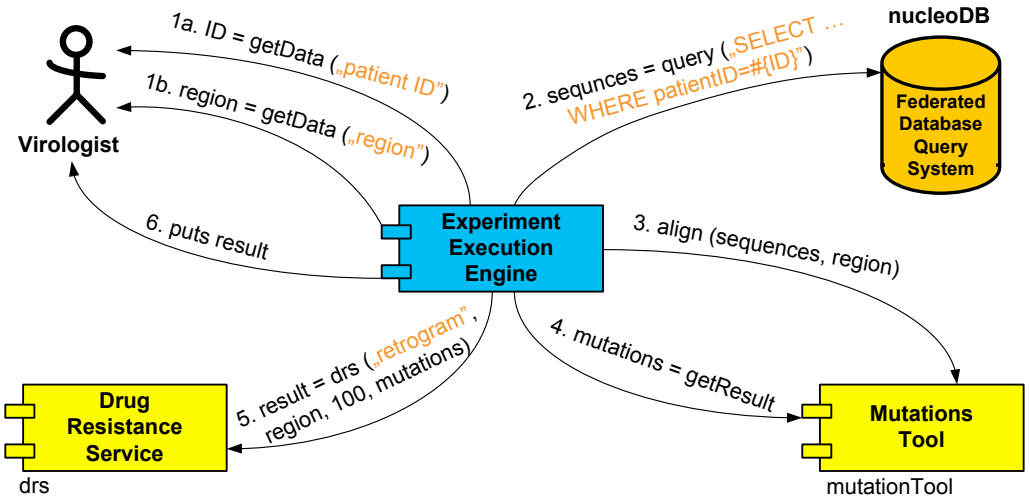


Figure 13: Interactions between components during execution of a sample experimental plan with source code was provided from listing 1 [46].

mutations. In addition, they can look through database schemas, execute SQL queries, sort results returned using assortment of criteria, as well as save data as XML, HTML, CSV or print the records. Database Browser is a user interface for DAS, the former being covered further in this section.

An alternative means in accessing data, as it was mentioned above, is to use GScript; this is plausible using Data Access Client (now in version 2). DAC is a library written in JRuby language, which additionally utilizes libraries coded in Java to obtain access to miscellaneous data sources, including databases, data sources available using WebDAV interface and data accessible through Virtual Laboratory Data Access Services – DAS. DAC is underpinned by Data Source Registry – DSR, currently implemented as MySQL database. A graphical front-end to DSR is the DSR plug-in of EPE environment. Architecture of data access in ViroLab has been presented in figure 14. I will begin broaching data access from browsing data sources, registering

Listing 1: Sample experimental plan (from [46]). See figure 13 illustrating interactions between ViroLab components during its execution.

```
patientID = DataRequester.new.getData("Provide patient\'s ID")                    #1a
region = DataRequester.new.getData("Region (\"rt\" or \"pro\")")                  #1b

nucleoDB = DACConnector.new("das",
       "virolab.hlrs.de:8081/wsrf/services/virolab/DataAccessService","","","")
sequences = nucleoDB.executeDistributedQuery(
       "select nucleotides from nt_sequence where
        patient_ii=#{patientID.to_s};")                                           #2
        mutationsTool = GObj.create("RegaDBMutationsTool")

mutationsTool.align(sequences, region)                                            #3
mutations = mutationsTool.getResult                                               #4

drs = GObj.create("DrugResistanceService")
result = drs.drs("retrogram", region, 100, mutatations)                           #5
puts result                                                                       #6
```

a new data source and then using those from GScript level. Programming environment of experiment developers – let us name them for brevity 'programmers' – is Experiment Planning Environment (EPE). A programmer uses EPE for coding experiments in GScript language, searching for grid services, cooperation with other developers and experiment users by correcting inaccuracies or errors they identify and taking into account suggestions they submit using Experiment Management Interface (EMI) feedback form. In addition, EPE allows for publishing experiments in Experiment Repository. As it was mentioned before, EPE contains a plug-in, which enables browsing of data sources, registering new ones and storing user credentials.

Data access workflow In order to be able to look through DSR records and add new entries, the programmer needs to login to Virtual Laboratory using EPE login form. After successful authentication and choosing the DSR-plug-in view, a categorized list of data sources appears. Programmer, by clicking particular data source may edit or view information, together with changing credentials that are linked to this data source. In the list view, they also have the possibility of adding a new source. A diagram illustrating the data access workflow has been provided in figure 15. By adding a new data source, programmer chooses its type ('structured' or 'unstructured'), then data source technology (e.g. PostgreSQL), as it is shown in figure 16, presenting one of the DSR plug-in forms. Depending on technology, programmer has the opportunity to provide varied information on particular data source. Some of it is typical and occurs often, e.g. URL or schema name. A field, that is always and will be always required with every data source, unless DAC architecture changes, is the data source 'handle', which is a symbolic name utilized when initiating a data source in experiment code. It is a means by which

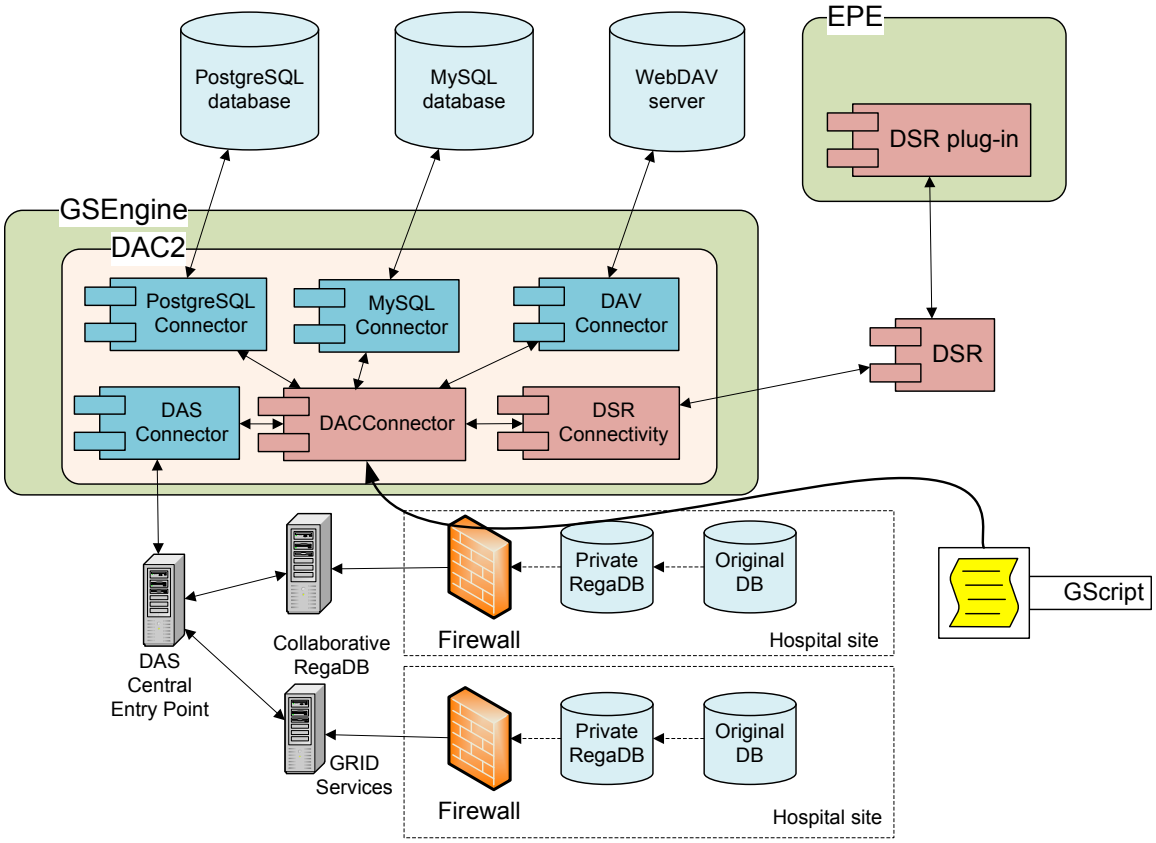


Figure 14: Architecture of data access in ViroLab.

a programmer makes a reference to data source in code. Furthermore, a programmer has the option of providing his or her credentials that are needed to access a particular data source (see figure 17). What's more, the interface allows for specifying whether the credentials supplied by user can be shared with other authenticated users. In DAC terminology, such credentials are called 'static'. After accepting changes, recently added data source becomes visible to other programmers. They can choose a data source from list and supply their own credentials, if they want to use it, but other programmers did not make their credentials static. A question arises: how to utilize a data source in experiment code? To this end, a programmer adds the following line:

```
require 'cyfronet/gridspace/dac2/dac_connector.rb'
```

at the beginning of their script. This gives programmers access to DACConnector class, which is exploited for instantiation of data sources. Using `DACConnector.new` method, programmer passes the data source handle, thus creating a new instance of specific data source connector. Virtual Laboratory, as a grid project, takes steps to make data access occur on Single Sign On (SSO) basis. Therefore, as it was mentioned earlier, programmer stores their credentials in the DSR and does not have to provide them when running a script. It suffices, that the programmer is logged in and the DSRConnectivity DAC module will download credentials from DSR. Conversely, if programmer did not provide particular data source credentials, when

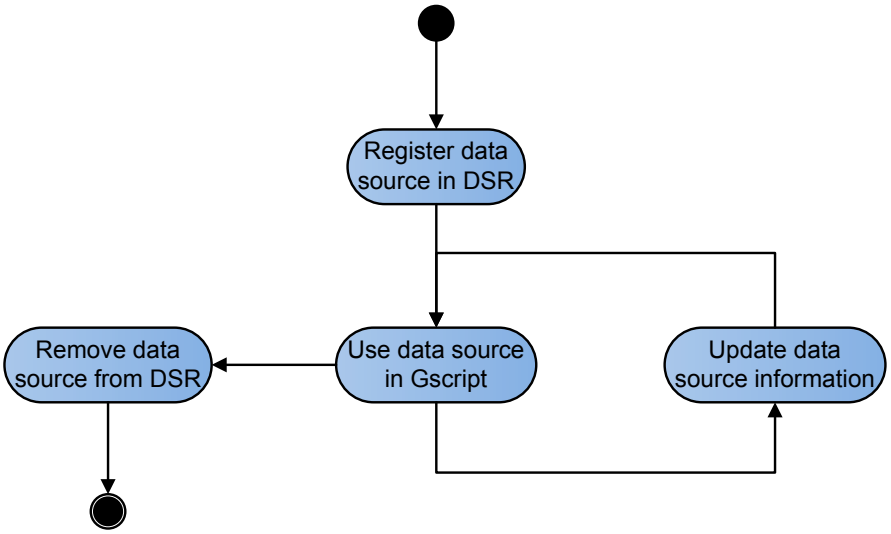


Figure 15: DAC2 data access workflow as described in the text.

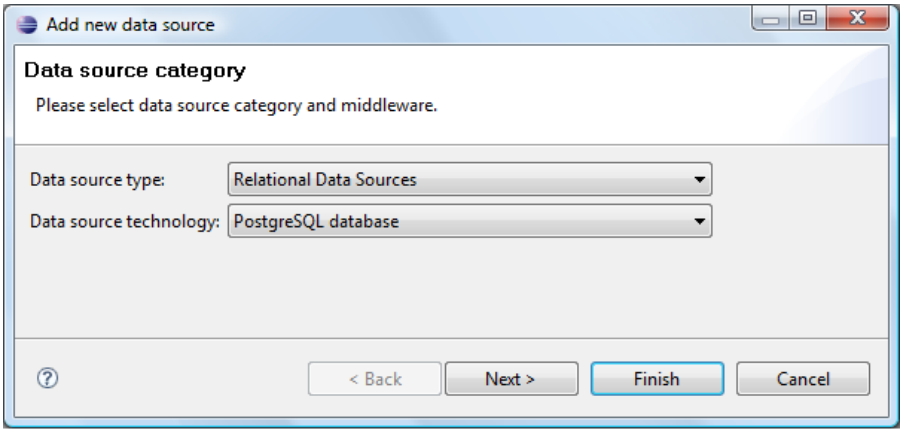


Figure 16: A DSR form that appears when adding a new data source.

credentials are required, the DACConnector checks if static credentials for this data source exist, i.e. if someone made their credentials available to other authenticated users. If such credentials subsist, DACConnector instantiates a connection to data source; otherwise an exception is thrown saying that static credentials have not been found and that programmer should provide their credentials as `DACConnector.new` method parameters. As the exception says; instead of using credentials stored in DSR, programmer, after providing data source handle in the first argument may pass login as second and password as third argument of the `new` method. In this way, data source instantiation is carried out in most cases. Additional API, taking into consideration LFC DS component developed as part of this thesis, is presented in appendix A.

In the case of Data Access Services (DAS) data source, Single Sign On (SSO) is not provided by DSR, but by DAS itself [21], i.e. it is sufficient, that a user holds a valid Shibboleth handle. Policy Decision Point (PDP) service decides whether an authenticated user may execute operations or query data. Programmer, as a second argument of DAS data source initialization, may provide alternative Shibboleth handle, if they want to perform operations on behalf of another

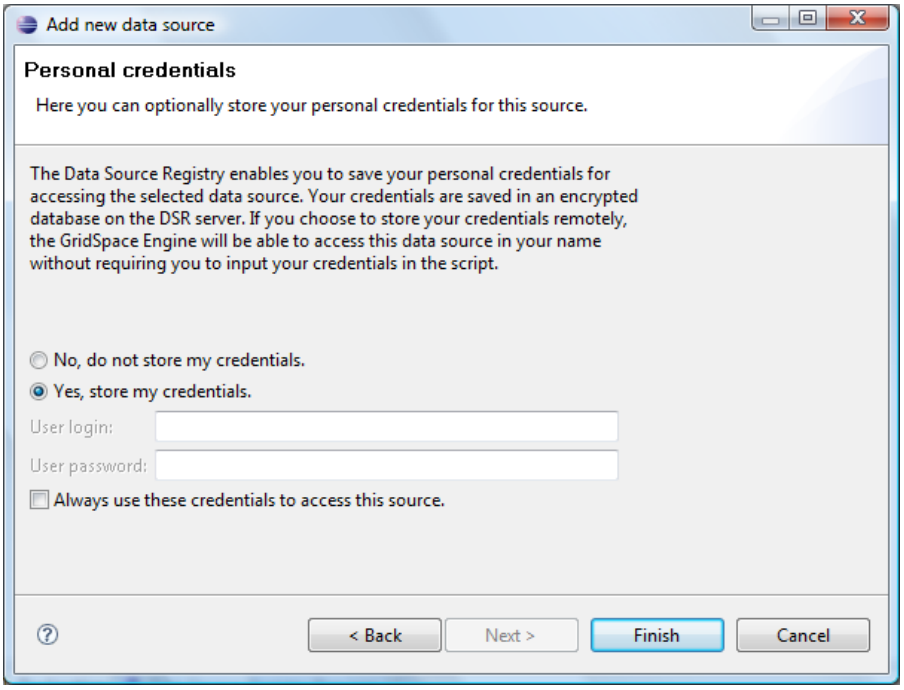


Figure 17: DSR form for providing data source credentials.

user. Apart from 1, 2 and 3-argument constructor, a 4-argument constructor is available when instantiating an LFC DS data source and the reader may peruse it in appendix A.2.

The DACConnector class, aside from constructor, provides methods that render operations on instantiated data source. Before the conception of LFC DS, the list of methods appeared as follows:

- `executeQuery(query)`
- `executeUpdate(query)`
- `storeFile(payload, filename)`
- `getFile(filename)`
- `deleteFile(filename)`

Devising LFC DS required adding new methods, which are discussed in detail in appendix A.3. DACConnector instantiates a data source on basis of handle supplied by user and information on the data source represented by handle that is returned by DSR. If it is, for instance, a WebDAV source, DACConnector creates an object of DAVDataSource type, which serves as a role of connector to WebDAV data source, i.e. translates invocations of DACConnector methods into invocations of WebDAV specific libraries that connect to WebDAV server. If it is a MySQL data source, a MySQLDataSource object is instantiated, which in turn is a connector to MySQL database, etc.

DACConnector, while instantiating a data source connector, passes to it information received from DSR. As a consequence, user does not have to supply this information in method invocations, as it occurred in previous Data Access Client version. After `DACConnector.new` method finishes successfully, reference to connector object is then preserved in `@source` object variable for further method invocations. If a user sends DACConnector an `executeQuery(query)` message (or in other words, executes `executeQuery(query)` method of DACConnector instance), the instance of DACConnector object will send that message to data source connector, whose reference, as mentioned before, it holds in `@source` variable. If the data source connector pointed by `@source` supports such a message, it performs appropriate operations and returns results or throws an exception, if operation failed for some reason. On the contrary, if connector does not support such a message, an exception is thrown indicating this. Connector hierarchy has been illustrated in figure 18.

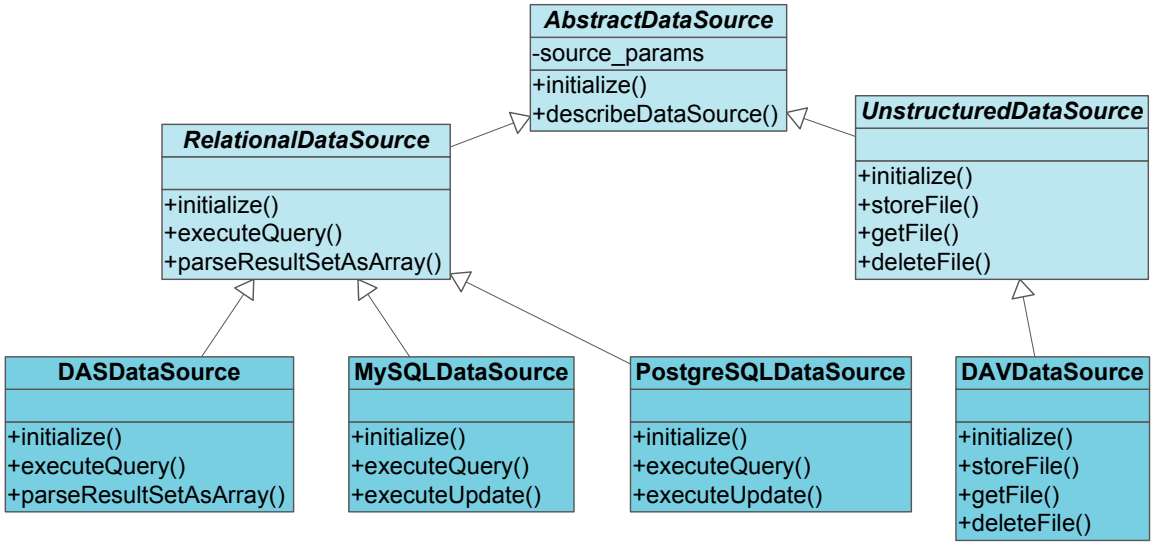


Figure 18: Data source connector hierarchy in DAC2.

Data Access Services (DAS) Virtual Laboratory Data Access Services deserve detailed explanation. Its mission is to provide integrated, secure access to patient databases of hospitals participating in the ViroLab consortium.

The DAS authors, while conceiving this component, faced a dilemma of how to render integrated access to databases belonging to different organizations. Paweł Płaszczak [184] asserts that volume of data was massive, not in terms of gigabytes, but in terms of its delicacy, since leakage of patient data would be undoubtedly a legal threat for hospitals. On the other hand, creating one central database was not considered, because, as Plaszczak asserted, medical institutions are equally possessive about their data, as software corporations are about their intellectual rights. Federated Single Sign On [156] was chosen as an alternative to Public Key Infrastructure (PKI). In Federated SSO, every organization is responsible for granting or revoking access to its data and for confirming identity of their members. Firstly, it allows

many organizations to join a project. Secondly, every organization has full control over its data. Nevertheless, the Federated SSO solution has advantages, but also some shortcomings, one of which is a possibility of data being stolen by a member of a trusted organization. Assel and Kanyocu [16, 19] considered this issue and to avert situations of this kind, they formulated a security policy which can be defined very precisely. They employed Access Control Markup Language (XACML)[44] for the purpose of managing data access policy and created a user-friendly interface for generating, uploading and modifying access policies. Apart from graphical interface dedicated to DAS it is relevant to mention ViroLab component for virtual organization management, which is available as a portlet through the ViroLab portal, whose graphical interface has been implemented by GridwiseTech [105], using Adobe Flex, thus taking advantage of capabilities provided by contemporary 3D graphics. The interface is based on virtual organization (VO) idea, which makes management of permissions more efficient.

For managing data access policies to DAS data sources, a modified version of Policy Decision Point (PDP) is utilized, which is is a product of TrustCoM, another European project [68]. PDP has been implemented as a web service and is responsible for controlling every data access request to DAS data. In addition, PDP makes decisions based on access policies, deciding whether a particular user can be given access to a particular resource or whether the user can question queries raised in this resource. The overall plan of this mechanism has been portrayed in figure 19. Meizner et al. [156] asserts that securing data access to DAS follows a two-step approach: a user interested in accessing DAS must first pass through ViroLab security policy defined by Security Assertion Markup Language (SAML), then through policy delineated in XACML. The dissertation author, based upon his knowledge, is of the opinion that in securing DAS data sources there should be mentioned an additional step in-between, i.e. aside from the steps mentioned in [156] and [16, 19] – identification by Identity Provider (IdP), authorization by ShibAuthAPI and consent of Policy Decision Point. Additionally, another need requires permission to access the data given by Data Source Registry. At present, DSR access policy is quite primitive: access to data source can be given only to its owner or to every authenticated ViroLab user with no intermediate access granting levels. Perhaps, in the future, capabilities of security policies of DSR will be extended. Furthermore, the dissertation author recommends any reader interested in security aspects of ViroLab, to take into account the publication of Meizner et al. [156] which describes this in detail. DAS has been based on Globus Toolkit [88], Open Grid Services Architecture Data Access Integration (OGSA—DAI) [10, 129] and the aforementioned security architecture of ViroLab. A challenge that DAS successfully unravelled, was integrating into ViroLab dispersed and heterogeneous data resources belonging to different institutions. One of the problems was the fact that data was stored in relational databases whose format was dissimilar, even though they were concerned with the same entities, i.e. patients and virus

[44] Access Control Markup Language (XACML) is a language standardized by Organization for the Advancement of Structured Information Standards (OASIS).

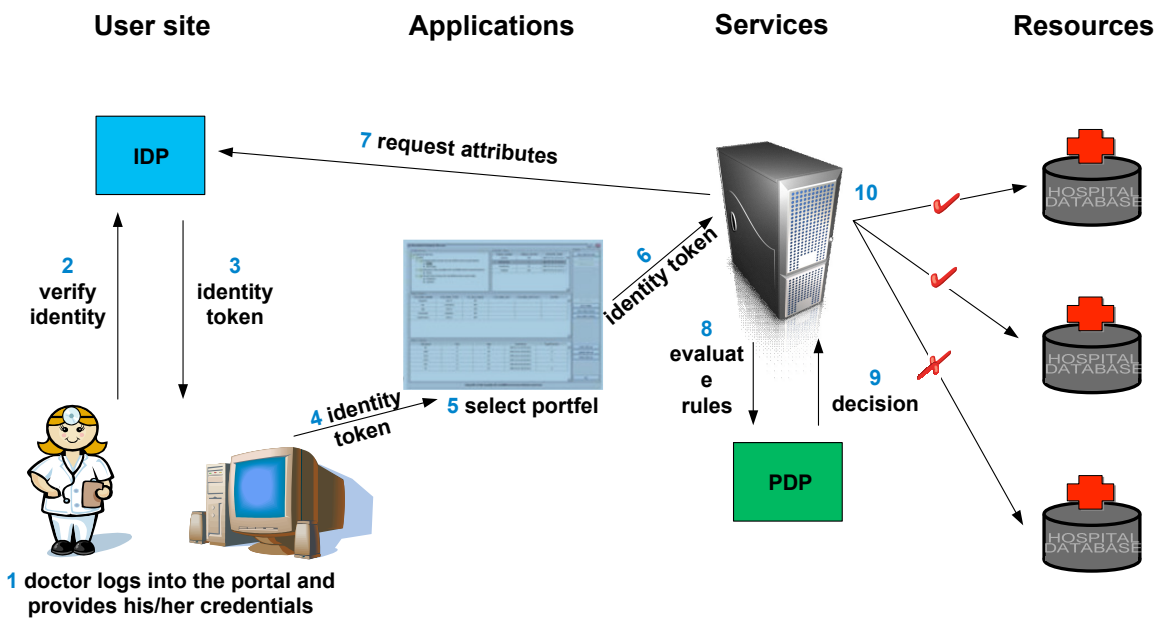


Figure 19: DAS security mechanisms [16, 19].

genotyp which patients are infected with. For instance, Catholic University of Rome in Italy, which has been involved in HIV resistance research since 1999, stores patient examination results and essays regarding drug invulnerability in dedicated relational databases based on Microsoft Access.

According to Assel et al. [18], the process of transforming databases could be very complex. In order to minimize the complexity, a common database scheme was chosen to be installed in every hospital participating' RegaDB HIV Data and Analysis Management Environment were chosen and the solution envisioned by DAS authors can be delineated as follows:

A hospital may use both private RegaDB installed behind its firewall, in a so called Demilitarized Zone (DMZ) or utilize collaborative RegaDB located with some trusted partner using encrypted connection (see figure 20). DAS consists of 3 subsystems responsible for the ensuing points: Data Resource Discovery, Data Access and Data Transformation. Data Resource Discovery virtualizes locations of data resources – applications reference data sources by logical names using Meta Query Language (MQL), so called by DAS authors. The Data Access Module provides interfaces enabling usage of relational and XML databases. A noteworthy fact is that access to relational and XML databases is possible using DAS component and using GSEngine Data Access Client (DAC). The choice is at the discretion of a user, although the thesis author recommends DAC, as it is a more general interface not limited to Virtual Laboratory, but useful in every project that employs GSEngine. A reader interested in DAS architecture can find more information in [18].

Data Source Registry DSR is a solution that aims at relieving programmers from remembering particulars of access to various data sources and strives to provide Single Sign On

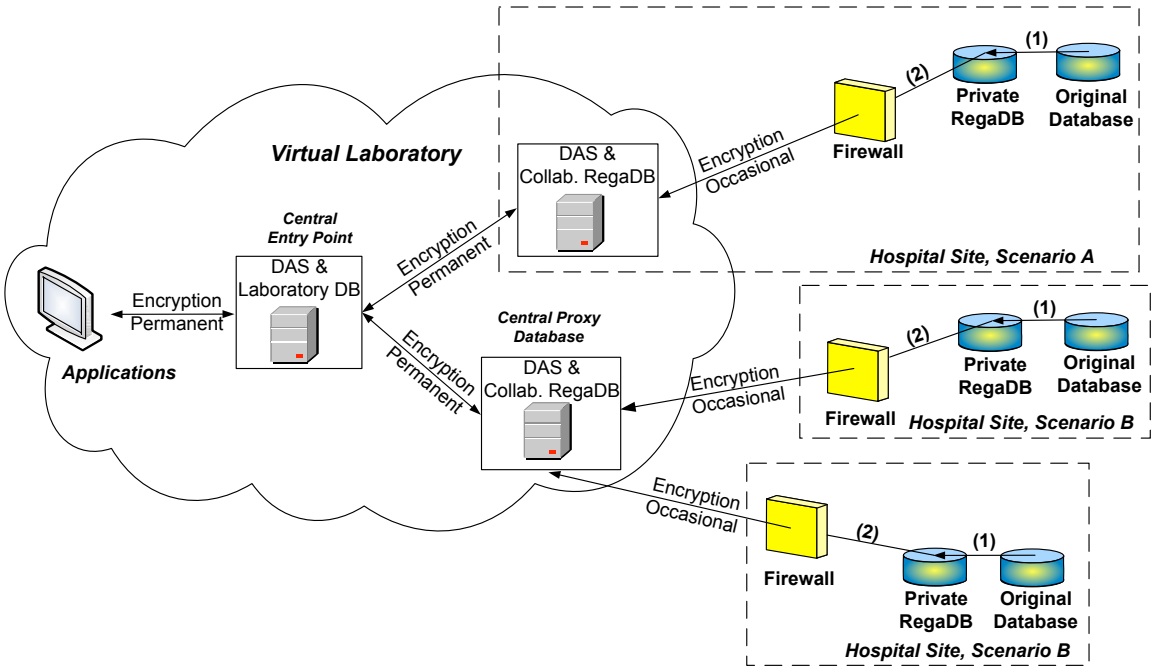


Figure 20: Data integration scenarios in ViroLab Data Access Services [18].

mechanism to these sources. In the first generation of DAC, the programmer had to provide every detail regarding a data source with which they wished to connect and can be seen in listing 1. DAC2 is a second version of DAC that was completely rebuilt to take advantage of DSR and was created by Piotr Nowakowski, the main author of DAC. Ideally, using DAC2 and DSR is sufficient to provide data source handle in order to be able to use a particular data source. On the other hand, DACConnector enables programmers to override some data that is stored in DSR during data source initialization or to provide this data in the case of its absence. An example of such data overriding can be seen in appendix A.2, where it is done using LFC DS constructors. With regard to DSR implementation, currently it is a secured MySQL database, although it may possibly be implemented as a software component in the future. Access to DSR is clearly divided in DAC source code and is performed only in `dsr_connectivity.rb` file, which makes the probable change of DSR implementation, in terms of source code update required, less costly.

3.5 Other projects based on GridSpace platform

Two undertakings can be referenced here: GREDIA and ChemPo.

GREDIA The aims of GREDIA are firstly, creation of middleware intended for business grid applications and secondly, production of two pilot applications: domain of journalism and area of banking [212]. The devised middleware comprises the following components: Application Execution Assistant – Appea [44, 45], Framework for Intelligent Virtual Organisations (FiVO) [137] and a data management layer, so called 'virtual work space' that binds together

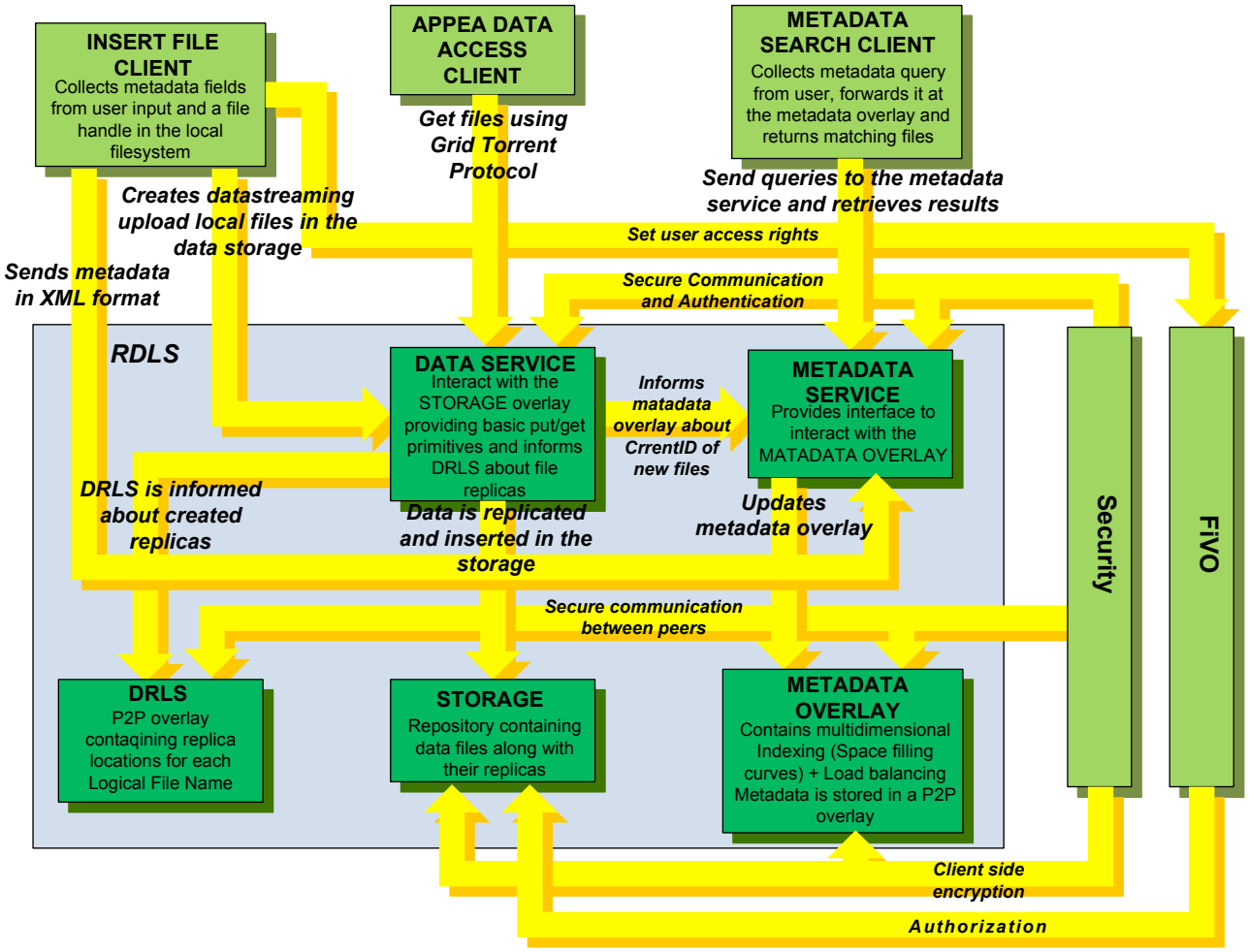


Figure 21: Structure of GREDIA middleware [133].

nodes participating in the project. This virtual workspace is a form secured from intrusion of Data Grid, in which there are written and annotated multimedia files, spreadsheets etc. that can be then discovered by users. Moreover, every user being a part of this virtual space is able to specify who can access data that he or she makes accessible. Thanks to the absence of central server and basing infrastructure on peer-to-peer architecture, the system has no single point of failure and provides fast data transfers. This is possible, because every node can not only be a consumer of services, such as data searching, but may also contribute storage capacity and data services [133]. The structure of GREDIA has been illustrated on figure 21. A reader interested in specifics of implementation can find further information in the publication of Asiki et al. [14].

The dissertation author believes the Appea framework previously mentioned has many common characteristics with Virtual Laboratory software. Let us look at its architecture shown in figure 22; similarity to figure 11 representing ViroLab structure is obvious. In place of Experiment Repository we can see Scenario Repository, also noticing existence of Grid Resource Registry. However, GREDIA accentuates supporting grid data management services with important points being, Distributed Replica Location Service (DRLS), which maps logical file

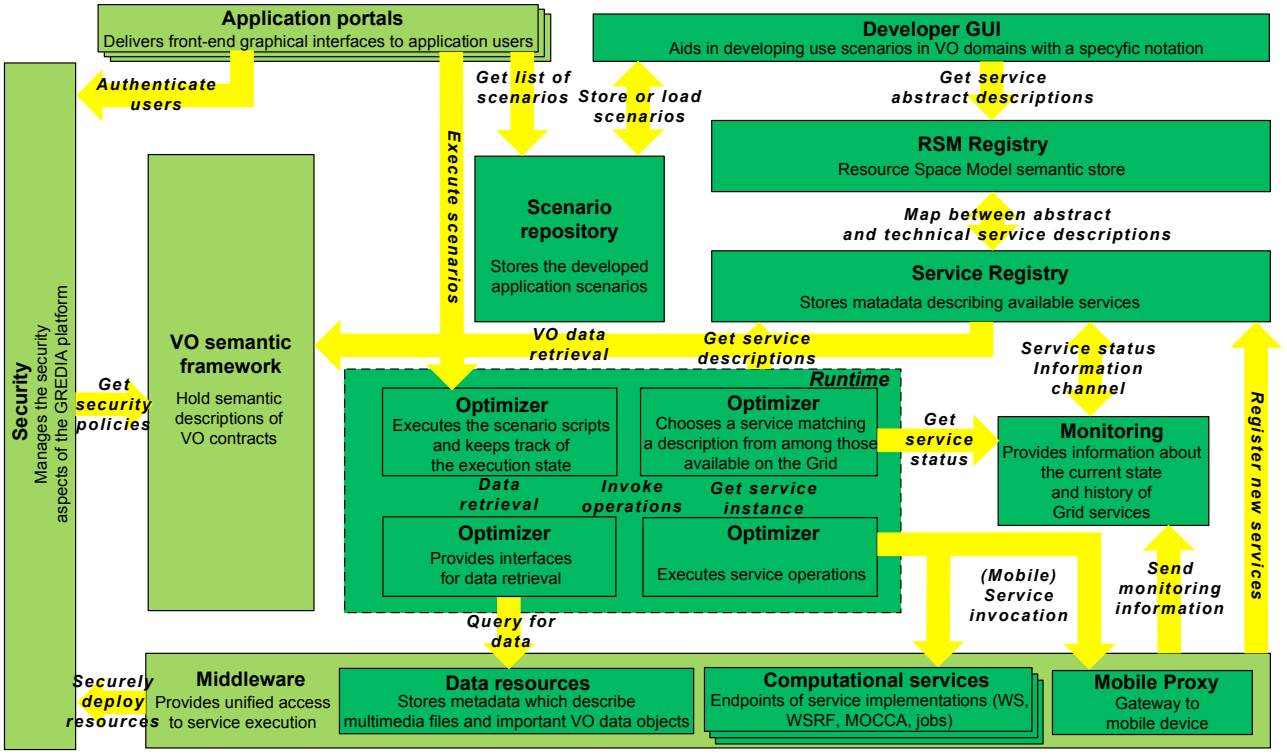


Figure 22: Architecture of Appea platform [44].

names to physical ones, Metadata Overlay, which is a P2P DHT[45]-based based overlay storing metadata files, a Metadata Service, which assigns metadata files to peers from the Metadata Overlay and processes Appea or web-client queries and finally Data Service, which stores data in Storage Overlay and returns streams enabling downloading data with GridTorrent protocol (see figure 23). Those services have been detailed in the article of Konstantinou et al. [133].

ChemPo According to Sterzel, Szepieniec and Harężlak [201, 202], Grid can be applied directly to conformational analysis, numerical frequency computations, zero point vibrational averaging, determination of chemical reaction paths or potential energy surfaces (PES) etc. all being computation intensive tasks or tasks that operate on large data sets. In order to help scientists make effective use of Grid resources, they built an environment for performing such chemical calculations on the Grid. The environment manages computational processes together with experimental data and strives not to distract scientists with technology and not to change their way of thinking. The project has a web portal front-end based on Web 2.0 techniques, e.g. Google Web Toolkit (GWT). The portal character of the project is also the origin of its name: ChemPo – Grid Web Portal for Chemists. Besides usage of GWT as a user interface technology, ChemPo makes extensive use of GSEngine for job management. Architecture of the project has been presented in figure 24. Currently, ChemPo enables usage of Gaussian, one of the most popular commercial chemical packages, although authors plan integration with

[45]Distributed Hash Table

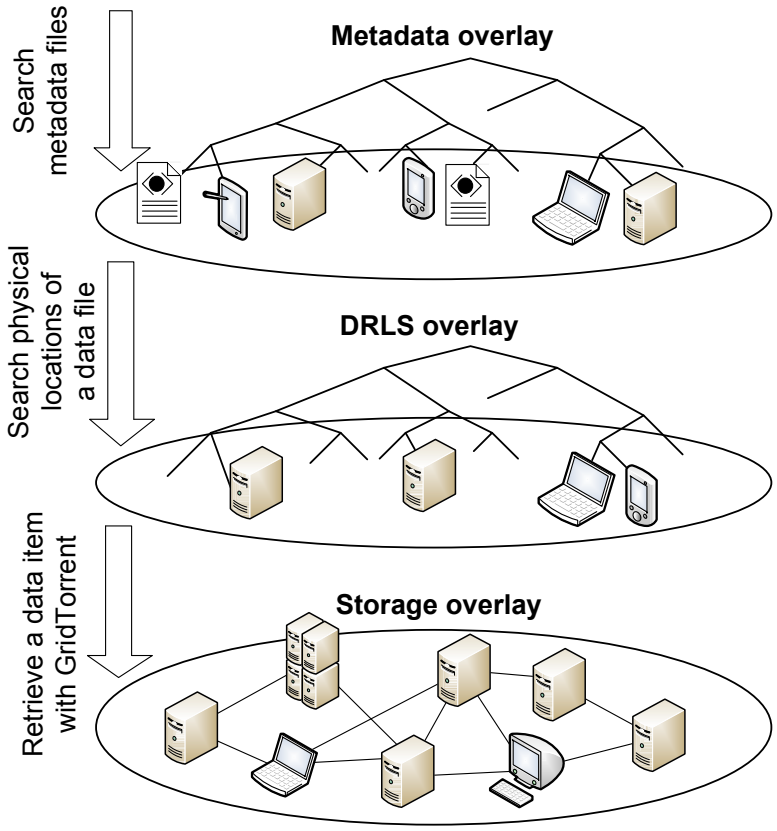


Figure 23: An overview of GREDIA data management services [14].

GAMESS and NAMD.

PL-Grid – Polish Infrastructure for Supporting Computational Science in the European Research Space PL-Grid is an emerging project throughout Poland whose purpose is to create a grid-computing infrastructure for scientists, besides local needs enabling future international collaboration. In the scope of the project, there are tools being created that allow for design and execution of scientific applications on computational resources using dispersed data. Inception of PL-Grid is an answer to e-IRG (e-Infrastructures Reflection Group) resolutions, which was established in 2003 by the European Commission to promote consistent formation of European Grid. Moreover, e-IRG objectives and PL-Grid realizes goals of national plans regarding the informatization and development of Poland [159]. Currently, the project is progressing into production stage – information on its website (http://www.plgrid.pl/) indicates that PL-Grid operation portal will be open soon (operation portal is a place where among other services, user account creation will be possible). Furthermore, PL-Grid represents Poland in European Grid Initiative (EGI), which is a stratagem and goal for a long-term sustainability of grid infrastructures in Europe [123]. Its approach is an establishment of a federated model bringing together National Grid Initiatives (NGI) to build a common Grid infrastructure, which will replace EGEE when its third phase ends in 2010. Planned computational power is 215 Tflops (about 5000 processors) while disk space provided by PL-Grid will

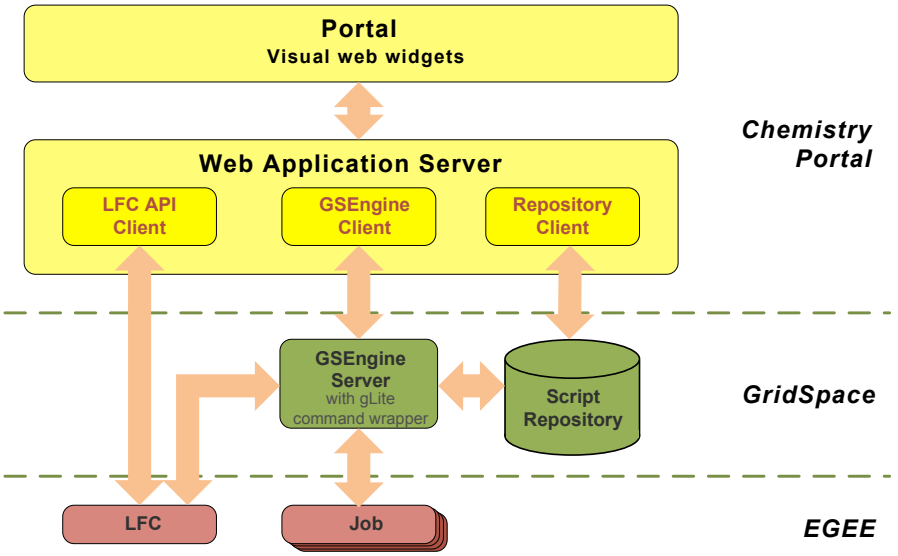


Figure 24: ChemPo architecture [202].

be 2500 TB. Moreover, two additional separate infrastructures are provided: for testing and development [131, 159]. Additionally, integration with local computer clusters belonging to various research establishments is possible.

