PROCEEDINGS OF SCIENCE

gridCake & gridCamp: Making the Grid easier

César SUÁREZ-ORTEGA*
Extremadura Research Center for Advanced Technologies (CETA-CIEMAT)
E-mail: cesar.suarez@ciemat.es

Francisco PRIETO-CASTRILLO
Extremadura Research Center for Advanced Technologies (CETA-CIEMAT)
E-mail: francisco.prieto@ciemat.es

José Miguel FRANCO-VALIENTE
Extremadura Research Center for Advanced Technologies (CETA-CIEMAT)
E-mail: josemiguel.franco@ciemat.es

The Grid has shown to be a valuable technology for researchers. However, its front-ends are usually designed for users with high computational skills and they are not very user-friendly. This fact typically causes contempt to final users, because they want to focus their efforts on their researches, skipping concepts not related with their studies.
Also, the tools to make software that can interact with Grid services are complex, generally poorly documented and sometimes they do not support latest Grid middleware versions. This makes very difficult to develop new user interfaces for these services friendlier than the existing ones.
This work will deal about the problems with existing Grid user interfaces and development tools, and it will describe the development of a very easy to use Java library called gridCake, and a web portal called gridCamp built with gridCake, which mixes Grid functionalities with social network features.

EGI Community Forum 2012 / EMI Second Technical Conference
26-30 March, 2012
Munich, Germany

*Speaker.

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

PoS(EGICF12-EMITC2)054

1. Introduction

The Grid [1] is a very useful technology, specially for researchers, because it makes possible to share computational resources between multiple centers to speed up their researches. The typical Grid front-end is a command line interface (CLI) which implies that final users have to memorize complex commands to start using the Grid services. This is a big limitation, because not all researchers have the skills needed to use a CLI without a previous training, and this fact has caused the rejection of this technology for many users.

The current work has two goals with the same motivation: to make the Grid easier. The first objective it is the development of a library easy to learn to soften the learning curve for beginner developers. The second objective it is the development of a simple web portal, aimed to improve the usability for the non-technical user, by mixing concepts from the "classic" Grid web portals and social networks, which are well known tools.

We have developed a Java library called gridCake which is in a prototype stage. It can interact with the basic services [2] in a very simple way, and it is perfect for educational purposes. Using gridCake, we have developed a web portal prototype called gridCamp. It is based on social network concepts, where users can work in a collaborative way, and make operations in a infrastructure easily in the same portal.

2. Related works and *state of the art*

Since the present work reports two different "products", the state of the art is described in two separate sections: gLite development tools and Grid web portals.

2.1 gLite development tools

This work has been developed and tested using the CETA-Ciemat's infrastructure, built using gLite 3.2 [3]. Because of this, our description of the state of the art of Grid development resources is limited to the experience obtained by using these resources.

Software libraries are the main, and almost unique, tool which can be used to build Grid-interacting software. Almost all of this software libraries are developed using some of the Commodity Grid Kits [4] (CoG Kits). CoG Kits are software libraries designed by the Globus Alliance, and they have been developed to interact with the Globus Toolkit services. These libraries are so popular because the main Grid middlewares are implemented over Globus Toolkit and they can interact with almost all the Grid "core" services; hence they can be used over the majority of Grid infrastructures. The main CoG Kits are the Python and Java libraries, but there are versions for other programming languages.

The CoG Kits have one drawback: their generic design makes their use complex. This is a big disadvantage for beginner developers who want to develop software for the Grid.

Many libraries exist in order to hide the complexity of CoG Kits, but generally they are poorly documented or they are incompatible with the newer versions of gLite. The most popular software libraries of gLite are the "official" Java library [18] and jLite [5]. The "official" library is complex, almost abandoned and poorly documented. These problems motivated the creation of jLite: jLite is a software layer over other libraries (mainly the Java CoG Kit and the "official" Java libraries)

PoS(EGICF12-EMITC2)054

to ease the Grid software development. jLite has a proper documentation, and its use it is so easy that a developer with some basic knowledge about Grid could start to develop Grid software. The only limitations of jLite are that some new versions of Grid services are not supported, and his development is almost ended.