As regards realization of this project, tasks are distributed among several Polish universities and research institutions that already manage main computing centres and with experienced personnel who have gained from earlier grid computing projects, with apportionment as follows:

- Project management – ACC CYFRONET AGH (Krakow)
- Hardware infrastructure – TASK[46] (Gdansk)
- Operations centre – ACC CYFRONET AGH (Krakow)
- Development of e-infrastructure software and user's tools – PSNC[47] (Poznan)
- Training and users' support – ICM[48] (Warsaw)
- Security of the infrastructure – WCSS[49] (Wroclaw)

From inception, PL-Grid is integrated with Worldwide Grid and in particular with systems being the result of EGEE and DEISA projects. Software of PL-Grid encloses (see figure 25):

- User tools, such as portals, systems for application monitoring, results visualization etc. Importantly they include

[46] Academic Computer Centre in Gdansk (CI TASK)
[47] Poznan Supercomputing and Networking Centre
[48] Interdisciplinary Centre for Mathematical and Computational Modelling
[49] Wroclaw Centre for Networking and Supercomputing

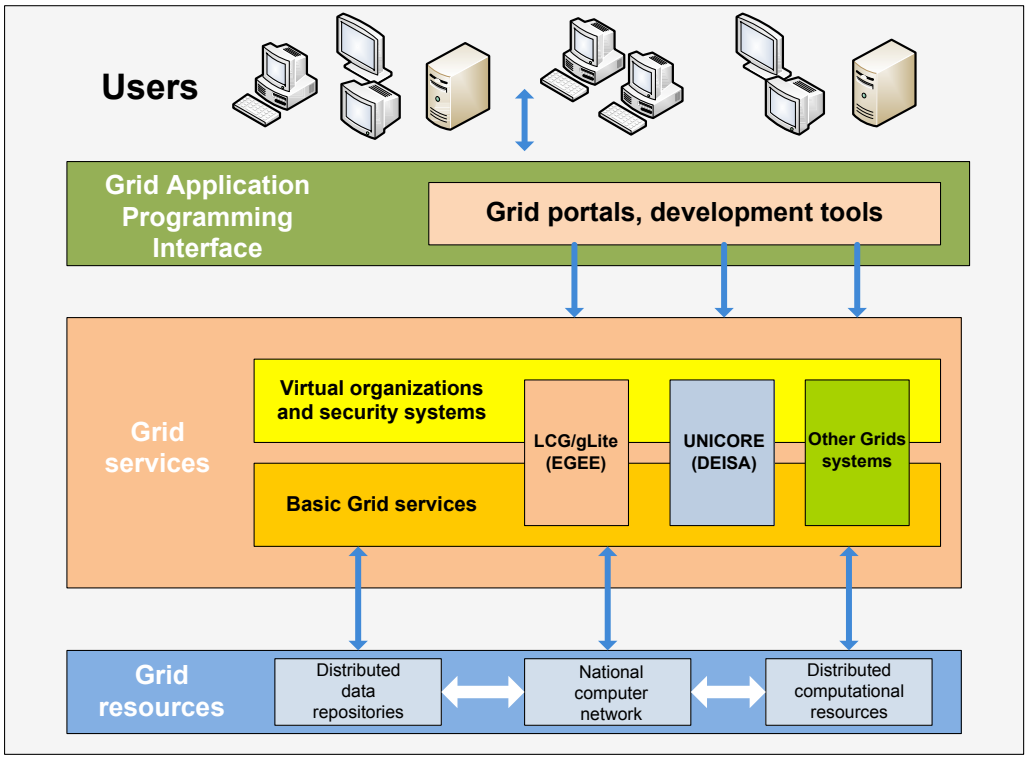


Figure 25: Structure of PL-Grid

  - Migrating Desktop – environment for job and file management together with results visualization.
  - Grid Commander – a file manager.
  - g-Eclipse – environment for operators and developers.
  - Vine Toolkit – tools for creating web applications.
  - The aforementioned GridSpace platform, which is used for constructing applications using a high-level scripting language. It will be one of the most supported user and developer tools. Furthermore, another Virtual Laboratory based on this platform is being built for PL-Grid project with many functionalities being similar to those found in ViroLab, e.g. provenance tracking system, portal, grid resource registry etc., although certain differences exist. An example is security system – at the time of writing the dissertation, it has not been decided whether it will be based on Shibboleth or on a different security system.

- Programming libraries.
- Virtual Organizations system: certificate and account subsystem, resources usage accounting and security subsystem. Particularly, FiVO - Grid Virtual Organisation Semantic Framework – software enabling VO contract negotiations. FiVO is one of the results of GREDIA project.

- Data management systems: metadata catalogues, replica management catalogues, file transfer service.
- Systems for managing jobs, monitoring of services and infrastructure, handling software licenses and administering local resources. Some of which include
  - GEMINI2 – a system for monitoring applications in a Grid environment.
  - X2R – a system for integration of relational database management systems, LDAP data sources and XML databases into an integrated semantic knowledge base.
  - StorMon – a system monitoring performance parameters of mass storage.
  - ACARM (Alert Correlation, Assessment and Reaction Module) whose purpose is collecting and correlating security alerts gathered by Intrusion Detection Systems (IDS) located in network infrastructure.

Specialized scientific software packages that are planned, will be supported and include those in the field of physics (Meep), numerical computations and simulations (MATLAB), biology (AutoDock, Gromacs, NAMD) and from the domain of quantum chemistry (ACES II, ADF, Dalton, GAMESS, Gaussian, Molcas, Molpro, NWChem, Siesta, TURBOMOLE).

3.6 Storage services in gLite

gLite data storage approach The initial user groups of gLite storage and catalogue services that authors of these services targeted, were High Energy Physics[50] and Biomedical communities [138]. They deemed that these communities store their data primarily in files, which, as already noticed in *Motivation*, may not be true due to many biomedical communities storing genomic data in relational databases. In the second rationale of providing file semantics to Grid storage, they put forward a view that these semantics are well understood by prospective consumers and providers of storage services as opposed to generic data objects, which can have many definitions among varied application groups.

They discarded an option of imposing distributed world-wide file system like AFS[51] on each site participating in EGEE grid; since they were of the opinion that gLite middleware should work with locally available hardware and software. As an alternative, Kunszt et al. [138] declared that to deal with peculiarities of an individual storage system, they required all Grid-aware storage to implement Storage Resource Manager (SRM) interface [194]. The collection of services providing file access and storage forms a 'gLite Storage Element' (SE). The ensuing constituents can summarize SE functionalities [138]:

[50] Abadie et al. [1] assert that Large Hadron Collider project, which is expected to be one of the main consumers of EGEE/WLCG Grid infrastructure, will generate 15Pb of data per year. Normally 1Pb/s generated by detectors will be reduced to ca. 100Mb/s by multi-level trigger systems. However, velocity of data generation may be up to 1.5Gb/s.

[51] Andrew File System

1. Storage back-end with related hardware and drivers
2. Implementation of Storage Resource Manager service
3. A transfer service
4. gLite File I/O service
5. Supplementary logging and security services.

File names Files can be replicated at many sites to hasten file access operations. Therefore a need for location-independent logical file names (LFN) arose, which can be assigned by users. Ideally, users would refer only to those logical names and never to physical names which contain location dependent information, such as which storage element holds a particular file or what protocol to use when accessing the file. The second requirement that became apparent from usage of replicas, was the necessity to have a mechanism that identifies which replicas represent the same file. A central service, managing unique identifiers could be employed. Nevertheless, authors provided a better decentralized solution: upon creation, each file obtains a unique, unalterable ID, termed Grid Unique ID (GUID). As a result, many replicas may represent the same file identified by a GUID. Applications may use either GUID or LFN to identify files.

Replicas are identified by Site URLs (SURL) [138], by some sources, e.g. [1, 47], termed the Storage URLs, although [138] uses StURL abbreviation for that purpose. SURL specifies which Storage Element to contact when accessing data and can be passed to SRM interface as an argument. Finally, Transport URL (TURL) is a filename giving necessary information to obtain or write to a particular replica, including protocol, hostname, port, path. TURLs are valid for a very short period of time after they have been retrieved.

To recapitulate, the following file names exist in gLite Grid:

- LFN – Logical File Name. LFNs are mutable, human readable names and exist in global, hierarchical namespace with each Virtual Organization having its own namespace.
- GUID – Global Unique Identifier. GUIDs are
  - constructed using UUID mechanism [141], which guarantee their uniqueness.
  - immutable – once a file obtains a GUID, neither GUID nor the file can be modified. If not, consistency will be lost.
- SURL – Site URL. SURLs, also denoted as Physical File Names (PFNs), indicate an instance of a replica and are accepted by SRM interface. According to Kunszt et al. [138] Storage URL (StURL) is a term used for an actual file name inside storage system, whilst Site URL (SURL) is a logical name.

- Symlinks – Symbolic links that point to another LFN. They have weak consistency, i.e. may point to nonexistent LFN and can create cycles. In addition, operations on target LFN do nothing to update symlinks.
- TURL – Transport URL. TURLs are valid URIs with all requisite information to access a physical file on a storage element.

One-to-many relationship exists between GUID and LFN, GUID and SURL and between LFN and Symlinks (see figure 26).

Formats of each file name in gLite according to [47] are as follows:

`LFN` `lfn:<any_string>`, e.g. `lfn:importantResults/Test1240.dat`
In the case of LCG File Catalogue, the filename format is
`lfn:/grid/<MyVO>/<MyDirs>/<MyFile>`

`GUID` `guid:<36_bytes_unique_string>`, e.g. `guid:38ed3f60-c402-11d7-a6b0-f53ee5a37e1d`

`SURL` `<sfn|srm>://<SE_hostname>/<some_string>`
In the case of `sfn` prefix, the format is
`sfn://<SE_hostname><SE_Accesspoint><VO_path><filename>`,
for instance `sfn://tbed0101.cern.ch/data/dteam/doe/file1`
On the other hand, SRM managed storage element (SE) often maintain a virtual file system, so such an assumption cannot be taken. An example of SURL of SRM managed SE could be: `srm://srm.cern.ch/castor/cern.ch/grid/dteam/doe/file1`

`TURL` `<protocol>://<some_string>`, e.g. `gsiftp://tbed0101.cern.ch/data/dteam/doe/file1`

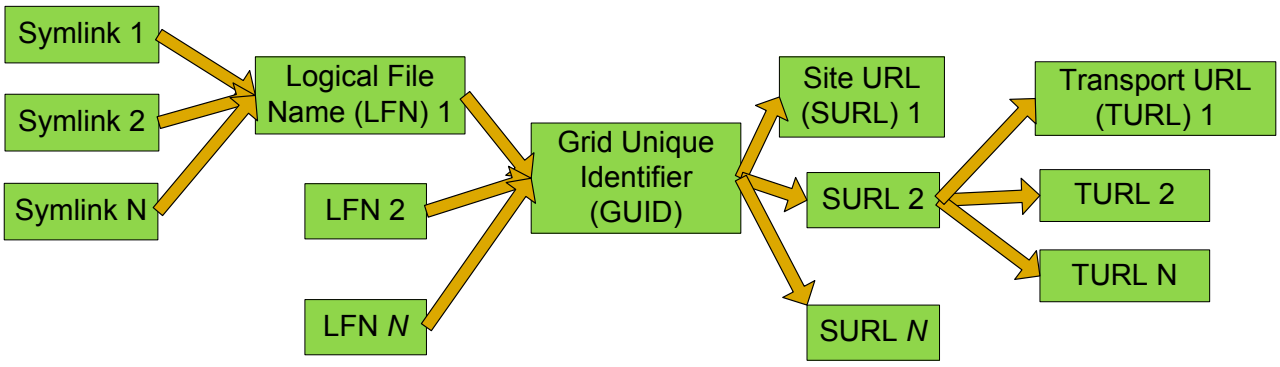


Figure 26: Filenames in gLite

Catalogue types Although most catalogues provide services coupled in one server, there are conceptually four types of catalogues (see figure 27):

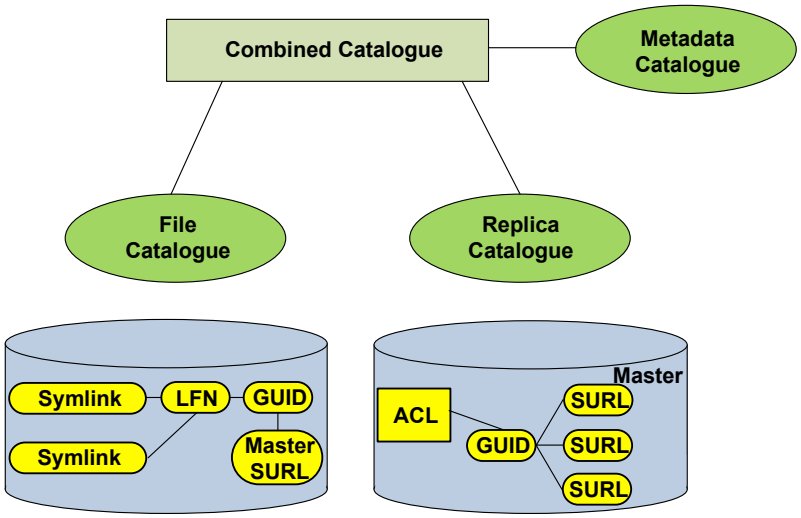


Figure 27: Catalogues in gLite [138]

1. File Catalogue – exposes operations such as creating directories, symlinks, renaming or deleting files and folders and registering grid files under logical file names. In brief, provides operations on LFN namespace.

2. Metadata Catalogue (MC) – MC interface provides operations to set, get and query metadata, i.e. some data connected with LFNs.

3. Replica Catalogue (RC) – RC manages list of replicas of a file identified by GUID.

4. Combined Catalogue – Unites operations on several catalogues to provide higher-level functionality, e.g. creating or deleting files. For instance, creating a new file involves storing replicas in storage elements, associating replicas' Site URLs to a particular GUID in replica catalogue and finally, creating a logical file name and connecting it with GUID in file catalogue. Combined catalogues must maintain a consistent state during all operations.

Usually, it makes sense not to divide these catalogues into separate entities. Therefore, combined catalogues are most often used. As regards security supported by catalogues, Access Control List (ACL) can be associated with files.

Catalogue implementations One of the results of European Data Grid (EDG) [192] project was a Replica Location Service (EDG RLS) [49], which is a catalogue based on web services model that enables management of distributed replicated data and related services. An example is enabling movement and replication of data, optimization of access etc. Architecture of RLS is divided into Local Replica Catalogue (LRC) and Replica Location Index (RLI). The former maintains information about replicas at a single site while the latter contains information retrieved from various LRCs that are updated occasionally and thus may not be up to date. An additional constituent, Metadata Catalogue Service allows users to define mapping between

LFNs and GUIDs. In addition, it stores information, such as file size, owner and creation date. Replica Optimization Service concentrates on the selection of best replica of file for a given job with regard to location, storage, latencies etc. Cameron et al. [49] admitted that they did not test their software under heavy load from concurrent clients – they narrowed down their performance tests to single LRC. Further testing [9] demonstrated that EDG RLS suffered from slow insertion and query rates which limit performance of the entire system [160].

Two projects were introduced to supersede EDG RLS due to its performance limitations: LCG File Catalogue (LFC) [1] produced by Data Management team at CERN for Worldwide LHC Computing Grid (LCG) project and File and Replica Management (FiReMan) [163] catalogue launched by Enabling Grid for E-sciencE (EGEE), an European Commission funded endeavour.

LFC is stateful and connection-oriented and offers increased performance compared to EDG RLS as Munro et al. [160] indicated. It supplies transactions API, which allows transactions to be started, committed or aborted. Furthermore, it allows for sessions to reduce the overhead of establishing SSL connection before each operation. Implementation of LFC has been done solely in C using multi-threading [1]. It was shown [160] that LFC is faster for single operations than FiReMan, probably due to its modest communication overhead compared to FiReMan SOAP API. Santos and Koblitz [191] argue that SOAP is 2 to 5 times slower than corresponding TCP implementation. On the other hand, when operations are executed in bulk, FiReMan comes first, which is most possibly caused by efficient use of Oracle database functionalities. LFC appears not to use specifics of particular database management systems, since tests using both Oracle and MySQL yield comparable results. On the contrary, FiReMan – a catalogue server whose logic is written mainly in Oracle PL/SQL stored procedures [163] sustains much better performance when its database back-end is Oracle than when it is MySQL [160], i.e. since stored procedures are written for Oracle, the usage of MySQL causes the whole logic to be executed within Tomcat servlet container. Another difference between the two catalogue implementations is that FiReMan, in contrast to LFC, follows a service-oriented approach – clients convey messages via SOAP over HTTP(S) with Axis application running inside Tomcat.

With regard to security, authentication to both catalogues is performed using X.509 grid certificates, which is acceptable as these certificates are standard security mechanism of gLite middleware. With respect to file permissions, both Access Control Lists (ACLs) and UNIX style file permission are supported by each of the two catalogues. In addition, the mentioned catalogues expose virtual hierarchical filesystem namespace, operations and file semantic as described in paragraphs earlier. Abadie et al. [1] report that LFC uses Virtual Organization Membership Service (VOMS) [53] for authorization; the same is true for FirReMan. Particularly, using ACLs a user can grant access for users and services specified by Distinguished

Name (DN)[52] or VOMS attributes (VO membership or groups)[53]. A comparison of authorization services in several grid middlewares, namely Globus Toolkit 4, gLite and UNICORE, can be found in [106].

With respect to means of access, LFC can be contacted using a C library with Python and Perl bindings or using a command line interface (CLI) somewhat analogous to UNIX shell commands [1]. As it was mentioned in *Motivation*, a web service Data Location Interface (DLI) is also available, although it does not support authentication and is read only – that is because it is not intended for end users but for Workload Management Service.

Storage elements As previously mentioned in *gLite data storage approach* paragraph, Storage Resource Manager (SRM) interface was conceived to make storage technologies transparent to VOs. In addition, SRM brings a web service interface to storage, providing functionality to upload files to a storage element, extract or delete them, e.g. `srmPut()`, `srmGet()`, `srmAdvisoryDelete()` [205]. Moreover, five constituents that make a storage element have been identified. There are at least 4 SE implementations: CASTOR 2, dCache, DPM and Medical Data Manager (MDM) [158].

'CASTOR 2' abbreviation identifies with *CERN Advanced STORage manager*. As Stewart et al. [205] indicate, it was designed around a mass storage tape system, and therefore is not appropriate for exploitation at sites without this facility. CASTOR 2 provides a single namespace for file management; supports *rfio*, *root* protocols for LAN access while SRM and gridftp are used with Wide Area Networks. A key component of CASTOR 2 is a stager that administers disk pools of tape system and facilitates access by using a scheduler plugin – LSF batch system scheduler is utilized for this purpose. A valuable feature of CASTOR is its capability to dynamically replicate frequently used files and to switch access to a less busy replica on a current open file. As regards CASTOR monitoring, it is performed using both LEMON (LHC Era Monitoring) and Oracle database management system logging features.

dCache is a storage element implementation developed by Deutsches Elektronen-Synchrotron (DESY) in collaboration with Fermilab. It endeavours to provide means for storing and retrieving large amounts of data among a number of heterogeneous server nodes. dCache exposes a single namespace view of all files under its administration. When a tape backend is connected to dCache, it becomes a hierarchical storage manager (HSM), i.e. when frequently used, it moves data from tape to disk and then back to tape. Former namespace used by dCache was PNFS (Perfectly Normal File System) while the current filesystem implementation used by dCache is Chimera. File access is possible using *dcap* (dCache access) protocol or xroot. WAN access is possible using GridFTP and SRM protocols. dCache load balances system by replicating frequently used files. An interesting feature from a reliability perspective is the ability for an administrator to control a number of replicas of each file, e.g. state that it must

[52]Distinguished Names (DNs) are found in the subject of a grid certificate.

[53]These attributes are found in VOMS enabled grid certificates.

be between n and m replicas available in each separate pool. With regard to installation of dCache, significant integration with WLCG YAIM[54] has been provided.

Disk Pool Manager (DPM), introduced by LCG project at CERN, puts emphasis on the ease of configuration and maintenance, with Tier-2 centres in mind. DPM is written entirely in C and shares much of nameserver code with CASTOR and it exposes gridftp for WAN transfers and rfio for LAN access. A database backend is required for DPM to operate. Both MySQL and Oracle are supported. Since DPM is mostly envisaged as storage element software with ease of installation and administration in mind, support for YAIM is provided.

In 2007, CASTOR was installed in CERN and 4 WLCG Tier-1 centres managing 50 million files and 5PB of storage and dCache was used in approximately 40 WLCG sites while DPM in 70 [205].

The last storage element software mentioned, the *Medical Data Manager (MDM)*, is a bridge between DICOM compliant storage and gLite middleware translating grid file read operations into DICOM transactions. The use of grid services enables unified view of data stored in dispersed DICOM servers [158].

File transfer service gLite File Transfer Service (FTS) is an infrastructure service intended to facilitate data movements. Users' transfers are assigned to channels, which are unidirectional links between sites. Channels may be dedicated or non-dedicated. A dedicated channel is a point-to-point link between sites while non-dedicated channel links group of sites. As regards data transfer requests, SOAP over HTTPS interface is provided to clients enabling submission of transfer jobs and polling for their statuses. FTS is backed by a MySQL or Oracle database, which is a central and critical component for it to operate as the state of the service is kept there. On the other hand, A SOAP server is stateless and can be load balanced. Other constituents of FTS are VO agents, which are daemons that apply VO-specific policies to transfer jobs, e.g. apply retry policy in case of transfer failure. Finally channel agents carry out actual transfers interacting with SRM and gridftp servers.

Information systems Information and monitoring service for gLite storage is provided by Berkeley Database Information Index (BDII), which keeps track of both static information, e.g. existing storage and computing elements, number of available CPUs, supported Virtual Organizations and dynamic information, for instance how much free space is available on a specified storage element or how many free CPUs a given computing element possesses. BDII is an LDAP-based information system, with services on the Grid publishing information about hosts under their administration in LDAP. Site BDII (SBDII) aggregates this information at

[54]YAIM – YAIM Ain't an Installation Manager. As http://yaim.info/ indicates, "The aim of YAIM is to provide a simple installation and configuration method that can be used to set up a simple Grid Site but can be easily adapted and extended to meet the need of larger sites."

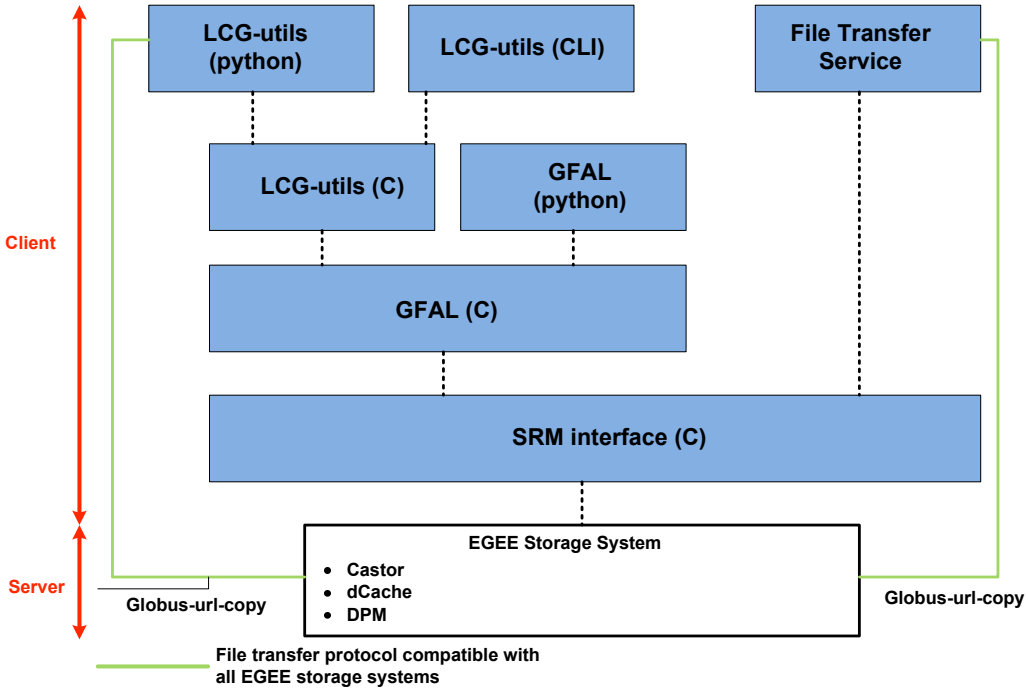


Figure 28: Client tools for interacting with gLite storage [1]

site level. Finally, Top Level BDIIs (TL BDII) query lower-level Site BDIIs creating complete view over the whole infrastructure. Unlike components mentioned in earlier paragraphs, communication with BDII does not require authentication – every client has read-only access.

Client utilities Figure 28 shows client tools for interacting with gLite storage, in particular

- LCG-utils: CLI interface, C library and Python and Perl module – highest level of abstraction enabling storing, replication, deletion and copying files.
- Grid File Access Library (GFAL) – a C library providing POSIX interface to storage on the Grid.
- File Transfer Service (FTS)
- SRM SOAP interface

Figure 29 depicts various operations performed with gLite components on execution of `gfal_open` function from GFAL library. Firstly, LFC catalogue is contacted to obtain list of replicas for a given GUID. Secondly, BDII is queried to acquire a version of SRM interface to use. Subsequently, TURL is taken from DPM storage element using SRM interface. Finally, file can be opened using one of the access protocols, e.g. gsirfio – a secure RFIO.

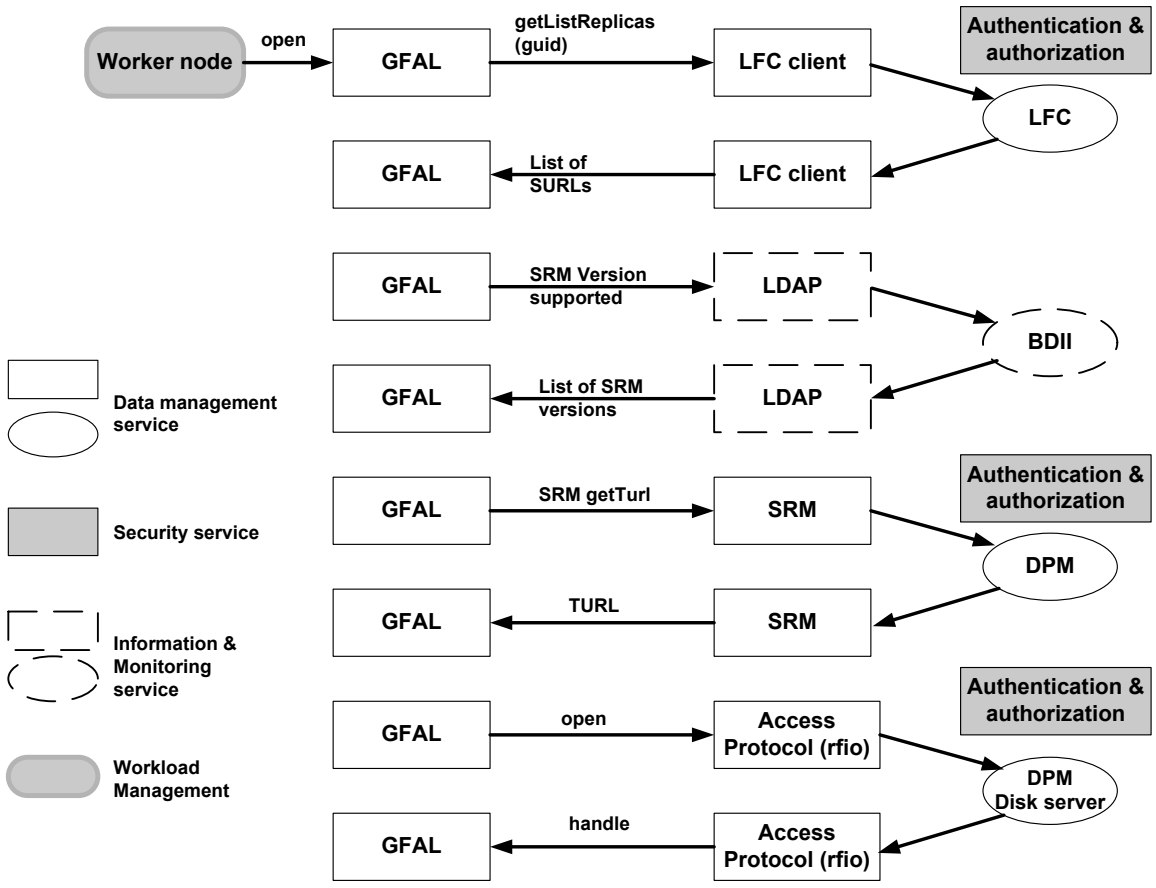


Figure 29: Execution of `gfal_open` function [1]

4 Needs to be addressed / Problems to be solved

The content of chapter 3, as recounted by the dissertation author, refers to achievements of Virtual Laboratory, its philosophy, concepts, middleware, components, services and architecture. Furthermore, some introductory information regarding addressed storage services, namely gLite data services, has been provided. This chapter will state the challenges that are to be addressed by the thesis author.

4.1 Providing access to EGEE/WLCG data sources

Access to EGEE storage services may be useful to projects, such as ViroLab or PL-Grid Virtual Laboratory, ChemPo and other Grid projects. However, it creates many difficulties and prerequisites. Firstly, in order to use the mentioned client utilities, a so-called gLite User Interface is required, i.e. a computer with gLite installed, configured and connected to EGEE/WLCG Grid. Furthermore, gLite is available only to certain distributions of Linux (e.g. Scientific Linux). Secondly, configuration of this software is not an easy task and requires substantial administrative and procedural work when attaching a computer to EGEE/WLCG grid. On the other hand, it is not possible to access EGEE/WLCG storage from computers without a valid gLite installation. Therefore, users usually obtain shell accounts on some gLite UI server in order to use EGEE services and use these services remotely, logging through SSH. Similarly, in order to programmatically access gLite storage services, software making use of these services must also be executed on gLite UI which is a restricted requirement. Additionally, every user needs to go through a long and error-prone procedure of obtaining a grid certificate, generating key pair and completing several request forms. This deters users from employing Grid services in their research work and additionally imposes a learning curve needed to work from Linux console and to use gLite command line interface

To paraphrase the Jargon File; from user's perspective, these actions may be perceived as a "pointless activity, which is actually necessary to solve a problem which solves a problem which, several levels of recursion later, solves the real problem you're working on." Therefore, it would be beneficial, if users could use these services without satisfying so many prerequisites. Moreover, since many powerful end-user tools exist inside Virtual Laboratories [122], integration with EGEE storage services would be useful, as there would be no need to switch between two environments: gLite command line and Virtual Laboratory tools.

4.2 Integration with the GridSpace Engine

Integration with the existing DAC2 infrastructure is one of the main requirements and the solution should be consistent with existing GridSpace Engine data access approach.

Further complexity occurs with the integration of new data source type into existing Data Source Registry schema, which does not take into account so many configuration options or

specific needs, regarding access to catalogues and storage elements, together with managing certificate information. Examination showed that DSR schema had to be reorganized to enable integration with EGEE/WLCG data sources. Nonetheless, merely altering database schema is only a small part of the work to be done, since database access layer had to be rewritten so as to take advantage of new schema. Changes to be made in database access layer were apparent in both DAC2 and DSR-plugin.

A supplementary need to be addressed was reorganization of DAC2 data access API, as it did not allow typical catalogue operations that were to be provided by new data source, nor did DAC2 supply constructors that could initialize new data source with a variety of credentials in case they are not downloaded from DSR. Therefore, modules that dispatch data access requests to diverse data source connectors had to be revised and altered.

4.3 Automation of certificate management

Management of users' certificates was a complex issue and difficult decisions had to be taken, so as not to compromise security. Another issue was automation of generation of proxy certificates, so that users would not have to create them manually. Additionally, credentials must be stored in DSR and proper credential must be provided when a Grid operation invocation occurs – a grid certificate, private key and private key passphrase must be conveyed if operation is the generation of a proxy certificate. On the contrary, if it is a Grid operation e.g. accessing LFC catalogue or accessing a file, proxy certificate must be sent; when proxy is not present or is expired, it should be generated and saved in DSR.

Finally, an important issue regarding certificate management is the need for communication encryption and maximum security of user files, especially private keys and certificates if they are stored temporarily.

4.4 Extending the DSR plug-in to enable registration of LFC data sources

A number of core requirements of the solution have already been identified, although satisfying all the mentioned requirements still will not make the solution usable if there is no means of adding, deleting and updating data source information. Of course, one could edit Data Source Registry contents directly using database utilities. Nevertheless, if we want to have a user-friendly solution, a graphical user interface to DSR LFC data sources is mandatory.

5 Related work

In chapter 4, the dissertation author put forward several goals that are to be achieved by this thesis project, together with discussion on possible solutions that could satisfy these requirements. This chapter will mention several projects that touch upon comparable substance as stated in this paper. In particular, 5.1 will recount other Grid-based virtual laboratories that were created, 5.2 tries to present various efforts that endeavor to make Grid service-oriented and 5.3 on the other hand, describes ways of how some Grid-projects handle data access and storage. Finally, 5.4 touches upon libraries providing access to gLite storage.

5.1 Other virtual laboratories

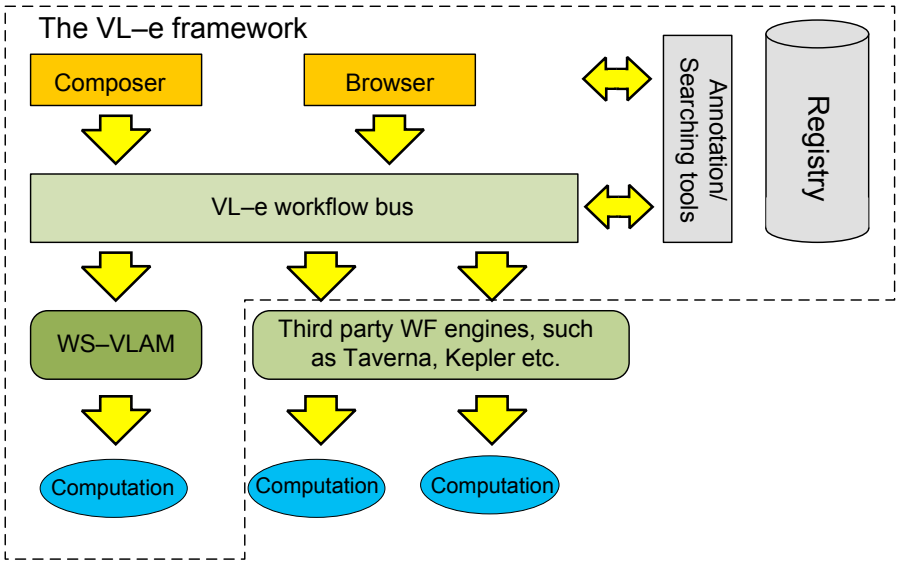


Figure 30: Virtual Laboratory for e-Science architecture (figure from [238])

Virtual Laboratory for e-Science (VL-e) VL-e [238] is a project, which aims at providing generic functionalities that support a wide range of specific e-Science application environments and the setting up of research infrastructure for evaluating diverse ideas. VL-e consortium is composed of a number of Dutch scientific and business partners. Some usage VL-e scenarios include modeling and managing workflow templates, browsing distributed resources, integrating third-party workflow systems and composing and executing application workflows. VL-e tools have been presented on figure 30 and they comprise of Virtual Resource Browser (VBrowser) to interactively access miscellaneous distributed resources, manipulate data, start applications and monitor resources. Furthermore it includes a FRIPS tool that supports interactive execution of parameter sweep applications, WS-VLAM workflow system, which enables scientists to design and monitor workflow execution and a server-side engine for scheduling and enactment of workflows – the workflow bus [237]. Some applications of VL-e include Real Time Monitor,

which tracks projects, participating in EGEE and WLCG grid, medical visualization application for planning shoulder replacement, Virtual Lab for functional MRI (VL-fMRI) applications to facilitate storage, analysis and sharing of fMRI data, Bird Avoidance Model that helps avoid collisions with birds by aircraft..

VLAB VLAB [168] virtual laboratory is a research project, which has been developed by Poznań Supercomputing and Networking Center since 2002. VLAB enables users to access scientific instruments connected using PIONEER optical network. Currently, this apparatus includes:

- 2 NMR spectrometers located in Institute of Bioorganic Chemistry of Polish Academy of Sciences and at Adam Mickiewicz University in Poznań
- a 32m radio telescope situated in Piwnice near Toruń possessed by Radio Astronomy Department of the Nicolaus Copernicus University in Torun, Poland and a second radio telescope positioned in Mexico City
- Freeze Atmospheric Dryer, which is a custom device built by Faculty of Process and Environmental Engineering of the Technical University of Łódz, Poland.

VLAB design is composed of 3 layers: Access, Grid and Monitoring. Access Layer encompasses components responsible for user interaction, including a web portal and a data input interface. Grid Layer communicates with grid middleware; in particular, delegates computational tasks to Globus toolkit and collects results together with response messages. Monitoring layer contains a scheduler, user account module and system monitoring component.

VLAB allows for so-called *dynamic measurement scenarios*, which are workflows specifying a set of computational tasks and experiments performed using remote accessible apparatus. Such a workflow is designed and submitted using Scenario Submission Application (SSA) and executed by Scenario Management Module. In addition to remote access to equipment, VLAB encompasses Digital Science Library (DSL) – a product of PROGRESS project. DSL is a distributed data management system allowing users to store results of experiments and associated materials. Apart from basic remote access to instruments, the ambition of VLAB is to give added value by combining results from several devices, e.g. radio telescopes, to provide higher resolution of the entire measurement.

myExperiment *myExperiment Virtual Research Environment for collaboration and sharing of experiments* [65, 67] was launched in November 2007, a project lead by University of Southampton and University of Manchester, which endeavors to provide "workflow bazaar" for workflow management systems and for other scientific assets, such as academic papers, Power-

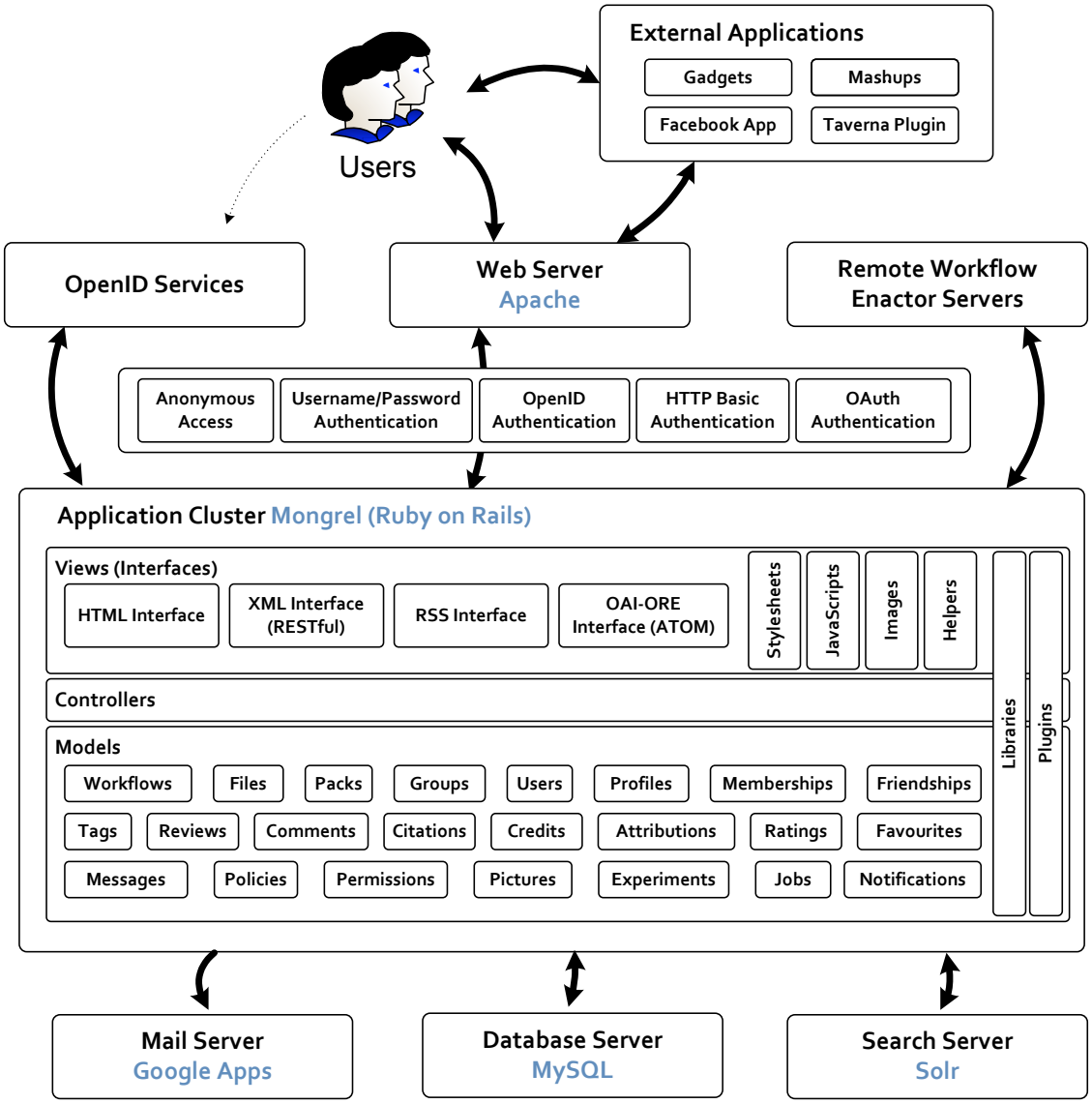


Figure 31: *myExperiment* architecture – figure shared on *myExperiment* website by David de Roure, myExperiment director, using Creative Commons Attribution-Share Alike 3.0 Unported License

Point slides, input and output data, service invocation logs, Visio diagrams[55] and other various types of files. Authors of myExperiment anticipate that workflows will become part of a scholarly knowledge cycle, i.e. a process of publishing scientific results and reusing these results by the scientific community. Community noticed scientists moving from writing stand-alone applications into reusable workflows; then, sharing workflows using emails, wikis, publishing them on personal websites, which was thought to be quite cumbersome. In order to streamline the sharing process, they created a collaborative environment, where scientists can distribute their workflows. myExperiment authors perceive workflows not only as a way of describing

[55]e.g. the figure 31 with myExperiment architecture is produced from a Visio drawing that has been shared by David de Roure, myExperiment director, using Creative Commons Attribution-Share Alike 3.0 Unported License

computational processes, but also as a means of communicating methodology and know-how and a method to avoid reinvention and propagation of best practices [66]. In order to convince scientists to publish their research work in myExperiment online system, they put emphasis on attributes that are important to them, namely credit, attribution and license. With regard to the design approach, a system was created that combines social networking, wikis and workflow sharing with reviewing and recommendation capabilities. In addition, users can inspect workflows by extracting their metadata, identifying services used and previewing these workflows graphically. A distinguishing feature of myExperiment is support for packs, which are groups of workflows, together with related files that are prepared for sharing. myExperiment service is both accessible using web pages and by utilizing its API. The usage of the latter enabled creation of Google Gadgets and Facebook applications. Additionally, researchers are equipped with a capability of enacting their workflows 'in the cloud' by submitting a collection of research objects to be processed remotely by myExperiment. An interesting feature of the project is 'social metadata', which is composed of attributions, creditations, favorites, ratings, reviews, citations, comments, tags and policies.

With regard to implementation, the main application of myExperiment is built using Ruby on Rails framework, while workflow enactment engines, database server, search server and mail server are external – main application connects to them. Authentication can be done by both using username/password method or via OpenID services. As regards interfaces, HTML, RESTful XML, RSS and ATOM interfaces are supplied with REST authentication being provided by OAuth library. Ruby on Rails application is deployed on Mongrel Cluster, while static content is served by Apache. The database system used is MySQL while search server is Solr. In addition, Nagios tools are employed for monitoring. Furthermore, myExperiment functionality has been extended by several projects, such as BioCatalogue, SKUA astronomy project and NEMU music analysis project, which was possible thanks to the open source character of the project. Usage statistics by De Roure et al. [66] show that over the period January–July 2008 myExperiment site received 60000 page views in 13500 visits by 8581 unique visitors, with workflows being downloaded 50934 times. Some other interesting figures are also presented in the referenced publication. myExperiment is definitely a successful endeavor that has attracted a quite large scientific community, especially from the field of bioinformatics.

myGrid myGrid is a middleware for Semantic Grid lead by University of Manchester, which enables biologists to perform and manage *in silico* experiment, and thereafter explore and exploit experiment results. myGrid goals are similar to those of ViroLab Virtual Laboratory, namely management of personal biological data and co-ordination of resources to manage virtual organizations of people, data, tools and machines. Research practice in which myGrid may be helpful, is where there is a need to repeatedly co-ordinate tools to produce results – "tasks that take minutes of computational time, actually take days to run manually"[204]. Stevens et al.

[204] presented their solution by a pilot application to explore William-Beuren syndrome (WBS) which is a rare disorder, a microdeletion in a region of human chromosome 7 characterized by a unique set of physical and behavioral features. As authors recall, before applying myGrid, results were obtained by manually interacting with bioinformatics services on the Web, manually copying results in order to form input of a subsequent task. Intermediate results were saved on a local file system, while their origin, relevance and status were noted by hand in a lab book. However, such an approach resulted in a rapid growth of files which are difficult to track manually and myGrid ambition is to solve this problem.

In order to shift the manual procedure into myGrid environment, several steps need to be followed. Firstly, all bioinformatics applications employed must be made available as a web service. Secondly, a user represents a bioinformatics process in a declarative way using Simple Conceptual Unified Flow Language (Scufl) in Taverna environment, which is a workbench enabling edition of Scufl workflows – both are projects created as part of myGrid endeavor. Thirdly, created workflows are enacted using Freefluo workflow enactment engine. Intermediate and final results are saved either in local file system or in myGrid information repository (mIR). In order support verification of origin or provenance of large sets of results, common experimental information model and automated provenance recording are utilized. The former adopts life science identifier (LSID), which is a class of universal resource name (URN). Workflow inputs, intermediate results and outputs; all are allocated an LSID. Afterwards, retrieval of metadata associated with items is possible. Alternatively, automatic provenance recording module records process provenance, which is a log describing which services were employed to generate data. Additionally, relationships between data are identified. For viewing and mining results, Haystack desktop application for browsing multiple views of Resource Description Framework (RDF) information is employed.

Linked Environments for Atmospheric Discovery (LEAD) LEAD [73] is a US National Science Foundation (NSF) funded venture whose objective is to create cyber infrastructure in mesoscale meteorology allowing for grid-based on-demand design and enactment of dynamic workflows in the domain of meteorology with ability of dynamic adaptation to changing requirements e.g. rapidly moving tornado or a flood. Foundations of LEAD are Web Portal, which is a major entry point to the applications, ARPS Data Assimilation System (ADAS) for data quality control and assimilation, myLEAD metadata catalogue service, Weather Research and Forecast (WRF) – atmospheric prediction and simulation model, ADaM (Algorithm Development and Mining) for mining data and Integrated Data Viewer (IDV) – a desktop application for visualization of a variety of multidimensional geophysical data. Principal components of LEAD built are the following subsystems: user, data, tools, orchestration, and grid; each responsible for one aspect of the system.

Kepler Kepler [5, 224] is a scientific workflow management system which allows for design, execution and deployment of workflows. Kepler equips a user with a library of reusable components, called actors, which perform computations e.g. signal processing or to provide access to data, for instance a relational database actor facilitates access to relational database. Input and output ports are defined for each actor and can be linked to a direct acyclic graph which specifies data flow between actors. A graphical user interface empowers users with an easy workflow construction mechanism. Generic Web service actors allow for utilization of services defined using Web Service Description Language (WSDL), while Grid service actors provide means for certificate-based authentication, Grid job submission and Grid-based data access using OGSA interfaces. Additionally, support for specialized data transformation actors, e.g. XSLT, XQuery, has been enabled while a *harvester* capability provides means of importing a whole set of related services from web pages or Universal Description Discovery and Integration (UDDI) repository.

Triana Triana [56, 224], a part of GridLab [130] project, is a graphical Problem Solving Environment (PSE) with basic unit of operation being a component – a Java class representing an algorithm or process with an identifying name, input and output ports, a number of optional name/value arguments and a single process method [224]. In order to write components in languages other than Java, apposite wrapping code must be provided. Triana is flow based – data arriving on the component input which triggers its execution. Multiple inputs indicate that execution will be suspended until all inputs arrive at the component, or if developer wishes, the execution will trigger immediately. Execution in Triana is decentralized with data and control flow messages being sent through communication pipes. Internal workflow representation is object based – each component or task has an accompanying Java object. Instead of common Directed Acyclic Graph model, Directed Cyclic Graph (DCG) model is used, i.e. cyclic connections are allowed within Triana language. As with many workflow engines, external format is an XML file. Triana provides interoperability with external workflow language representations such as BPEL4WS through pluggable language readers and writers.

Other workflow systems Aforementioned systems are only a few of the many workflow design and enactment environments. Some alternative implementations include [224]:

- *Condor DAGMan*, which uses Direct Acyclic Graph (DAG) to represent a set of tasks – nodes symbolize tasks while edges symbolize the dependencies. DAGMan submits jobs to Condor in an order specified by DAG and processes results. DAGMan is base workflow scheduler used by other Grid workflow systems, for instance Chimera, Pegasus and P-GRADE.
- *UNICORE.* In this project launched by the German Ministry for Education and Research, DAG model is used for job description. UNICORE has 3-tier architecture: user, server

and batch subsystem level. In user level, users create jobs independent from system where jobs will run. UNICORE server level tackles managing resources, execution of jobs and returning results to users while batch subsystem tier tackles destination systems with their batch systems and storage.

- *Chimera and Pegasus in GriPhyN.* GriPhyN is a US National Science Foundation project to support large-scale data management in physics experiments, for instance gravitational wave physics or high energy physics. GriPhyN proposes concepts of an abstract workflow and a concrete workflow. Chimera is a virtual data system combining virtual data catalogue with Virtual Data Language (VDL) interpreter that translates user requests into data definition and query operation on database. Chimera is used to produce Abstract Workflows (AW) that are specified using DAG XML description (DAX) language. Pegasus, on the other hand, is exploited to map AW onto computational Grids, therefore creating a Concrete Workflow (CW). Subsequently, CWs, which are executables combined with runtime information, are submitted to DAGMan for enactment.

- Many more workflow systems not discussed here exist, e.g. GridAnt, ASKALON, GridFlow, GSFL, BPEL, McRunJob, Symphony, P-GRADE, ScyFlow, GALE, WebFlow, Collaborative Application Specification Tool (CAST), Grid-WFS. For a comprehensive list and comparison of grid workflow systems, the reader is counseled to consider [233].

gLite gLite [47] is software of EGEE and WLCG projects. Both share much of the infrastructure with former, aiming to create a geographically distributed computing infrastructure available to computational scientists, while latter aspires to provide infrastructure for simulation, analysis and processing of data of Large Hadron Collider (LHC) experiments. Authentication is provided using X.504 certificates, while authorization services are supported by Virtual Organization Membership Service (VOMS). gLite is job-oriented – jobs are specified using Job Description Language (JDL) which is based on Classified Advertisement (ClassAd) language. Jobs are submitted to resource brokers that route them to particular computing elements, which are usually cluster farm. User may interrogate status of a job, cancel it or retrieve results after its completion. Data management has already been discussed in section 3.6 of chapter 3.

Projects recounted in this section are just examples of virtual laboratories that enable performing of *in-silico* experiments. They range by degree in which they support computational scientist with their job; many are workflow-based, since numerous Grid practitioners believe that workflow-based software is the best way of utilizing Grid capabilities.

5.2 Attempts to make the Grid service-oriented

Historically, Grid was not service-oriented, but rather job-oriented. Modern approaches try to follow service-oriented paradigm, since it decreases costs of integration of software components

that were developed in isolation from each other and enables faster adaptation to changing requirements by employing on-demand computing model. In addition, wrapping legacy systems with service-oriented interfaces helps avoid rewriting of existing code.

Open Grid Services Architecture (OGSA) OGSA [92, 144] is composed of 3 main elements: Open Grid Services Infrastructure (OGSI), OGSA services and OGSA schemas. A basic building block of OGSA Grid is a Grid Service, which is a Web service that conforms to a set of conventions regarding its interface definition and behavior defined by OGSA. Grid Services are addressable, potentially stateful and transient. Grid Services are created using `createService` factory method which returns an invariant Grid Service Handle (GSH) and initial Grid service reference (GSR), which may change over service's lifetime.

Open Grid Services Architecture Data Access and Integration (OGSA-DAI) The goal of OGSA-DAI [10] is to bring a consistent service interface for data access and integration to databases exposed to Grid, concealing dissimilarities of these systems such as database driver technology, formatting and delivery mechanisms. OGSA-DAI achievements include consistent access to multiple database paradigms: relational and XML, support for incremental and bulk data delivery from services and files, full integration with existing Grid authentication and data transport as well as ongoing standardization. OGSA-DAI facilitates easier design of federation middleware hiding much of heterogeneity of underlining data resources. On top of OGSA-DAI, OGSA-DQP (distributed query service) has been built which enables distributed queries over nodes obtained from the Grid.

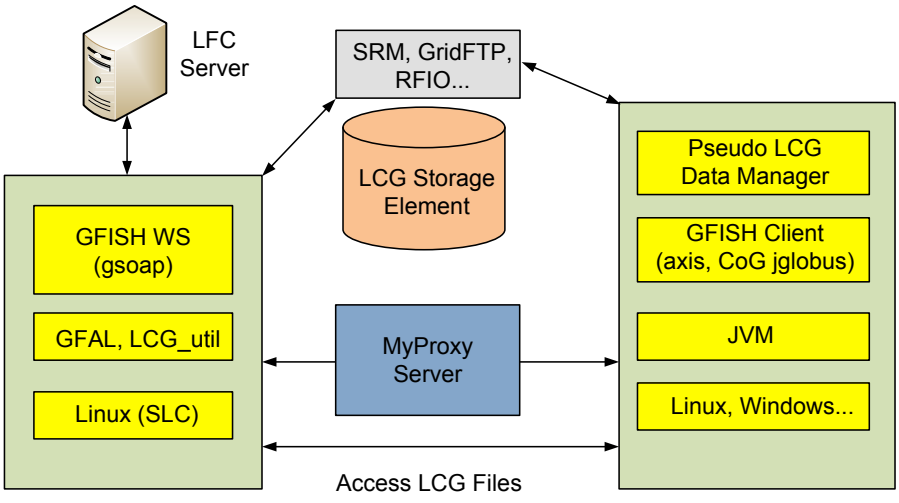


Figure 32: Grid File Sharing System (GFISH) architecture [232]

Grid File Sharing System (GFISH) Yaodong et al. [232] have developed GFISH (Grid File Sharing system), which includes a server providing a web service API for the LFC catalogue and a related Java client with Grid user credentials retrieved from a MyProxy server. They

implemented the server using gSOAP, while utilizing Axis on the client side, thus introducing significant transmission overhead. GFISH [231, 232] is a noteworthy project, because its goals are similar to objectives of this thesis project. In particular, GFISH ambition is to provide pseudo LCG file access commands, such as `lcg-cp`, `lcg-cr` etc. (see figure 32). A more recent paper [231] indicates that the authors enhanced their solution by dividing communication into two channels and currently metadata operations, e.g. listing directories, moving or replicating files and locating physical file address is performed through GFISH WS server, while actual data transfer is achieved using GridFTP protocol embedded in CoG jglobus tool.

5.3 Data access and persistence in Grid projects

As previously mentioned in *'Organization of the thesis'* section of chapter 2.3, an overwhelming majority of Grid projects still store data in relational, XML or occasionally, object databases located outside of Grid, which is of no interest from the thesis point of view. Nevertheless, some projects take advantage of Grid-enabled storage and data management services with a few examples being broached in this section.

ATLAS experiment ATLAS project has developed its own distributed data management system, termed Don Quijote (DQ2) [100, 205] which manages file-based data of all types. In particular event data, conditions data together with user-defined file sets and groups file-based data into datasets with a set of catalogues storing information of their location, constituent files and metadata. These include the following catalogues: dataset repository, which is a catalogue of datasets, dataset selection catalogue, dataset content catalogue and data location catalogue. Datasets posses a changeability state – they can be open or frozen (locked permanently) with data subscription service enabling users and sites to acquire data updates in an automated way via 'subscriptions' to mutable datasets. In addition to the catalogues mentioned, according to Stewart et al. [205] LFC is queried very frequently. Don Quijote is deployed in Tier-0 and 10 Tier-1 sites that participate in ATLAS experiment.

EUChinaGrid One of the ambitions of EUChinaGrid was to find proteins of potential pharmacological application. Genomic part of this project had an objective to identify stretches of genomic sequences of potential biological function that are not present in known protein and genetic databases. Piwowar et al. [179] report that they used LFC catalogue available for their VO to store input data for experiment in order to enable access to this data on all machines participating in computation. Sequences being the focus of experiment were grouped in sets of about 100000, stored on a storage element and registered in LFC catalogue. Main script automated the necessary work, including data transformation, execution of BLAST and copying output files to destination. A dedicated portal was developed as a user interface enabling selection of appropriate files and job submission. Piwowar et al. [179] assert that they were able

to carry out the whole experiment in 38 hours with average resource consumption of 126 CPUs.

InteliGrid Within the scope of InteliGrid project, a Document Management System has been developed [71], which is based on grid middleware services, namely OGSA-DAI and Grid Authorization Service (GAS). OGSA-DAI has been employed to integrate various back-end document storage systems: relation database management systems and WebDAV based servers.

PALADIN Paladin [101–103] project is interesting from this dissertation point of view, since it addresses the issue of dynamic data source integration in Grid environment and more importantly, because it developed a data source registry, a component that is difficult to find in existing projects, but is also apparent in our Virtual Laboratory architecture. In particular, Göres and Dessloch [103] developed a Paladin data source registry for registration and discovery of data sources. Paladin DSR implements Paladin Metamodel (PMM) [101], which is based on "typed, attributed multigraphs to represent atomic and complex features, types and relationships" [103] of data source schemas. Paladin DSR stores information on data sources to be exploited in schema matching process, which is performed by ScheMaF framework, which is also a part of Paladin project. On the other hand, our Virtual Laboratory Data Source Registry (DSR) is intended for storing data source information together with accompanying credentials, in order to automate access to these data sources. Although, purposes and realizations of both solutions are different, the general idea of storing structured information on data sources is similar.

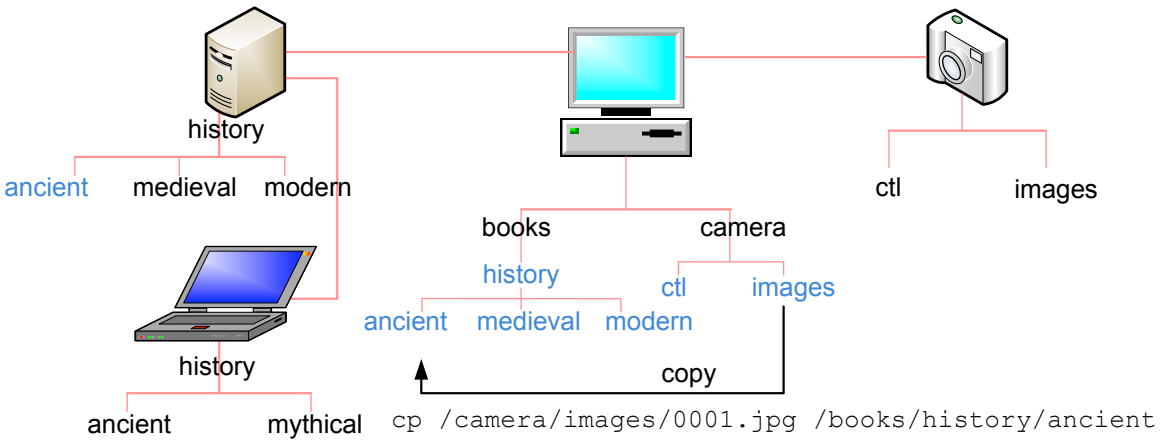


Figure 33: Inferno namespace exporting and importing (figure created on basis of presentation from Inferno website)

Grid, cloud computing and distributed file systems Apart from LFC and FiReMan that may be perceived as a distributed file system, several others provide file or replica catalogue functionality. These include DFSgc [3], Globus Data Replication Service (DRS), Giggle [54],

IGOR file system [6] and Grid Virtual Directory System (VDS) [59]. In addition, numerous large scale distributed file systems exist, e.g. Lustre filesystem, Hadoop Distributed File System, Google File System [99], IBM General Parallel File System [124], Microsoft DFS and Sun ZFS. An interesting example of distributed file system is the file system of Inferno operating system [72], which enables single-rooted namespace over a variety of resources connected to the network, specifically computers, databases, cameras and others.

5.4 Libraries providing access to gLite data resources

lcg_util C API This is a C library that provides the same functionality as LCG command line programs, which in fact, invoke `lcg_util API` functions. According to Burke et al. [16], it should cover most needs of user applications. `lcg_util API` interacts with LFC catalogue and is independent from underlying technology. `lcg_util API` functions begin with `lcg_` prefix, e.g. `lcg_cp`, `lcg_cr`. In addition to simple command, there are `lcg_util` functions that use buffer for complete error messages – they are formed by following function name with an 'x', e.g. `lcg_cpx`, functions with timeout – formed by following function name with 't' or functions with both functionalities – their names are followed by 'xt'.

Grid File Access Library (GFAL) GFAL C is high-level library bringing POSIX style interface for input/output operations on Grid files, concealing interactions with Storage Resource Managers, Storage Elements and LFC catalogue. However, it is lower-level that `lcg_util`. GFAL function names begin with *gfal* prefix, e.g. `gfal_read`, `gfal_close`. A user can supply GUID, LFN, SURL and TURL names as arguments to GFAL.

Both GFAL and `lcg_util` API need certain environment variables to be set, if they are to contact storage elements and LFC catalogue: `LCG_GFAL_VO` – a Virtual Organization name, `LCG_GFAL_INFOSYS` – list of BDII hostnames and ports separated by commas, `LFC_HOST` – LFC server address.

Lower-lever APIs gLite also provides some lower-level APIs, although their use is discouraged [47]. They include `LFC_client` API, RFIO API, gsidcap API, edg-gridftp Globus API, SRM API, edg-rm, edg-rmc and edg-rlc APIs (see figure 34).

Java lcg_util and GFAL API wrappers SEE-Grid and Gilda project developed wrappers to aforementioned C libraries using Java Native Interface (JNI). Gilda GFAL Java API (also termed APIGFAL) provides GFAL functionalities through three classes: GFalDirectory, GFalFile and GFalUtilities, while File Management Java API provides means to interact with gLite storage on higher level allowing not only for data access, but also for LFC catalogue manipulation. SEE-Grid File Management Java API exposes LFC operations with following

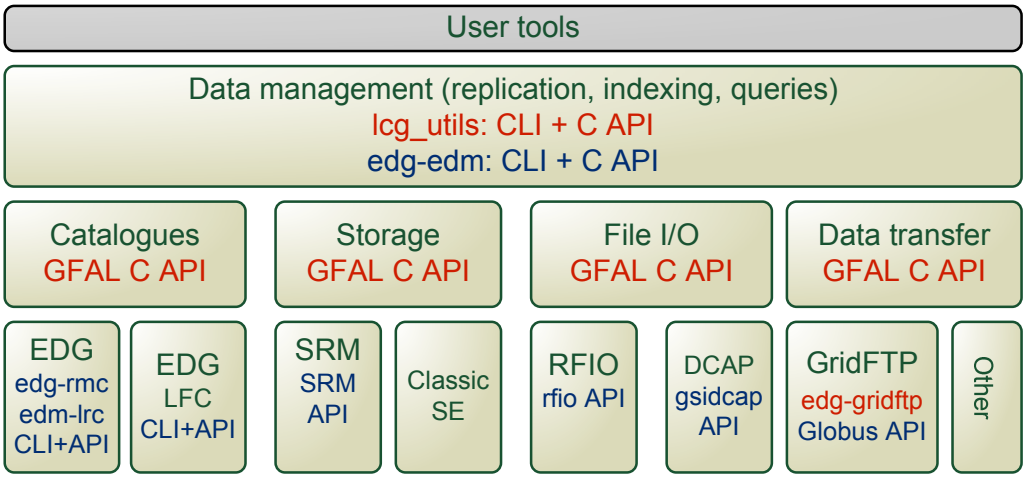


Figure 34: gLite data management application and command line interfaces – blue color indicates those that are depreciated [47]

classes: LFCDataStorage, LFCAliasItem, LFCDirectoryItem, LFCFileItem and a few helper classes, such as LFCFileMode, SEList and ItemIterator.

ChemPo LFC command wrappers ChemPo [202] project provided wrappers for SEE-Grid Java File Management API calls. ChemPo executes every data access or data management command in a separate Java Virtual Machine (JVM), thus enabling commands to run with different set of environment variables, which in turn enables to act on behalf of another user.

6 General software requirements

This chapter provides background for chapter 7 – *Detailed requirements* and following chapters of dissertation. It defines the environment in which component being designed will have to operate, external modules and software it will need to communicate and users it will serve. Furthermore, it contains an overview of the component functions. However, it does not provide information about its decomposition to modules nor implementation. This is the purpose of chapter 8.

6.1 Scope

The thesis concerns the ***LFC Data Source (LFC DS)*** component, which will enable Virtual Laboratory users to access and manage data administered by EGEE/WLCG storage services and in particular LCG File Catalogue and storage elements, by using GScript Virtual Laboratory language. Moreover it will manage their EGEE/WLCG credentials and provide means for registering EGEE/WLCG data sources. LFC DS is intended to provide only core functionality leaving out advanced gLite features, such as replication.

Benefits of using LFC DS for GScript developers can be abbreviated as follows:

- short learning curve
- interoperability with other JRuby code
- less effort put into programming EGEE/WLCG data access
- integration with Virtual Laboratory software

The main advantage for Virtual Laboratory users is support for large storage infrastructure and access to files that are present on gLite storage currently.

6.2 Product perspective

As the *Recommended Practice for Software Requirements Specifications* [236] suggests, this subsection should connect requirements of a larger system with functionality of component being developed. Therefore, it is noteworthy that access to EGEE infrastructure is one of the design goals for Virtual Laboratory project (see figure 10). Most of the envisaged components have already been created. However, one of the very few features that still needs to be added is integrated access to EGEE/WLCG data sources and this is the focal purpose of LFC Data Source component. With regard to components and modules and systems with which LFC DS will have to be integrated, in the scope of Virtual Laboratory include (see figure 35):

- Data Access Client, version 2 (DAC2)

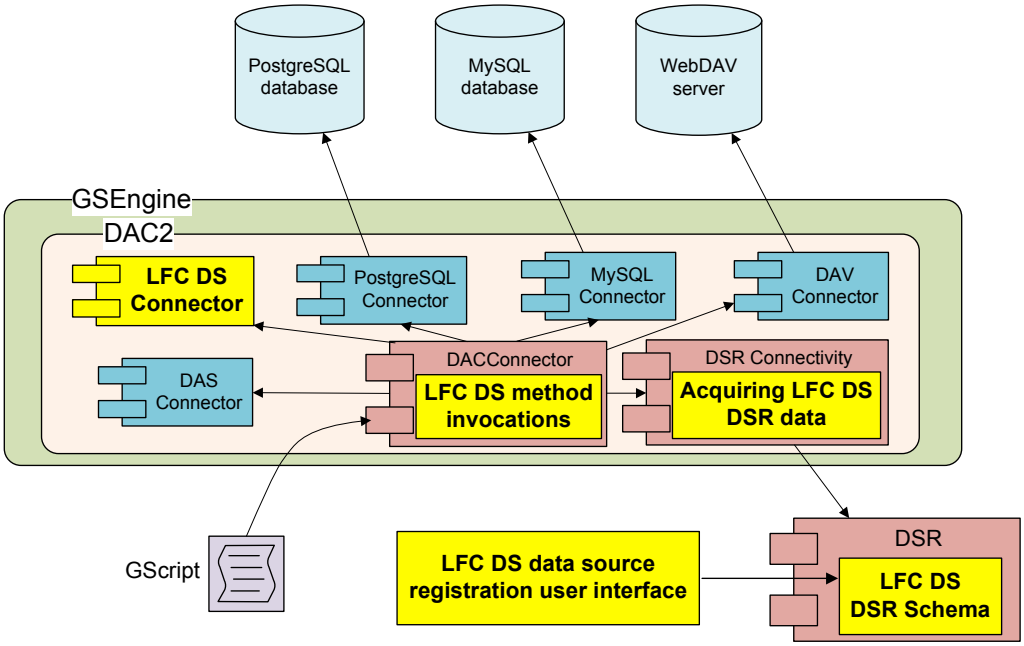


Figure 35: LFC DS (indicated by yellow color) in the context of Virtual Laboratory

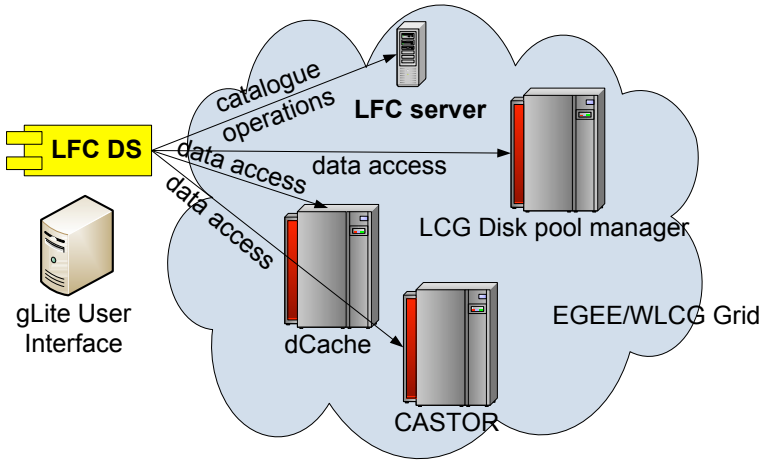


Figure 36: LFC DS in the realm of EGEE/WLCG Grid

- Data Source Registry
- One of user interfaces if there is a decision that an integrated LFC DS user interface is desirable
- Security system

Furthermore, LFC DS will have to operate in the realm of EGEE/WLCG Grid (see figure 36), performing data management and data access operations. Virtual Laboratory and EGEE/WLCG Grid are separate systems and the choice how to carry out simultaneous operation in these two distinct worlds is a design decision to be considered.

6.3 Product functions

Use case diagram on figure 37 which summarizes foreseen functions of LFC DS.

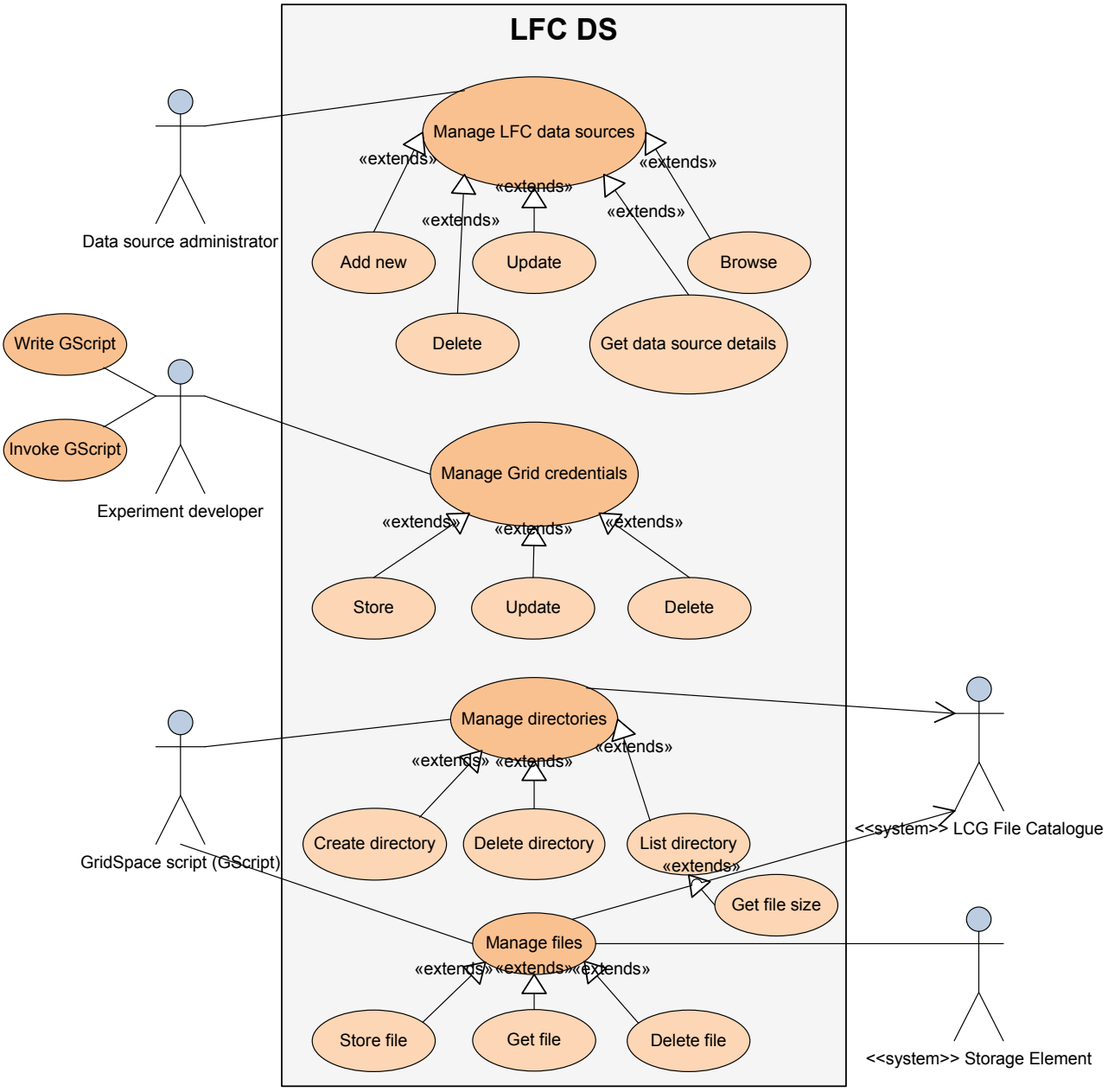


Figure 37: LFC DS Use Case diagram

6.4 User characteristics

Final users of LFC DS are mostly computational scientists, who have some programming knowledge. However, their main field of expertise is their scientific domain, such as bioinformatics or computational chemistry. They know fundamentals of JRuby programming syntax and are acquainted with Virtual Laboratory environment. In addition, they are often in contact with a professional computer scientist, who helps them solve encountered technical problems, e.g. if

they come across an exception they do not understand. On the other hand, GridSpace Engine software and an accompanying LFC DS component are installed by a system administrator who has considerable knowledge on Virtual Laboratory, GSEngine environment, UNIX command line and networking. Nevertheless, interaction of system administrator with LFC DS is limited to installing and configuring the software, starting services and unraveling problems encountered during its operation.

6.5 Constraints

Interview with Virtual Laboratory team members identified several restrictions LFC DS must follow:

- LFC DS module must not incorporate too many dependencies into GSEngine
- LFC DS should not store temporary files on a server where GSEngine operates
- Devised API must be simple
- LFC DS must automatically manage user credentials

Another consideration that is beyond question, is that the transfer of private keys and other sensitive data must be encrypted and any temporary files containing sensitive data should be kept no longer than are needed. Furthermore, a legal constraint the software must follow is that the libraries used by the project must fall under FLOSS[56] category in order to conform to copyright policy of GridSpace Engine, as mentioned in section 3.1.

6.6 Assumptions and dependencies

Assumptions of the following requirement specifications are as follows:

- Access to gLite UI with all gLite libraries in place is provided.
- Virtual Laboratory infrastructure is established, in particular there is access to Data Source Registry, security components and DAC2 data access layer.
- Finally, an imperative assumption is that at least some users possess a valid Grid credentials enabling them to use EGEE/WLCG Grid.

[56]Free Libre Open Source

7 Detailed requirements

The following chapter provides detailed requirements for LFC DS component with each requirement holding a unique ID composed of string UI, SI, FR, SC, NF indicating user interface requirement (UI), software interface requirement (SI), functional requirement (FR), a user scenario (SC) or a non-functional requirement (NF) respectively, followed by '-' and a number.

7.1 Functional requirements

With regard to functionalities of EGEE storage services that need to be provided to Virtual Laboratory, the dissertation author had several discussions with Cyfronet development team members and came to the conclusion that paramount goals of the devised API for accessing EGEE/WLCG storage resources have simplicity and accessibility. This is also an approach used in DAC2 data access architecture, where configuration of data sources is required only once and can be done by a qualified person. Such a configured data source can then be used by a number of computational scientists who do not need to specify every detail concerning the data source as this information is downloaded from DSR – this mechanism eliminates the burden of remembering various endpoint URLs and technology dependent information. Similar ideas should be employed when designing services for accessing EGEE/WLCG grid. Moreover, actual file operations that need to be supported include, obtaining a file represented by logical file name from Grid, storing a file in Grid filesystem, obtaining the size of a file, creating a directory in a file catalogue and deleting a file or directory. File permissions are not required to be supported, as they may intimidate users – a transparent access without file permissions is a better solution in this case. Additionally, the user should not be required to specify which storage element is used for accessing a replica or saving a file. If there is no automatic optimization mechanism employed, there needs to a default storage element used for each EGEE/WLCG data source.

Another step in collecting requirements was a review of existing code accessing EGEE storage resources within ChemPo project which gave the author valuable information on actual servers being contacted and services being used and in particular, our VO uses LCG File Catalogue as a grid file catalogue. Additionally, VOMS enabled grid certificates are employed. Requisite configuration parameters that were used by ChemPo software were location of certificate repositories, VOMS directory, VOMS server, Site Berkeley Database Information Index host, LFC host, storage element URL to be used, locations of user private keys, grid certificates, proxy certificates, passwords and Virtual Organization name to be used. The usage of LFC implied that FiReMan web-service interface could not be used and another solution would have to be envisaged. In chapter 8, various possibilities of integration of these services using existing gLite client utilities into GridSpace Engine are discussed.

As already mentioned in chapter 4, one of the issues is automation of generation of proxy

certificates, so that users would not have to create them manually together with managing users' credentials. Therefore, software requirements must address this issue.

However, this does not limit the requirements of certificate management. Many users wishing to take advantage of Grid resources will not want to go to the trouble of generating proper key pair, applying for a certificate or waiting several weeks or even months for a certificate to arrive. Often, the need to perform computation is urgent and there is no time for these activities. Furthermore, many non-computer scientists are intimidated by procedures when applying for and receiving grid certificates which creates doubts as to whether users will follow the procedure correctly. A solution in such a situation would be for GridSpace data access system to allow authenticated user to perform Grid operations without the need to provide a certificate; the system would use other user's certificate who had agreed that their certificate could be used by other members of the group involved. Of course, this is a security compromise; however when used within VO boundaries and only to authenticated users, any tracing of damage caused by data operations on behalf of user who shared the certificate will be limited to authenticated users. Therefore, such a mode of operation makes sense only when the number of scientists within a collaboration is small. To scale this solution into larger user quantity, a tight control over who can perform operations using a particular certificate is of paramount importance. However, due to complications of the whole DAC2 infrastructure that would be implied by such a modification and because such a feature is not essential to the goals that are to be achieved by thesis project, these functionalities are left for future work as this facet is only necessary if adaptation of the solution to larger virtual organizations is mandatory.

Table 4: Functional requirements

REQ ID	Requirement Description
FR-1	Provision access to EGEE/WLCG storage resources
FR-1.1	Support obtaining a file represented by logical file name from Grid
FR-1.2	Support storing a file in Grid filesystem
FR-1.3	Support obtaining the size of a file
FR-1.4	Support creating a directory in a file catalogue
FR-1.5	Support deleting a file or directory
FR-1.6	Support listing directories
FR-1.7	Support checking for existence of files and directories
FR-1.8	Omit support for file permissions
FR-1.9	User should not be required to provide storage element name in method invocations – it should be remembered by system for each LFC data source.
FR-1.10	Let users refer to files only using logical file names (LFN)
FR-1.11	Use LCG File Catalogue for catalogue operations

Table 4: Functional requirements (continued)

REQ ID	Requirement Description
FR-1.12	Support standard gLite storage elements: dCache, CASTOR and DPM
FR-2	Create a mechanism that eliminates the burden of remembering various endpoint URLs and technology dependent information.
FR-3	Manage users' credentials using Data Source Registry (DSR)
FR-4	Allow for credentials sharing
FR-4.1	Allow to mark credential as being available to other authenticated users
FR-4.2	If credentials are not found for current user, automatically search and use credentials marked for sharing
FR-5	Automate Grid proxy generation

7.2 User interfaces

Users are will be able to interact with LFC DS using a GScript based interface, which is described in 7.3 and a graphical user interface (GUI) that satisfies requirements delineated in table 6. Their rationale includes management of user Grid credentials and management of information regarding LFC data sources. The following terms are used within a list of user interface requirements:

- *private key* – a user private key that can be used to generate a Grid proxy certificate
- *proxy certificate* – a temporarily generated certificate that allows for authentication and authorization in gLite environment
- *grid certificate* – a certificate signed by Certificate Authority (CA) confirming particular user's identity and that he or she is entitled to use Grid services

All requirements presented in table 5 are to be verified by manual examination of GUI.

Table 5: User interface requirements

REQ ID	Requirement Description
UI-1	Ability to upload or remove a private key or to check whether it has been uploaded.
UI-2	Ability to upload or remove a grid certificate or to check whether it has been uploaded.

Table 5: User interface requirements (continued)

REQ ID	Requirement Description
UI-3	Ability to upload or remove a proxy certificate or to check whether it has been uploaded.
UI-4	Ability to set passphrase that can be used to decrypt private key, remove it from system or check whether it has been set.
UI-5	Ability for the user to decide whether his or her credentials can be used for data access by other authenticated users.
UI-6	Capability to add new LFC data source with necessary information enabling other LFC DS components to utilize it.
UI-7	Capability to edit or delete existing LFC data sources.
UI-8	Checking for simple errors in fields provided by user, before submitting them to system
UI-8.1	checking whether user has provided mandatory fields
UI-8.2	verifying that host names provided by user are valid
UI-8.3	checking validity of fields for input of numbers
UI-10	Providing contextual help.
UI-11	Capability of browsing the list of LFC data sources.
UI-12	Securing user interface by Virtual Laboratory security system.
UI-13	Integration with existing Virtual Laboratory components for registering and updating of data source information.

7.3 Software interfaces

Software interface requirements are listed in table 6 and they refer to DACConnector methods that will be accessible from GScript code. When perusing these requirements, the reader is advised to note the following statements and conventions:

- When a requirement refers to a path, it is meant to be an LFC catalogue path without `"/grid/vo_name/"` part (last "/" may or may not be left out) as specified in SI-0.
- A convention for distinguishing class and instance methods from *Programming Ruby: The Pragmatic Programmers' Guide, Second Edition* [214] has been employed, namely `ClassName.method_name` is used to indicate a class method while `ClassName#method_name` is used to denote an instance method.
- DACConnector instance method invocations presented here apply only to DACConnector class instances initialized with a data source handle that refers to an LFC data source.

- If a requirement makes reference to a user, it denotes a user who executes GridSpace script invoking DACConnector methods or a script developer depending on context
- *handle-name* is a data source handle referring to an LFC data source

All requirements presented in table 6 are to be verified using test methods created using a chosen testing framework.

Table 6: Software interface requirements

REQ ID	Requirement Description	Rationale & Comments
SI-1	LFC DS methods represent paths by providing only the text after `"/grid/vo_name/"` part, where `vo_name` is the LFC catalogue of data source being used. Last "/" of `"/grid/vo_name/"` text may be present or omitted.	Allows users to type less when writing script by reusing information stored by LFC DS.
SI-2	If a valid credential is stored for a user or some other user of Virtual Laboratory had agreed that their certificate could be used by other members of collaboration, invocation of method `DACConnector.new(handle-name)` should initialize DACConnector object with a reference to an LFC DS connector enabling a script to perform subsequent LFC DS method invocations. If user credentials are not present, an exception should be thrown and DACConnector should not be initialized.	Enables initialization of LFC DS in a fully automated way without explicitly specifying credentials by a user.
SI-3	`DACConnector.new(handle-name, password)`, where password is a passphrase to Grid private key stored for a user, should initialize DACConnector object with a valid reference to an LFC DS connector if all credentials, with possible exception of passphrase and proxy certificate, are stored for a user. If they are not stored, an exception should be thrown and DACConnector should not be initialized.	Enables script to use credentials previously provided by user when passphrase to private key has not been provided.

Table 6: Software interface requirements (continued)

REQ ID	Requirement Description	Rationale & Comments
SI-4	`DACConnector.new(handle-name, proxy)`, where proxy contains contents of a valid user's proxy certificate represented by JRuby string, should initialize DACConnector object with a valid reference to an LFC DS connector object regardless whether credentials are stored for a user or not.	Allows script to initialize data source connector using a proxy certificate, so that there is no need to have user's Grid credentials previously provided to LFC DS in order to use its functionality.
SI-5	`DACConnector.new(handle-name, userkey, usercert, key-passphrase)`, where userkey contains private key used for Grid proxy certificate generation, usercert is user's Grid certificate and key-passphrase is passphrase to decrypt userkey, should initialize DACConnector object with a valid reference to an LFC DS connector object regardless whether credentials are stored for a user or not. userkey, usercert and key-passphrase are represented by JRuby strings.	Enables script to utilize LFC DS functionality in case of credentials not being previously provided by user, but when access to these credentials is possible from GridSpace Script.
SI-6	`DACConnector#createDirectory(path)` or `DACConnector#create_directory(path)`, where path is a string being constructed by concatenating existing directory name followed by a slash "/" and by a directory to be created, should attempt to create directory specified by path in data source's LFC catalogue in "`/grid/vo_name`" directory, where `vo_name` is data source's Virtual Organization, returning true on success and false otherwise.	Enables creation of LFC directories.

Table 6: Software interface requirements (continued)

REQ ID	Requirement Description	Rationale & Comments
SI-7	`DACConnector#createDirectory(path,child_directory)` or `DACConnector#create_directory(path,child_directory)`, where path is an existing directory and `child_directory` is a directory to be created in path folder, should attempt to create directory specified by `path+"/"+child_directory` in data source's LFC catalogue under "`/grid/vo_name`" folder, where `vo_name` is data source's Virtual Organization, returning true on success and false otherwise.	Enables creation of LFC directories.
SI-8	`DACConnector#delete(path)` or `DACConnector#deleteFile(path)` attempts to delete file or directory specified by path returning true on success and false otherwise.	Allows for deletion of files and directories.
SI-9	`DACConnector#isDirectory(path)`, `DACConnector#is_directory(path)` and `DACConnector#directory?(path)` check whether directory denoted by path exists in data source's LFC catalogue and return true or false respectively.	Enables checking for directory existence without listing parent folder and testing whether the directory in question contained in returned listing.
SI-10	`DACConnector#exist?(path)`, `DACConnector#exist(path)`, `DACConnector#exists(path)` and `DACConnector#exists?(path)` check whether an item denoted by path exists in data source's LFC catalogue and return true or false respectively.	Enables checking for directory existence without listing parent folder.
SI-11	`DACConnector#file?(path)`, `DACConnector#is_file(path)` and `DACConnector#isFile(path)` check whether an item denoted by path exists in data source's LFC catalogue and represents a file returning true or false respectively.	Enables checking for directory existence without listing parent folder and testing whether the file in question is contained in returned listing.

Table 6: Software interface requirements (continued)

REQ ID	Requirement Description	Rationale & Comments
SI-12	`DACConnector#get_file(path)` and `DACConnector#getFile(path)` obtains a file denoted by path from Grid storage returning it as a Java byte array. Methods throw an exception in case of file unavailability.	1. Allows for loading a file directly into a variable in GScript code. 2. There is no need to remember to close the file when writing GScript code.
SI-13	`DACConnector#list_files(path)` and `DACConnector#listFiles(path)` return a list of directory items of directory denoted by path allowing for: • getting item name by `ClassOfDirectoryItem#get_name` and `ClassOfDirectoryItem#getName` methods • checking whether an item is a directory or a file by using `ClassOfDirectoryItem#is_directory` and `ClassOfDirectoryItem#isDirectory` methods where `ClassOfDirectoryItem` is some internal class representing directory items which is not required to be exposed to the user.	Allows for directory listings.

Table 6: Software interface requirements (continued)

REQ ID	Requirement Description	Rationale & Comments
SI-14	`DACConnector#open(path,mode)`, `DACConnector#open_file(path,mode)` and `DACConnector#openFile(path,mode)` each with an optional Ruby block with one block argument attempt to open a particular Grid file for reading or writing depending on mode, which may be one of the following values • `:r, :read, "r", "read"` indicate opening for reading • `:w, :write, "w", "write"` denote opening for writing. If an optional block is supplied, a Ruby IO object is passed to code contained in a block. After code is executed, Grid file is closed. On the other hand, if block has not been supplied, method should return Ruby IO object to the caller leaving responsibility of closing the file to invoking script. If file is opened for reading, data is streamed to machine invoking GridSpace script as a result of invocation of Ruby IO stream reading methods. Conversely, invocations Ruby IO stream writing methods on a file opened for writing will cause data to be streamed out of GSEngine machine.	1. Allows for Ruby-like file access. 2. Enables access to large files.