2.2 Grid web portals

The typical Grid front-end is a command line interface (CLI) which forces final users to memorize complex commands to start using the Grid services. Nowadays one of the most emergent solutions for this drawback is the development of web portals able to interact with Grid infrastructures [6]. These tools have a lot of advantages:

- They allow non-technical user to use the Grid easily and without installing any additional software in their computers.
- A user interface friendlier than classic CLI front-end.
- They avoid the necessity of any network configuration.
- They are accessed through a simple web browser. There are not restrictions related to operating systems.
- They can hide the Grid internals, so the user only needs to learn a few concepts about Grid services in order to use an infrastructure.
- Some of them work without any Grid user interface, so the user does not have to setup anything in his workstation.

Most of the current Grid web portals are built specifically for particular projects (for example: GridChem [7], NASA QuakeSim [8], Biomedical Informatics Research Network [9], etc.). However, there are some projects developing generic Grid web portals. For example, GENIUS [10] is a Grid web portal for gLite, but it seems to be discontinued, or Web4Grid [11], one of the last web portal developed for gLite. Moreover, there are frameworks to develop Grid web portals like Grid Portal Development Kit [12] (GPDK).

Currently, there are a lot of projects developing Grid web portals and researches using them, so it seems that these tools would be an essential pillar in the forthcoming Grid.

3. Description of the work

3.1 Motivation

The main objective of our work is to develop a Grid web portal easy to use and configure compatible with gLite middleware. We want that the portal could be deployed easily, so the portal avoids any user interfaces. Because of that, the Grid web portal cannot use a command wrapper to interact with the middleware, hence it needs to be developed over some Grid programming library. In our previous researches, we have experienced some technical issues with the CETA-Ciemat's infrastructure and the main gLite programming libraries, jLite and the "official" ones, so we have decided to develop a new library over both.

PoS(EGICF12-EMITC2)054

3.2 gridCake: An easy gLite Java library

gridCake is our gLite 3.2 compatible Java library. It is developed using jLite and some of the gLite “official" libraries and its main objective is to hide the complexities of the libraries mentioned before and to be compatible with the last version of gLite middleware. Some minimum requirements are set:

- The new library has to be able to work with the “core" Grid services [2]: work management (WMS), storage elements (gridFTP), authentication (VMS), credential delegation (MyProxy) [13].
- The public API has to be as easy as it can be possible.
- The library configuration has to be minimum.
- The library has to be able to work without a Grid User Interface.
- The documentation has to be complete and easy to understand.

Because of these targets, we decided to make a Java library (see Figure 1). This decision is the most logical, because we wanted to use the jLite library as the main pillar, and it has been developed using Java as main programming language. Besides, Java is a programming language with a lot of support, and it can be used to develop a web application (for our Grid web portal).

In the early stages of the development, we discovered that the gLite “official" programming libraries did not work with a reliable stability in the Grid infrastructure used during the development at CETA-Ciemat. A Grid library without the job management support is almost useless, so we decided to manage the jobs by using directly a single computing element. This is not the ideal implementation of the library, but this option permits us to develop a functional prototype of the library and, by extension, a functional prototype of the Grid web portal.

The computing elements of CETA-Ciemat are configured with CREAM software [14] which has a fully functional Java library and a SOAP interface, so the integration of our library with a CREAM service was easy, even though its poor documentation.

The resulting library, named gridCake, has the following functionalities:

- Creation of Grid credentials.
- Delegation of Grid credentials.
- Upload of Grid credentials to a MyProxy server.
- Grid credentials download from a MyProxy server.
- Upload of files to a storage element using GridFTP.
- Download of files from a storage element using GridFTP.
- Removal of files in a storage element using GridFTP.
- Sent of jobs to a CREAM computing element.