Table 6: Software interface requirements (continued)

REQ ID	Requirement Description	Rationale & Comments
SI-15	`DACConnector#store_file(payload, path)` and `DACConnector#storeFile(payload, path)` attempt to store contents of Java byte array payload into a Grid file denoted by path returning true on success and false otherwise.	1. Allows for storing contents of a variable in a Grid file. 2. There is no need to remember to close the file when writing GScript code.
SI-16	`DACConnector#zero?(path)` checks whether a file denoted by path exists and has length of zero bytes returning true or false respectively.	
SI-17	`DACConnector#size?(path)`, `DACConnector#size(path)` and `DACConnector#getSize(path)` retrieve size of a Grid file specified by path.	Allows for retrieving file sizes.
SI-18	Enabling consistence between LFC DS and DAC2 data access infrastructure, i.e. making sure that DAC2 data source do not use different method names for similar operations that LFC DS provides.	

Table 6: Software interface requirements (continued)

REQ ID	Requirement Description	Rationale & Comments
SI-19	User should not be exposed to functions providing Grid credential management, with the exception of providing them within specific constructors (see SI-2, SI-3 and SI-4). In particular, proxy certificates should be generated and managed without user intervention. Therefore, GScript software interfaces providing credential management should not be created.	1. Concealing particulars of Grid credential management. 2. Reducing number of steps required to access Grid data.

7.4 Performance requirements

Communication overhead If network communication needs to be employed, protocols which introduce significant performance overhead, such as SOAP, should be avoided for transmission of file contents. On the other hand, they are acceptable for sending commands to be invoked, since such operations are less costly in terms of data transmitted.

Command execution overhead Because execution of data management commands, especially access to storage elements, is time consuming and no significant requirements regarding command execution overhead must to be met. Furthermore, even a small execution overhead will not make an entire command execution much faster.

Number of simultaneous data access requests to be supported Quantity of simultaneous data access requests should only be limited by hardware capabilities. However, for the purpose of system validation, this figure should be at least 5.

Ability to access large files Support for filesize of at least 1Gb must be provided by LFC DS server with no constraints by available memory.

7.5 Software system attributes

Security An important issue regarding certificate management is maximum security of user files, especially private keys and certificates if they are stored temporarily or sent over network. In particular, if a need for storing temporary files arises, access rights should be set appropriately, and sensitive files deleted as soon as they are not required. Furthermore, if sensitive data, such user credentials, needs to be sent over network, strong encryption should be used.

Maintainability Since the system is meant to be used and extended in the context of emerging Grid projects, such as PL-Grid, complete documentation of its design and functions is expected. However, this requirement is mostly met by the dissertation itself. Moreover, it would be beneficial, if certain components of the system were reusable. Such functionality would be helpful for other projects, if only some part of LFC DS functions would be of their interest, e.g. only access to EGEE/WLCG storage without automatic credentials management.

In addition, a proven solution for managing project dependencies would be helpful to ease incorporating LFC DS into other projects or to adapt LFC DS when the environment in which it operates changes. Furthermore, logging of operations should also be possible to track any problems that may occur.

Portability Most importantly, LFC DS must not compromise portability of GridSpace Engine which is platform-independent. If portability across platforms is not possible, LFC DS

should be split into parts that are portable and parts that are platform-reliant, so that those that are not portable would be external to GridSpace Engine.

Testability Tests should be provided in order to check validity of software installation.

Summary of non-functional requirements Sections 7.4 and 7.5 identified non-functional requirements of the project. They are abridged in table 7.

Table 7: Synopsis of LFC DS non-functional requirements

REQ ID	Requirement Description	Verification Method
NF-1	Efficient communication protocol used for data streaming, if a need to employ network communication arises.	Verification test & code inspection
NF-2	At least 5 simultaneous data access requests supported	Verification test
NF-3	Ability to access large files (at least 1Gb) must be supported by LFC DS server.	Verification test
NF-4	Strong communication encryption if sensitive data is sent over network	Code inspection
NF-5	If temporary files are used, the system should • set appropriate file permissions if they contain sensitive data • delete them as soon as they are not required	Code inspection
NF-6	Complete documentation	Documentation inspection
NF-6.1	Software requirements specification	
NF-6.2	Design description	
NF-6.3	Documentation of user interfaces	
NF-6.4	Documentation of software interfaces	
NF-6.5	Documentation of reusable components that are artifacts of the project	
NF-6.6	Installation guide	
NF-7	Logging of operations	Code inspection
NF-8	Support for managing dependencies	Code inspection
NF-9	Not to compromise portability of GridSpace Engine	Code inspection

Table 7: Synopsis of LFC DS non-functional requirements (continued)

REQ ID	Requirement Description	Verification Method
NF-10	Provide some modularity and reusability enabling incorporation of only some parts of the functionality into other projects.	Code inspection
NF-11	Tests enabling verifying validity of LFC DS installation	Code inspection

8 Design description

Firstly, this chapter examines various design decisions that could satisfy requirements stated in chapters 6 and 7 together with considerations on their applicability and value. After identifying advantages and disadvantages of each solution, decisions are made on which to use. Subsequently, section 8.4 shows how the software will be structured and how it will operate.

8.1 Design decisions

With regard to providing access to EGEE storage resources an uncomplicated service-oriented access to these services could be the solution. However, it is not provided with default gLite installation. A service-oriented approach has been successfully applied to a number of legacy applications, e.g. COBOL, CL, ILE or RPG programs on IBM i (formerly known as IBM System i or iSeries) saving many man-hours invested in these applications. If such an approach had not been employed, these applications would have to be rewritten from scratch imposing immense development cost. Therefore, it is highly probable that such a solution will also succeed in the scope of our projects. However, SOAP protocol involves too much overhead communication and perhaps its usage would be arguable if transferred data were highly structured and small in volume. Nevertheless, when dealing mostly with files, which may be quite large, a more compact protocol is advised as already indicated in requirement NF-1.

With respect to integration with GridSpace Engine and the placement of LFC DS in data source hierarchy, after a consultation with Cyfronet members, it was decided that the new GScript data source connector would fall under *Unstructured data sources* category in DAC2 data access layer (see figure 18). With regard to configuration previously mentioned in section 4.2; all data source configuration will be stored in Data Source Registry.

An interesting issue, is the means of access to gLite storage services – several alternatives were considered that finally led to decision of employing service-oriented paradigm – this paragraph presents the reasoning. Firstly, data access could be performed directly from a library which is a dependency of DAC2 connector. Although simple in realization, it would impose unacceptable requirements onto GridSpace Engine and would limit its installation environment to Linux with gLite installed, which would definitely be too high a compromise (see NF-9). Another option would be to require the user to possess a valid account on a gLite UI and to provide some means of remote command execution, e.g. a user would upload a private key generated to allow access to their account. GridSpace Engine would then execute gLite commands, logging into user's account using the uploaded credentials. Although SSH provides means for limiting available commands, when logging using a particular key, so that in the case of a private key being stolen, actions that are possible would be restricted and many users would show reluctance. Furthermore, users still would have to perform some intricate procedures of obtaining access to gLite UI, generating and adding private keys (see FR-3 and FR-5)

not only to access the Grid, but also to utilize envisaged Virtual Laboratory gLite data access service. Although it is an advancement compared with formerly mentioned work-around, it is not the ultimate solution for satisfying prospective users. An architectural choice that could satisfy the needs, without compromising GridSpace portability and usability, would be to create a dedicated server installed on a gLite UI that would act as a data server. However, it involves many implementation complications, an example of which would be the sending of large files which would induce the need to incorporate some kind of streaming which would avoid `OutOfMemory` errors (NF-3). Additionally, there should be a utility to concurrently generate certificates without access to actual users' accounts, i.e. only one account on a gLite UI would be utilized. However, many distinct certificates belonging to various users would have to be generated and managed, which is impossible from a single process, since certificate locations are controlled using environment variables – therefore, a multi-process application is the only option to implement this idea. Furthermore, tracing errors will be difficult as with any remote application accessing native functionality which is composed of multiple processes. Despite implementation difficulties, it appears to be the only plausible choice of solving all concerns.

With regard to credentials management, initially MyProxy [134, 164] server was considered as a solution. Nevertheless, it was discarded by other team members and it was decided that certificate information will be stored in Data Source Registry (see 4.3 and FR-3). As already denoted in 4.3 and by FR-5 one of the issues was automation of generation of proxy certificates, so that users would not have to create them manually. Normally, they are produced using `voms-proxy-init` command with Virtual Organization name provided with `-voms` options. Locations of user private key, grid certificate and path where generated proxy certificate will be stored, are specified using environment variables. It is impossible to produce proxy certificates concurrently from the same process and it is also impossible to invoke commands that access gLite data sources within a single process. Therefore, as already mentioned, a multi-process application had to be created which executed Grid operations in separate processes. Moreover, a supervising process will have to take care of file locations where certificates will be stored and proper environment variables set for each new process spawned and each variable pointing to valid user credentials, i.e. existing certificate and private key files.

Additionally, proper credentials must be provided when a Grid operation invocation occurs – a grid certificate, private key and private key passphrase must be conveyed if operation is the generation of a proxy certificate. On the contrary, as Grid operation is e.g. accessing LFC catalogue or accessing a file, proxy certificate must be sent; when proxy is not present or is expired, it should be generated and saved in DSR (FR-5).

In order to satisfy requirement NF-4, Transport Layer Security (TLS) or tunnelling may be used. Nevertheless, an inherent feature of TLS is the necessity to manage server certificates. Therefore, tunnelling was chosen as a solution providing communication encryption. Mechanism implemented to fulfill NF-5 requirement is planned to restrict UNIX file permissions

immediately after creation of a temporary file and deletion of the file when a server method that is using it finishes.

Since integration with existing infrastructure is a key requirement (UI-13), instead of writing a standalone application, one of the Virtual Laboratory user interfaces ought to be extended. A portlet in Virtual Laboratory portal could satisfy this need but on the other hand, portal is mostly indented for the final end-users, so presenting them with somewhat obscure options would confuse rather than help them. Another option is to extend Experiment Planning Environment, which is intended for experiment developers who are acquainted, to a certain extent with computer science. However, since they are often computational and not computer scientists, any possible means of providing simiplicity and complicity should be pursued. Moreover, integrating graphical interface of registry of LFC data sources into EPE will bring this interface close to the environment where experimental code employing LFC data sources is produced, making this user interface helpful when searching for existing LFC data sources, updating, deleting or adding new ones during a development cycle. Additionally, existing DSR-plugin that already manages existing data sources of various kinds, mostly relational databases and sources accessible using WebDAV interface, is a most promising target of integration. Nevertheless, wizards for adding and updating the aforementioned data sources are quite monolithic and therefore not easy to extend. Furthermore, they manage information, such as host name, port, schema name, username and password, but these values are totally different for LFC data sources. Therefore, alternatives for consideration are, rewriting these wizards from scratch taking into account new data source type or creating a separate wizard with distinct invocation mechanism. The former is time consuming while the latter will compromise consistency. Therefore, a hybrid approach is needed: to maintain invariable invocation mechanism while amending monolithic design into a modular one, so that rewrite of already created wizards will not be necessary. Apart from changes in user interface, many changes must be done to database layer of DSR-plugin in order to sustain current capabilities of managing existing types of data sources, since DSR schema need to be changed, as already indicated in earlier sections. Additionally, new methods in database access layer must be added to supply functionalities necessary to new LFC data source wizard.

With regard to documentation requirements, NF-6.1, NF-6.2, NF-6.4 and NF-6.5 will be satisfied by respective chapters of dissertation, while NF-6.3 and NF-6.6 will be fulfilled both by dissertation and by appropriate wiki pages on Virtual Laboratory website.

To summarize, figure 38 delineates conceptual view onto proposed design.

8.2 Organization of *Design description*

Subsequent chapter is organized by means of views[57] and viewpoints as recommended by [114]. At first, design stakeholders and their concerns are identified. Each viewpoint specifies design

[57]The terms *design concern, design element, design stakeholder, view* and *viewpoint* were defined by [114].

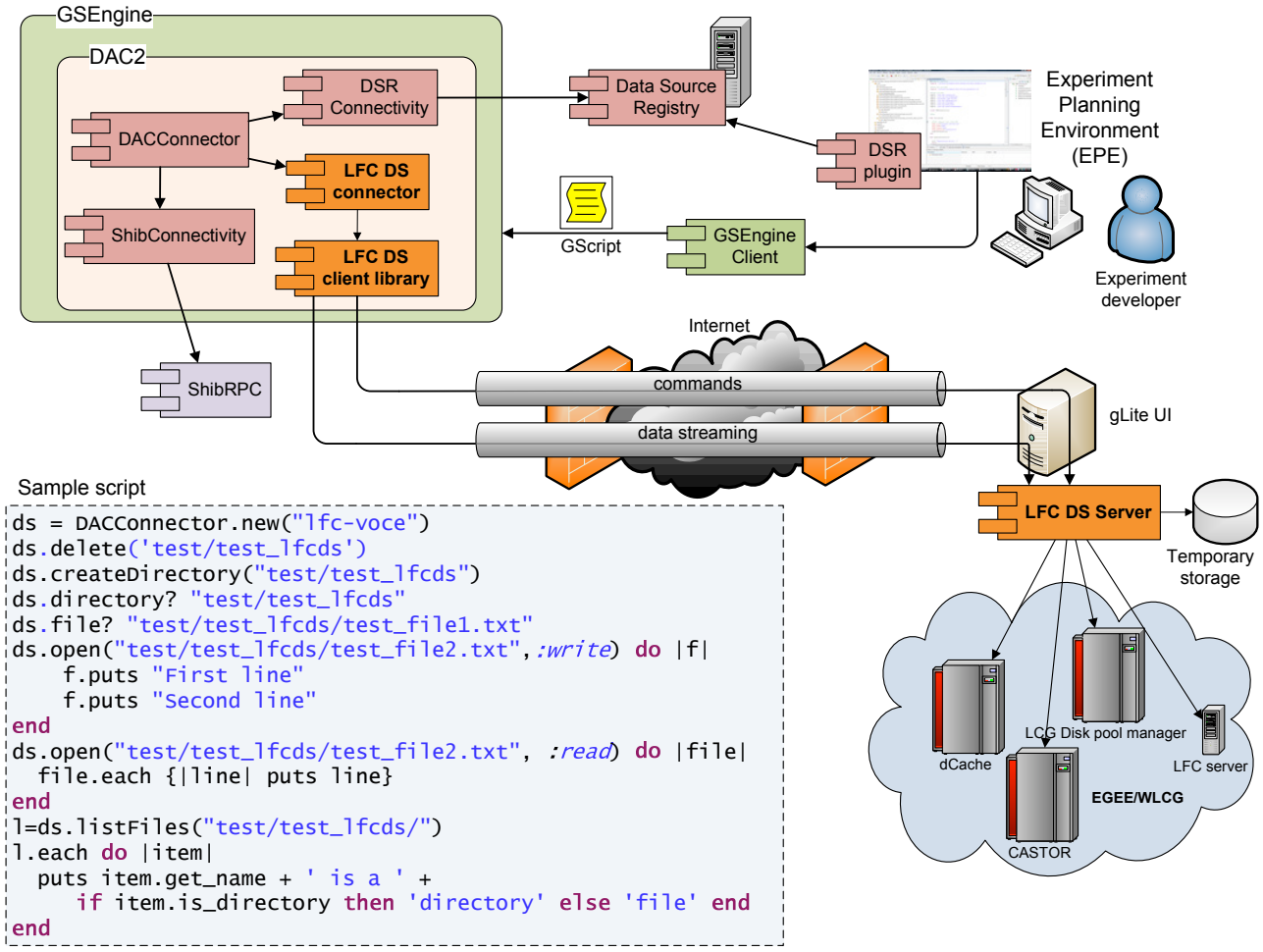


Figure 38: Conceptual view onto proposed design of LFC DS

concerns being its topic, design elements that are defined or used by the viewpoint and a set of conventions stating how design will be conveyed, including design language. Consequently, design views address design concerns from a specified design viewpoint. Maier et al. [146] allude to analogy of a view from civil engineering domain – buildings have several views: front, top, side, electrical, plumbing, floor plans etc.

8.3 Identified stakeholders and design concerns

Design concerns are identified by DC-*number* strings. Tables 8 and 9 indicate design stakeholders, design concerns and addressing viewpoints.

8.4 Design views

Design views are governed by apposite design viewpoints, each defined in relevant sections of [114] as shown in table 10.

Identification	Design concern	Addressing view
DC-1	Composition and modular assembly of systems in terms of sybsystems and components	Composition
DC-2	Static structure, reuse of types and implementations	Logical
DC-3	Interconnection, sharing, and parameterization	Dependency
DC-4	Persistent information	Information
DC-5	Service definition, service access	Interface
DC-6	Object communication and messaging	Interaction

Table 8: Design concerns and views addressing them

Stakeholder	Design concern
Developer who wants to modify or extend LFC DS	all provided
Developer wanting to incorporate some of the reusable components into their work	DC-1, DC-3, DC-4 and DC-5
Developer adapting LFC DS to different environment	DC-3 and DC-4

Table 9: Identified stakeholders and their design concerns

Viewpoint	Specification
Composition	[114, section 5.3]
Logical	[114, section 5.4]
Dependency	[114, section 5.5]
Information	[114, section 5.6]
Interface	[114, section 5.8]
Interaction	[114, section 5.10]

Table 10: Design viewpoints specifications

8.4.1 Composition

LFC DS solution (figure 39) is composed of DACConnector (figure 43), which is a GSEngine component that enables data access and management from GScript code, EPE DSR Plugin (figures 42, 55, 56, 57 and 58) – an EPE plugin that allows for browsing and registration of data sources and user Grid credentials, Data Source Registry (DSR) (figure 54) which stores information on data sources and credentials, and LFCDS Server (figure 40), which is a gateway enabling access to EGEE/WLCG Grid.

Moreover, DACConnector includes a reference to LFCDS library (figure 41) which connects to LFCDS server, a reference to LFCDS connector represented by JRuby class `LFCDataSource` (figure 49), which is a class dedicated to managing access to LFC data sources and usage of Grid credentials, and to DSRConnectivity instance, that encapsulates methods for communicating with DSR. Both DACConnector and DSRConnectivity were extended with methods specific to LFC DS.

In a similar instance, EPE DSR Plugin exploits its own DSRConnectivity module also dedicated to communications with DSR; EPE DSRConnectivity was expanded with a richer set of methods than DAC2 DSRConnectivity. In particular, DAC2 DSRConnectivity is mostly responsible for reading data – with one exception being updating proxy certificates. On the other hand, EPE DSRConnectivity must handle not only reading data, but also data updates and registering new data sources and credentials. A distinct part of EPE DSR Plugin is LFCDS Form – a dedicated form for registering LFC data sources and uploading Grid credentials.

As far as DSR is concerned, its additional constituent is LFCDS Schema, i.e. a schema that is dedicated to storing information regarding Grid data sources and credentials.

Figure 39 depicts the aforementioned components with relationships of inclusion and usage.

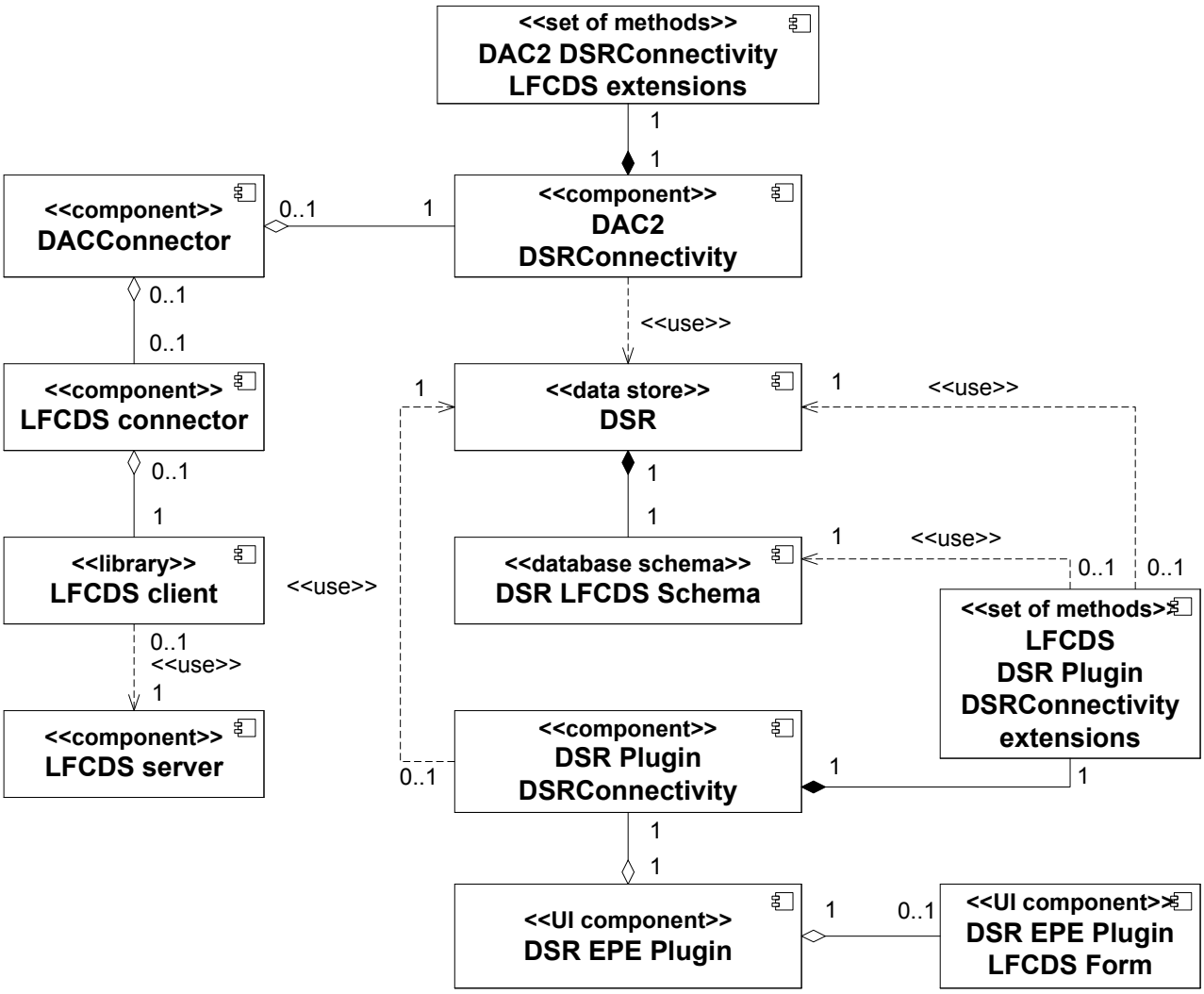


Figure 39: Composition of LFC DS system. DACConnector, DAC2 DSRConnectivity, DSR EPE Plugin, DSR Plugin DSRConnectivity and DSR are components that existed before creation of LFC DS

8.4.2 Logical

This clause presents the division of components into classes and then depicts their internal subdivision into methods. Firstly, class diagram on figure 40 shows the structure of LFCDS server. As illustration suggests, `LfcDsServer` class (also see figure 45) plays the main role in the operation of LFCDS server. `LfcDsServer` is a consumer of services provided by other entities. In particular, it uses `LfcDsProperties` for reading values of properties stored in standard Java properties files. In the case of LFCDS server the properties file name is `server.properties`. However, in current implementation server administrator, who is the only person who will modify it, provides its values by substituting appropriate values in `lfcds.properties` file read by Maven [166], which propagates these changes to two configuration files: `server.properties` and `test.properties` saving administrator the encumbrance of keeping both files up to date, since many properties they use are common.

On the other hand, *DacLfcCommands* (see figure 44) class is a class extending `LfcCommands` type from ChemPo project. `LfcCommands`, not presented on the drawing, supplies wrappers with specific Grid operations, such as downloading a file or sending a file to storage element and registering same in LFC catalogue. `LfcCommands` class achieves it by using `LfcExecutor` class from ChemPo project, that executes each command in a separate Java Virtual Machine (JVM) with specific UNIX environment (see figures 61 and 62). In addition to extending `LfcCommands`, `DacLfcCommands` class provides two methods that were not provided by `LfcCommands`, in particular `delete`, `getSize` and `exists`.

`LfcDsServer` implements *ILfcCommands* interface (see figure 44) which specifies a set of operations that LFCDS server provides. Apart from data access and management commands, `ILfcCommands` is used for generation of proxy certificates and retrieving certificate attributes. These attributes are stored in *UserProxyDetails* object and sent to calling client. *StoreFileBean*, *PathInputBean* and *LongOutputBean* are Java beans that transfer specific data when sent by `LfcExecutor` to `DacLfcCommands`, whereas *LfcCommonParametersBean* is used to encapsulate common data that may be useful in most data access and management Grid commands; namely the user proxy certificate, which is used by Grid software for authentication and authorization purposes, LFC host, indicating LCG File Catalogue server to be contacted, Site BDII (see section 3.6), which is a server that informs Grid File Access Library (GFAL)[58] about particulars of storage elements (see figure 29), Virtual Organization Name and path which is a common argument of data management commands. Furthermore, *LfcDsItem* class envelops information about items retrieved from LFC directory, their path and whether they are files or directories – other information, such as file permissions is omitted as it is required by FR-1.8.

An important component in LFCDS server structure is *LfcDsOutputStream*, which was developed as a part of larger data streaming scheme. Its role is to remotely invoke a LFCDS server method that sends file to Grid and deletes temporary file on server when streaming

[58] GFAL works underneath software exploited by LFCDS server

file to LFCDS server finishes. If it is not possible, it throws an *LfcDsException* from client library, which in turn passes it to LFCDS connector notifying user about the problem. If `LfcDsOutputStream` had not been used and default callback of RMIIO library had been utilized, a message about the problem would have not been conveyed – when callback executes, there is no way of returning information. `LfcDsOutputStream` is utilized on client side; however, it is present in the server class hierarchy, because it is one of the contents of `StoreFileBean` sent to client library when it invokes *storeFileInit* method of the server (see figure 62 that addresses client↔server interaction during writing file to Grid).

`LfcDsException` is not only raised when sending a file to Grid fails, but it is instantiated whenever a problem with input data occurs that was not detected by LFCDS client or a server side method which encounters difficulties performing requested action. The aforementioned exception is also raised by *LfcDsClient*, a principal class of LFC client library, if it meets some impediments connecting LFCDS server or it detects a mistake in user's request. As can be seen in figure 41, `ILfcCommands` interface is used by both `LfcDsOutputStream` (the connection with `ILfcCommands` has not been shown on figure 40) and `LfcDsClient`. For these two classes `ILfcCommands` defines methods which can be remotely invoked on `LfcDsServer`. LFCDS client library does not define any additional classes. However, there is an enormous difference between LFCDS client and LFCDS server, when it relates to dependencies required (compare figures 50 and 52 illustrating dependencies of these two components in terms of Maven artifacts). Because of LFCDS lightweight library, it can be incorporated into software that could benefit from a communication with LFCDS server – LFCDS solution is not limited to GSEngine and its DAC2 data access layer. In fact, it can be used by any Java application.

Figure 42 depicts EPE DSR-Plugin classes that play a role in the operation of LFC DS solution. *LfcDsEditForm* is a graphical user interface form created using Visual Editor [174] user interface builder with several other functionalities added manually. These include validation of inputs, dynamically disabling and enabling buttons, changing group and button captions depending on the context in which the form was invoked, i.e. whether it was a request for edition of existing LFC data source or addition of a new data source of this type and naturally, application logic. Initially, the form was invoked with separate buttons and menu commands. However, Piotr Nowakowski, main developer of EPE DSR-Plugin, replaced previous design which composed of one wizard into "two-form" approach, i.e. when user request registering a new file (figure 57) or edition of existing one, the *SelectSourceTypeDialog* form enabling a user to choose a data source type (structured or unstructured) and technology appears (figure 57). This form, in turn, invokes apposite form responsible for managing registration of concrete data sources and credentials. Thus, he enabled inclusion of other data source wizards in an integrated way and only one button and menu item suffices to invoke registration dialog of any type of data source that emerges. *PasswordDialog* pops up when user clicks `[Set]` button near the password label (it would be visible on screenshots if Grid credentials were not loaded). A reference to

`DSRConnectivity` is passed by *SelectSourceTypeDialog* to `LfcDsEditForm`, which then uses it for searching, updating, deleting and adding new entries of LFC data sources, Grid credentials and server connections. Methods added to EPE DSR Plugin `DSRConnectivity` in order to enable these operations are delineated in figure 48. On the other hand, `ShibConnectivity` plays a role in LFC DS operation by providing a *user handle* to `LfcDsEditForm`, which enables identification of the user in context of DSR.

DAC2 (see figure 43), a data access layer of Virtual Laboratory is logically decomposed into several classes, each yielding autonomous functionalities – in particular, similar to EPE DSR Plugin, it delegates connectivity responsibilities to *DSRConnectivity* (figure 49) and *ShibConnectivity* or *ChempoConnectivity* classes, this time written in JRuby language [86]. `ChempoConnectivity` was added as part of the dissertation to enable access to DAC2 data access layer within ChemPo custom built GSEngine. Methods provided by both `ChempoConnectivity` and `ShibConnectivity` are as follows: `getParams`, `getRawHandle` and `getUserHandle`. While `getParams` is the necessary initialization of `ShibConnectivity`, in `ChempoConnectivity` that method does nothing – the common interface was left untouched in order to decrease code changes necessary, so that `getParams` method is invoked regardless of security provider. A careful reader may discern similarity between `DACConnector` methods shown in figure 49 and those specified in table 6. Indeed, this is the same set of methods – `DACConnector` is the central class of DAC2, which provides GScript developers interface for data access. As part of the thesis project, `DACConnector` interface was significantly expanded. In particular, all methods and aliases specified for LFC DS component in table 6, with the exception of `initialize`, `getFile`, `storeFile` and `deleteFile` (which remained for compatibility) were added. Earlier, when the majority of data sources were relational, most operations were performed using `executeQuery` and `executeUpdate` methods, and therefore, such a rich API had not been mandatory. However, with the introduction of LFC connector operations, such as directory creation, deletion of directories and files (achieved using single `delete` method), together with methods to be used for data streaming, such a need arose and `DACConnector` API was extended. In addition, several alias methods with Ruby style notation, such as `exist?`, file? and `zero?` with a question mark at the end which indicates that a method returns logical value or methods with an underscore instead of usual Java *camelCaseNotation*. This makes the API more Ruby-like. However, a change that made LFC DS method invocations most Ruby-oriented was the introduction of block argument into `openFile` method. In consequence, `DACConnector#open` method gives the impression of being standard Ruby `open` invocation executed on a Ruby `File`. Moreover, since `DACConnector#open` method returns a subclass of Ruby IO converted from Java InputStream or OutputStream depends upon whether a file was opened for reading or writing. Therefore, a complete impression of standard Ruby IO is given; thus shortening the learning curve significantly for developers or computational scientists already knowing Ruby.

`SourceParameters`, another class of DAC2, is a bean containing data that is passed by

DACConnector to connector objects. SourceParameters methods relevant to LFC DS have been depicted in figure 49. On the other hand, for DAC2 DSRConnectivity all methods have been shown, since only getCertData, updateProxy and getStaticCertData have been appended. On the other hand, both EPE DSR Plugin and DAC2 DSRConnectivity class method set required not only augmentation, but also refactoring. This is the result of changing DSR structure. However, such change was mandatory in order to enable registration of LFC data source, which had totally dissimilar information needs, a fact that can be observed by analyzing the current DSR schema illustrated on figure 54. Finally, *LFCDataSource* supplies concrete implementation of LFC DS connector. A feature built into LFCDataSource component is a mechanism for checking whether a certificate is valid – it uses LFC DS server getProxyDetails method for this purpose. If a proxy Grid certificate is present in DSR and valid (it is assumed no longer valid, if it has less than an hour to expire), the LFCDataSource utilizes it for Grid operations. Otherwise, a new proxy is generated and saved in DSR using DACConnector#generateProxy method, as shown in figure 59.

With regard to internal logical organization of each of the aforementioned design entities, in terms of methods they provide and private variables they contain, the simplest organization is of bean files: StoreFileBean, LongOutputBean, PathInputBean, UserProxyDetails, LfcCommonParametersBean and SourceParameters – they contain a single private field with accompanying get and set methods for these variables. On the other hand, LfcDsOutputStream implements methods defined by itsOutputStream superclass. DacLfcCommands is a class, whose functionality is mostly provided by a class higher in inheritance hierarchy – LfcCommands. However, as it was mentioned, three specific methods are implemented in it – as with LfcCommands, LfcExecutor from ChemPo project has been utilized to achieve the functionality of executing commands in separate JVM. LfcDsException is a standard exception class extending Java Exception. In contrast to these Java classes, LfcDsServer has a more complex structure. It was designed in such a way, that it should not impose any threading problems. In particular, the only variable its methods share, is the log used for logging its functions (see NF-7). Several private auxiliary static methods: getTempProxyFile, parametersBeanToLfcConfiguration, createTempDirectory, cleanAfterOperations and restrictFilePermission serve other methods by providing them with common functionality. Because log is the only object-level variable, LfcDsServer may be safely shared by many clients without worrying about concurrency problems. The methods exposed to clients are those that were specified by ILfcCommands interface. Each of them has its own logic, but a common scheme of operation is creating DacLfcCommands instance and invoking one of its methods, catching exceptions, logging them and wrapping by LfcDsException. Often temporary files are stored during execution of these methods with createTempDirectory, restrictFilePermissions and cleanAfterOperations static methods being used. Most often, temporary files and directories are deleted when methods finish (regardless of exceptions that occur). However, with getFile, storeFileInit and StoreFileFinish,

i.e. methods that incorporate streaming; the responsibility to delete temporary file is transfered to callback – such a method was used for `getFile` method, or a Java special purpose `OutputStream`, namely `LfcDsOutputStream`, whose `close` method causes associated temporary file to be sent to Grid; subsequently directory and files that are no longer required are deleted. The `main` method of `LfcDsServer` configures Cajo library class `Remote` with endpoints specified in `server.properties` file. Later on, the `main` method binds a newly created instance of `LfcDsServer` so that its methods can be invoked by remote clients. The `parametersBeanToLfcConfiguration` is a static method executed by many other `LfcDsServer` methods – it translates an LFC DS bean with configuration: `LfcCommonParameters` into ChemPo `LfcCommands` valid `LfcConfiguration`. One of its roles is to create temporary files with proxy certificates, that were passed to server as byte arrays and store filenames in `LfcConfiguration`, which can be used by ChemPo LFC command wrappers.

As regards `LfcDsClient`, its main responsibility is to abstract server communication. A long constructor provides `LfcDsClient` with information on server endpoints and specific data access configuration, that will not have to be provided with each method invocation. During its initialization carried out by `LfcDsClient` constructor, it creates a `TransparentItemProxy` item from Cajo framework that enables communication with LFC DS server. Subsequent method invocations utilize both `LfcDsClient` data stored in its private field and parameters supplied by user. It is noteworthy that client library automatically translates the *path* provided by calling object (e.g. user's script) into valid LFC path, i.e. when user specifies *some_path* as path, then `LfcDsClient` adds `/grid/vo_name/` to this path. If user supplied path begins with a slash - '/', the character is removed.

The internal structure of `LfcDsEditForm` is quite simple. It contains many user-interface private building methods, such as `createCredentialGroup`, `createServersGroup` or `createComboConnList`, some utility methods, such as `isHandleUnique`, `isDataSourceNameUnique`, `validateConnData`, `connExists`, but the main logic is contained in button callback methods, which are created during building user interface and therefore, they are not visible on the diagram. Public methods of `LfcDsEditForm` class are intended for communication with calling code, e.g. the `showDialog` method causes `LfcDsEditForm` dialog to be created and displayed. The method returns 0 on success, and other values otherwise – it is a convention used by other wizards in DSR EPE Plugin.

Both EPE and DAC2 `DSRConnectivity` are classes that were extended to satisfy need of LFC DS and new DSR schema. `DSRConnectivity` in either case is a class encapsulating SQL code in several methods, each dedicated to one purpose. Private methods of this class are utilities used to make other methods' bodies shorter by reusing some functionality.

Hopefully, this clause gave the reader deep insight into logical decomposition of LFC DS into components → classes → and methods and how they are reused among the design entities. The next clause will present how LFC DS design entities depend on each other and on external

resources, mainly software libraries.

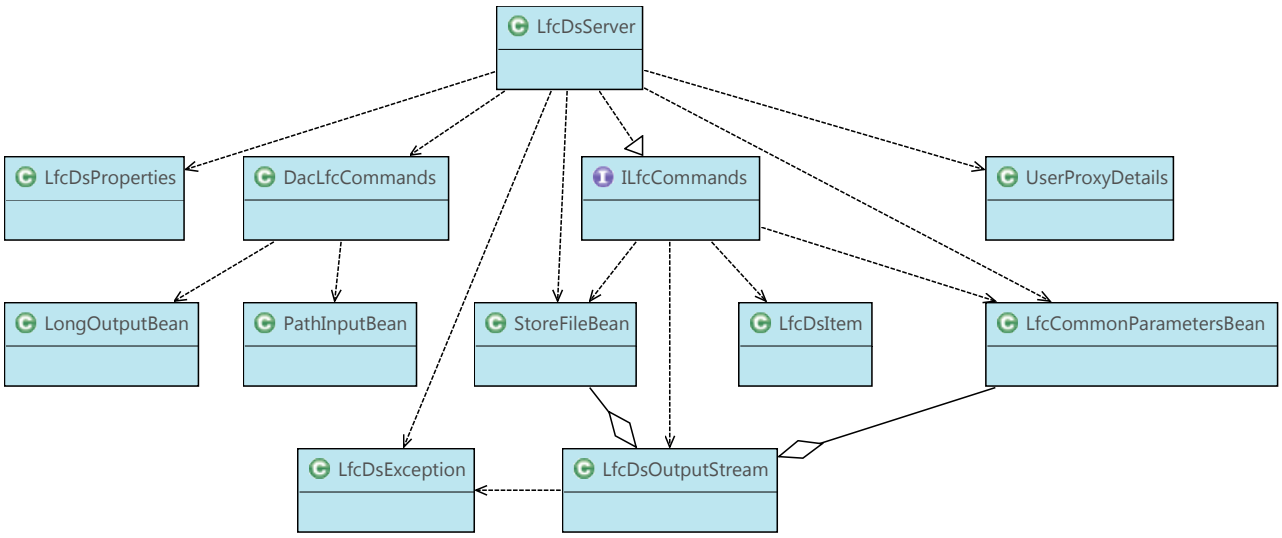


Figure 40: Logical view onto LFCDS server component

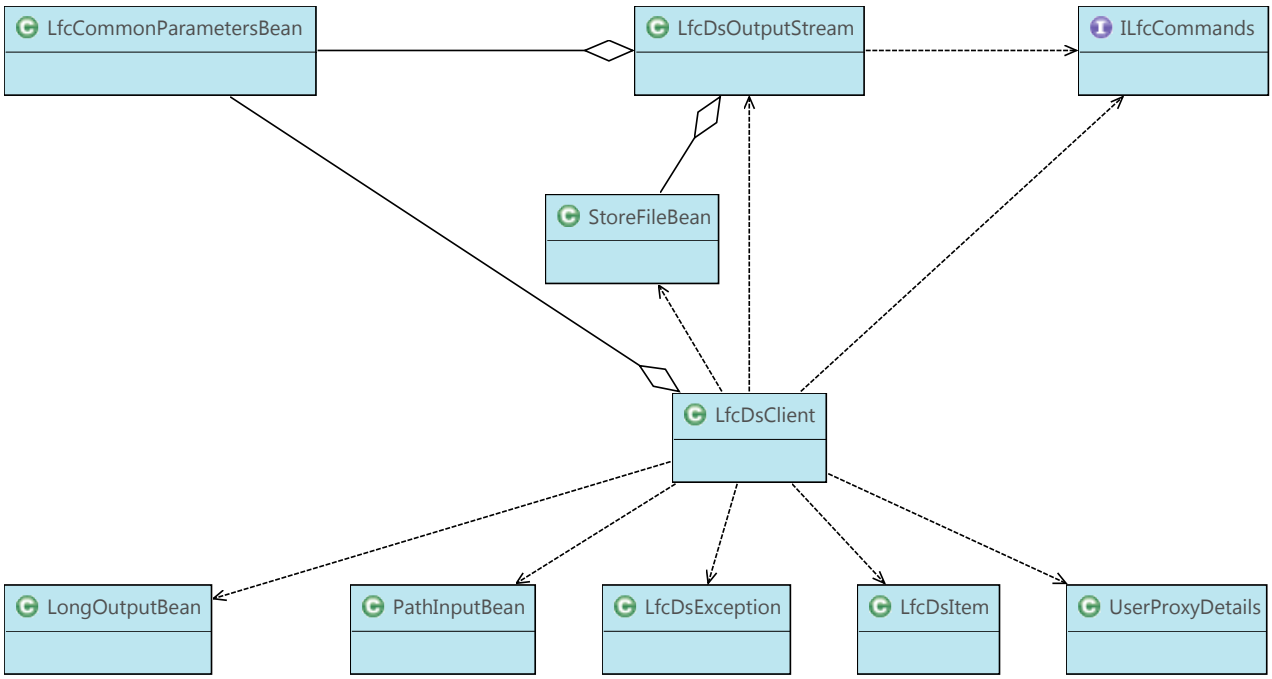


Figure 41: Logical view onto LFCDS client library

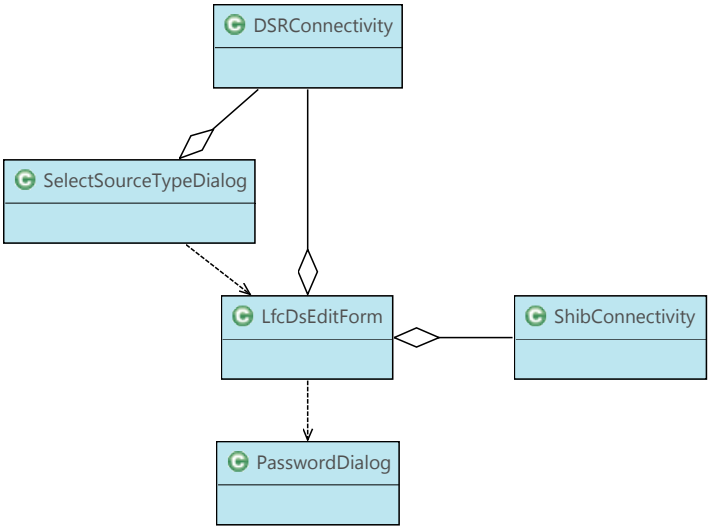


Figure 42: Class diagram DSR EPE Plugin LFCDS Form. Classes not directly connected to operation of LFC DS were excluded from diagram.

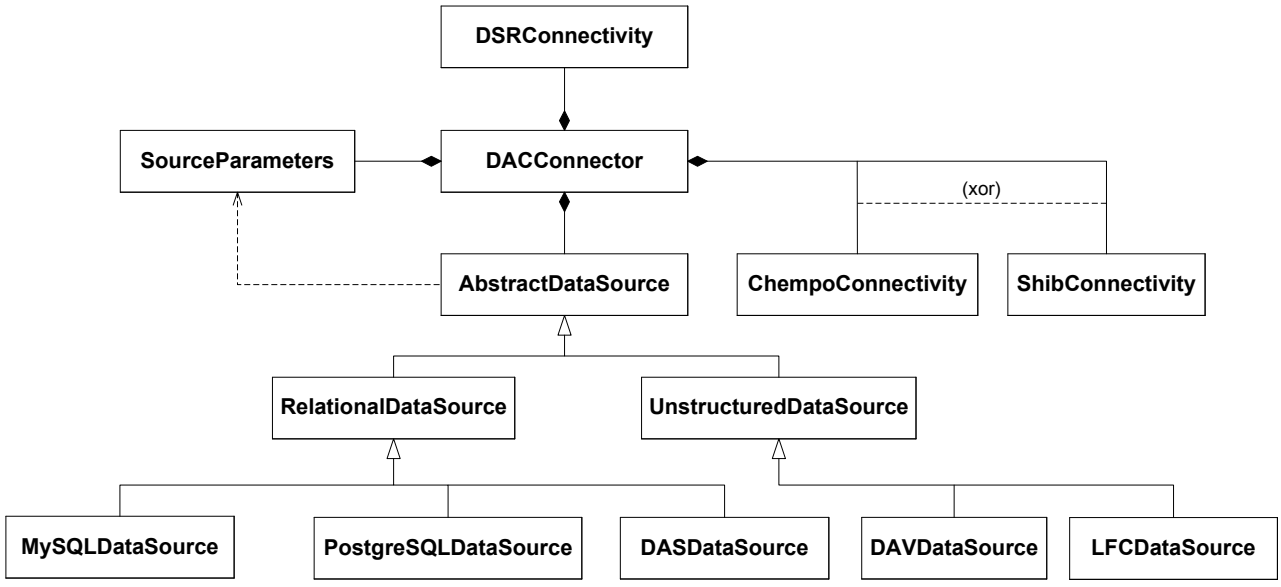


Figure 43: DAC2 class diagram after integration with LFC DS. Classes not directly related to LFC DS are omitted.