PoS(EGICF12-EMITC2)054

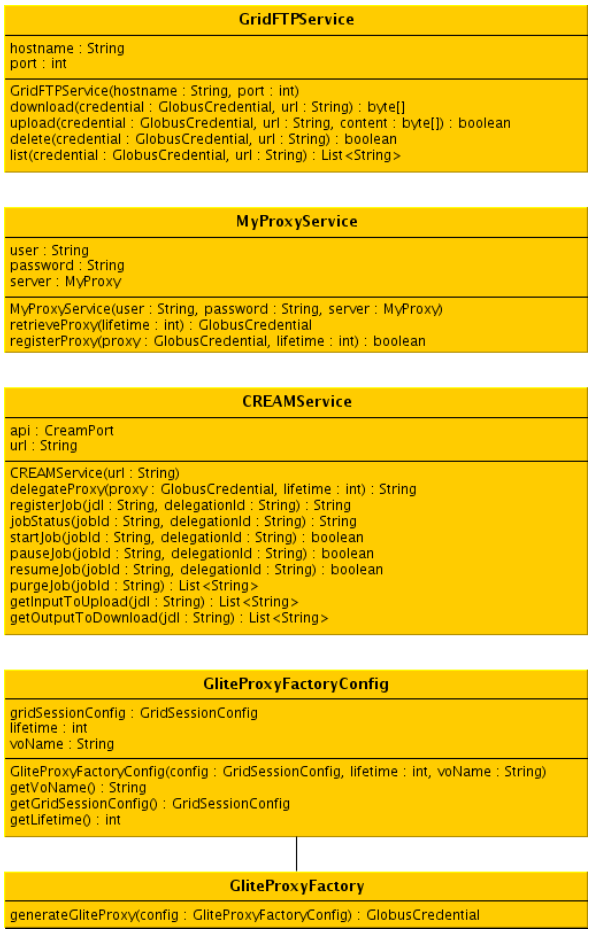

Figure 1: Class diagram of gridCake with all the exposed methods of public classes. ***GridFTPService*** manages the data of storage elements using the GridFTP protocol. **MyProxyService** interacts with a MyProxy server to save Grid credentials and retrieve it as short time proxies. **CREAMService** can manage the submitted jobs to a computing element configured with CREAM. **GliteProxyFactory** creates Grid credentials using the class **GliteProxyFactoryConfiguration**, which contains the entire configuration needed to use the Grid (for example, the configuration of the Virtual Organization).

- Automatic upload of job inputs to a storage element if it is specified in the job description file (JDL).
- Monitor of statuses of jobs.
- Resume and pause of jobs.
- Download of the job output.
- Batch and filtered operations with jobs. For example, it is possible to pause all jobs started one day ago.

The library has already been open-sourced with an Apache License, and it can be downloaded in the web page of gridCake [15].

PoS(EGICF12-EMITC2)054

Also, the library is fully documented using Javadoc, and there is some examples of its use in a wiki [16].

3.3 gridCamp: A gLite web portal for collaboration

Once we had a fully functional prototype of a Java Grid library, we started to search a suitable web Java framework for our needs. After some research, we choose Google Web Toolkit (GWT) to build our Grid portal. We selected Google Web Toolkit because is well documented and because we have some positive previous experiences with it.