Figure 44: Class diagrams: LfcDsProperties, LongOutputBean, PathInputBean, LfcDsItem, StoreFileBean, LfcDsOutputStream, UserProxyDetails, DacLfcCommands and ILfcCommands.

Figure 45: Class diagrams: LfcCommonParametersBean, LfcDsException and LfcDsServer.

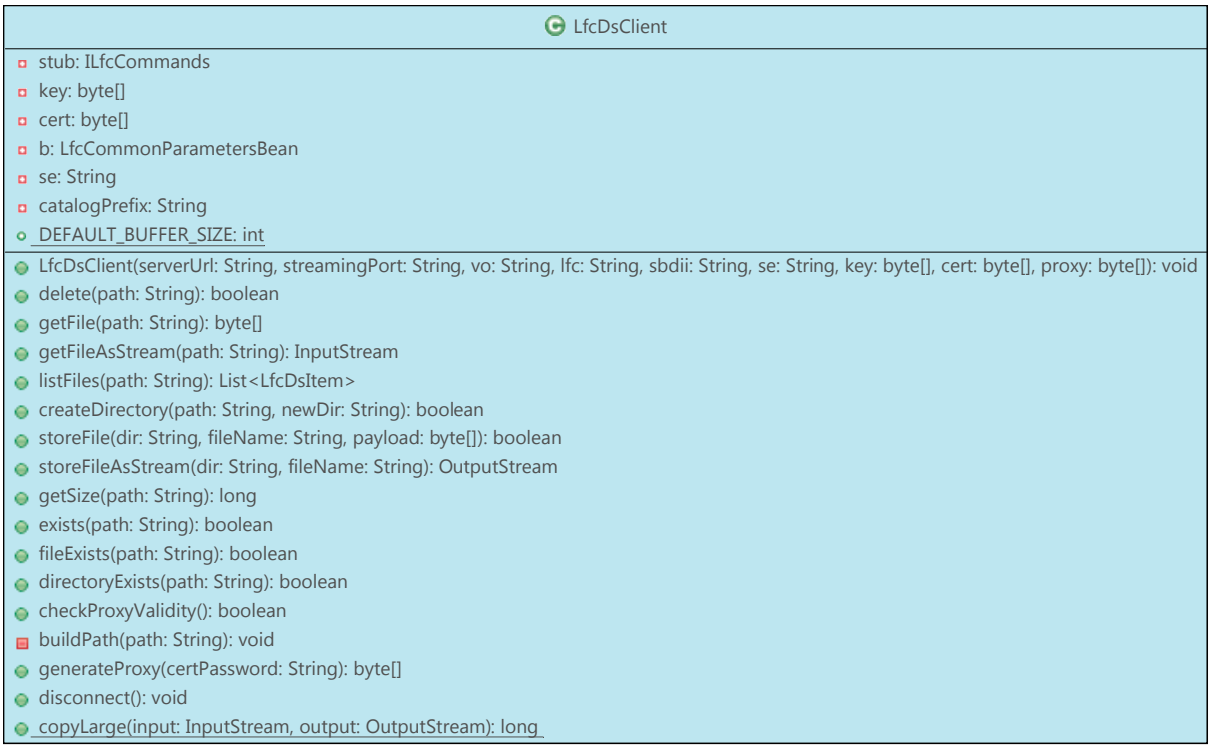


Figure 46: Class diagram: LfcDsClient

Figure 47: Class diagram: LfcDsEditForm and PasswordDialog. For LfcDsEditForm private attributes were omitted for brevity.

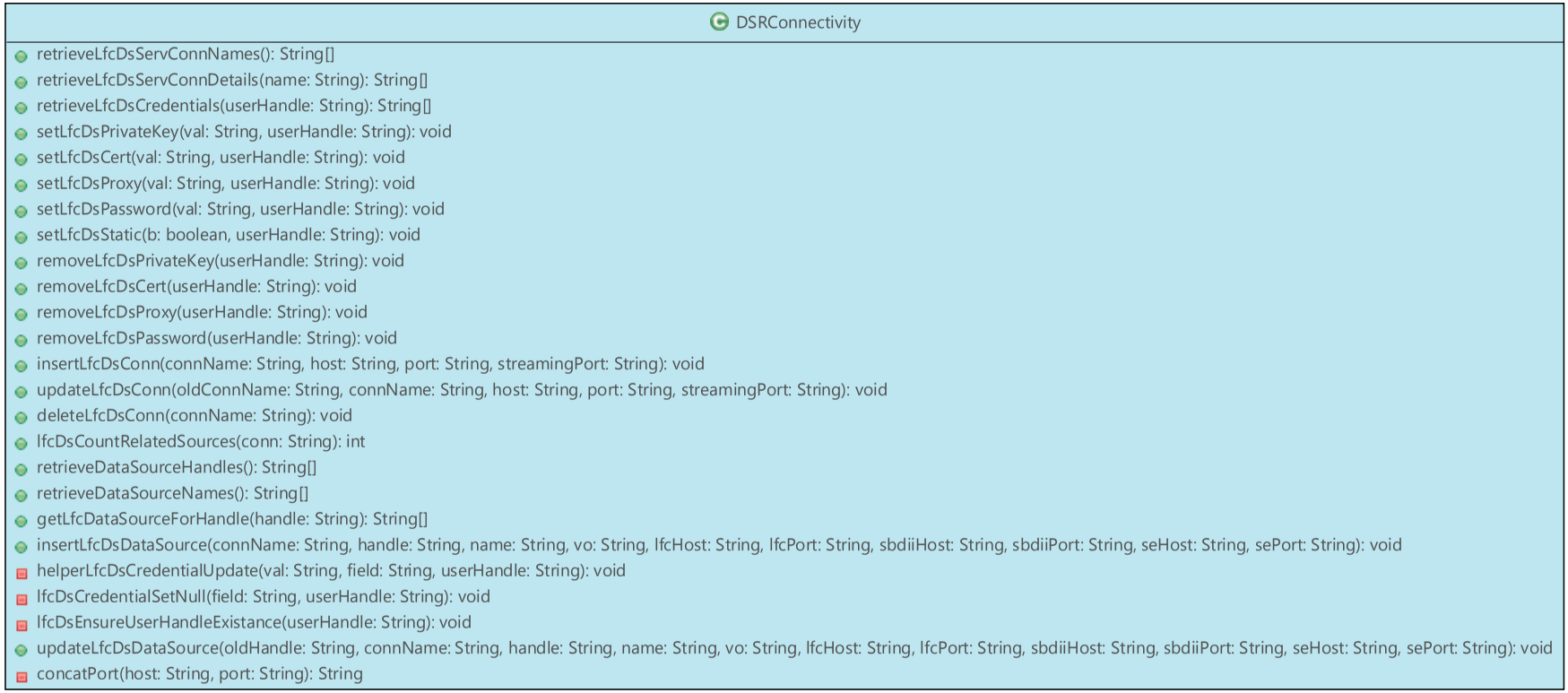


Figure 48: Class diagram: DSR Plugin DSRConnectivity – private attributes were omitted for brevity. In addition, only added methods are shown; modified methods or those that existed previously are excluded.

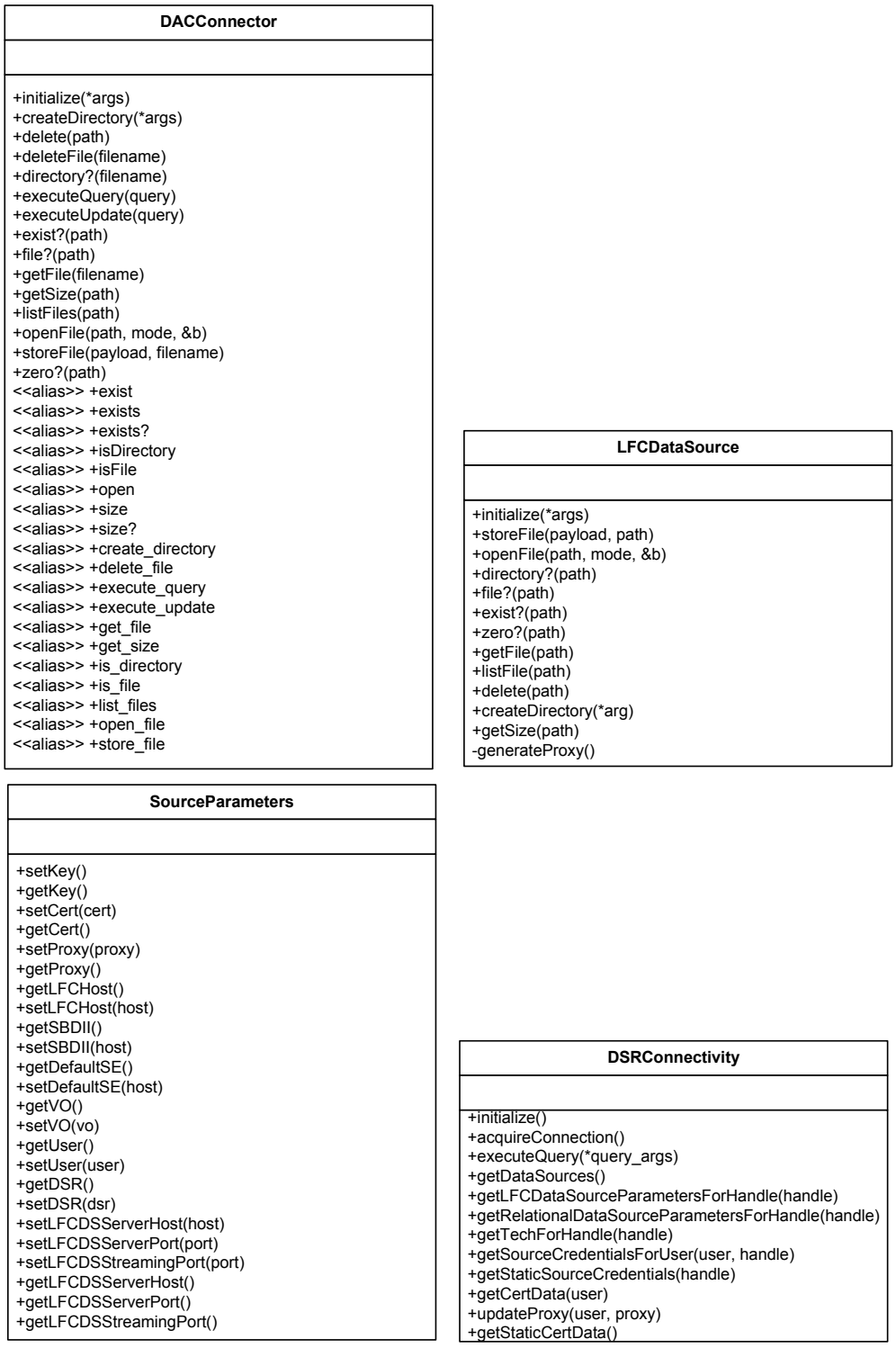


Figure 49: Class diagrams: DACConnector, DACConnector, SourceParameters, and DAC2 DSRConnectivity

8.4.3 Dependency

Figure 51 illustrates dependencies among design entities and services they provide to each other. A component diagram notation [165] has been chosen. Going from the left the reader may notice two communication libraries: Cajo and RMIIO both being RMI-based frameworks. During prototyping phase, Cajo library was chosen for overall communication, since it was discovered that using this library it is relatively simple to have communicating applications run behind firewalls. RMIIO is used for the same purpose, i.e. in order to facilitate communication through firewalls. Note, that on figures 61 and 62 there are no server→client callbacks – all communication is initiated by client; even when server sends data to client (figure 61), this functionality is accomplished by pulling data by client, not by sending it by server. Another rationale for using RMIIO is that it provides fault tolerant streaming, a valuable feature which RMIIO achieves by multiple retry requests in case of communication errors. Both libraries are Open Source, which makes them suitable for integration with LFC DS project.

An interesting fact shown in figure 51 is that `ShibConnectivity` instance connects to `ShibRPC` while `ChempoConnectivity` – an alternative implementation of security mechanism does not communicate with any ChemPo specific security mechanism. This is because `ChempoConnectivity` uses `GS_USER_ID` which holds a unique user identifier that is used to distinguish users.

The main interest of figure 51 are interfaces that each component requires and provides and how the fulfillment of these needs is attained by interconnecting components.

On the other hand, figures 50, 52 and 53 represent dependency graphs of components in terms of requisite Maven artifacts from Cyfronet Maven repository and their scope (compilation or test). EPE DSR Plugin has been omitted, since it does not use Maven for dependency management. However, it also has dependencies, which include the following plugins: `cyfronet.gridspace.api` – version 0.4.0, `cyfronet.gridspace.gisde.auth` – version 1.1.3, `cyfronet.gridspace.voconfig.plugin.preferences` – version 0.6.0, `org.eclipse.ui`, and `org.eclipse.core.runtime`.

In particular, artifacts related to *JSAGA*, *CoG* and *VOMS Java API* shown in figure 52 are utilized for manipulating Grid certificates, while *LFC API* from ChemPo project wraps SEE-Grid Java File Management library providing means for accessing Grid data sources and managing entries in LFC Catalogue.

DAC2 dependencies portrayed in figure 53 are mostly those related to accessing various types of data sources, including Virtual Laboratory Data Access Service (DAS), eXist Native XML Database, MySQL, HSQLDB, PostgreSQL. Dependencies added by LFC DS client are also apparent. However, they are not many, as can also be seen on figure 50 – small number of dependencies incorporated into GSEngine is required 4-th constraint listed in section 6.5. None of artifacts required by LFC DS client library is platform dependent. Thus, requirement NF-9 has been met. Furthermore, by using Maven for compilation and dependencies management in 3 of 4 LFC DS collaborating components: DAC2, LFC DS server and LFC DS client requirement

NF-8 is partly met. Taking into consideration the fact that Eclipse, which is a platform of EPE DSR-Plugin, has its own mechanism of managing dependencies, it can be said, that the requirement NF-8 has been fulfilled in its entirety.

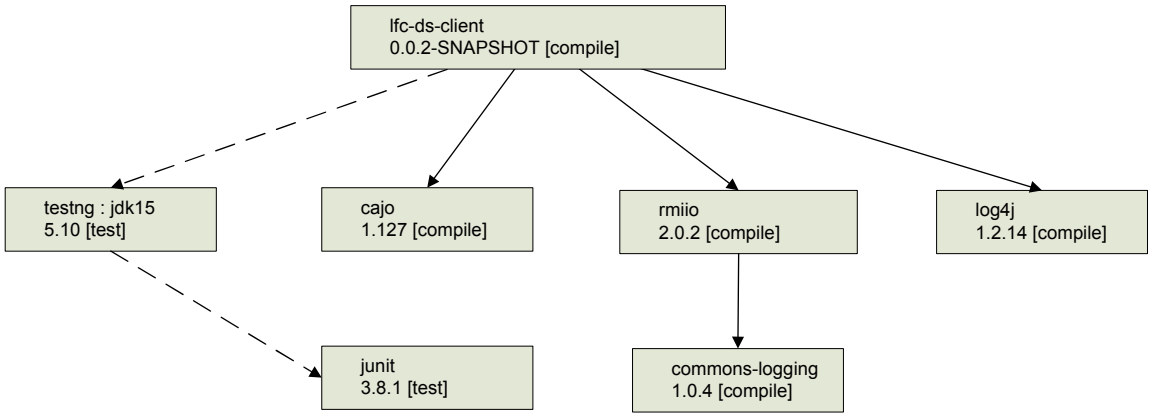


Figure 50: LFCDS client library – dependency graph

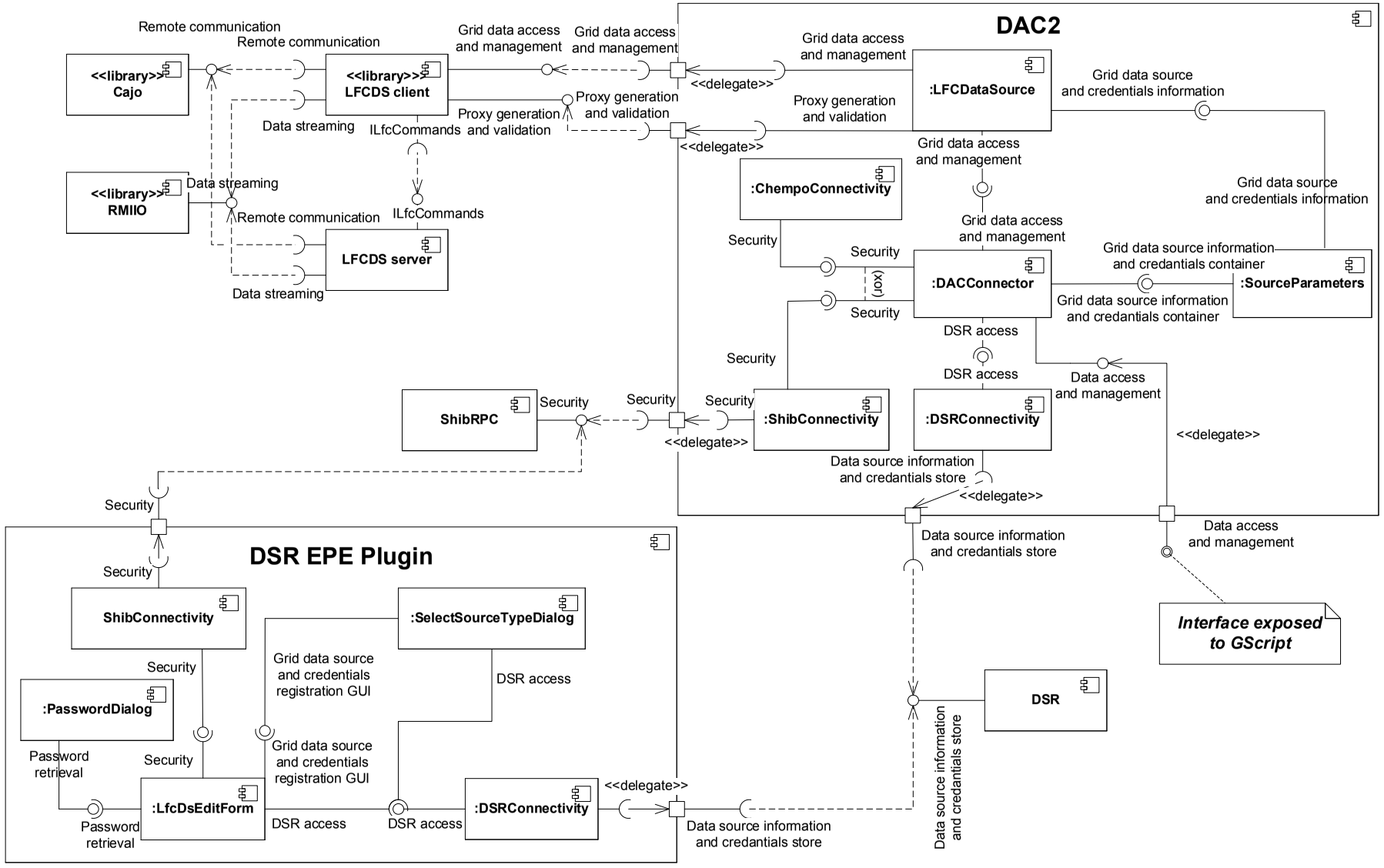


Figure 51: Component diagram depicting dependencies between system components

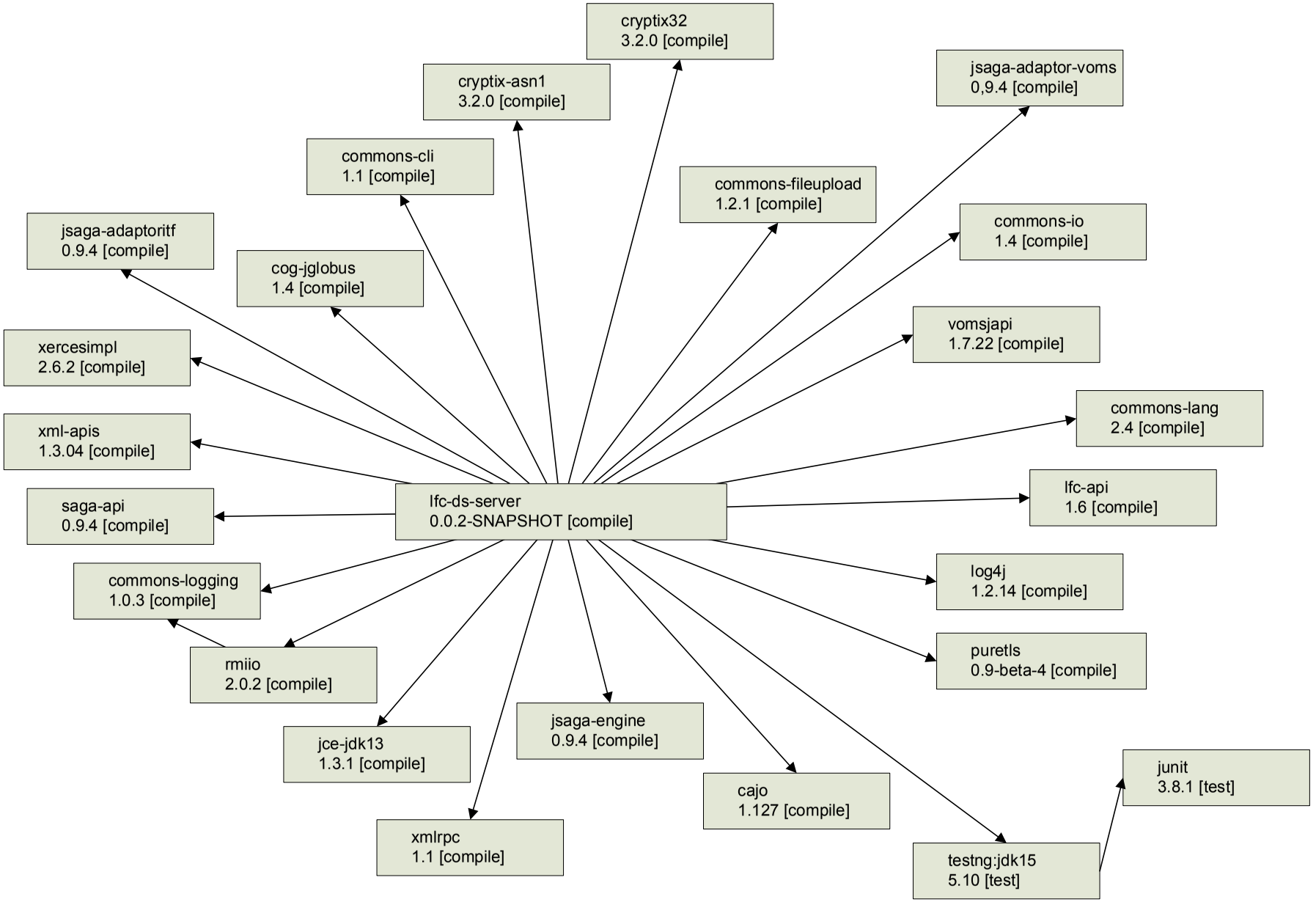


Figure 52: LFCDS server – dependency graph

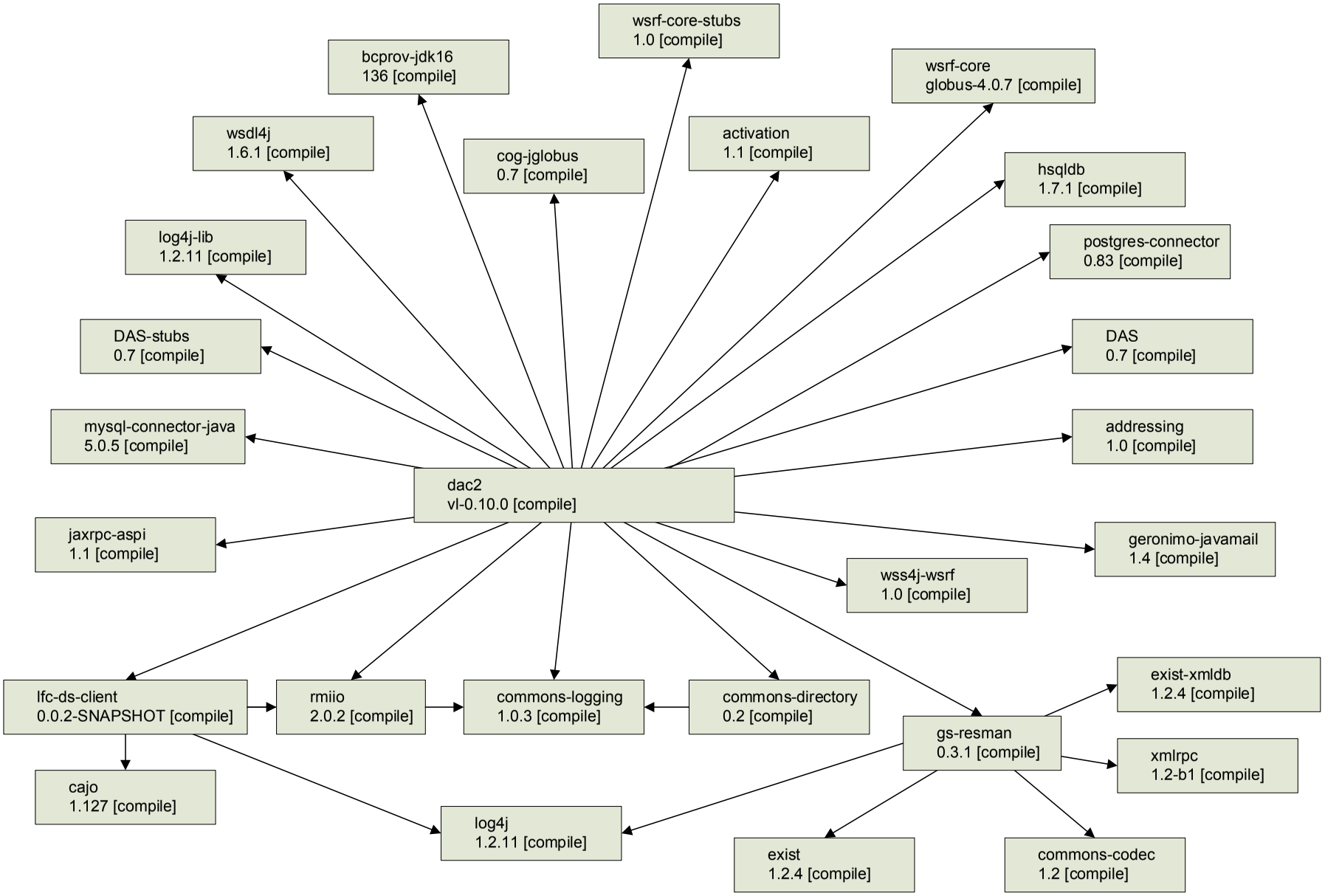


Figure 53: DAC2 – dependency graph

8.4.4 Information

This clause contains specification of data that is stored for the purpose of LFC DS operation. In particular, figure 54 delineates Data Source Registry database schema. Before introduction of LFC DS, *DataSources* table contained all the information needed by data sources. During adaptation of DSR for incorporation of new data source type, the *DataSources* table has been split into *RelationalDataSource* and *LFCDataSources* which incorporated some reorganization of primary and foreign key relationship. However, data movement from earlier to newer schema has been successful. Moreover, in order to be consistent with naming, *DataSourceCredentials* table has been renamed to *RelationalDataSourceCredentials*.

Furthermore, *LFCDSConnections* table has been added which maintains information about various database servers. Additionally, *LFCCertData* stores user Grid credentials.

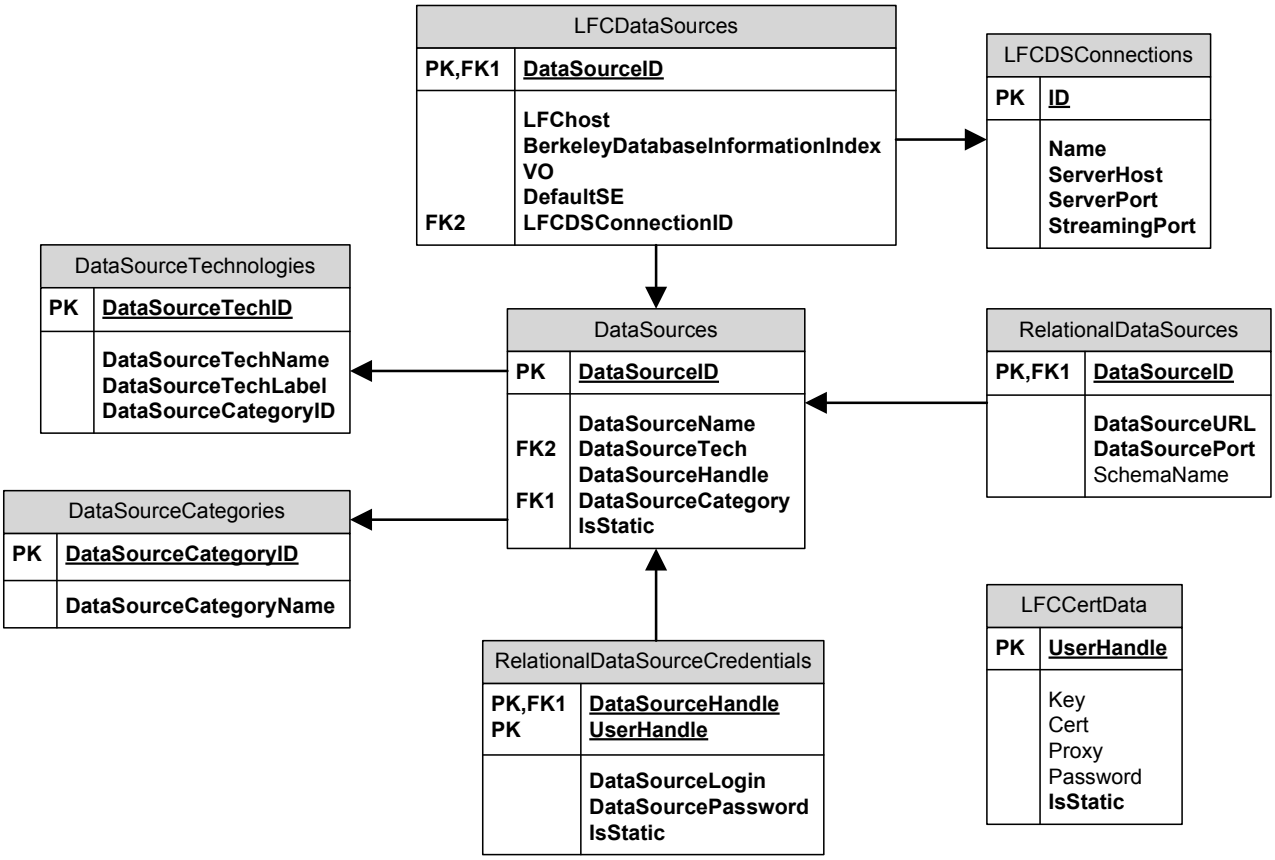


Figure 54: DSR – database schema

8.4.5 Interface

Figure 55 illustrates a user interface that enables registration of LFC data sources. It is invoked by DSR-EPE Plugin, when a user requests creation or edition of LFC data source. Explanation of meaning of "LFC data source parameters" fields, namely "LCG File Catalogue", "Berkeley Database Information Index" and "Storage element" has been provided in 3.6. "Your credentials" group allows for uploading and removing grid user credentials from DSR together with specifying whether they are available to other authenticated users. LFC DS Server connection is a connection to LFC DS Server running somewhere on gLite UI. Normally, user chooses the server to use from a list. If there is no server he or she intends to use, they can add a new entry. LFC DS Server connection information is usually conveyed to user by administrator who installed LFC DS. Figure 56 demonstrates validation mechanisms incorporated into the form, figure 57 illustrates DSR EPE Plugin view onto data sources registered in Virtual Laboratory. Finally, 58 presents data source type selection form that is invoked when user requests adding a new data source. User interface forms shown in 57 and 58 were created by Piotr Nowakowski.

With regard to software interfaces, they have already been specified in 7.3.

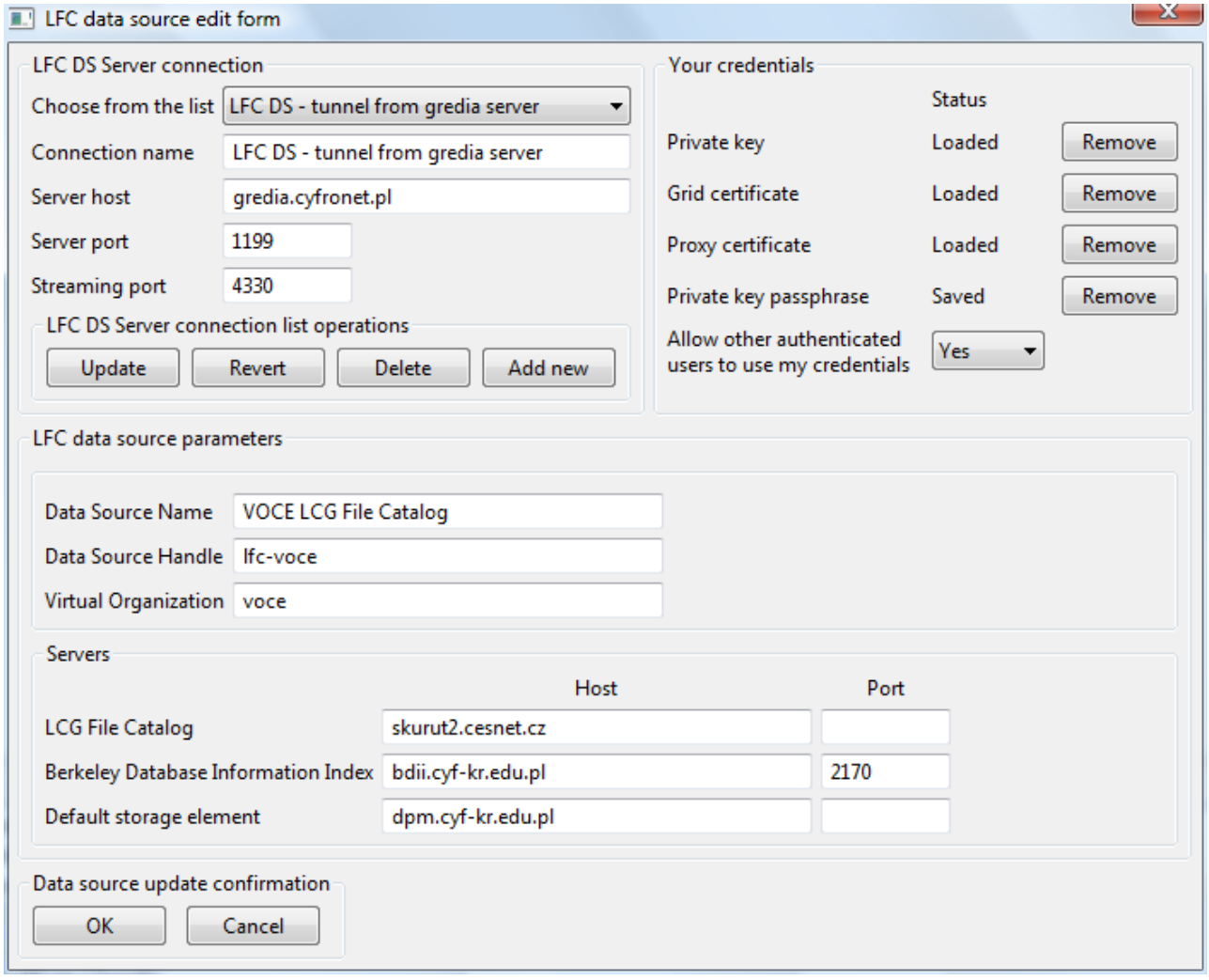


Figure 55: User interface for registering LFC data sources

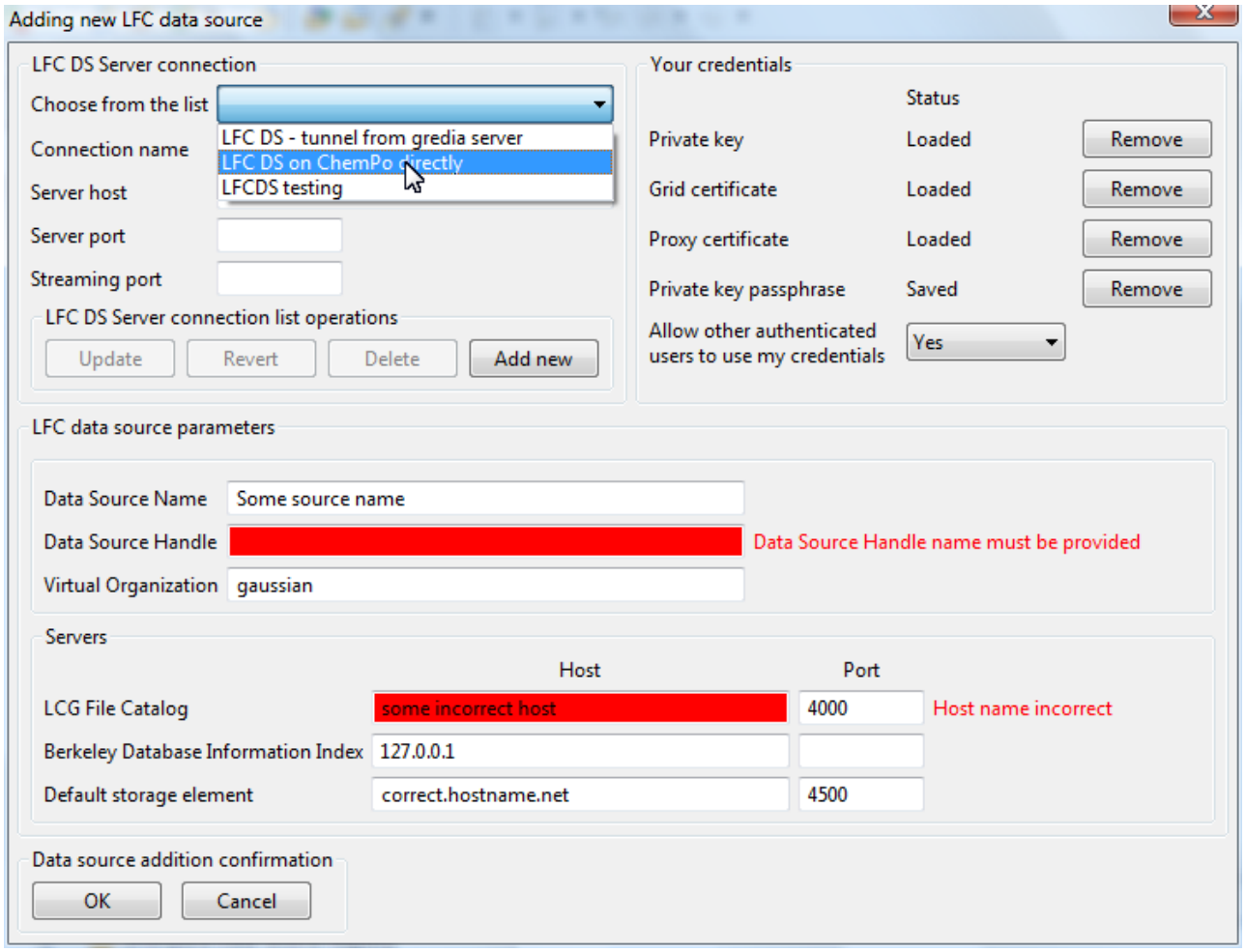


Figure 56: Demonstration of DSR EPE Plugin LFC DS Edit Form validation mechanisms

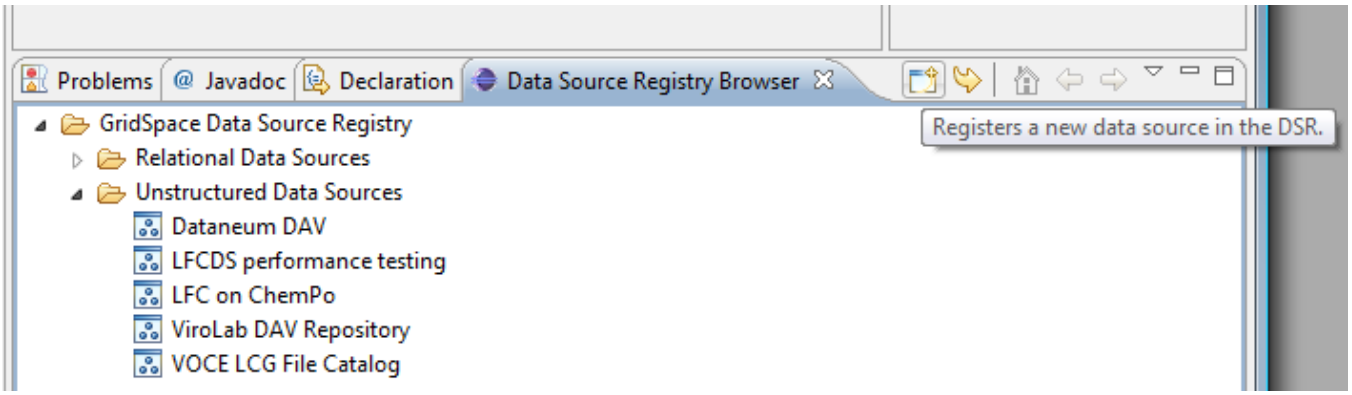


Figure 57: Tree view onto data sources registered in Virtual Laboratory

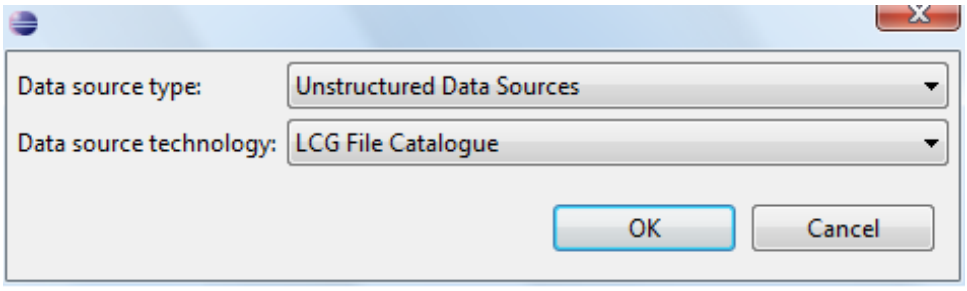


Figure 58: Data source selection form

8.4.6 Interaction

Interaction is one of the more interesting aspects of LFC DS. Before performing data access and management operations, LFC DS connector must initialize with Grid credentials. If a Grid proxy certificate is present in DSR, it is used for LFC and data access operations. If it is not it is generated and stored in DSR (see figure 59).

Subsequently, a user may execute commands listed in table 6. Figure 60 depicts interaction of LFC DS components when a command does not require streaming. On the other hand, when executing one of the *open* methods, the interaction scheme is different. Figure 61 presents simplified sequence diagram of *getFile* method execution, while figure 62 addresses the case of sending a file to Grid. All classes, with the exception of `RemoteOutputStream`, `RemoteInputStream`, and `LfcWorker` have been discussed in Logical design view. `RemoteOutputStream` and `RemoteInputStream` are classes of RMIIO library providing streaming functionalities while `LfcWorker` is a ChemPo class that effectuates actual Grid data access code that it receives from `LfcCommands` that communicates with it via a socket.