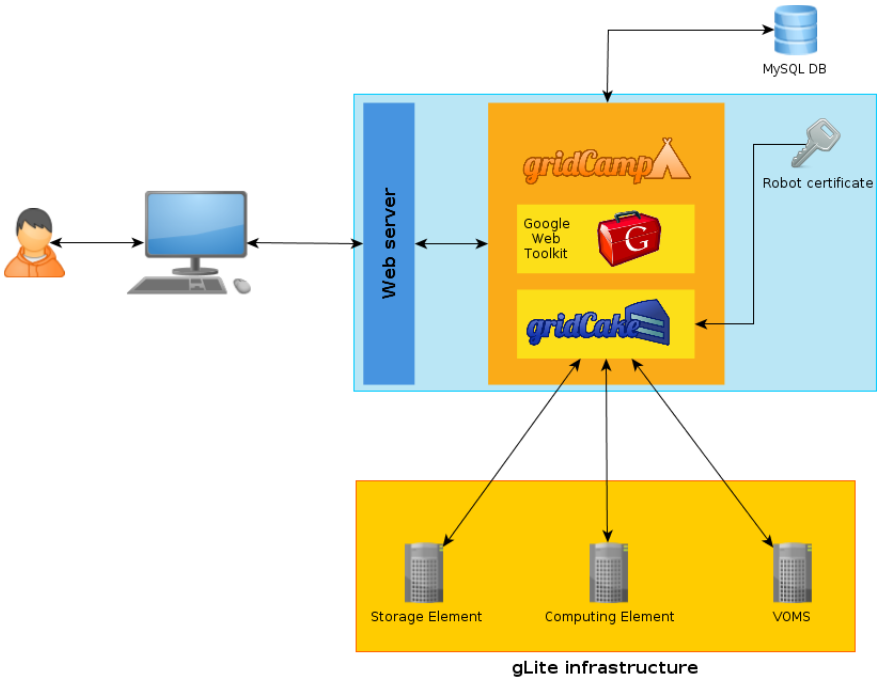

Figure 2: gridCamp architecture. gridCamp can be accessed only using a web browser, which connects to a server with the platform deployed. gridCamp has been developed using GWT, hence it can be deployed on any web server compatible with Java. All related data with users and projects of gridCamp are stored in a MySQL database. gridCake has been used to implement all the functionalities needed in the interaction with gLite services. For Grid authentication it is used a robot certificate stored in the same server than gridCamp.

Our main objective when we started to develop the Grid web portal was to make a generic web portal easy to use. A lot of Grid web portals we have tested have complex front-ends, and this could lead non-technical users to give up their projects. Therefore we decided to make an extra effort to design a simple front-end for our Grid web portal. Also, we decided that the web portal had to be reusable, so it had to be suitable for all kind of projects.

Instead of focusing the portal design on the capabilities of Grid for massive storage and computation, we decided to focus the design on the collaborative potential of Grid technology. Grid technology allows to researchers to collaborate regardless of their location, and our portal could reinforce this fact acting as a meeting point for users.

PoS(EGICF12-EMITC2)054

This philosophy shares many features with social networks: both act as a meeting point for users, both have suitable facilities for collaborative work (message boards, chats, etc.) and both implement ways to share content between users easily. In our case, the content to share is research information; generally, the results of Grid jobs. Therefore, we consider that the idea of mixing the classic Grid web portal with some social networks features could be interesting.

With all these ideas in mind, we had developed gridCamp, a generic Grid web portal with some social features. Its major characteristics are:

- The users have to register in gridCamp only by specifying a set of personal data; a username and a password, which would be his credentials. GridCamp has a robot certificate common for all users, so they can interact with Grid services without any certificate (see complete architecture in Figure 2).
- The users could organize themselves by creating *Projects*.
- Every *Project* has a message board to share information with every associated member (see Figure 3). This information
- All the jobs on gridCamp have to be sent associated with a *Project*. All associated members can look up all the project jobs, regardless of the user who sent the job.
- When a user sends a job, gridCamp analyzes the job JDL, and it determines the input files needed to be uploaded. The user could specify these files by using a simple HTML input, instead of having to manage with gridFTP. This way, gridCamp is the responsible to put in the correct place these files.
- When a job finishes, the user can download all the job outputs by clicking on a link. gridCamp manages the output retrieval from the storage elements like a usual download at the user's web browser.
- Each user has a public profile with his account information, and the projects where he participates.
- Users could be *friends*. If two users are friends, they can send messages each other.

The main difficulties found to implement were all related with security. This is the most controversial issue in a Grid web portal design, because there are multiple options to solve it, but a definitive solution is still missing. The most interesting options are:

- To include inside the Grid web portal a valid Grid certificate called “robot certificate". This is the easiest option to implement, though it is very insecure.
- To use a MyProxy to store the user's Grid credentials. This is a secure option, but the user has to create and upload his credentials to the MyProxy server “by hand".
- GridCertLib [17] is the latest option, which permits a secure storage of Grid credentials. It was created to solve the mentioned limitations.

Because of its simplicity, gridCamp security is managed through a robot certificate.
Although gridCamp is not public, we hope to publish it in the near future.

PoS(EGICF12-EMITC2)054

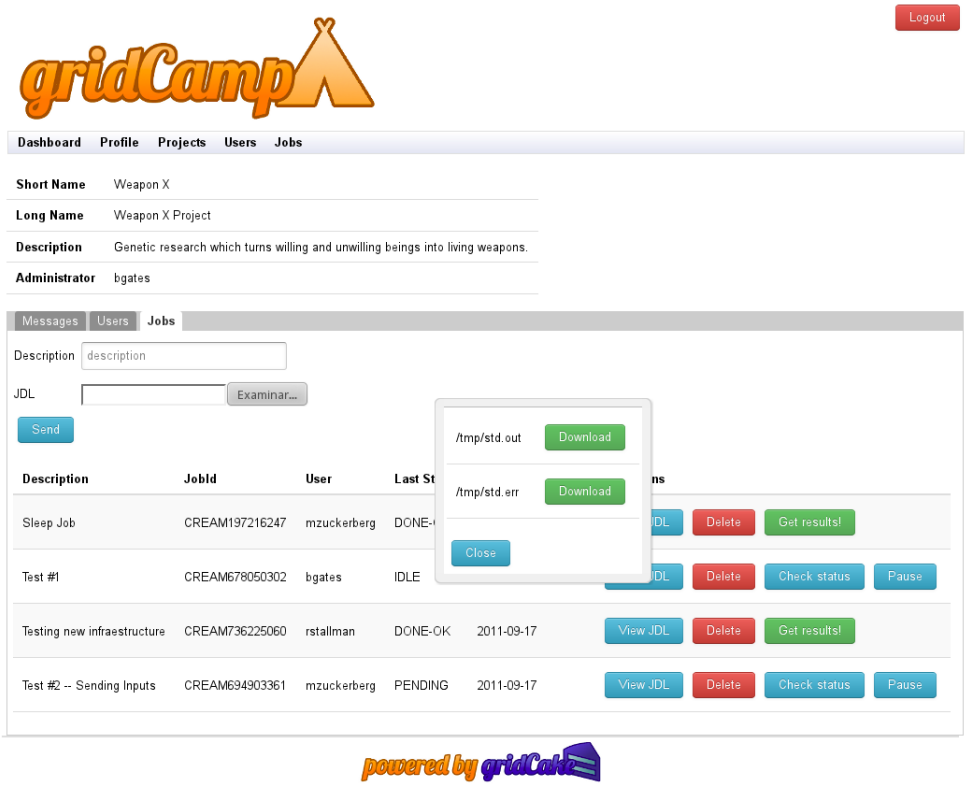

Figure 3: Screenshot of a project page on gridCamp. All the information of the current project is shown in the upper part. The different sections of the project page (the message board, the user list and the job list) are organized using tabs. On the current section appears the list of sent jobs and their related information. The floating dialog shows the output of the finished jobs, which can be downloaded clicking on the download buttons.

4. Conclusions and future work

4.1 Conclusions

During the development of the present work, two different “products" have been developed with the same final objective: to ease the interaction with the Grid for inexperienced users, whether they are final users or new Grid developers. In the previous work analysis, we detected that the Grid is a mature technology, but it could be improved to a great extent. Maybe the main feature to improve is the barriers of entry in Grid because they are very hard, specially for the non-technical users. This is an important issue, because the Grid has been designed to host multi-disciplinary research projects.

Grid web portals are an excellent tool to link users to Grid technologies. They are easy to use, and the can be executed in any operating system, because they only need a web browser. Also, they avoid the need of network configuration, opening ports, installing additional libraries, etc. Currently, there are a lot of developments aimed to build Grid web portals, but a lot of them are complex and immature. Also, the current Grid development tools are complex and (generally)

PoS(EGICF12-EMITC2)054

poorly documented, so the amateur Grid developer has to discover how to use the library; a very frustrating task which could cause disregard for the development of Grid tools.

Due to these facts, we can conclude that to improve the Grid usability, we need better user interfaces and front-ends. The built of these improvements is impossible without excellent development tools, to encourage developers and designers to use the Grid for their software projects. With this idea on mind, we have created gridCake, a Java library enabled to interact with the basic Grid services as easy as possible. With this library, we have developed a small generic Grid web portal, called gridCam, with a design focused on its usability.

Although both tools are prototypes, they are fully functional and they mask the Grid complexities to both end-users and developers. gridCamp is adequate for educational purposes: it covers almost all basic Grid services, it is fully documented and its simplicity is perfect for students. Also, the functionality given by gridCake allows its use at small research groups or in Grid tutorials.

4.2 Future work

gridCake and gridCamp are still prototypes, and they need a lot of improvements that we hope we can develop in the future.

4.2.1 gridCake

- **Additional Grid services support**: Currently, gridCake supports some basic Grid services: VOMS, MyProxy, GridFTP and CREAM, but there are some other important services not supported such as monitoring and the file catalog services. Also the support of EMI middleware is being studied.