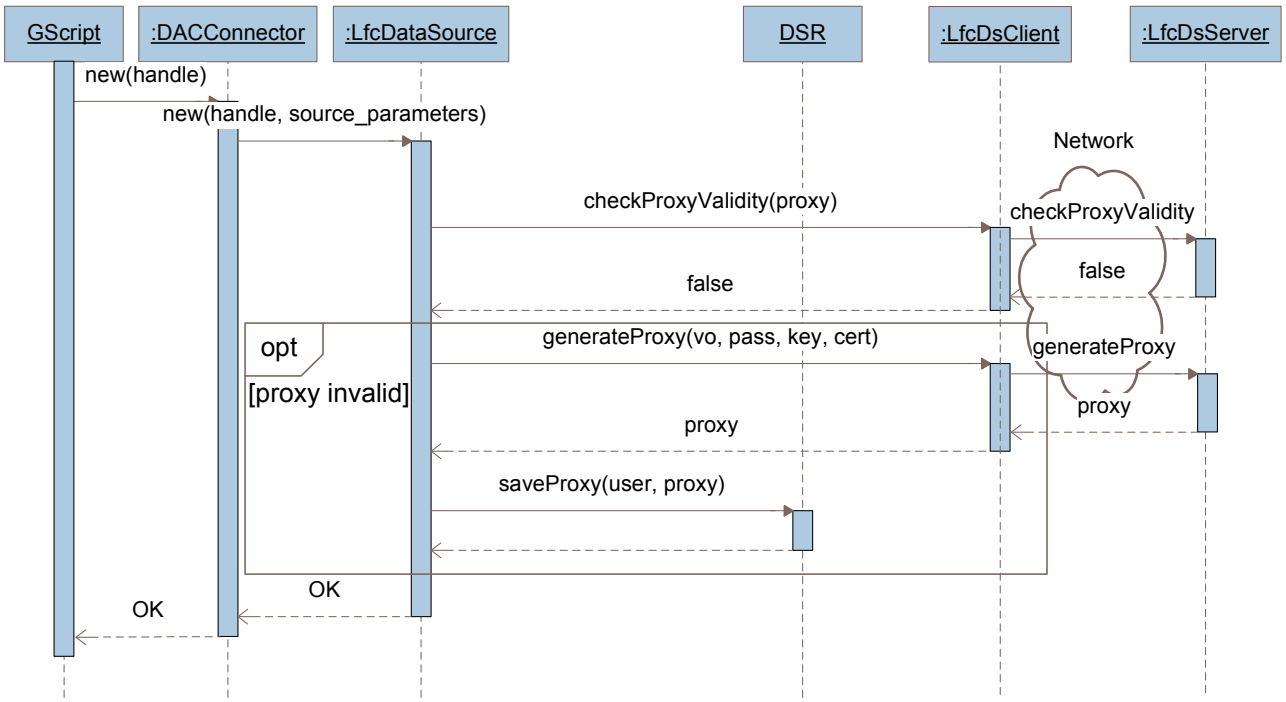


Figure 59: Initialization of LFC DS connector – sequence diagram

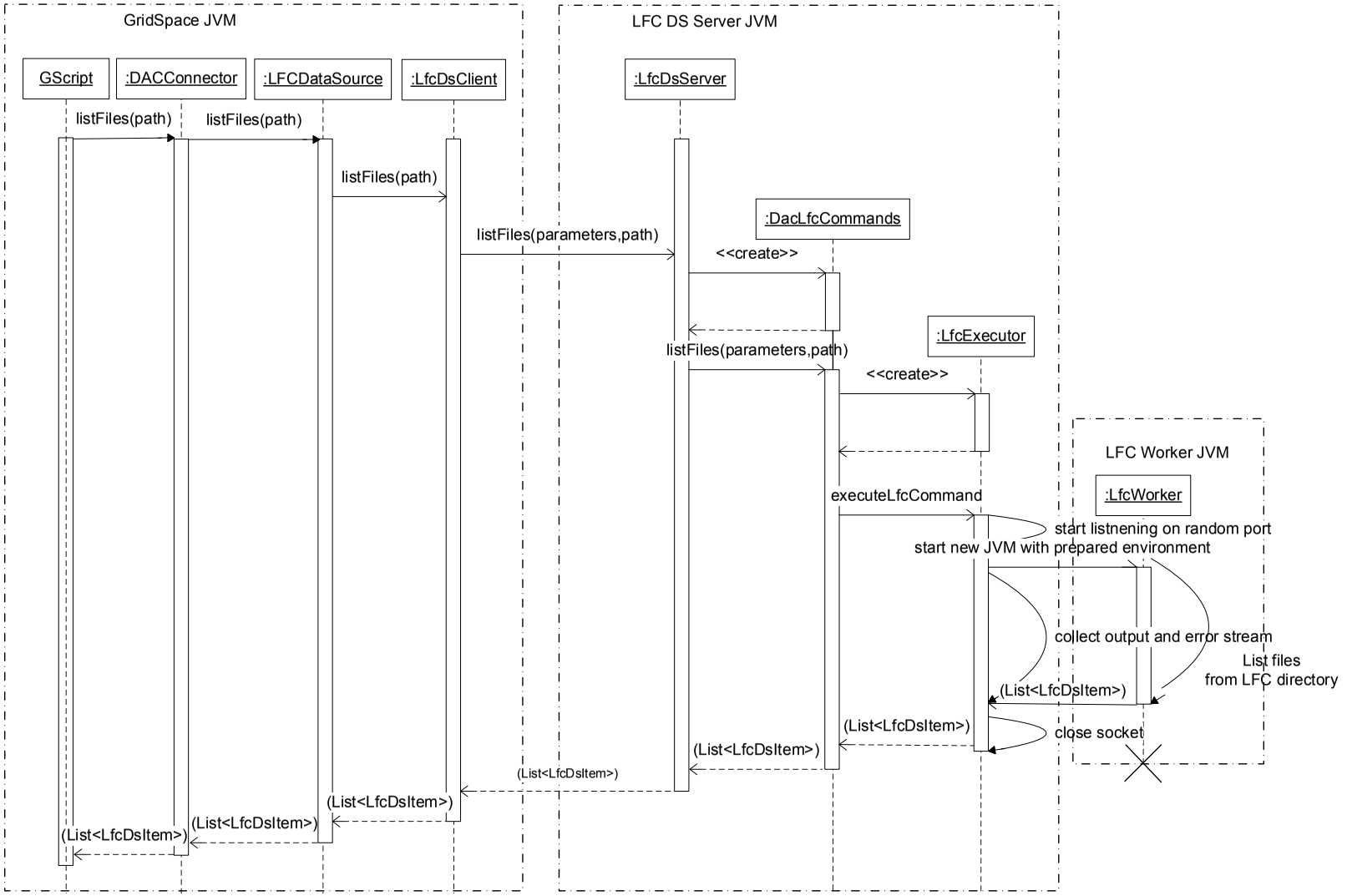


Figure 60: A sample LFC command – in this case, *listFiles* command

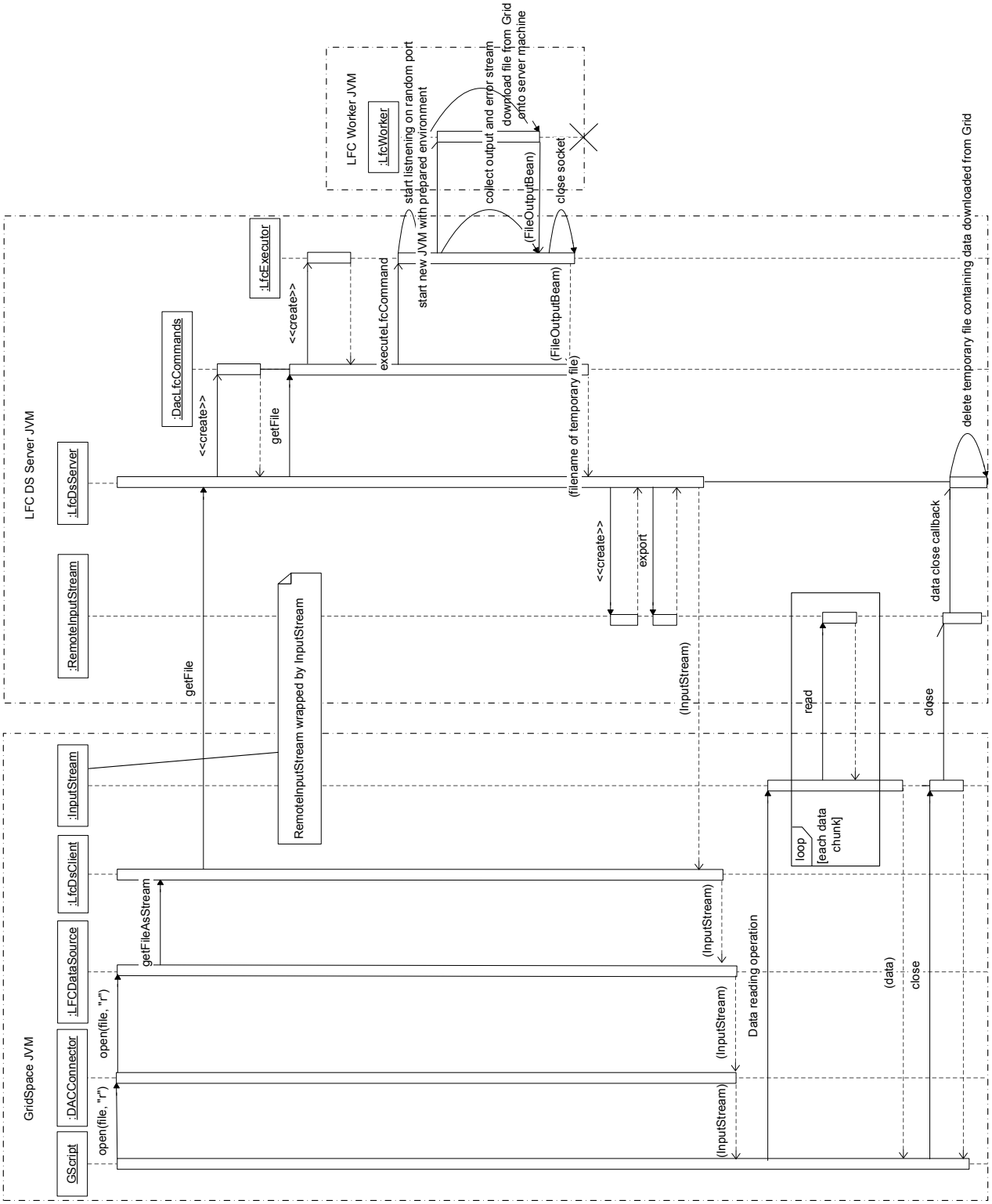


Figure 61: Reading file from Grid – sequence diagram

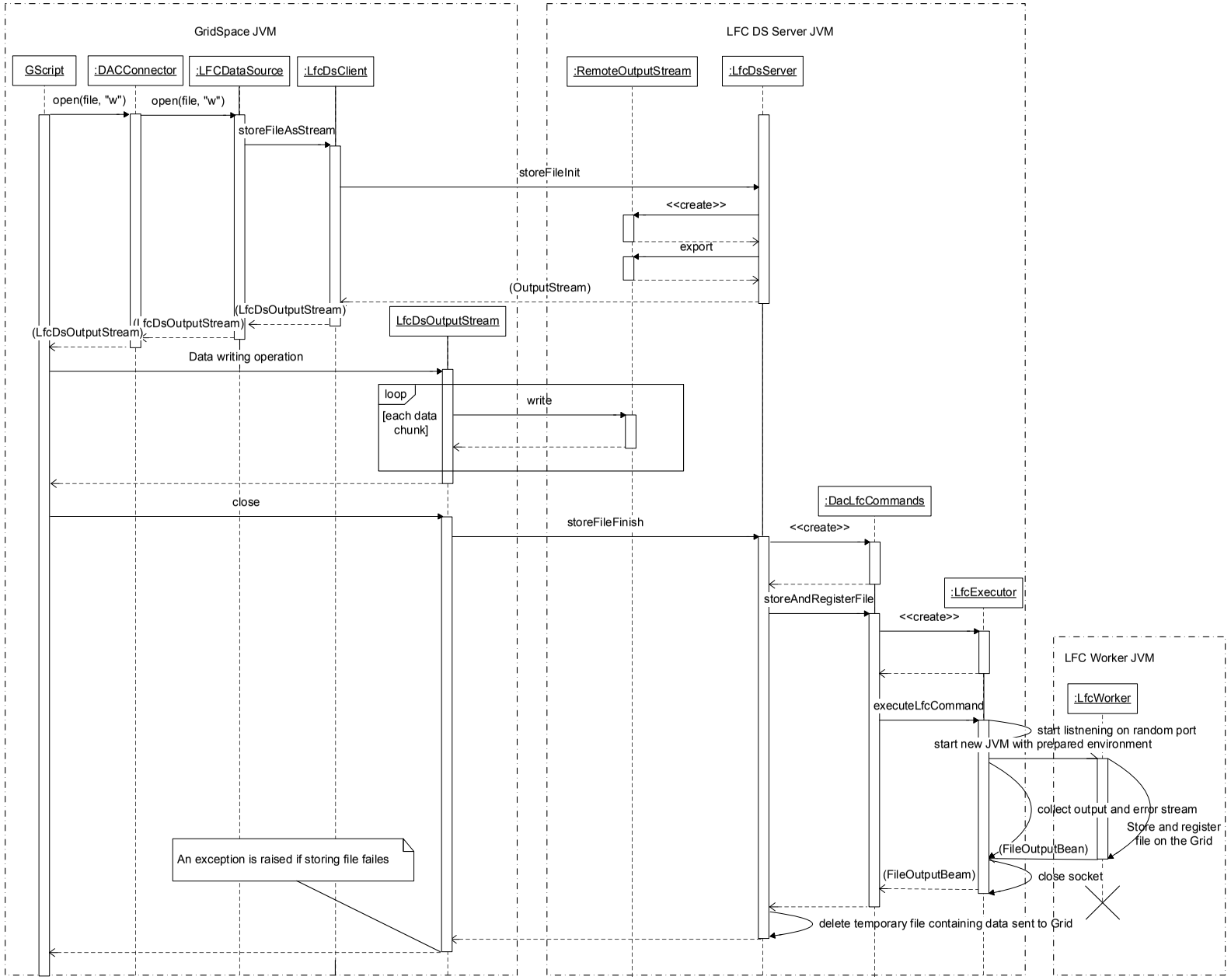


Figure 62: Sending file to Grid – sequence diagram

9 Verification and validation

Verification and validation has been performed both using client library and using GScript client code. GScript client was tested both as a standalone library and in conjunction with GSEngine, i.e. by executing code using GSEngine interpreter. LFCDS server, LFCDS connector and client Java library were run on ChemPo server (`chempo.grid.cyfronet.pl`). A test over WAN connection has also been performed with the following configuration: LFCDS server running on EGEE CESNET gLite UI in Czech Republic (host: `ui1.egee.cesnet.cz`), tunneling commands were executed on GREDIA server (`gredia.cyfronet.pl`) and GScript client was run on ChemPo machine.

9.1 Functional tests

Approach Functional tests with high granularity were possible only when testing LFC DS client Java library using TestNG testing framework, which enables specifying test dependencies. In the case of LFC connector it was not possible. Therefore, one large test was executed instead. Listing included below provides code used for testing LFC connector interaction with LFC DS server.

LFC connector functional test

```
1  # Author: Marek Pomocka
2
3  require 'cyfronet/gridspace/dac2/dac_connector.rb'
4
5  ###
6  ### This is a test file for LFC Data Source connector.
7  ###
8
9  def LFCDSTest(ds)
10   # File names can start with or without a slash. Both are mapped to /grid/vo_name/path
11   puts "delete 'mpomocka/test_lfcds' = #{ds.delete('mpomocka/test_lfcds')}"
12   #puts "createDirectory '/','mpomocka' = #{ds.createDirectory('/','mpomocka')}"
13   puts "createDirectory('mpomocka/test_lfcds') finished "+(ds.createDirectory("mpomocka/
         test_lfcds")==true ? "successfully":"unsuccessfully")
14   puts "directory? '/mpomocka/test_lfcds' = #{ds.directory? "/mpomocka/test_lfcds"}"
15   puts "createDirectory('/mpomocka/test_lfcds','test_dir) finished "+(ds.createDirectory("/
         mpomocka/test_lfcds","test_dir")==true ? "successfully":"unsuccessfully")
16   puts "directory? '/mpomocka/test_lfcds' = #{ds.directory? "/mpomocka/test_lfcds"}"
17   puts "file? '/mpomocka/test_lfcds/test_file1.txt' = #{ds.file? "/mpomocka/test_lfcds/
         test_file1.txt"}"
18   puts "storeFile 'mpomocka/test_lfcds/test_file1.txt' command finished "+(ds.storeFile("
         TEST file 1 cOnTeNtS".to_java_bytes, "mpomocka/test_lfcds/test_file1.txt")==true ? "
         successfully":"unsuccessfully")
19   puts "file? '/mpomocka/test_lfcds/test_file1.txt' = #{ds.file? "/mpomocka/test_lfcds/
         test_file1.txt"}"
20   puts "file? '/mpomocka/test_lfcds/test_file2.txt' = #{ds.file? "/mpomocka/test_lfcds/
         test_file2.txt"}"
```

```
21    f = ds.open("/mpomocka/test_lfcds/test_file2.txt",:write)
22    f.puts "First line of the file file 2"
23    f.puts "Second line of the file file 2"
24    f.close
25    ds.open("/mpomocka/test_lfcds/test_file3.txt",:w) do |f|
26      f.puts "Another way to write to a file"
27      f.puts "Note that close is not necessary"
28    end
29    puts "exist? '/mpomocka/test_lfcds/test_file2.txt' = #{ds.exist? "/mpomocka/test_lfcds/
          test_file2.txt"}"
30    puts "getFile '/mpomocka/test_lfcds/test_file1.txt' = #{String.from_java_bytes ds.getFile(
          "mpomocka/test_lfcds/test_file1.txt")}"
31    puts "test_file2.txt contents:"
32    f = ds.open("/mpomocka/test_lfcds/test_file2.txt", :read)
33    f.each {|line| puts line}
34    f.close
35    ds.open("/mpomocka/test_lfcds/test_file3.txt", "r") do |file|
36      file.each {|line| puts line}
37    end
38    puts "getSize /mpomocka/test_lfcds/test_file1.txt "+ds.getSize("mpomocka/test_lfcds/
          test_file1.txt").to_s
39    puts "getSize /mpomocka/test_lfcds/test_file2.txt "+ds.getSize("mpomocka/test_lfcds/
          test_file2.txt").to_s
40    l=ds.listFiles("/mpomocka/test_lfcds/")
41    l.each do |item|
42    puts item.get_name + " is a " + if item.is_directory then "directory" else "file" end
43    end
44    puts "delete command executed on a file finished "+(ds.delete("mpomocka/test_lfcds/
          test_file2.txt")?"successfully":"unsuccessfully")
45    puts "delete command executed on a directory finished "+(ds.delete("mpomocka/test_lfcds")?
          "successfully":"unsuccessfully")
46  end
47
48  begin
49    # 1 argument: handle - obvious
50    # 2 arguments: handle and password to the private key - useful if a user does not want to
51    #     keep password in the DSR
52    # 2 arguments: handle and proxy - if someone has not provided their credentials
53    #     in the DSR, but want to use the data source.
54    # Note: these two method above are distinguished by the length of the second argument
55    #     (if more than 300 bytes, it is assumed to be a proxy)
56    # 4 arguments: handle, private key, grid certificate and password to the private key
57    #     - useful if one wants to use the LFC data source, but not registered their
58    #     credentials in the DSR _and_ has not generated the proxy - proxy is being saved
59    #     in the DSR if the user has an entry in the database
60
61    ### One argument constructor - everything is in the DSR
62    ds = DACConnector.new("lfcds-test");
63    puts "Successfully instantiated LFC data source (1 arg)"
64    LFCDSTest(ds)
65
66    ## 2 argument constructor - handle and password to the private key
67    ds = DACConnector.new("lfcds-test","your_password")
```

```
68    puts "Successfully instantiated LFC data source (2 args)"
69    LFCDSTest(ds)
70
71    ## 2 argument constructor - handle and proxy
72    ds = DACConnector.new("lfcds-test",IO.read("C:/Users/Marek/Documents/cert/x509up_u506"))
73    puts "Successfully instantiated LFC data source (2 args - 2nd one a proxy)"
74    LFCDSTest(ds)
75
76    ## 4 argument constructor - handle, private key, grid certificate and password to the
          private key
77    ds = DACConnector.new("lfcds-test",IO.read("C:/Users/Marek/Documents/cert/userkey.pem"),
78      IO.read("C:/Users/Marek/Documents/cert/usercert.pem"), # change to file names stored in
            your computer
79      "your_password")
80    puts "Successfully instantiated LFC data source (4 args)"
81    LFCDSTest(ds)
82  end
```

Output of this script is as follows (for brevity product of lines 66-81 has been omitted):

```
Successfully instantiated LFC data source (1 arg)
delete 'mpomocka/test_lfcds' = false
createDirectory('mpomocka/test_lfcds') finished successfully
directory? '/mpomocka/test_lfcds' = true
createDirectory('/mpomocka/test_lfcds','test_dir) finished successfully
directory? '/mpomocka/test_lfcds' = true
file? '/mpomocka/test_lfcds/test_file1.txt' = false
storeFile 'mpomocka/test_lfcds/test_file1.txt' command finished successfully
file? '/mpomocka/test_lfcds/test_file1.txt' = true
file? '/mpomocka/test_lfcds/test_file2.txt' = false
exist? '/mpomocka/test_lfcds/test_file2.txt' = true
getFile '/mpomocka/test_lfcds/test_file1.txt' = TEST file 1 cOnTeNtS
test_file2.txt contents:
First line of the file file 2
Second line of the file file 2
Another way to write to a file
Note that close is not necessary
getSize /mpomocka/test_lfcds/test_file1.txt 20
getSize /mpomocka/test_lfcds/test_file2.txt 61
test_dir is a directory
test_file1.txt is a file
test_file2.txt is a file
test_file3.txt is a file
delete command executed on a file finished successfully
delete command executed on a directory finished successfully
```

On the other hand, the subsequent listing includes TestNG test case that was utilized for functional test of LFC DS client interacting with the server.

Functional test of LFC DS client library interacting with LFC DS server

```
1  package cyfronet.gridspace.dac2.lfcds;
2
3  import java.io.ByteArrayOutputStream;
4  import java.io.File;
5  import java.io.FileInputStream;
6  import java.util.List;
7  import org.apache.log4j.Logger;
8
9  import cyfronet.gridspace.dac2.lfcds.client.LfcDsClient;
10 import cyfronet.gridspace.dac2.lfcds.exceptions.LfcDsException;
11 import org.testng.annotations.*;
12
13 /**
14  * @author Marek Pomocka
15  *
16  */
17 public class LfcDsServerTest {
18         private static final Logger log = Logger.getLogger(LfcDsServerTest.class);
19         private static final String USER_CERT = TestProperties.getInstance().getProperty("
               user.cert");
20         private static final String USER_KEY = TestProperties.getInstance().getProperty("
               user.key");
21         private static final String CERT_PASSWORD = TestProperties.getInstance().getProperty
               ("cert.password");
22         private static final String USER_DIR = TestProperties.getInstance().getProperty("
               user.directory");
23         private static final String TEST_DIR = TestProperties.getInstance().getProperty("
               test.directory");
24         private static final String TEST_PATH = USER_DIR+"/"+TEST_DIR;
25         private static final String TEST_FILE_CONTENTS = "TEST file contents";
26         static LfcDsClient cl;
27
28         private ByteArrayOutputStream certBytes;
29         private ByteArrayOutputStream keyBytes;
30
31         @BeforeSuite
32         void setUp() throws Exception {
33                 FileInputStream certFile=new FileInputStream(new File(USER_CERT));
34                 FileInputStream keyFile=new FileInputStream(new File(USER_KEY));
35                 certBytes=new ByteArrayOutputStream();
36                 keyBytes=new ByteArrayOutputStream();
37                 LfcDsClient.copyLarge(certFile,certBytes);
38                 LfcDsClient.copyLarge(keyFile,keyBytes);
39                 certFile.close();
40                 keyFile.close();
41         }
42
43         @Test
```

```
44          public void testServerConnection() throws LfcDsException {
45                  log.info("Testing server connection");
46                  log.info("Connection parameters:");
47                  log.info("  user.host = "+TestProperties.getInstance().getProperty("client.
                        host"));
48                  log.info("  user.port = "+TestProperties.getInstance().getProperty("client.
                        port"));
49                  cl = new LfcDsClient("//"+TestProperties.getInstance().getProperty("client.
                        host")+":"+
50                                  TestProperties.getInstance().getProperty("client.port")+"/
                                      LfcDsServer",
51                                  TestProperties.getInstance().getProperty("streaming.port"),
52                                  TestProperties.getInstance().getProperty("user.vo"),
53                                  TestProperties.getInstance().getProperty("lfc.host"),
54                                  TestProperties.getInstance().getProperty("sbdii.host"),
55                                  TestProperties.getInstance().getProperty("se.url"),
56                                  keyBytes.toByteArray(), certBytes.toByteArray(), null);
57                  // First method to try whether connection works.
58                  // Furthermore, it deletes earlier test artifacts if there are any
59                  log.info("Trying to remove earlier test directory (if exists)");
60                  log.info("Directory " +(cl.delete(TEST_PATH)?"":"not")+" removed");
61                  log.info("Server connection working");
62          }
63          @Test (dependsOnMethods={"testServerConnection"})
64          public void testProxyGeneration() throws LfcDsException {
65                  log.info("Testing proxy generation");
66                  assert cl.checkProxyValidity() == false;
67                  cl.generateProxy(CERT_PASSWORD);
68                  assert cl.checkProxyValidity() == true;
69                  log.info("Proxy generation passed");
70          }
71          @Test (dependsOnMethods={"testProxyGeneration"})
72          public void testDirectoryExists() throws LfcDsException {
73                  log.info("Testing 'directoryExists' method");
74                  assert cl.directoryExists(USER_DIR) == true;
75                  assert cl.directoryExists("asojdfioasjfrpFASKFAJSLDFJA/FASIDFJAS324234") ==
                        false;
76                  log.info("'directoryExists' method test passed");
77          }
78          @Test (dependsOnMethods={"testProxyGeneration"})
79          public void testExists1() throws LfcDsException {
80                  log.info("Testing  method 'exists' -- test 1");
81                  assert cl.exists(USER_DIR) == true;
82                  assert cl.exists("asojdfioasjfrpFASKFAJSLDFJA/FASIDFJAS324234") == false;
83                  log.info("'exists' method test 1 passed");
84          }
85          @Test (dependsOnMethods={"testDirectoryExists"})
86          public void testCreateDirectory() throws LfcDsException {
87                  log.info("Testing directory creation");
88                  assert cl.createDirectory(USER_DIR, TEST_DIR) == true;
89                  assert cl.createDirectory(USER_DIR, TEST_DIR) == false;
90                  assert cl.directoryExists(TEST_PATH) == true;
91                  log.info("Directory creation test passed");
```

```
92          }
93          @Test (dependsOnMethods={"testCreateDirectory","testExists1"})
94          public void testStoreFile() throws LfcDsException {
95                  log.info("Testing file creation");
96                  assert cl.storeFile(TEST_PATH, "test_file1.txt", TEST_FILE_CONTENTS.getBytes
                        ()) == true;
97                  assert cl.directoryExists(TEST_PATH+"/"+"test_file1.txt") == false;
98                  assert cl.exists(TEST_PATH+"/"+"test_file1.txt") == true;
99                  log.info("File creation test passed");
100         }
101         @Test (dependsOnMethods={"testStoreFile"})
102         public void testFileExists() throws LfcDsException {
103                 log.info("Testing method 'fileExists'");
104                 assert cl.fileExists(TEST_PATH+"/"+"test_file1.txt") == true;
105                 assert cl.fileExists(TEST_PATH) == false;
106                 assert cl.fileExists("asfjaskfjaskdfjRRU3242394/FASDKFczxlkcjz/asfasd") ==
                        false;
107                 log.info("'fileExists' method test passed");
108         }
109         @Test (dependsOnMethods={"testStoreFile"})
110         public void testExists2() throws LfcDsException {
111                 log.info("Testing  method 'exists' -- test 2");
112                 assert cl.exists(TEST_PATH+"/"+"test_file1.txt") == true;
113                 log.info("'exists' method test 2 passed");
114         }
115         @Test (dependsOnMethods={"testStoreFile"})
116         public void testGetFile() throws LfcDsException {
117                 log.info("Testing method 'getFile'");
118                 String s=new String(cl.getFile(TEST_PATH+"/"+"test_file1.txt"));
119                 assert s.equals(TEST_FILE_CONTENTS);
120                 log.info("'getFile' method test passed");
121         }
122         @Test (dependsOnMethods={"testStoreFile"})
123         public void testGetSize() throws LfcDsException {
124                 log.info("Testing method 'getSize'");
125                 long l=cl.getSize(TEST_PATH+"/"+"test_file1.txt");
126                 assert l == TEST_FILE_CONTENTS.length();
127                 log.info("'getSize' method test passed");
128         }
129         @Test (dependsOnMethods={"testStoreFile"})
130         public void testListFiles() throws LfcDsException {
131                 log.info("Testing method 'listFiles'");
132                 cl.storeFile(TEST_PATH, "test_file2.txt", "Test 2 file -- contents".getBytes
                        ());
133                 cl.createDirectory(TEST_PATH, "test_dir1");
134                 List<LfcDsItem> l=cl.listFiles(TEST_PATH);
135                 assert l.size() == 3;
136                 for(LfcDsItem item: l) {
137                         assert item.getName().equals("test_file1.txt") || item.getName().
                                equals("test_file2.txt") ||
138                                         item.getName().equals("test_dir1");
139                         if (item.getName().equals("test_file1.txt"))
140                                 assert item.isDirectory()==false;
```

```
                        if (item.getName().equals("test_file2.txt"))
                                assert item.isDirectory()==false;
                        if (item.getName().equals("test_dir1"))
                                assert item.isDirectory()==true;
                }
                log.info("'listFiles' method test passed");
        }
        @Test (dependsOnMethods={"testListFiles","testGetSize","testGetFile",
                        "testExists2","testFileExists"})
        public void testDeleteFile() throws LfcDsException {
                log.info("Testing method 'deleteFile'");
                assert cl.fileExists(TEST_PATH+"/"+"test_file2.txt") == true;
                assert cl.delete(TEST_PATH+"/"+"afasdfasdfr243142") == false;
                assert cl.delete(TEST_PATH+"/"+"test_file2.txt") == true;
                assert cl.fileExists(TEST_PATH+"/"+"test_file2.txt") == false;
                log.info("'deleteFile' method test passed");
        }
        @Test (dependsOnMethods={"testListFiles"})
        public void testDeleteEmptyDirectory() throws LfcDsException {
                log.info("Testing deletion of empty directory");
                assert cl.directoryExists(TEST_PATH+"/"+"test_dir1") == true;
                assert cl.delete(TEST_PATH+"/"+"test_dir1") == true;
                assert cl.directoryExists(TEST_PATH+"/"+"test_dir1") == false;
                log.info("Deletion of empty directory succeeded");
        }
        @Test (dependsOnMethods={"testDeleteFile","testDeleteEmptyDirectory"})
        public void testDeleteDirectoryWithContents() throws LfcDsException {
                log.info("Testing deletion of directory with contents");
                assert cl.directoryExists(TEST_PATH) == true;
                assert cl.delete(TEST_PATH) == true;
                assert cl.directoryExists(TEST_PATH) == false;
                log.info("Deletion of directory with contents succeeded");
        }
        @AfterSuite
        void tearDown() throws Exception {
                cl.delete(TEST_PATH);
                cl.disconnect();
                log.info("Client disconnected");
        }
}
```

Results

Test	Methods Passed	Scenarios Passed	# skipped	# failed	Total Time	Included Groups	Excluded Groups
Command line test	14	14	0	0	77.2 seconds		

Class	Method	# of Scenarios	Time (Msecs)
Command line test — passed			
cyfronet.gridspace.dac2.lfcds.LfcDsServerTest "Command line test"	testCreateDirectory	1	3478
	testDeleteDirectoryWithContents	1	7431
	testDeleteEmptyDirectory	1	4761
	testDeleteFile	1	7258
	testDirectoryExists	1	2503
	testExists1	1	2501
	testExists2	1	1519
	testFileExists	1	4052
	testGetFile	1	8654
	testGetSize	1	1270
	testListFiles	1	11681
	testProxyGeneration	1	8892
	testServerConnection	1	895
	testStoreFile	1	12270

Figure 63: Verification tests – TestNG report

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running TestSuite
0    INFO  LfcDsServerTest  - Testing server connection
3    INFO  LfcDsServerTest  - Connection parameters:
3    INFO  LfcDsServerTest  -   user.host = chempo.grid.cyfronet.pl
4    INFO  LfcDsServerTest  -   user.port = 2000
204  INFO  LfcDsServerTest  - Trying to remove earlier test directory (if exists)
891  INFO  LfcDsServerTest  - Directory not removed
892  INFO  LfcDsServerTest  - Server connection working
900  INFO  LfcDsServerTest  - Testing proxy generation
9791 INFO  LfcDsServerTest  - Proxy generation passed
9798 INFO  LfcDsServerTest  - Testing 'directoryExists' method
12300 INFO  LfcDsServerTest  - 'directoryExists' method test passed
12302 INFO  LfcDsServerTest  - Testing directory creation
15779 INFO  LfcDsServerTest  - Directory creation test passed
15781 INFO  LfcDsServerTest  - Testing  method 'exists' -- test 1
18280 INFO  LfcDsServerTest  - 'exists' method test 1 passed
18282 INFO  LfcDsServerTest  - Testing file creation
30551 INFO  LfcDsServerTest  - File creation test passed
30553 INFO  LfcDsServerTest  - Testing  method 'exists' -- test 2
32071 INFO  LfcDsServerTest  - 'exists' method test 2 passed
32075 INFO  LfcDsServerTest  - Testing method 'fileExists'
36123 INFO  LfcDsServerTest  - 'fileExists' method test passed
36125 INFO  LfcDsServerTest  - Testing method 'getFile'
44777 INFO  LfcDsServerTest  - 'getFile' method test passed
44779 INFO  LfcDsServerTest  - Testing method 'getSize'
46048 INFO  LfcDsServerTest  - 'getSize' method test passed
46053 INFO  LfcDsServerTest  - Testing method 'listFiles'
57734 INFO  LfcDsServerTest  - 'listFiles' method test passed
57736 INFO  LfcDsServerTest  - Testing deletion of empty directory
62497 INFO  LfcDsServerTest  - Deletion of empty directory succeeded
62500 INFO  LfcDsServerTest  - Testing method 'deleteFile'
69758 INFO  LfcDsServerTest  - 'deleteFile' method test passed
69762 INFO  LfcDsServerTest  - Testing deletion of directory with contents
77190 INFO  LfcDsServerTest  - Deletion of directory with contents succeeded
78179 INFO  LfcDsServerTest  - Client disconnected
Tests run: 14, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 78.869 sec
```

Figure 64: Test log from verification tests

9.2 Performance tests

Approach Performance of LFC DS was assessed using client Java library and LFCDS GScript connector. Listing below presents the test code exploited when testing interaction of client Java library with LFCDS server.

Performance test of LFC DS client library interacting with LFC DS server

```
1  package cyfronet.gridspace.dac2.lfcds;
2
3  import java.io.ByteArrayOutputStream;
4  import java.io.File;
5  import java.io.FileInputStream;
6  import java.io.InputStream;
7  import java.io.OutputStream;
8  import java.io.PrintStream;
9  import java.util.Random;
10
11 import org.apache.log4j.Logger;
12
13 import cyfronet.gridspace.dac2.lfcds.client.LfcDsClient;
14 import org.testng.annotations.*;
15
16 /**
17  * @author Marek Pomocka
18  *
19  */
20 public class PerformanceTest {
21         private static final Logger log = Logger.getLogger(LfcDsServerTest.class);
22         private static final String USER_CERT = TestProperties.getInstance().getProperty("
               user.cert");
23         private static final String USER_KEY = TestProperties.getInstance().getProperty("
               user.key");
24         private static final String CERT_PASSWORD = TestProperties.getInstance().getProperty
               ("cert.password");
25         private static final String USER_DIR = TestProperties.getInstance().getProperty("
               user.directory");
26         private static final String TEST_DIR = TestProperties.getInstance().getProperty("
               test.directory");
27         private static final String TEST_PATH = USER_DIR+"/"+TEST_DIR;
28         static LfcDsClient cl;
29
30         private ByteArrayOutputStream certBytes;
31         private ByteArrayOutputStream keyBytes;
32
33         @Test
34         public void testPerformance() throws Exception {
35                 FileInputStream certFile=new FileInputStream(new File(USER_CERT));
36                 FileInputStream keyFile=new FileInputStream(new File(USER_KEY));
37                 certBytes=new ByteArrayOutputStream();
38                 keyBytes=new ByteArrayOutputStream();
39                 LfcDsClient.copyLarge(certFile,certBytes);
40                 LfcDsClient.copyLarge(keyFile,keyBytes);
```

```
41                  certFile.close();
42                  keyFile.close();
43                  log.info("Connection parameters:");
44                  log.info("  user.host = "+TestProperties.getInstance().getProperty("client.
                        host"));
45                  log.info("  user.port = "+TestProperties.getInstance().getProperty("client.
                        port"));
46                  cl = new LfcDsClient("//"+TestProperties.getInstance().getProperty("client.
                        host")+":"+
47                                  TestProperties.getInstance().getProperty("client.port")+"/
                                      LfcDsServer",
48                                  TestProperties.getInstance().getProperty("streaming.port"),
49                                  TestProperties.getInstance().getProperty("user.vo"),
50                                  TestProperties.getInstance().getProperty("lfc.host"),
51                                  TestProperties.getInstance().getProperty("sbdii.host"),
52                                  TestProperties.getInstance().getProperty("se.url"),
53                                  keyBytes.toByteArray(), certBytes.toByteArray(), null);
54                  cl.generateProxy(CERT_PASSWORD);
55                  cl.delete(TEST_PATH);
56                  assert cl.createDirectory(USER_DIR, TEST_DIR) == true;
57
58                  PrintStream ps=new PrintStream(new File("performance_test_results.txt"));
59                  ps.println("# LFCDS performance test results");
60                  ps.println("# file size, sending time, downloading time");
61                  for (int sz=1;sz<=2048;sz*=2) {
62                          ps.print(""+sz+" ");
63                          log.info("Sending file -- " + sz + "MB");
64                          Random r = new Random();
65                          byte[] mb=new byte[1024*1024];
66                          long start = System.currentTimeMillis();
67                          OutputStream os=cl.storeFileAsStream(TEST_PATH, "test_big_file");
68                          long totalRandTime=0;
69                          for (int i=0;i<sz;++i) {
70                                  long t1=System.currentTimeMillis();
71                                  r.nextBytes(mb);
72                                  long t2=System.currentTimeMillis();
73                                  totalRandTime+=t2-t1;
74                                  os.write(mb);
75                          }
76                          os.close();
77                          long elapsedTimeMillis = System.currentTimeMillis()-start;
78                          log.info("" + sz + "MB file sent in "+elapsedTimeMillis+"
                                miliseconds, total rand time "+
79                                          totalRandTime);
80                          log.info("Downloading file -- " + sz + "MB");
81                          ps.print(""+elapsedTimeMillis+" "+totalRandTime+" ");
82                          start = System.currentTimeMillis();
83                          InputStream is=cl.getFileAsStream(TEST_PATH +"/"+"test_big_file");
84                  byte[] buffer = new byte[4*1024];
85                  long count = 0;
86                  int n = 0;
87                  while (-1 != (n = is.read(buffer))) {
88                      count += n;
```

```
89                  }
90                  is.close();
91                  elapsedTimeMillis = System.currentTimeMillis()-start;
92                  log.info("" + sz + "MB file retrieved in "+elapsedTimeMillis+" miliseconds")
                        ;
93                  ps.print(""+elapsedTimeMillis);
94                  ps.println();
95                  cl.delete(TEST_PATH +"/"+"test_big_file");
96                  }
97                  ps.close();
98          }
99  }
```

On the other hand, the following listing presents code used for assessing performance of GScript connector interacting with LFCDS server.

LFC connector performance test

```
1  # Author: Marek Pomocka
2
3  require 'cyfronet/gridspace/dac2/dac_connector.rb'
4  include Java
5
6  def test_streaming_performance(ds)
7    test_dir="mpomocka_temp"
8    test_file="test_big_file"
9    ds.delete test_dir
10   ds.create_directory test_dir
11   file_sizes=(0..11).collect {|x| 2**x }
12   bytes = Java::byte[1024*1024].new
13   r=java.util.Random.new()
14   buf=String.new
15   File.open("performance_test_results.txt","w") do |test_results|
16     test_results.puts "file format: size in MB, sending time, "+
17       "random text generating time, downloading time"
18     file_sizes.each do |file_size|
19       test_results.print file_size.to_s + " "
20       start_time = Time.now
21       text_generating_time=0
22       ds.open(test_dir+"/"+test_file, "w") do |f|
23         file_size.times do
24           t1=Time.now
25           r.nextBytes(bytes)
26           s=String.from_java_bytes bytes
27           t2=Time.now
28           text_generating_time+=(t2-t1)
29           f.write s
30         end
31       end
32       end_time = Time.now
33       test_results.print((end_time - start_time).to_s + " ")
34       test_results.print(text_generating_time.to_s + " ")
35       start_time = Time.now
```