- **Job management**: The gridCake job management could be considered as partial: the library only works with a single computing element, ignoring the work management system (WMS). The WMS support is mandatory if we want to use gridCake in production environments.

- **Documentation**: The current documentation of gridCake includes a lot of examples in the library wiki and the code is fully documented using Javadoc, but it could be improved with some examples of functional code and FAQs.

4.2.2 gridCamp

- **More collaborative features**: The current collaborative features of gridCamp are very limited: it only includes message boards on the users and projects profiles. In the future we are going to include some other helpful tools for collaboration: forums, chats, diaries, etc.

- **GridCertLib based security**: GridCertLib seems to be the most perfect solution for security in Grid web portals, but the current implementation of gridCamp works using a robot certificate. This practice is risky, because if a server which hosts gridCamp is hacked, all the users' activities could be affected, because all are being managed with the same certificate stored in the hosting server.

- **To improve the Grid related functionalities**: gridCamp manages the submission and monitoring of Grid jobs, but it does not use all capabilities of gridCake: gridCamp could be extended to give support to data management via GridFTP.

PoS(EGICF12-EMITC2)054

- **JDL generator**: In order to send a job, the user has to upload a hand-made JDL file. The development of a JDL web visual generator could save a lot of user's effort.
- **Job workflow support**: The way of sending jobs on gridCamp is straightforward: the user can only send them one by one, and it cannot define the dependencies between them. The support of any of the existing workflow languages could improve a lot the user's productivity in their tasks at Grid.

5. Acknowledgments

CETA-CIEMAT acknowledges the support received from the European Regional Development Fund (ERDF-FEDER) through its Operational Programme "Knowledge-based Economy".

References

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kauffman, 1999.

[2] E. Laure, S. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, and Others, "Programming the grid with glite," *Computational Methods in Science and Technology*, vol. 12, no. 1, pp. 33–45, 2006.

[3] "glite - lightweight middleware for grid computing." `http://glite.cern.ch`.

[4] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A java commodity grid kit," *Concurrency and Computation: Practice and Experience*, pp. 643 – 662, 2001.

[5] O. Sukhoroslov, "jlite: A lightweight java api for glite,"

[6] M. P. Thomas, J. Burryss, L. Cinquini, G. Fox, D. Gannon, L. Gilbert, G. von Laszewski, K. Jackson, D. Middleton, R. Moore, M. Pierce, B. Plale, A. Rajasekar, R. Regno, E. Roberts, D. Schissel, A. Seth, and W. Shroeder, "Grid portal architectures for scientific applications," *Journal of Physics: Conference Series 16*, pp. 596 – 600, 2005.

[7] "The gridchem project." `https://www.gridchem.org`.

[8] "Nasa jpl quakesim portal." `http://complexity.ucs.indiana.edu:8282`.

[9] "Biomedical informatics research network (birn)." `http://www.nbirn.net`.

[10] A. Andronico, R. Barbera, A. Falzone, P. Kunszt, G. L. Re, A. Pulvirenti, and A. Rodolico, "Genius: a simple and easy way to access computational and data grids," *Future Generation Computer Systems 19*, pp. 805 – 813, 2003.

[11] P. Colet and A. Tugores, "Web interface for generic grid jobs: Web4grid," 2011.

[12] J. Novotny, "The grid portal development kit," *Concurrency And Computation: Practice and Experience*, pp. 1129 – 1444, 2002.

[13] J. Novotny, S. Tuecke, and V. Welch, "An online credential repository for the grid: Myproxy,"

[14] C. Aiftimiei, P. Andreetto, S. Bertocco, S. D. Fina, and Others, "Using cream and cemonitor for job submission and management in the glite middleware,"

[15] "gridcake - a glite 3.2 java library easy as a piece of cake!." `http://www.github.com/csuarez/gridcake`.

PoS(EGICF12-EMITC2)054

[16] "gridcake wiki." `https://github.com/csuarez/gridcake/wiki`.

[17] R. Murri, P. Kunszt, S. Maffioletti, and V. Tschopp, "Gridcertlib: A single sign-on solution for grid web portals," 2011.

[18] "WMProxy Java API at EUAGWiki." `http://www.euasiagrid.org/wiki/index.php/WMProxy_Java_API`.

PoS(EGICF12-EMITC2)054