```
36        ds.open(test_dir+"/"+test_file, "r") do |f|
37          file_size.times { f.read(1024*1024,buf) }
38        end
39        end_time = Time.now
40        ds.delete(test_dir+"/"+test_file)
41        test_results.print((end_time - start_time).to_s + " ")
42        test_results.puts
43        test_results.flush
44      end
45    end
46  end
47
48  ds = DACConnector.new("lfcds-test");
49  puts "Successfully instantiated LFC data source"
50  test_streaming_performance(ds)
```

Java client library test results Figures 65, 66 together with table 11 illustrate results of Java client library↔LFCDS server performance tests. Both client and server were located on ChemPo machine.

GScript LFC connector test results Figures 67, 68 and table 12 show results of GScript LFC connector↔LFCDS server performance tests. As with Java client library test, both client and server were located on ChemPo machine. It is noteworthy, that upload and download times of both Java client library and GScript LFC connector are comparable.

Communication over WAN An additional performance test of LFC connector has been performed over Wide Area Network. In particular, LFCDS server was located in CESNET networking center in Czech Republic, while GScript client was run on ChemPo machine situated in ACC Cyfronet. Tunneling was performed by GREDIA server also situated in ACC Cyfronet. Figures 69 and 70 together with table 13 demonstrate results of the tests.

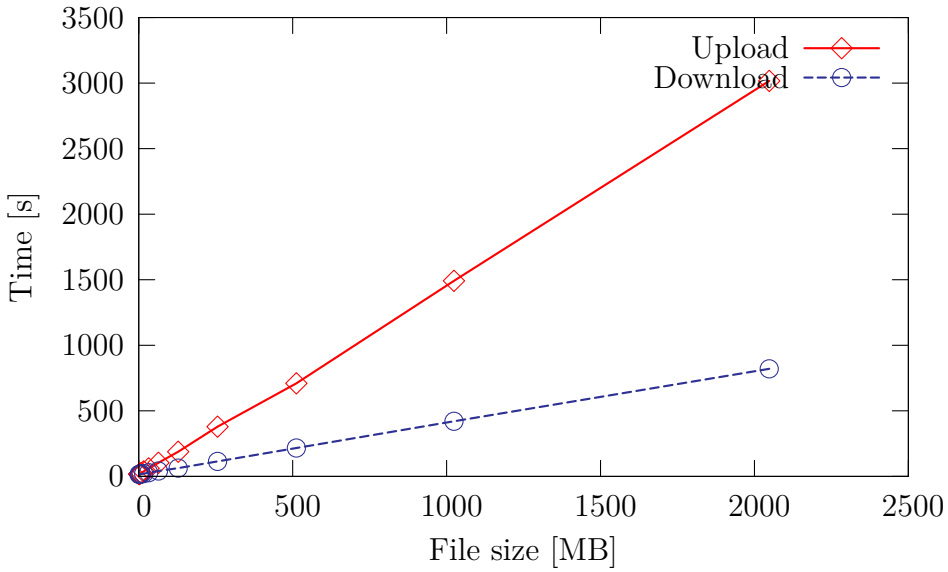


Figure 65: LFCDS Java client library↔LFCDS server performance test: sending and retrieving file from Grid – linear scale

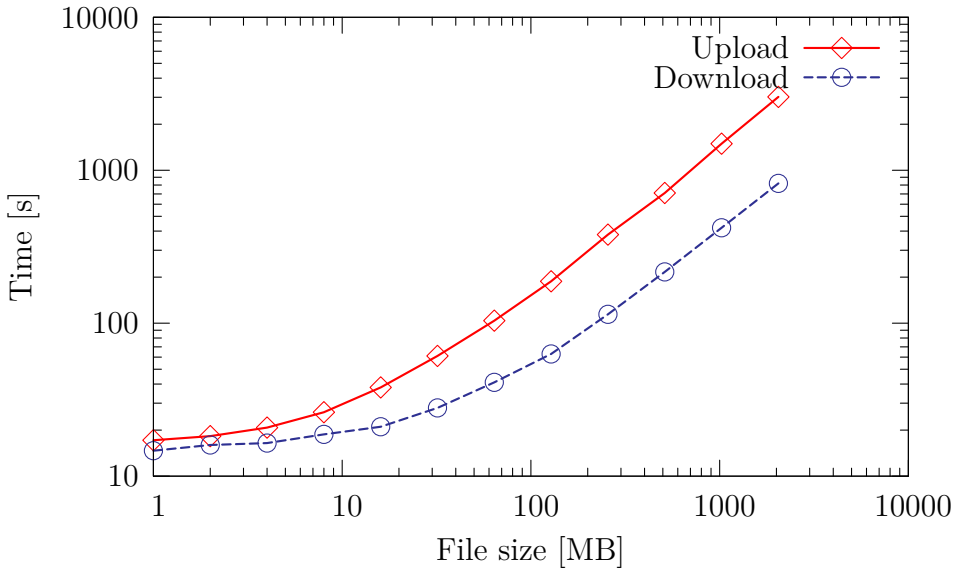


Figure 66: LFCDS Java client library↔LFCDS server performance test: sending and retrieving file from Grid – logarithmic scale

File size [MB]	Upload time [s]	Download time [s]
1	17.169	14.669
2	18.261	15.990
4	20.763	16.460
8	26.184	18.751
16	38.035	21.047
32	61.108	27.929
64	103.873	41.050
128	187.853	62.932
256	379.492	114.283
512	709.921	216.484
1024	1491.528	420.444
2048	3016.697	820.362

Table 11: LFCDS Java client library↔LFCDS server performance test

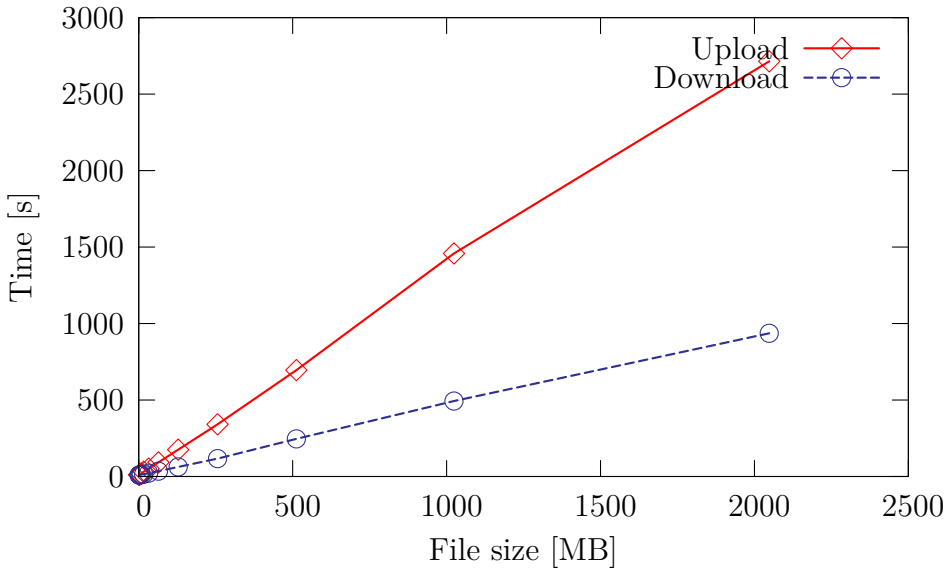


Figure 67: GScript LFC connector↔LFCDS server performance test: sending and retrieving file from Grid – linear scale

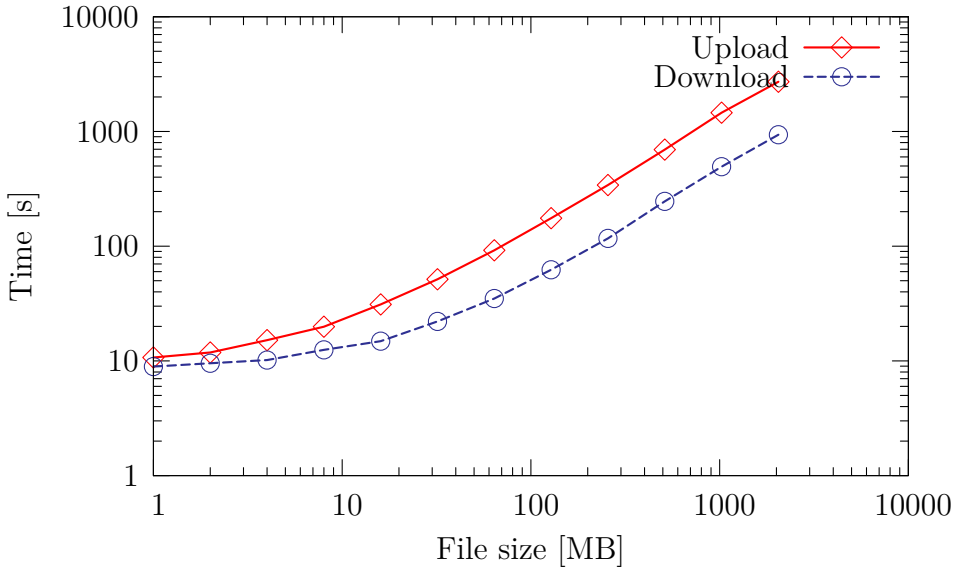


Figure 68: GScript LFC connector↔LFCDS server performance test: sending and retrieving file from Grid – logarithmic scale

File size [MB]	Upload time [s]	Download time [s]
1	10.705	8.925
2	11.842	9.537
4	15.173	10.170
8	19.859	12.483
16	31.093	14.865
32	51.466	22.100
64	92.058	34.960
128	175.523	62.315
256	341.087	116.957
512	695.070	245.934
1024	1458.043	493.427
2048	2714.133	936.395

Table 12: GScript LFC connector↔LFCDS server performance test

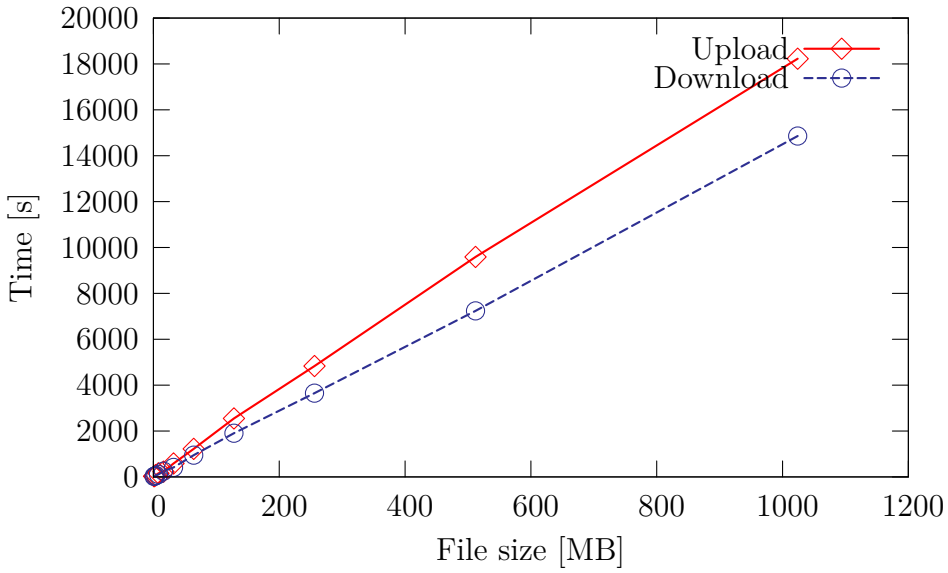


Figure 69: GScript LFC connector↔LFCDS server performance test over WAN: sending and retrieving file from Grid – linear scale

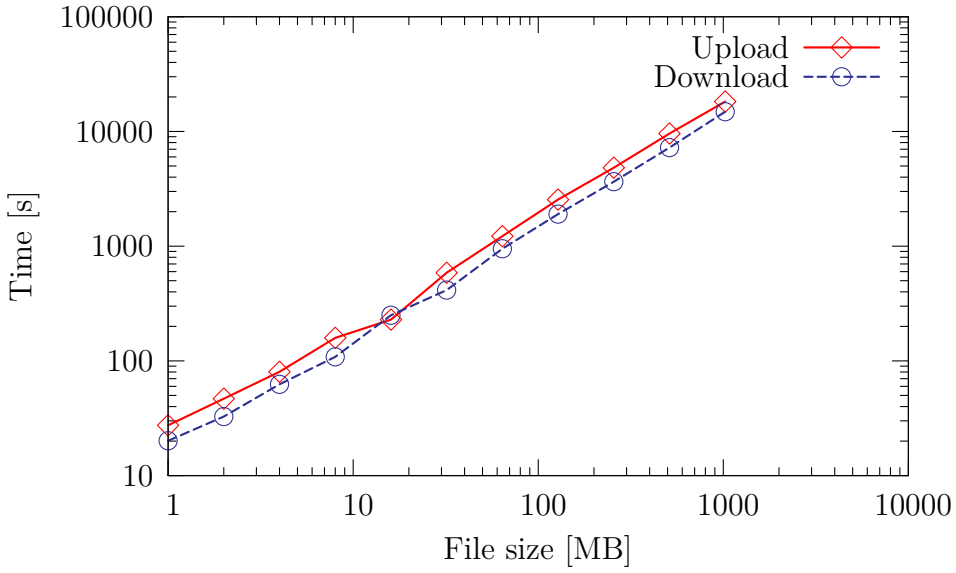


Figure 70: GScript LFC connector↔LFCDS server performance test over WAN: sending and retrieving file from Grid – logarithmic scale

File size [MB]	Upload time [s]	Download time [s]
1	27.451	20.121
2	46.827	32.728
4	80.361	62.482
8	158.904	108.517
16	229.096	250.209
32	586.697	414.049
64	1222.293	949.451
128	2549.077	1904.185
256	4831.703	3650.851
512	9588.798	7237.884
1024	18226.778	14857.206

Table 13: GScript LFC connector↔LFCDS server performance test over WAN

10 Conclusions

10.1 Summary

Nearing to the end of this dissertation, it is noteworthy that goals delineated in chapters 6 and 7 have been successfully achieved and that a significant level of usability has been attained. Tests shown that file upload and download time depends linearly on file size, with scalability up to 2Gb and probably more, although larger file uploads and downloads have not been tested. In addition, validation tests supplied with LFC DS product may help system administrators validate their installation of LFC DS software detecting problems early, before their installation is deployed into production.

LFC DS adds high value to GridSpace Engine allowing for comfortable and efficient access to Grid data sources, eliminating the burden of managing various technology dependent information and automatically managing user credentials. At the moment, LFC DS software is being integrated by ChemPo computational chemistry team into their in-silico experiments utilizing Gaussian software package. The role of LFC DS in this project is to enable searching Gaussian catalogue, processing of Grid files that are results of experiments, and downloading them in order to be visualized in GridSpace environment. More applications among the scientific community are anticipated since LFC DS software has shown to be efficient and reliable while at the same time not compromising simplicity; LFC DS promises to make scientific work more productive by helping researchers focus on real scientific problems, not the technology they use.

10.2 Future work

Future extensions of LFC DS and DAC2 layer should address expressiveness of security polices, in order to make the software suitable for larger collaborations as the current "all or nothing" security policy is limited only to small groups, probably up to 10 persons as already mentioned in chapter 7. In addition, providing Web Service API could be beneficial to projects written in languages other than Java, since current communication mechanism is available to Java platform exclusively. Although performance is satisfactory, some scalability and performance improvements may also be pursued.

Another feature, may not be completely necessary, but fascinating in terms of functionalities, would be a provision of pseudo memory-mapped files (abbreviated *mmap*). A native memory mapped file feature for local files has already been provided by MMAP Ruby gem, which is only available on UNIX machines. An example of distributed filesystem implementation in which memory-mapped file support has been supplied is IBM General Parallel File System (GPFS). In the case of LFC DS, a pseudo mmap is feasible, since Ruby allows for operator overloading. Such an implementation would use `[ ]` operator to access remote Grid in a similar way as local

memory. Depending on chosen architecture and client→server→Grid interaction mechanism, it could also provide means for simpler construction of parallel applications, communicating by using the same files – in such a case a server would host file chunks or entire file downloads from Grid, while clients would access the file caching its contents in local memory and propagating changes of fragments of this file that are shared by other clients. On file closed by all clients, the file would be propagated back to Grid storage. Parallel applications that could take advantage of this technique would be, for instance, cellular automata and differential equations solvers communicating boundary data to each other. Probably, performance would not be able to compete with HPC machines, but with careful design the solution could scale to very large files. Another application of such functionality would be database management systems that access data files mostly in record manner and frequently use `mmap` function if it is provided by operating system.

Stand on the shoulders of giants

11 References

[1] L. Abadie, P. Badino, J.-P. Baud, J. Casey, A. Frohner, G. Grosdidier, S. Lemaitre, G. Mccance, R. Mollon, K. Nienartowicz, D. Smith, and P. Tedesco. Grid-Enabled Standards-based Data Management. In *Mass Storage Systems and Technologies, 2007. MSST 2007. 24th IEEE Conference on*, pages 60–71, Sept. 2007. doi: 10.1109/MSST.2007.4367964.

[2] W. Alda, M. Białoskórski, R. Górecki, and J. Rybicki. Grid Approach to Heat Transfer Simulation in Atomistic-continuum Model. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[3] Carlos de Alfonso, Miguel Caballer, José V. Carrión, and Vicente Hernández. DFSgc: Distributed File System for Multipurpose Grid Applications and Cloud Computing. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'08, October 2008*, Krakow, Poland, 2008. ACC-Cyfronet AGH.

[4] William Allcock, John Bresnahan, Rajkumar Kettimuthu, Michael Link, Catalin Dumitrescu, Ioan Raicu, and Ian Foster. The Globus Striped GridFTP Framework and Server. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 54, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 1-59593-061-2. doi: http://dx.doi.org/10.1109/SC.2005.72.

[5] I. Altintas, E. Jaeger, Kai Lin, B. Ludaescher, and A. Memon. A Web service composition and deployment framework for scientific workflows. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 814–815, July 2004. doi: 10.1109/ICWS.2004.1314956.

[6] B. Amann, B. Elser, Y. Houri, and T. Fuhrmann. IgorFs: A Distributed P2P File System. In *Peer-to-Peer Computing , 2008. P2P '08. Eighth International Conference on*, pages 77–78, Sept. 2008. doi: 10.1109/P2P.2008.19.

[7] David P. Anderson. BOINC: A System for Public-Resource Computing and Storage. In *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2256-4. doi: http://dx.doi.org/10.1109/GRID.2004.14.

[8] David P. Anderson, Eric Korpela, and Rom Walton. High-Performance Task Distribution for Volunteer Computing. In *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, pages 196–203, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2448-6. doi: ttp://dx.doi.org/10.1109/E-SCIENCE.2005.51.

[9] J. Andreeva, A. Anjum, T. Barrass, D. Bonacorsi, J. Bunn, P. Capiluppi, M. Corvo, N. Darmenov, N. DeFilippis, F. Donno, G. Donvito, G. Eulisse, A. Fanfani, F. Fanzago, A. Filine, C. Grandi, J.M. Hernandez, V. Innocente, A. Jan, S. Lacaprara, I. Legrand, S. Metson, H. Newman, D. Newbold, A. Pierro, L. Silvestris, C. Steenberg, H. Stockinger, L. Taylor, M. Thomas, L. Tuura, T. Wildish, and F. VanLingen. Distributed Computing Grid Experiences in CMS. *Nuclear Science, IEEE Transactions on*, 52(4):884–890, Aug. 2005. ISSN 0018-9499. doi: 10.1109/TNS.2005.852755.

[10] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, N.P.C. Hong, B. Collins, N. Hardman, A.C. Hume, A. Knox, M. Jackson, et al. The Design and Implementation of Grid Database Services in OGSA-DAI. *Concurrency and Computation: Practice & Experience*, 17(2):357–376, 2005.

[11] K. Appel and W. Haken. A proof of the four color theorem. *Discrete Math*, 16(2):179–180, 1976.

[12] K. Appel and W. Haken. The solution of the four-color-map problem. *Scientific American*, 237(4):108–121, 1977.

[13] Owen Appleton and Diter Kranzlmüller. EGEE - Status and Future of the World's Largest Multi-Science Grid. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'05, November 2005*, Krakow, Poland, 2005. ACC-Cyfronet AGH.

[14] Athanasia Asiki, Katerina Doka, Ioannis Konstantinou, Antonis Zissimos, and Nectarios Koziris. A Distributed Architecture for Multi-Dimensional Indexing and Data Retrieval in Grid Environments. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[15] Athanasia Asiki, Katerina Doka, Ioannis Konstantinou, Antonis Zissimos, Dimitrios Tsoumakos, Nectarios Koziris, and Panayiotis Tsanakas. A grid middleware for data management exploiting peer-to-peer techniques. *Future Gener. Comput. Syst.*, 25(4):426–435, 2009. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2008.09.005.

[16] Matthias Assel and Onur Kalyoncu. Dynamic Access Control Management for Distributed Biomedical Data Resources. In Paul Cunningham and Miriam Cunningham, editors,

eChallenges e-2008 Conference, Collaboration and the Knowledge Economy: Issues, Applications, Case Studies, pages 1592–1599. IOS Press, October 2008.

[17] Matthias Assel, Bettina Krammer, and Aenne Loehden. Management and Access of Biomedical Data in a Grid Environment. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[18] Matthias Assel, Bettina Krammer, and Aenne Loehden. Data Access and Virtualization within ViroLab. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[19] Matthias Assel, Onur Kalyoncu, and Yi Pan. Approaching Fine-grain Access Control for Distributed Biomedical Databases within Virtual Environments. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'08, October 2008*, Krakow, Poland, 2008. ACC-Cyfronet AGH.

[20] Matthias Assel, Piotr Nowakowski, and Marian Bubak. Integrating and Accessing Medical Data Resources within the ViroLab Virtual Laboratory. In *ICCS '08: Proceedings of the 8th international conference on Computational Science, Part III*, pages 90–99, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-69388-8. doi: http://dx.doi.org/10.1007/978-3-540-69389-5_12.

[21] Matthias Assel, David van de Vijver, Pieter Libin, Kristof Theys, Daniel Harężlak, Breanndann O Nuallain, Piotr Nowakowski, Marian Bubak, Anne-Mieke Vandamme, Stijn Imbrechts, Raphael Sangeda, Tao Jiang, Dineke Frentz, and Peter Sloot. A Collaborative Environment Allowing Clinical Investigations on Integrated Biomedical Databases. In Tony Solomonides, Martin Hofmann-Apitius, Mathias Freudigmann, Sebastian Claudius Semler, Yannick Legré, and Mary Kratz, editors, *Proceedings of HealthGrid 2009, Studies in Health Technology and Informatics*, volume 147, pages 51–61. IOS Press, 2009. doi: 10.3233/978-1-60750-027-8-51.

[22] J. Astalos, Ł. Flis, M. Radecki, and W. Ziajka. Performance Improvements to BDII - Grid Information Service in EGEE. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[23] J. Austin, R. Davis, M. Fletcher, T. Jackson, M. Jessop, B. Liang, and A. Pasley. DAME: Searching Large Data Sets Within a Grid-Enabled Engineering Application. *Proceedings of the IEEE*, 93(3):496–509, March 2005. ISSN 0018-9219. doi: 10.1109/JPROC.2004.842746.

[24] Zoltán Balaton, Gabor Gombás, Péter Kacsuk, Adam Kornafeld, József Kovács, Csaba Attila Marosi, Gabor Vida, Norbert Podhorszki, and Tamás Kiss. SZTAKI Desktop Grid: a Modular and Scalable Way of Building Large Computing Grids. In *IPDPS*, pages 1–8. IEEE, 2007.

[25] Bartosz Baliś, Marian Bubak, Michał Pelczar, and Jakub Wach. Provenance Querying for End-Users: A Drug Resistance Case Study. In *ICCS '08: Proceedings of the 8th international conference on Computational Science, Part III*, pages 80–89, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-69388-8. doi: http://dx.doi.org/10.1007/978-3-540-69389-5_11.

[26] Bartosz Baliś, Marian Bubak, and Michał Pelczar. From Monitoring Data to Experiment Information - Monitoring of Grid Scientific Workflows. In *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, pages 77–84, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3064-8. doi: http://dx.doi.org/10.1109/E-SCIENCE.2007.36.

[27] Bartosz Baliś, Marian Bubak, Michał Pelczar, and Jakub Wach. Provenance Tracking and Querying in ViroLab. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[28] Bartosz Baliś, Marian Bubak, and Jakub Wach. User-Oriented Querying over Repositories of Data and Provenance. In *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, pages 187–194, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3064-8. doi: http://dx.doi.org/10.1109/E-SCIENCE.2007.81.

[29] Bartosz Baliś, Marian Bubak, Michał Pelczar, and Jakub Wach. Provenance Tracking and Querying in the ViroLab Virtual Laboratory. In *CCGRID '08: Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid*, pages 675–680, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3156-4. doi: http://dx.doi.org/10.1109/CCGRID.2008.83.

[30] J. Bart and A. Weisbecker. Services in Fraunhofer Enterprise Grids. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[31] Bartosz Kryza and Łukasz Dutka and Renata Słota and Jacek Kitowski. Supporting Knowledge-based Dynamic Virtual Organizations with Contracts. In Paul Cunningham and Miriam Cunningham, editors, *Expanding the Knowledge Economy: Issues, Applications, Case Studies*, Amsterdam, The Netherlands, 2007. IOS Press.

[32] Tomasz Bartyński. Remote execution of delegated operations with support for automatic selection among multiple communication protocols. Master's thesis, AGH University of Science and Technology in Krakow, Poland, 2008.

[33] Tomasz Bartyński, Maciej Malawski, and Marian Bubak. Invocation of Grid Operations in the ViroLab Virtual Laboratory. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[34] Tomasz Bartyński, Maciej Malawski, Tomasz Gubała, and Marian Bubak. Universal grid client: Grid operation invoker. In Roman Wyrzykowski, editor, *Parallel Processing and Applied Mathematics, 7th International Conference, PPAM 2007, Gdansk, Poland, September 2007, Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2007.

[35] Jean-Philippe Baud, James Casey, Sophie Lemaitre, Caitriana Nicholson, David Smith, and Graeme Stewart. LCG Data Management: From EDG to EGEE . In *UK eScience All Hands Meeting Proceedings, Nottingham, UK*, 2005.

[36] K. Benedyczak, A. Nowiński, K. S. Nowiński, and P. Bała. Interactive Visualization Using the UNICORE Grid Middleware. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[37] D. Bernholdt, S. Bharathi, D. Brown, K. Chanchio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, P. Fox, et al. The Earth System Grid: Supporting the Next Generation of Climate Modeling Research. *Proceedings of the IEEE*, 93(3):485–495, 2005.

[38] I. Bird and R.W.L. Jones. LHC computing grid: Technical design report. Technical report, CERN. Geneva. LHC Experiments Committee; LHCC, 2005.

[39] Christophe Blanchet, Alexis Michon, Krystyna Zakrzewska, and Richard Lavery. Grid Solving a Bioinformatics Challenge: a First Step to Anchoring the Nucleosome. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[40] P. Brezany, I. Janciak, A. Wöhrer, and A M. Tjoa. GridMiner: A Framework for Knowledge Discovery on the Grid – from Vision to Design and Implementation. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[41] P. Brezany, I. Janciak, and A. M. Tjoa. Data Mining on the Grid: Perspective from the GridMiner Experience. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors,

Proceedings of Cracow Grid Workshop - CGW'05, November 2005, Krakow, Poland, 2005. ACC-Cyfronet AGH.

[42] P. Brezany, I. Janciak, and A. Min Tjoa. GridMiner: a Fundamental Infrastructure for Building Intelligent Grid Systems. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 150–156, Sept. 2005. doi: 10.1109/WI.2005.68.

[43] Marian Bubak, Tomasz Gubała, Marek Kasztelnik, Maciej Malawski, Piotr Nowakowski, and P.M.A. Sloot. Collaborative virtual laboratory for e-health. In P. Cunningham and M. Cunningham, editors, *Expanding the Knowledge Economy: Issues, Applications, Case Studies, eChallenges e-2007 Conference Proceedings*, pages 537–544, Amsterdam, 2007. IOS Press. ISBN 978-1-58603-801-4. URL http://www.science.uva.nl/research/scs/papers/archive/Bubak2007a.pdf.

[44] Marian Bubak, Daniel Harężlak, Piotr Nowakowski, Tomasz Gubała, and Maciej Malawski. Appea: A Framework for the Design and Development of Business Applications on the Grid. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[45] Marian Bubak, Daniel Harężlak, Piotr Nowakowski, Tomasz Gubała, and Maciej Malawski. Appea: A Platform for Developments and Execution of Grid Applications. In P. Cunningham and M. Cunningham, editors, *Expanding the Knowledge Economy: Issues, Applications, Case Studies, eChallenges e-2007 Conference Proceedings*, pages 123–130, Amsterdam, 2007. IOS Press. ISBN 978-1-58603-801-4.

[46] Marian Bubak, Tomasz Gubała, Maciej Malawski, Bartosz Baliś, Włodzimierz Funika, Tomasz Bartyński, Eryk Ciepiela, Daniel Harężlak, Marek Kasztelnik, Joanna Kocot, Dariusz Król, Piotr Nowakowski, Michał Pelczar, Jakub Wach, Matthias Assel, and Alfredo Tirado-Ramos. Virtual Laboratory for Development and Execution of Biomedical Collaborative Applications. In *CBMS '08: Proceedings of the 2008 21st IEEE International Symposium on Computer-Based Medical Systems*, pages 373–378, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3165-6. doi: http://dx.doi.org/10.1109/CBMS.2008.47.

[47] Stephen Burke, Simone Campana, Elisa Lanciotti, Patricia Méndez Lorenzo, Vincenzo Miccio, Christopher Nater, Roberto Santinelli, and Andrea Sciabà. gLite 3 User Guide. *Manual Series, Worldwide LHC Computing Grid*, 2009.

[48] J. Cala, L. Czekierda, M. Nowak, and K. Zieliński. The Practical Experiences with Deployment of Advanced Medical Teleconsultation System over Public IT Infrastructure. In

Computer-Based Medical Systems, 2008. CBMS '08. 21st IEEE International Symposium on, pages 349–354, June 2008. doi: 10.1109/CBMS.2008.130.

[49] D. Cameron, J. Casey, L. Guy, P. Kunszt, S. Lemaitre, G. McCance, H. Stockinger, K. Stockinger, G. Andronico, W. Bell, et al. Replica management in the european datagrid project. *Journal of Grid computing*, 2(4):341–351, 2004.

[50] D. Caromel, C. Delbe, A. Di Costanzo, and M. Leyton. ProActive: an integrated platform for programming and running applications on grids and P2P systems. *Computational Methods in Science and Technology*, 12(1):69–77, 2006.

[51] H. Casanova, G. Obertelli, F. Berman, and R. Wolski. The AppLeS Parameter Sweep Template: User-level middleware for the Grid\ m {1}. *Scientific Programming*, 8(3): 111–126, 2000.

[52] H. Casanova, F. Berman, T. Bartol, E. Gokcay, T. Sejnowski, A. Birnbaum, J. Dongarra, M. Miller, M. Ellisman, M. Faerman, et al. The Virtual Instrument: Support for Grid-Enabled Mcell Simulations. *International Journal of High Performance Computing Applications*, 18(1):3, 2004.

[53] Alfieri Cecchini, R. Alfieri, R. Cecchini, V. Ciaschini, Á. Frohner, A. Gianoli, K. Lőrentey, and F. Spataro. VOMS, an Authorization System for Virtual Organizations. In *In Proceedings of the 1st European Across Grids Conference, Santiago de Compostela*, pages 13–14, 2003.

[54] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney. Giggle: A Framework for Constructing Scalable Replica Location Services. In *Supercomputing, ACM/IEEE 2002 Conference*, pages 58–58, Nov. 2002. doi: 10.1109/SC.2002.10024.

[55] A. A. Chien. Architecture of a commercial enterprise desktop Grid: the Entropia system. *Grid Computing: Making the Global Infrastructure a Reality*, pages 337–350, 2003.

[56] David Churches, Gabor Gombas, Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor, and Ian Wang. Programming scientific and distributed workflow with Triana services: Research Articles. *Concurr. Comput.: Pract. Exper.*, 18 (10):1021–1037, 2006. ISSN 1532-0626. doi: http://dx.doi.org/10.1002/cpe.v18:10.

[57] Eryk Ciepiela. Monitoring of Component-Based Applications. Master's thesis, AGH University of Science and Technology in Krakow, Poland, 2007.

[58] Eryk Ciepiela, Joanna Kocot, Tomasz Gubała, Maciej Malawski, Marek Kasztelnik, and Marian Bubak. GridSpace Engine of the ViroLab Virtual Laboratory. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[59] M. Ciglan, B. Simo, M. Maliska, P. Slizik, and L. Hluchy. Grid Virtual Directory System (VDS) – User Centric Approach to Data Management in Medigrid Project. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'05, November 2005*, Krakow, Poland, 2005. ACC-Cyfronet AGH.

[60] ACC Cyfronet. DAC2 GForge site. https://gforge.cyfronet.pl/projects/dac2/, July 2009.

[61] ACC Cyfronet. GridSpace. http://gs.cyfronet.pl/, June 2009.

[62] ACC Cyfronet. GSEngine User Manual for version 0.8.x. http://virolab.cyfronet.pl/trac/vlruntime/wiki/GSEngineUserManual-0.8, June 2009.

[63] ACC Cyfronet. ViroLab Glossary. http://virolab.cyfronet.pl/trac/vlvl/wiki/Glossary, July 2009.

[64] A. T. Das and B. Berkhout. Efficient extension of a misaligned tRNA-primer during replication of the HIV-1 retrovirus. *Nucleic Acids Res.*, 23:1319–1326, Apr 1995.

[65] D. De Roure, C. Goble, and R. Stevens. Designing the myExperiment Virtual Research Environment for the Social Sharing of Workflows. In *e-Science and Grid Computing, IEEE International Conference on*, pages 603–610, Dec. 2007. doi: 10.1109/E-SCIENCE.2007.29.

[66] D. De Roure, C. Goble, J. Bhagat, D. Cruickshank, A. Goderis, D. Michaelides, and D. Newman. myExperiment: Defining the Social Virtual Research Environment. In *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, pages 182–189, Dec. 2008. doi: 10.1109/eScience.2008.86.

[67] David De Roure, Carole Goble, and Robert Stevens. The design and realisation of the myexperiment virtual research environment for social sharing of workflows. *Future Generation Computer Systems*, 25(5):561–567, May 2009. ISSN 0167739X. doi: 10.1016/j.future.2008.06.010.

[68] T. Dimitrakos, M. Wilson, and S. Ristol. TrustCoM-A Trust and Contract Management Framework enabling Secure Collaborations in Dynamic Virtual Organisations. *ERCIM News*, 59:59–60, 2004.

[69] M. Dolenc, V. Stankovski, and Z. Turk. InteliGrid Project: A Vision of Engineering on the Grid. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[70] M. Dolenc, Z. Turk, P. Katranuschkov, K. Kurowski, and M Hannus. Towards Grid Enabled Engineering Collaboration Environment. *Proceedings of the Tenth International Conference on Civil, Structural and Environmental Engineering Computing, B.H.V. Topping (Editor), Civil-Comp Press*, 2005.

[71] M. Dolenc, K. Kurowski, M. Kulczewski, and A. Gehre. InteliGrid Document Management System: an Overview. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[72] S. Dorward, R. Pike, D.L. Presotto, D. Ritchie, H. Trickey, and P. Winterbottom. Inferno. In *Compcon '97. Proceedings, IEEE*, pages 241–244, Feb 1997. doi: 10.1109/CMPCON.1997.584718.

[73] K.K. Droegemeier, D. Gannon, D. Reed, B. Plale, J. Alameda, T. Baltzer, K. Brewster, R. Clark, B. Domenico, S. Graves, E. Joseph, D. Murray, R. Ramachandran, M. Ramamurthy, L. Ramakrishnan, J.A. Rushing, D. Weber, R. Wilhelmson, A. Wilson, M. Xue, and S. Yalda. Service-oriented environments for dynamically interacting with mesoscale weather. *Computing in Science & Engineering*, 7(6):12–29, Nov.-Dec. 2005. ISSN 1521-9615. doi: 10.1109/MCSE.2005.124.

[74] G. Duckeck and Roger W. L. Jones. ATLAS computing: Technical design report. Technical report, CERN. Geneva. LHC Experiments Committee; LHCC, 2005.

[75] L. Dusseault. RFC 4918: HTTP Extensions for Web Distributed Authoring and Versioning 11 (WebDAV). Technical report, RFC, IETF, June 2007.

[76] L. Dutka, K. Korcyl, K. Zieliński, J. Kitowski, R. Słota, W. Funika, K. Bałos, L. Skital, and B. Kryza. Interactive European Grid Environment for HEP Application with Real Time Requirements. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[77] P. Dóbé, R. Kápolnai, and I. Szeberényi. Saleve: Supporting the Deployment of Parameter Study Tasks in the Grid. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[78] M. Eigen. Error catastrophe and antiviral strategy. *Proc. Natl. Acad. Sci. U.S.A.*, 99: 13374–13376, Oct 2002.

[79] J. Falkner, , and A. Weisbecker. Integration of Applications in MediGRID. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[80] Zoltan Farkas, Robert Lovas, and Peter Kacsuk. CancerGrid: Enterprise Desktop Grid Solution with Workflow Support for Anti-Cancer Drug Design. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[81] Zoltan Farkas, Robert Lovas, and Peter Kacsuk. CancerGrid: Enterprise Desktop Grid Solution with Workflow Support for Anti-Cancer Drug Design. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[82] G. Fedak, C. Germain, V. Neri, and F. Cappello. XtremWeb: a generic global computing system. In *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, pages 582–587, 2001. doi: 10.1109/CCGRID.2001.923246.

[83] Laurence Field. Berkeley Database Information Index V5. https://twiki.cern.ch/twiki//bin/view/EGEE/BDII, July 2009.

[84] Travis Fischer, John Hughes, and Andy van Dam. Milton. Master's thesis, Brown University, Providence, R.I., 2009.

[85] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A Directory Service for Configuring High-performance Distributed Computations. In *Proceedings of the 6th IEEE Symposium on High Performance Distributed Computing*, pages 365–375. IEEE Computer Society Press, 1997.

[86] David Flanagan and Yukihiro Matsumoto. *The Ruby Programming Language*. O'Reilly, 2008. ISBN 9780596516178.

[87] I. Foster. What is the grid? a three point checklist. *GRID today*, 1(6):22–25, 2002.

[88] I. Foster. Globus toolkit version 4: Software for service-oriented systems. *Journal of Computer Science and Technology*, 21(4):513–520, 2006.

[89] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of High Performance Computing Applications*, 11(2):115, 1997.

[90] I. Foster and C. Kesselman. Knowledge Integration: In Silico Experiments in Bioinformatics. In *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[91] I. Foster, C. Kesselman, et al. The grid: blueprint for a future computing infrastructure, 1999.

[92] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, June 2002.

[93] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, et al. The open grid services architecture. *The Grid2: Blueprint for a New Computing Infrastructure*, pages 215–257, 2004.

[94] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, et al. The open grid services architecture, version 1.0. In *Global Grid Forum*, volume 29, 2005.

[95] Ákos Frohner on behalf of the Grid DM Team. Medical Data Management. In *CERN - JRA1 All Hands meeting*, 2007. Presentation slides.

[96] Włodzimierz Funika and Piotr Pęgiel. GScript Editor as Part of the ViroLab Presentation Layer. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[97] Włodzimierz Funika, Daniel Harężlak, Dariusz Król, Piotr Pęgiel, and Marian Bubak. User Interfaces of the Virolab Virtual Laboratory. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[98] F. Gagliardi and M.-E. Begin. EGEE - providing a production quality grid for e-science. In *LGDI '05: Proceedings of the 2005 IEEE International Symposium on Mass Storage Systems and Technology*, pages 88–92, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7803-9228-0. doi: http://dx.doi.org/10.1109/LGDI.2005.1612472.

[99] S. Ghemawat, H. Gobioff, and S.T. Leung. The Google file system. *ACM SIGOPS Operating Systems Review*, 37(5):29–43, 2003.

[100] Santiago Gonzalez de la Hoz, Luis March Ruiz, and Dietric Liko. Experience with Atlas Distributed Analysis Tools. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[101] Jürgen Göres. Pattern-based information integration in dynamic environments. In *Database Engineering and Application Symposium, 2005. IDEAS 2005. 9th International*, pages 125–134, July 2005. doi: 10.1109/IDEAS.2005.42.

[102] Jürgen Göres. Towards dynamic information integration. *Lecture Notes in Computer Science*, 3836:16, 2005.

[103] Jürgen Göres and Stefan Dessloch. Discovering data sources in a dynamic Grid environment: Research Articles. *Concurr. Comput. : Pract. Exper.*, 19(16):2109–2124, 2007. ISSN 1532-0626. doi: http://dx.doi.org/10.1002/cpe.v19:16.

[104] Open Science Grid. Virtual Data Toolkit. http://vdt.cs.wisc.edu/, July 2009.

[105] GridwiseTech. GridwiseTech in the ViroLab Project. http://www.gridwisetech.com/virolab, 2009.

[106] C. Grimm and M. Pattloch. Use Cases for Authorization in Grid-Middleware. *D-Grid Technical Report, Version*, 1.3, September 2006.

[107] T. Gubała and M. Bubak. GridSpace – Semantic Programming Environment for the Grid. In Roman Wyrzykowski, Jack Dongarra, Norbert Meyer, and Jerzy Wasniewski, editors, *Parallel Processing and Applied Mathematics: 6th International Conference, PPAM 2005 Poznan, Poland, September 11-14, 2005 Revised Selected Papers (Lecture Notes in Computer Science)*, Secaucus, NJ, USA, 2006. Springer-Verlag New York, Inc. ISBN 3540341412.

[108] Tomasz Gubała, Bartosz Baliś, Maciej Malawski, Marek Kasztelnik, Piotr Nowakowski, Matthias Assel, Daniel Harężlak, Tomasz Bartyński, Joanna Kocot, Eryk Ciepiela, Dariusz Krol, Jakub Wach, Michał Pelczar, Wlodzimierz Funika, and Marian Bubak. ViroLab Virtual Laboratory. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[109] Tomasz Gubała, Marek Kasztelnik, Maciej Malawski, and Marian Bubak. Development and execution of collaborative application on the virolab virtual laboratory. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[110] S.M. Hammer, J.J. Eron Jr, P. Reiss, R.T. Schooley, M.A. Thompson, S. Walmsley, P. Cahn, M.A. Fischl, J.M. Gatell, M.S. Hirsch, et al. Antiretroviral treatment of adult HIV infection: 2008 recommendations of the International AIDS Society-USA panel. *Jama*, 300(5):555, 2008.

[111] M. Hardt, N.V. Ruiter, and M. Zapf. Interactive Grid-Access for Ultrasound CT. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[112] K. S. Harris, W. Brabant, S. Styrchak, A. Gall, and R. Daifuku. KP-1212/1461, a nucleoside designed for the treatment of HIV by viral mutagenesis. *Antiviral Res.*, 67: 1–9, Jul 2005.

[113] Tony Hey and Anne Trefethen. The Data Deluge: An e-Science Perspective. *Grid computing-making the global infrastructure a reality*, pages 809–824, 2003.

[114] Rich Hilliard (editor). IEEE Standard for Information Technology—-Systems Design—-Software Design Descriptions. *IEEE STD 1016-2009*, pages c1–40, 2009. doi: 10.1109/IEEESTD.2009.5167255.

[115] A. G. Hoekstra, S. F. Portegies Zwart, M. Bubak, and P. M. A. Sloot. Towards Distributed Petascale Computing. *Arxiv preprint astro-ph/0703485*, 2007.

[116] Stephen J. Huffman (Editor). IEEE Standard Glossary of Computer Networking Terminology. *IEEE Std 610.7-1995*, Jun 1995.

[117] IEEE Standards Board. IEEE Standard Glossary of Computer Applications Terminology. *ANSI/IEEE Std 610.2-1987*, May 1987.

[118] IEEE Standards Board. IEEE Standard Glossary of Data Management Terminology. *IEEE Std 610.5-1990*, Aug 1990.

[119] IEEE Standards Board. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, Dec 1990.

[120] M.A. Inda, A.S.Z. Belloum, M. Roos, D. Vasunin, C. de Laat, L.O. Hertzberger, and T.M. Breit. Interactive Workflows in a Virtual Laboratory for e-Bioscience: The SigWin-Detector Tool for Gene Expression Analysis. In *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, pages 19–19, Dec. 2006. doi: 10.1109/E-SCIENCE.2006.261103.

[121] T. Jackson, J. Austin, M. Fletcher, and M. Jessop. Delivering a grid enabled distributed aircraft maintenance environment (DAME). In *Proceedings of the UK e-Science All Hands Meeting*, 2003.

[122] Tomasz Jadczyk. Bioinformatics Applications in the Virtual Laboratory. Master's thesis, AGH University of Science and Technology in Krakow, Poland, 2009.

[123] Bob Jones. EGEE status and plans. In *HEPiX Spring 2008*, CERN, Geneva, Switzerland, May 2008. Presentation slides.

[124] T. Jones, A. Koniges, and R.K. Yates. Performance of the IBM general parallel file system. In *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, pages 673–681, 2000. doi: 10.1109/IPDPS.2000.846052.

[125] U. Jovanovič, J. Močnik, M. Novak, G. Pipan, and B. Slivnik. Using Ant Colony Optimization for Collaborative (Re)Search in Data Grids. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[126] K. Shechtman and M. Vainstein and M. Bercovier. Matlab on grid: a progress report. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'05, November 2005*, Krakow, Poland, 2005. ACC-Cyfronet AGH.

[127] P. Kacsuk, A. Marosi, J. Kovács, Z. Balaton, G. Gombás, G. Vida, and Á. Kornafeld. SZTAKI Desktop Grid - a Hierarchical Desktop Grid System. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[128] P. Kacsuk, G. Sipos, A. Tóth, Z. Farkas, G. Kecskeméti, and G. Hermann. Defining and Running Parametric Study Workflow Applications by the P-GRADE Portal. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[129] K. Karasavvas, M. Antonioletti, M.P. Atkinson, N.P.C. Hong, T. Sugden, A.C. Hume, M. Jackson, A. Krause, and C. Palansuriya. Introduction to OGSA-DAI Services. *Lecture Notes in Computer Science*, 3458:1–12, 2005.

[130] Gabrielle Allen Kelly, Kelly Davis, Konstantinos N. Dolkas, Nikolaos D. Doulamis, Tom Goodale, Thilo Kielmann, André Merzky, Jarek Nabrzyski, Juliusz Pukacki, Thomas Radke, Michael Russell, John Shalf, and Ian Taylor. Enabling Applications on the Grid – A GridLab Overview. *International Journal of High Performance Computing Applications*, 17:449–466, 2003.

[131] Jacek Kitowski. Structure and Status of National Grid Initiative in Poland. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'08, October 2008*, Krakow, Poland, 2008. ACC-Cyfronet AGH.

[132] Joanna Kocot and Iwona Ryszka. Optimization of Grid Application Execution. Master's thesis, AGH University of Science and Technology in Krakow, Poland, 2007.

[133] Ioannis Konstantinou, Katerina Doka, Athanasia Asiki, Antonis Zissimos, and Nectarios Koziris. Gredia Middleware Architecture. In Marian Bubak, Michał Turała, and Kazi-

mierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[134] D. Koufil and J. Basney. A credential renewal service for long-running jobs. In *Grid Computing, 2005. The 6th IEEE/ACM International Workshop on*, pages 6 pp.–, Nov. 2005. doi: 10.1109/GRID.2005.1542725.

[135] D. Kranzlmüller, H. Rosmanith, P. Heinzlreiter, and M. Polak. Interactive Virtual Reality on the Grid. In *Distributed Simulation and Real-Time Applications, 2004. DS-RT 2004. Eighth IEEE International Symposium on*, pages 152–158, Oct. 2004. doi: 10.1109/DS-RT.2004.25.

[136] Bartosz Kryza, Łukasz Dutka, Renata Słota, Jan Pieczykolan, and Jacek Kitowski. GVOSF: Grid Virtual Organization Semantic Framework. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[137] Bartosz Kryza, Łukasz Dutka, Renata Słota, and Jacek Kitowski. Supporting Management of Dynamic Virtual Organizations in the Grid through Contracts. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[138] P. Kunszt, P. Badino, A. Frohner, G. McCance, K. Nienartowicz, R. Rocha, and D. Rodrigues. Data storage, access and catalogs in gLite. In *LGDI '05: Proceedings of the 2005 IEEE International Symposium on Mass Storage Systems and Technology*, pages 166–170, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7803-9228-0. doi: http://dx.doi.org/10.1109/LGDI.2005.1612487.

[139] M. Lamanna. The LHC computing grid project at CERN. *Nuclear Inst. and Methods in Physics Research, A*, 534(1-2):1–6, 2004.

[140] E. Laure, S. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, et al. Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12(1):33–45, 2006.

[141] P.J. Leach and R. Salz. UUIDs and GUIDs. *IETF draft specification*, 1998.

[142] LITBIO. Laboratory for Interdisciplinary Technologies in Bioinformatics. http://www.litbio.org/, July 2009.

[143] D. Lorenz, P. Buchholz, C. Uebing, W. Walkowiak, and R. Wismüller. Online Steering of HEP Grid Applications. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[144] Z. Luo, J. Zhang, and R.M. Badia. Service Grid for Business Computing. In M. P. Bekakos, G. A. Gravvanis, and H. R. Arabnia, editors, *Grid Technologies Emerging from Distributed Architectures to Virtual Organizations*. WIT Press, 2006.

[145] Grzegorz M. Wójcik and Wiesław A. Kamiński. Liquid State Machines and Large Simulations of Mammalian Visual System. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[146] M.W. Maier, D. Emery, and R. Hilliard. Software architecture: introducing IEEE Standard 1471. *Computer*, 34(4):107–109, Apr 2001. ISSN 0018-9162. doi: 10.1109/2.917550.

[147] M. Malawski, D. Kurzyniec, and V. Sunderam. MOCCA - Towards a Distributed CCA Framework for Metacomputing. In *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Joint Workshop on High-Performance Grid Computing and High-Level Parallel Programming Models - HIPS-HPGC, April 4-8, 2005, Denver, Colorado, USA*, page 174a. IEEE Computer Society Press, 2005.

[148] M. Malawski, T. Szepieniec, M. Kochanczyk, M. Piwowar, and I. Roterman-Konieczna. The Quest for Pharmacology Active Never Born Proteins within the EUChinaGRID Project. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[149] M. Malawski, T. Szepieniec, M. Kochanczyk, M. Piwowar, and I. Roterman. An approach to protein folding on the grid – EUChinaGrid experience. *Bio-Algorithms & Med-Systems – BAMS*, 2007.

[150] Maciej Malawski. *Component-based methodology for programming and running scientific applications on the grid*. PhD thesis, AGH University of Science and Technology in Krakow, Poland, 2008.

[151] Maciej Malawski, Marian Bubak, Michał Placek, Dawid Kurzyniec, and Vaidy Sunderam. Experiments with distributed component computing across Grid boundaries. In *Proceedings of the HPC-GECO/CompFrame workshop in conjunction with HPDC 2006*, pages 109–116, Paris, France, June 2006. URL http://www.icsr.agh.edu.pl/mambo/docman/task,doc_download/gid,17/Itemid,69/.

[152] Maciej Malawski, Joanna Kocot, Eryk Ciepiela, Iwona Ryszka, and Marian Bubak. Optimization of application execution on the virolab virtual laboratory. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[153] Maciej Malawski, Tomasz Gubała, Marek Kasztelnik, Tomasz Bartyński, Marian Bubak, Francoise Baude, and Ludovic Henrio. High-Level Scripting Approach for Building Component-Based Applications on the Grid. In Marco Danelutto, Paraskevi Fragopoulou, and Vladimir Getov, editors, *Making Grids Work*. Springer Publishing Company, Incorporated, 2008.

[154] Maciej Malawski, Tomasz Bartyński, and Marian Bubak. Invocation of operations from script-based Grid applications. In *Future Generation Computer Systems*. Elsevier, 2009. doi: 10.1016/j.future.2009.05.012. In Press, Accepted Manuscript.

[155] Martin Maliska, Branislav Simo, and Ladislav Hluchý. The Workflow Engine for the CROSSGRID Flood Forecasting Application. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[156] Jan Meizner, Maciej Malawski, Eryk Ciepiela, Marek Kasztelnik, Daniel Harężlak, Piotr Nowakowski, Dariusz Król, Tomasz Gubała, Włodzimierz Funika, Marian Bubak, Tomasz Mikołajczyk, Paweł Płaszczak, Krzysztof Wilk, , and Matthias Assel. ViroLab Security and Virtual Organization Infrastructure. In Y. Dou, R. Gruber, and J. Joller, editors, *APPT 2009, Advanced Parallel Processing Technologies 8th International Symposium, Rapperswil, Switzerland, Proceedings, LNCS 5737*, pages 230–245. Springer-Verlag Berlin Heidelberg, August 24-25 2009.

[157] Sun Microsystems. Sun's Network.com Renders Computer-Animated Movie "Big Buck Bunny". http://www.sun.com/aboutsun/pr/2008-06/sunflash.20080602.1.xml, July 2009.

[158] J. Montagnat, D. Jouvenot, C. Pera, A. Frohner, P. Kunszt, B. Koblitz, N. Santos, and C. Loomis. Bridging clinical information systems and grid middleware: a Medical Data Manager. *Studies in health technology and informatics*, 120:14, 2006.

[159] Zofia Mosurska and Kazimierz Wiatr. PL-Grid - Koncepcja budowy ogolnopolskiej infrastruktury Gridowej [PL-Grid - building the Polish grid infrastructure - in Polish]. *Biuletyn Informacyjny Pracowników AGH*, 170, October 2007.

[160] C. Munro, B. Koblitz, N. Santos, and A. Khan. Measurement of the LCG2 and Glite File Catalogue's Performance. *Nuclear Science, IEEE Transactions on*, 53(4):2228–2232, Aug. 2006. ISSN 0018-9499. doi: 10.1109/TNS.2006.877857.

[161] L. A. Napolitano, D. Schmidt, M. B. Gotway, N. Ameli, E. L. Filbert, M. M. Ng, J. L. Clor, L. Epling, E. Sinclair, P. D. Baum, K. Li, M. L. Killian, P. Bacchetti, and J. M. McCune. Growth hormone enhances thymic function in HIV-1-infected adults. *J. Clin. Invest.*, 118:1085–1098, Mar 2008.

[162] David Newman. OAuth – myExperiment. http://wiki.myexperiment.org/index.php/Developer:OAuth, September 2008.

[163] Krzysztof Nienartowicz. gLite FiReMan. In *Very Large Data Bases – VLDB*, 2006. Presentation slides.

[164] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the Grid: MyProxy. In *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, pages 104–111, 2001. doi: 10.1109/HPDC.2001.945181.

[165] Object Management Group. *Unified Modeling Language (OMG UML), Superstructure. V. 2.2*. Object Management Group, 2009.

[166] T. O'Brien, J. Casey, B. Fox, B. Snyder, J. Van Zyl, and E. Redmond. *Maven: The Definitive Guide*. Sonatype, 2008.

[167] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20 (17):3045–3054, 2004. ISSN 1367-4803. doi: http://dx.doi.org/10.1093/bioinformatics/bth361.

[168] M. Okon, D. Kaliszan, M. Lawenda, D. Stoklosa, T. Rajtar, N. Meyer, and M. Stroinski. Virtual Laboratory as a Remote and Interactive Access to the Scientific Instrumentation Embedded in Grid Environment. In *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, pages 124–124, Dec. 2006. doi: 10.1109/E-SCIENCE.2006.261057.

[169] S.D. Olabarriaga, A.J. Nederveen, and B.O. Nuallain. Parameter Sweeps for Functional MRI Research in the "Virtual Laboratory for e-Science" Project. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 685–690, May 2007. doi: 10.1109/CCGRID.2007.82.

[170] T. Olas and R. Wyrzykowski. Method for Mapping FEM Computations onto Cluster Grid Architectures. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[171] Tomasz Olas and Roman Wyrzykowski. Porting Thermomechanical Applicationsto CLUSTERIX Environment. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[172] Richard Olejnik, Bernard Toursel, Marek Tudruj, Eryk Laskowski, and Iyad Alshabani. Optimized Java Computing as an Application for Desktop Grid. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2005. ACC-Cyfronet AGH.

[173] Richard Olejnik, Bernard Toursel, Marek Tudruj, Eryk Laskowski, Iyad Alshabani, and Lukasz Maśko. DG-ADAJ: a Java Computing Platform for Desktop Grid. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'05, November 2005*, Krakow, Poland, 2005. ACC-Cyfronet AGH.

[174] D. Orme and J. Winchester. The Eclipse Visual Editor. Creating Eclipse-based GUI builders. *Dr Dobb's Journal-Software Tools for the Professional Programmer*, pages 73–75, 2006.

[175] Eva Pajorová, Ladislav Hluchý, and Ján Astaloš. 3D Geo-visualization Service for Grid-oriented Applications of Natural Disasters. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[176] Daniel Pasztuhov and Imre Szeberenyi. New Approach to Design UI for Grid Applications. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[177] Michał Pelczar. Recording application executions enriched with domain semantics of computations and data. Master's thesis, AGH University of Science and Technology in Krakow, Poland, 2008.

[178] Jan Pieczykolan, Lukasz Dutka, Krzysztof Korcyl, Tomir Kryza, and Jacek Kitowski. Grid support for A Toroidal LHC ApparatuS (ATLAS). In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[179] Monika Piwowar, Tomasz Szepieniec, Ewa Matczyńska, and Irena Roterman. Identification of "Never Born" Protein Traces in Human Chromosome 1 with Using Grid Environment – Preliminary Analysis. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[180] Monika Piwowar, Tomasz Szepieniec, and Irena Roterman. Massive Identification of Similarities in DNA Materials Organized in Grid Environment. *Bio-Algorithms & Med-Systems – BAMS*, 2007.

[181] Martin Polak, Dieter Kranzmüller, and Jens Volkert. GVid – A Dynamic Grid Videoservice for Advanced Visualization. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[182] Marek Pomocka. System "VirtualRenderer" do Renderowania Filmów Komputerowych Oparty o Technologie Gridowe – ["VirtualRenderer" – a System for Rendering Computer Films based on Grid Technologies]. In Leszek Kurcz and Andrzej Gołdasz, editors, *Sesje Studenckich Kół Naukowych. Materiały XLV Sesji Pionu Hutniczego: streszczenia referatów; program Sesji; informacje o kołach naukowych – [Sessions of Students' Scientific Circles]*, Krakow, Poland, May 2008. AGH University of Science and Technology, Wydawnictwo Studenckiego Towarzystwa Naukowego.

[183] S. Portegies Zwart, S. McMillan, S. Harfst, D. Groen, M. Fujii, B.Ó. Nualláin, E. Glebbeek, D. Heggie, J. Lombardi, P. Hut, et al. A multiphysics and multiscale software environment for modeling astrophysical systems. *New Astronomy*, 14(4):369–378, 2009.

[184] Paweł Płaszczak. Securing highly distributed data collections. http://bigdatamatters.com/bigdatamatters/2009/07/web-applications-security.html, July 2009.

[185] Tomáš Rebok. DiProNN: Distributed Programmable Network Node Architecture. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[186] A. Rodriguez, D. Sulakhe, E. Marland, V. Nefedova, N. Maltsev, M. Wilde, and I. Foster. A Grid-enabled service for high-throughput genome analysis. In *Workshop on Case Studies on Grid Applications*, 2004.

[187] Jan Ruthe, Grzegorz M. Wójcik, Wiesław A. Kamiński, Dorota Stanisławek, Michał Żukowski, and Marek Falski. New System of Parallel and Biologically Realistic Neural Simulation. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[188] K. Rycerz, M. Bubak, M. Malawski, and P. Sloot. A Framework for HLA-based Interactive Simulations on the Grid. *Simulation*, 81(1):67, 2005.

[189] K. Rycerz, M. Bubak, M. Malawski, and P. Sloot. Grid Support for HLA-Based Collaborative Environment for Vascular Reconstruction. In *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, pages 48–48, Dec. 2006. doi: 10.1109/E-SCIENCE.2006.261132.

[190] S. K. Sadiq, D. Wright, S. J. Watson, S. J. Zasada, I. Stoica, and P. V. Coveney. Automated molecular simulation based binding affinity calculator for ligand-bound HIV-1 proteases. *J Chem Inf Model*, 48:1909–1919, Sep 2008.

[191] N. Santos and B. Koblitz. Metadata services on the Grid. *Nuclear Inst. and Methods in Physics Research, A*, 559(1):53–56, 2006.

[192] B. Segal, L. Robertson, F. Gagliardi, and F. Carminati. Grid computing: the European Data Grid Project. In *Nuclear Science Symposium Conference Record, 2000 IEEE*, volume 1, pages 2/1 vol.1–, 2000. doi: 10.1109/NSSMIC.2000.948988.

[193] Sulev Sild, Andre Lomaka, and Uko Maran. OpenMolGRID: QSAR/QSPR Application in Grid Environment. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[194] A. Sim, A. Shoshani, P. Badino, O. Barring, JP Baud, F. Donno, M. Litmaath, T. Perelmutov, D. Petravick, E. Corso, et al. The Storage Resource Manager Interface Specification Version 2.2. In *Open Grid Forum*, 2007.

[195] Branislav Simo, Viera Sipkova, Martin Gazak, and Ladislav Hluchý. Interactive Air Pollution Simulation in int.eu.grid. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[196] Peter M. A. Sloot, Alfredo Tirado-Ramos, Ilkay Altintas, Marian Bubak, and Charles Boucher. From Molecule to Man: Decision Support in Individualized E-Health. *Computer*, 39(11):40–46, 2006. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/MC.2006.380.

[197] Peter M.A. Sloot, Alfredo Tirado-Ramos, Gokhan Ertaylan, Breanndan O Nuallain, D. Van de Vijver, Charles A. Boucher, and Marian Bubak. VIROLAB: a Distributed Decision Support System for Viral Disease Treatment. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[198] PMA Sloot, P.V. Coveney, G. Ertaylan, V. Müller, CA Boucher, and M. Bubak. HIV decision support: from molecule to man. *Philosophical Transactions A*, 367(1898):2691, 2009.

[199] DA Stainforth, T. Aina, C. Christensen, M. Collins, N. Faull, DJ Frame, JA Kettleborough, S. Knight, A. Martin, JM Murphy, et al. Uncertainty in predictions of the climate response to rising levels of greenhouse gases. *Nature*, 433(7024):403–406, 2005.

[200] V. Stankovski, M. Swain, V. Kravtsov, T. Niessen, D. Wegener, M. Rohm, J. Trnkoczy, M. May, J. Franke, A. Schuster, and W. Dubitzky. Digging Deep into the Data Mine with DataMiningGrid. *Internet Computing, IEEE*, 12(6):69–76, Nov.-Dec. 2008. ISSN 1089-7801. doi: 10.1109/MIC.2008.122.

[201] Mariusz Sterzel and Tomasz Szepieniec. Enabling Commercial Chemical Software on EGEE Grid – Gaussian VO. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[202] Mariusz Sterzel, Tomasz Szepieniec, and Daniel Harężlak. Grid Web Portal for Chemists. In *EGEE User Forum*, Catania, Italy, March 2009. Presentation slides.

[203] R. D. Stevens, A. J. Robinson, and C. A. Goble. myGrid: personalised bioinformatics on the information grid. *Bioinformatics*, 19 Suppl 1:i302–304, 2003.

[204] RD Stevens, H.J. Tipney, CJ Wroe, TM Oinn, M. Senger, PW Lord, CA Goble, A. Brass, and M. Tassabehji. Exploring Williams-Beuren Syndrome Using myGrid, 2004.

[205] Graeme A Stewart, David Cameron, Greig A Cowan, and Gavin McCance. Storage and data management in EGEE. In *ACSW '07: Proceedings of the fifth Australasian symposium on ACSW frontiers*, pages 69–77, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc. ISBN 1-920-68285-X.

[206] Ileana Stoica, S Kashif Sadiq, Catherine V Gale, and Peter V Coveney. Virtual Physiological Human research initiative: the future for rational HIV treatment design? *Future HIV Therapy*, 2(5):419–425, 2008. doi: 10.2217/17469600.2.5.419. URL http://www.futuremedicine.com/doi/abs/10.2217/17469600.2.5.419.

[207] Alberto Sánchez Jr., María S. Pérez Jr., Pierre Gueant, and José M. Peña Pilar Herrero. DMGA: A Generic Brokering-Based Data Mining Grid Architecture. In Werner Dubitzky, editor, *Data Mining Techniques in Grid Computing Environments*, pages 201–219. Wiley, 2008. doi: 10.1002/9780470699904.ch12.

[208] D. Talia. The Open Grid Services Architecture: where the grid meets the Web. *Internet Computing, IEEE*, 6(6):67–71, Nov/Dec 2002. ISSN 1089-7801. doi: 10.1109/MIC.2002.1067739.

[209] Andrzej Tarczyński, Tamas Kiss, Gabor Tersztyanszki, Thierry Delaitre, Dongdong Qu, and Stephen Winter. Application of grid computing for designing a class of optimal periodic nonuniform sampling sequences. *Future Gener. Comput. Syst.*, 24(7):763–773, 2008. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2008.02.005.

[210] A. Thandavan, C. Sahin, and V. N. Alexandrov. Experiences with the Globus Toolkit on AIX and Deploying the Large Scale Air Pollution Model as a Grid Service. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'04, December 2004*, Krakow, Poland, 2004. ACC-Cyfronet AGH.

[211] The British Library, Leipzig University Library, St Catherine's Monastery, and The National Library of Russia. Codex Sinaiticus. http://www.codexsinaiticus.org/, July 2009.

[212] The GREDIA Consortium. The GREDIA Project Grid Enabled Access to Rich Media Content. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[213] The ViroLab Consortium. ViroLab, a Virtual Laboratory for Decision Support in Viral Diseases Treatment. http://www.virolab.org/, July 2009.

[214] Dave Thomas, Chad Fowler, and Andy Hunt. *Programming Ruby: The Pragmatic Programmers' Guide, Second Edition*. Pragmatic Bookshelf, October 2004. ISBN 0974514055.

[215] Keith Thomson. RE: [Globus-discuss] "gsiftp" vs. "gridftp". http://www.globus.org/mail_archive/discuss/2003/04/msg00380.html, 2003.

[216] A. Tirado-Ramos. *Collaboratories on the Grid, Collaborative Software Architectures for Interactive Biomedical Applications*. PhD thesis, University of Amsterdam, 2007.

[217] A. Tirado-Ramos, P.M.A. Sloot, and M. Bubak. Grid-based Interactive Decision Support in BioMedicine. *Grid Computing for Bioinformatics and Computational Biology*, page 225, 2007.

[218] Viet D. Tran and Ladislav Hluchý. Application Management in Earth Science. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[219] TrueArt. Products – Plug-ins – VirtualRender. http://www.trueart.pl/?URIType=Directory&URI=Products/Plug-ins/VirtualRender, July 2009.

[220] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X. 509 public key infrastructure (PKI) proxy certificate profile. *RFC3820, June*, 2004.

[221] M. Šterk, I. Leben, E. Milošev, and G. Pipan. "River Soca Project" – Interactive Visualization of Massive Amount of Data with a Grid-based Engine. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[222] A. Uyar, W. Wu, H. Bulut, and G. Fox. Service-oriented architecture for a scalable video-conferencing system. In *Pervasive Services, 2005. ICPS '05. Proceedings. International Conference on*, pages 445–448, July 2005. doi: 10.1109/PERSER.2005.1506564.

[223] Jakub Wach. Collection and Storage of Provenance Data. Master's thesis, AGH University of Science and Technology in Krakow, Poland, 2008.

[224] L. Wang, W. Jie, and H. Zhu. State-of-the arts: workflow management for Grid computing. In M. P. Bekakos, G. A. Gravvanis, and H. R. Arabnia, editors, *Grid Technologies Emerging from Distributed Architectures to Virtual Organizations*. WIT Press, 2006.

[225] R. Watson, S. Maad, , and B. Coghlan. Multiscale Multimodal Visualization on a Grid. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2006. ACC-Cyfronet AGH.

[226] Adianto Wibisono, Zhiming Zhao, Adam Belloum, and Marian Bubak. Towards a Virtual Laboratory for Interactive Parameter Sweep Applications on the Grid. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'07, October 2007*, Krakow, Poland, 2007. ACC-Cyfronet AGH.

[227] M. Widenius, D. Axmark, and P. DuBois. *MySQL reference manual*. O'Reilly & Associates, Inc. Sebastopol, CA, USA, 2002.

[228] Stephen Wolfram. *The Mathematica Book, Fifth Edition*. Wolfram Media, 2003.

[229] J.C. Worsley and J.D. Drake. *Practical PostgreSQL*. O'Reilly Media, Inc., 2002.

[230] C. Wroe, C. Goble, M. Greenwood, P. Lord, S. Miles, J. Papay, T. Payne, and L. Moreau. Automating Experiments Using Semantic Data on a Bioinformatics Grid. *Intelligent Systems, IEEE*, 19(1):48–55, Jan-Feb 2004. ISSN 1541-1672. doi: 10.1109/MIS.2004.1265885.

[231] Cheng Yaodong, Gang Chen, Yongjian Wang, and Shuaijie Wang. Deploying HEP Applications on Multiple Grid Infrastructures. In *Grid and Cooperative Computing, 2008. GCC '08. Seventh International Conference on*, pages 632–641, Oct. 2008. doi: 10.1109/GCC.2008.78.

[232] Cheng Yaodong, Wang Lu, Liu Aigui, and Cheng Gang. Sharing LCG files across different platforms. In *Journal of Physics: Conference Series*, volume 119, page 062024. Institute of Physics Publishing, 2008.

[233] Jia Yu and Rajkumar Buyya. A Taxonomy of Workflow Management Systems for Grid Computing. Technical report, Journal of Grid Computing, 2005.

[234] Jia Yu and Rajkumar Buyya. A Taxonomy of Scientific Workflow Systems For Grid Computing. *SIGMOD Rec.*, 34(3):44–49, 2005. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/1084805.1084814.

[235] Kurt D. Zeilenga. Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. *RFC4510, June*, 2006.

[236] Valerie E. Zelenty (Editor). IEEE Recommended Practice For Software Requirements Specifications. *IEEE Std 830-1998*, Oct 1998.

[237] Zhiming Zhao, S. Booms, A. Belloum, C. de Laat, and B. Hertzberger. VLE-WFBus: A Scientific Workflow Bus for Multi e-Science Domains. In *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, pages 11–11, Dec. 2006. doi: 10.1109/E-SCIENCE.2006.261095.

[238] Zhiming Zhao, A. Belloum, M. Bubak, and B. Hertzberger. Support for Cooperative Experiments in VL-e: From Scientific Workflows to Knowledge Sharing. In *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, pages 329–330, Dec. 2008. doi: 10.1109/eScience.2008.120.

[239] Mikhail Zhizhin, Eric Kihn, Vassily Lyutsarev, Sergei Berezin, Alexey Poyda, Dmitry Mishin, Dmitry Medvedev, and Dmitry Voitsekhovsky. Environmental Scenario Search and Visualization. In *GIS '07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, pages 1–10, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-914-2. doi: http://doi.acm.org/10.1145/1341012.1341047.

A LFC Data Source – User guide

The LFC Data Sources allow you to access EGEE / WLCG storage resources with a simple Ruby API.

A.1 Data access workflow: *registering the data source, storing credentials, using the data source from a script*

The data access workflow is as follows:

1. Register data source in Data Source Registry. Information needed includes
   a Connection details to the LFC Data Source server – a gateway to EGEE/WLCG data resources
   b Addresses and ports of following servers: LCG File Catalog (LFC) server, Berkeley Database Information Index (BDII) and default storage element, which will be used to store new files.
   c Your Virtual Organization name.
2. Optionally, you can store your credentials in the DSR. This will allow you and other users (if you permit) to access the data sources without specifying credentials in the script. In order to make your credentials usable by other users, you must specify your credentials as static. Information needed to enable credential-free access from the script is either:
   a Your grid proxy certificate – note that proxy certificate is usually valid for only one day, so this is a short-term solution. On the other hand, it allows you to use Single Sign On authentication when accessing data sources without the need to store your private key, grid certificate and password in the DSR.
   b Another option is to store your complete credentials in the DSR. These include private key, password to private key, and grid certificate. If your private key is encrypted with a passphrase that you do not want other people to see (e.g. system administrator), you may encrypt your private key with another passphrase of your choice for this purpose:

   ```
   openssl rsa -in userkey.pem -des3 -out userkey.pem.new
   ```

   c The third option is to store only the grid certificate and private key in the DSR, but without the passphrase. In order to use the data source, you will have to provide the password in the constructor of `DACConnector`.
3. Once you have provided information about the data source and credentials (or if someone else has done it for you) you may access the LFC data sources in your GScript files by providing data source handle and optionally credentials.

A.2 DACConnector LFC DS specific constructors

LFC Data Source provides the constructors shown below. They are usually called by the `DACConnector.new` command. Note that every argument passed to the new method is a `String`. In the following examples the `lfc-voce` or `lfc-egee` string will be used as an example of a handle. `ds` will be a local variable holding the reference to the instantiated LFC DS connector.

1-argument constructor You provide only the data source handle. Grid credentials must be stored in the DSR in order to use this constructor. As noted before, the credentials may be yours or other users who declared them static, i.e. permitted other authenticated users to use them. For example:

```
ds = DACConnector.new("lfc-voce")
```

2-argument constructor, *second argument: password* The first parameter is the data source handle. The second is the private key passphrase. This is useful when you choose the option *2.c*, i.e. stored only the private key and grid certificate but did not save the private key passphrase.

```
ds = DACConnector.new("lfc-voce", "your_passphrase")
```

2-argument constructor, *second argument: proxy certificate* You may also provide a String with a proxy certificate as a second argument. The LFC Data Source connector will distinguish passwords from proxy certificates by their length. Anything that is longer than 300 characters will be assumed to be a proxy certificate. Example:

```
ds = DACConnector.new("lfc-egee", IO.read("C:/x509up_u506"))
# IO.read used to load a file into a String
```

or

```
ds = DACConnector.new("lfc-egee ", IO.read("/tmp/x509up_u506"))
```

4-argument constructor: *handle, private key, grid certificate and private key passphrase* You will probably use this constructor if you do not have a valid proxy or credentials stored in the DSR. You may also be interested in this method if you want to override your DSR credentials. Note, that if some or all your credentials are stored in the DSR, the "side effect" of using this method will be storage of a new generated proxy in the DSR. However, if your credentials are not stored in the DSR, the created proxy certificate will not be stored there. All of the arguments are Strings. To easily load contents of a file into a String, you may use the IO.read method shown before as in this example:

```
ds = DACConnector.new("lfc-voce", IO.read("C:/userkey.pem"), \
          IO.read("C:/usercert.pem") "your_password")
```

or

```
ds = DACConnector.new("lfc-voce", \
          IO.read("/home/username/.globus/userkey.pem"), \
          IO.read("/home/username/.globus/usercert.pem"), \
          "your_password")
```

A.3 LFC Data Source methods

At this stage, we have created an instance of the LFC Data Source. Now we can invoke methods that operate on files and LCG File Catalog. A useful point worth noting is that with the LFC data source connector, all paths begin with `/grid/vo_name/`. However, you do not have to provide this prefix in your commands. For instance, this path

`/grid/voce/username/important_project/experiment_data`

would be expressed as follows (there are two possibilities):

`username/important_project/experiment_data`

or

`/username/important_project/experiment_data`

The beginning slash is optional.

The LFC methods are accessible both using the `camelCase` notation and `ruby_notation`. They also have numerous aliases listed here and you are welcome to use whichever name you prefer. Methods may return a `DAC2Exception` if the LFC DS connector detects an error in the invocation parameters. If the LFC DS connector does not see any problems with the parameters, it passes the invocation to *LFC DS client* – a Java library used to connect to the LFC DS Server. If this java client cannot connect to the LFC DS server or receives an exception from the server, it returns this exception to you as an `LfcDsException`. The LFC DS server will throw an exception if for some reason it cannot execute the requested operation, e.g. it cannot retrieve contents of a file. You may prevent some exceptions by checking the existence of files or directories. This checking is not done on the client side as the communication with the LFC catalog, although faster than access to storage elements, is still a noticeable performance hit.

Knowing the constructors used by the LFC DS connector, path specification convention and exceptions that may be thrown, let us move onto the description of methods implemented by LFC DS connector.

LFC DS connector methods

`createDirectory(path)`

`createDirectory(path, child_directory)`

Aliases: `create_directory`

Creates a new directory specified by path (one argument version) or creates a directory of the name `child_directory` in the `parent_directory`. Returns true on success, false otherwise. Examples:

```
# use of an alias
ds.create_directory "some_directory/another_directory"
# use of the beginning slash '/'
ds.createDirectory "/some/lengthy/path/some_directory"
# two argument example
ds.create_directory "some/lengthy/path", "some_directory"
# two argument example with parentheses
ds.create_directory("some/long/path", "some_directory")
```

`delete(path)`

Aliases: `deleteFile` – for backward compatibility with scripts that use `deleteFile`

Deletes file or directory. Returns true on success, false otherwise. Example:

```
ds.delete("some/long/path/some_directory")
ds.delete "some/long/path/some_file"
```

`directory?(path)`

Aliases: `isDirectory, is_directory`

Returns true if the item denoted by the path exists and is a directory; false otherwise.

```
ds.directory?("some/path/file") # would return false
ds.isDirectory "some/path/directory"  # would return true if the directory
    exists
```

`exist?(path)`

Aliases: `exist, exists, exists?`

Returns true if the item represented by the path passed as an argument exists; false otherwise.

`file?(path)`

Aliases: `isFile, is_file`

Returns true if the item indicated by the path passed as an argument exists and is a file; false otherwise.

`getFile(path)`

Aliases: `get_file`

Returns a Java byte array representing the contents of a file. If the file does not exist an exception is thrown. In order to convert the java byte array to string you may use the `String.from_java_bytes` method as in the following example:

```
String.from_java_bytes \
    ds.getFile("some/directory/test_lfcds/test_file1.txt")
```

Although streaming is used to download the contents of the file, creation of large byte array objects may cause `OutOfMemory` errors. If you are accessing files of several hundreds of megabytes, you are advised to use the `openFile` method, which is, on the other hand, very convenient as it returns Ruby IO object.

Note that changes in the array returned will not be reflected in the file unless you save them using `storeFile` method (although such a functionality called memory mapped file might be very useful).

`getSize(path)`

Aliases: `size?, size, get_size`

Returns the size of the file represented by path – this information is retrieved from the LFC catalog. Examples:

```
ds.size? "/some/path/some_big_file.dat"
ds.getSize("some/long/path/some/other/file.mov")
```

`listFiles(path)`

Aliases: `list_files`

Returns a list of `LfcDsItem` objects. Each of these items respond to `is_directory` (or `isDirectory`) method which allows you to get information whether the item represents a directory or a file. In addition, each of the items responds to `getName` (or `get_name`) method which returns the base name of a file, i.e. without the directory part. You may iterate through the returned list in order to list available files in a directory. Example:

```
l=ds.listFiles("/foo/bar/test_lfcds/")
l.each do |item|
  puts item.get_name + " is a " + \
     if item.is_directory then "directory" else "file" end
end
```

The execution of the script above might yield the following results:

```
Test_file1.txt is a file
Test_dir is a directory
Test_file2.txt is a file
```

`openFile(path, mode) { optional block }`

Aliases: `open`, `open_file`

The mode parameter can be one of the following values:

- `:r, :read, "r", "read"` – indicate that the file will be opened in a *read* mode
- `:w, :w, "w", "write"` – means that the file will be opened in a *write* mode

Neither the LFC DS connector nor the LFC DS server do not support *read-write* mode. You must choose whether you would prefer *read* from a file or *write* to a file. If the file denoted by the path already exists, an exception is thrown. You must delete the previous version of a file before you attempt to write a new version.

The path indicates the location of file to open. In the case of opening file to be read, the file you requested is downloaded from Grid into the LFC DS server (not the same server which runs GSEngine, but another that could be installed on an alternative machine). This method returns a remote input stream for this file, which is converted to a Ruby IO stream by the LFC connector. After you finish reading the file, you release it by invoking the close method of the returned Ruby stream. The close method causes the temporary file stored in the temporary directory on LFC DS server to be deleted. If you forget to do this, it will be removed when LFC DS is restarted some time in the future. If you use optional block, the file will be closed for you automatically, when the block ends; so it may be preferred option to use the openFile method with a block. Example:

```
# the openFile method used with the alias "open" and a block argument
ds.open("/foo/bar/test_lfcds/test_file3.txt", "r") do |file|
  file.each {|line| puts line}
end
# example of a file opened and closed explicitly
f = ds.open("/foo/bar/test_lfcds/test_file2.txt", :read)
f.each {|line| puts line}
f.close
```

The file is streamed to you by the LFC DS server, after downloading from the Grid so you should be able to access very large files using the methods described above. Nevertheless, the machine on which LFC DS server runs must have enough storage in order to hold the file in a temporary directory.

As regards writing a file, the commands are similar. The difference is that, as opposed to a file opened for reading, a file opened for writing is first streamed to LFC DS server. The LFC DS server then writes the stream to a file temporary directory. When you close remote stream, the file is sent to the Grid and registered in the LFC catalog. If for some reason, the file cannot be stored or registered in LFC, an `LfcDsException` is thrown. A typical situation when this may occur is when you attempt to write to a file that is already

registered in the LFC catalog. As with a file opened for reading, the return value of the `openFile` method opened for writing is a reference to a remote stream you can use to manipulate the file. An example:

```
f = ds.open_file "foo/bar/test_lfcds/test_file2.txt", :write
f.puts "First line of the file file 2"
f.puts "Second line of the file file 2"
f.close # remember to close the stream
# openFile invoked with a block
ds.openFile("foo/bar/test_lfcds/test_file2.txt", :w) do |f|
  f.puts "Another way to write to a file"
  f.puts "Note that close is not necessary"
end # here you do not have to close the stream - it is done for you
```

As you can see in the example above, you do not have to close the file explicitly if you use a block argument, i.e. in this example the code between `do` and `end`. You could also use `{` and `}` if you prefer; although the curly braces are often used for one line block argument:

```
ds.open("foo/bar/test_lfcds/test_file2.txt","w") \
   { |file| file.puts("A short file") }
```

`storeFile(payload, path)`

Aliases: `store_file`

This method stores a file whose contents are passed as java bytes in a payload parameter. As with the `openFile` method, contents of file are first streamed to the LFC DS server and stored in a temporary directory; next they are sent to the Grid and registered in LFC catalog using the path specified by the client. True is returned when all of those operations succeed, false otherwise. Examples:

```
# Note the "to_java_bytes" method which enables you
# to turn a Ruby String into Java bytes array
ds.storeFile("TEST file 1 contents".to_java_bytes, \
   "foo/bar/test_lfcds/test_file1.txt")
ds.store_file "TEST file 2 contents".to_java_bytes, \
   "foo/bar/test_lfcds/test_file2.txt"
```

If you are sending large files to the Grid the `openFile` method may be more suitable, as creating large byte arrays may cause `OutOfMemory` errors.

`zero?(path)`

Returns true if file indicated by the path exists and has length of 0 bytes. Example:

```
ds.zero? "/foo/bar/some/path/empty_file.txt"
```

Abstract accepted for Cracow Grid Workshop 2009 (CGW'09) Conference

Integrating EGEE Storage Services with the Virtual Laboratory

Marek Pomocka (1), Piotr Nowakowski (2), Marian Bubak (3,4)
(1) Faculty of Physics and Applied Computer Science AGH, Krakow, Poland
(2) ACC CYFRONET AGH, Krakow, Poland
(3) Institute of Computer Science AGH, Krakow, Poland
(4) Informatics Institute, University of Amsterdam, The Netherlands

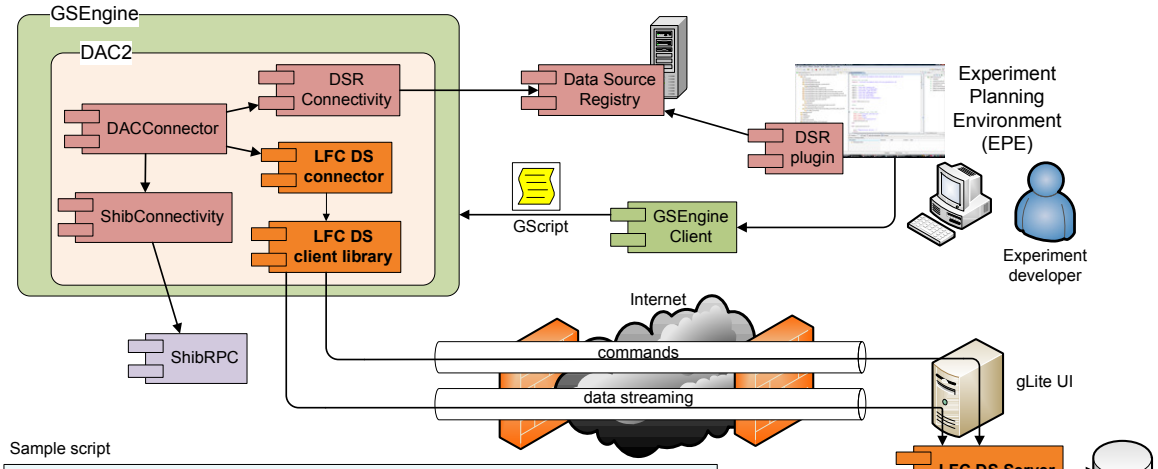
The advent of Grid technologies has enabled research at a pace not achievable using earlier methods, which facilitates easier access to high-end computing and data resources. However, employing Grids in scientific work is still a domain of highly skilled researchers, able to tackle the complexity of the Grid environment. Although there have been successful endeavors that strive to provide a mature scientific environment [1, 2, 3, 4] for scientific disciplines not normally related to computer science, fundamental obstacles still prevent scientific communities from adopting Grids. These include the complexity of Grid security solutions, such as Grid Security Infrastructure (GSI) and intricate access to core Grid services, e.g. data catalogues and storage resources.

Our work aims to minimize the learning curve for access to Grid data services, specifically to LCG File Catalogue (LFC) storage elements and GSI, concealing most technical details. The API we have devised creates an abstraction of working with local files with no intervening GSI, i.e. with no Grid certificate-related operations, although the user works with files stored on the Grid with all GSI mechanisms in place. As regards other projects that deal with comparable issues, the *Credential Mapping Service* [5] allows mapping one security system onto another, e.g. Kerberos authentication tokens onto GSI certificates. Similarly, in our solution, Shibboleth handles are automatically mapped to GSI certificates, relieving users from the burden of managing their own credentials. Furthermore, Yaodong et al. [6] have developed GFISH (Grid File Sharing system), which includes a server providing a web service API for the LFC catalogue and a related Java client with Grid user credentials retrieved from a MyProxy server. They implemented the server using gSOAP, while utilizing Axis on the client side, thus introducing significant transmission overhead. Our approach is also service-oriented, however we relied on RMI-based protocols and libraries, namely the Cajo library for overall communication and RMIIO for streaming. To provide secure transmission, our solution employs SSH tunneling; thus we avoid the need to generate server certificates and to manage keystores (which is an inherent feature of Transport Layer Security). Our development effort did not commence from scratch. Instead, we build on previous work, such as ChemPo [1] LFC command wrappers and the data access infrastructure prepared for the ViroLab [3, 4] project, specifically DAC2 [7] and Data Source Registry (DSR). We have extended DSR so that it is able to store Grid user credentials and information on new data source types, prepared a server that acts as a gateway between DAC2 and EGEE/WLCG, developed a client library that communicates with this server and, finally, developed a new DAC2 GScript [8] interface which makes use of the aforementioned components.

The result of our work is a new convenient API for managing and accessing files on the Grid, which automates certificate management and mimics local file access and directory operations, e.g. the user requesting a file from the Grid is handed a Ruby IO reference that points to a remote input or output stream. Last but not least, the client API is independent of the gLite software, which makes it more accessible to end users and does not impose additional dependencies on the GridSpace Engine [8] – the Virtual Laboratory [3, 4] runtime. Future work might include providing fine-grained security. In addition, further tailoring of the API to specific scientific scenarios may prove very valuable.

Acknowledgements
This work has been partly supported by the European Commission ViroLab Project [43] Grant 027446, Polish SPUB-M grant, the AGH grant 11.11.120.777, and ACC CYFRONET-AGH grant 500-08, as well as the Polish national PL-Grid project.

References
1. Mariusz Sterzel, Tomasz Szepieniec, and Daniel Harężlak. Grid Web Portal for Chemists. In EGEE User Forum, Catania, Italy, March 2009. Presentation slides.
2. Mariusz Sterzel and Tomasz Szepieniec. Enabling Commercial Chemical Software on EGEE Grid – Gaussian VO. In Marian Bubak, Michał Turała, and Kazimierz Wiatr, editors, *Proceedings of Cracow Grid Workshop - CGW'06, October 2006*, Krakow, Poland, 2007. ACC-Cyfronet AGH.
3. PMA Sloot, P.V. Coveney, G. Ertaylan, V. Müller, CA Boucher, and M. Bubak. HIV decision support: from molecule to man. Philosophical Transactions A, 367(1898):2691, 2009.
4. Marian Bubak et al., Virtual Laboratory for Collaborative Applications, In: M. Cannataro (Ed.) Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine and Healthcare, Information Science Reference, 2009, IGI Global
5. Mehran Ahsant, Esteban Talavera Gonzalez, Jim Basney, "Security Credential Mapping in Grids," ares, pp.481-486, 2009 International Conference on Availability, Reliability and Security, 2009
6. C. Yaodong, W. Lu, L. Aigui, and C. Gang. Sharing LCG files across different platforms. In Journal of Physics: Conference Series, volume 119, page 062024. Institute of Physics Publishing, 2008.
7. Matthias Assel, David van de Vijver, Pieter Libin, Kristof Theys, Daniel Harezlak, Breanndann O Nuallain, Piotr Nowakowski, Marian Bubak, Anne-Mieke Vandamme, Stijn Imbrechts, Raphael Sangeda, Tao Jiang, Dineke Frentz, and Peter Sloot: A Collaborative Environment Allowing Clinical Investigations on Integrated Biomedical Databases. In Tony Solomonides, Martin Hofmann-Apitius, Mathias Fredigmann, Sebastian Caludius Semler, Yannick Legre, and Mary Kratz: Healthgrid Research, Innovation and Business Case; Proceedings of HealthGrid 2009, Studies in Health Technology and Informatics, vol 147, IOS Press, ISSN 0926-9630, pp 51 -61
8. M. Malawski, T. Bartynski, and M. Bubak, "Invocation of operations from script-based grid applications," Future Generation Computer Systems, vol. In Press, Accepted Manuscript, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.future.2009.05.012



Sample script

```
ds = DACConnector.new("lfc-voce")
ds.delete('test/test_lfcds')
ds.createDirectory("test/test_lfcds")
ds.directory? "test/test_lfcds"
ds.file? "test/test_lfcds/test_file1.txt"
ds.open("test/test_lfcds/test_file2.txt",:write) do |f|
    f.puts "First line"
    f.puts "Second line"
end
ds.open("test/test_lfcds/test_file2.txt", :read) do |file|
  file.each {|line| puts line}
end
l=ds.listFiles("test/test_lfcds/")
l.each do |item|
  puts item.get_name + ' is a ' +
     if item.is_directory then 'directory' else 'file' end
end
```

Figure 1: Conceptual view of our solution together with a sample script accessing data and operating on LFC catalogue