Rapid Prototyping of Science Gateways in the Brazilian National HPC Network

Bruno F. Bastos, Vinicius M. Moreira and Antonio Tadeu A. Gomes
National Laboratory for Scientific Computing (LNCC)
Petrópolis-RJ, 25651–075, Brazil
Phone: +55-24-22336202
Email: bfbastos@lncc.br, vmacedo@lncc.br, atagomes@lncc.br

***Abstract*—Arguably, an important amount of scientific software development time is likely to be employed on user interfaces. In particular, science gateways have gained increasing interest from the e-Science community because of their convenience to hide the complexity of the underlying resources that give support to the management of scientific data and to the execution of scientific applications. Based on our previous experience with the development of science gateways for diverse application domains in the Brazilian National HPC Network (SINAPAD), we have devised a rapid prototyping strategy to lower the barrier for scientific application developers to launch new science gateways. In this paper we present such strategy, which is based on two main tools. The first tool implements a gateway engine that can be configured by a small set of XML files. Such files completely define the desired functionality of an specific science gateway in such an engine. The gateway engine also offers other features not commonly found in related technologies, such as file sharing, data provenance tracking, and restricted anonymous access to underlying computational resources. The second tool implements both an editor and a packager for the aforementioned engine, allowing the developer to rapidly deploy and launch a new science gateway in ordinary Web application containers. In this paper we present our results with the use of both tools in the SINAPAD network. We also discuss about the current limitations of such tools, as well as how we have been dealing with such limitations to provide a more comprehensive toolset to developers.**

I. Introduction

Science gateways allow researchers to interact in a convenient way (using mainly Web-based technologies) with diverse computational and data resources—*e.g.* HPC clusters, mass storage servers, computational grids and public/private clouds—, aiming at the management of scientific data and the execution of scientific applications on such resources. In some cases, such gateways also provide collaboration tools that allow researchers to share scientific data with the remainder of its scientific community.

The primary purpose of a science gateway is to allow researchers to increase their productivity by concentrating mainly on the subject of their research, and not on the details of the computational tools that are offered him to conduct such research. Ideally, none of such tools should be dealt with by the researcher itself, but instead by a **scientific application developer**. By such developer we mean the staff responsible for building, deploying, and (sometimes) optimizing the scientific software the researcher is interested in using.

Historically, the development of science gateways has been typically based on two main solutions. In the first solution, tools specifically devised for helping with the development of science gateways are used. In the second solution, content management tools are adapted to properly interface with the computational and data resources employed by researchers. Crucially, such solutions, in their current form, require a substantial effort from the scientific application developer, whether it be for development, for configuration, or for deployment of science gateways.

In this paper we propose a toolset specifically built for the rapid prototyping of application-specific science gateways. By **application-specific** we mean the provisioning of *narrow* interfaces to researchers, *i.e.* interfaces that are adapted to the particularities of specific scientific applications, as opposed to interfaces that expose typical science gateway services (such as job submission and monitoring) without regard to such particularities. In contrast to the aforementioned solutions, the purpose of the proposed toolset is to provide the simplest environment as possible for configuring such gateways, without any development or deployment effort by the scientific application developer. Moreover, the toolset offers some specific features not commonly found in the aforementioned solutions and which can be easily enabled or disabled individually for each gateway through configuration, such as: (i) file sharing between gateway users, (ii) support for provenance tracking of jobs' data inputs and outputs, and (iii) support for restricted anonymous access to gateway services.

The proposed toolset comprises two tools. The first tool, called PortEngin, implements a **gateway engine** that is deployed on a Web server like any ordinary Web application. Such engine is configured by a small set of XML files that describe the interface to be expected by the user of the intended scientific application, as well as the enabling of the specific gateway features mentioned above. The PortEngin tool is built upon the basic services for job and data management provided by the CSGrid middleware, an instantiation of the CSBase framework [1] that has been specifically customized for the Brazilian National HPC Network (SINAPAD)[1] to integrate its geographically-distributed, highly-heterogenous computational and data resources. This paper also describes some extensions to the CSBase framework that we have implemented for it to cope with the specific science gateway features mentioned above.

The second tool, called PortEditor, aims at aiding the scientific application developer in the edition of the XML files that

[1] http://www.lncc.br/sinapad

set up the PortEngin tool. Besides its edition capabilities, the PortEditor tool is also responsible for packaging the configured engine, so that it can be easily deployed and launched by the developer on a typical Web application container.

The remainder of this paper is organized as follows. Section II describes some of the most popular solutions for the development of science gateways found in the literature. Section III presents the overall architecture of the proposed toolset. It also provides some background on the CSGrid middleware technology, which is needed for understanding the *modus operandi* of the PorEngin tool, and describes some extensions that we have implemented in our CSGrid instantiation of the CSBase framework for it to cope with the specific needs of science gateway technologies such as the one proposed herein. Section IV illustrates and discusses about the use of the proposed toolset in some portals currently on operation in the SINAPAD network. Section V provides some concluding remarks as well as our perspectives on future work.

II. Related work

Historically, the development of science gateways has been typically based on two main solutions. In the first solution, specific tools for the development of science gateways are used. Among the many examples of this approach, the most probably known are Gridsphere,[2] used in TeraGrid (now XSEDE),[3] and the GENIUS Grid Portal,[4] used in the EGI infrastructure.[5] Science gateways that use this kind of tool implement environments that abstract away the details of the computational resources in which scientific applications are executed, providing a single access point to such resources. Such gateways, in its simplest form, are built for general use, offering secure access to computational resources, job submission and monitoring and file management operations, but exposing these services without regard to the particular needs of specific scientific communities. The development of application-specific gateways is possible and has been done in this type of tool, but such approach requires a substantial development and configuration effort from the scientific application developer [2].

In the second solution, content management tools are adapted to interface with the computational resources in which the scientific applications are run. A commonly found example of this approach is by employing Liferay.[6] Science gateways that use this type of tool implement highly effective environments for collaboration and data sharing. Nevertheless, adjusting such tools to properly interface with the underlying computational resources depends on a considerable development effort that does not receive direct support from such tools [3]. In this direction, some projects, such as Vine Toolkit[7] and EnginFrame,[8] have provided support for the integration of these tools in the context of grid computing and have been used to provide application-specific science gateways to the particular needs of some communities [4]. However, the complexity of the architecture resulting from this integration makes the process of configuring and deploying these gateways considerably difficult and prone to errors.

[2]http://www.gridsphere.org/gridsphere/gridsphere

[3]http://www.xsede.org

[4]http://egee.cesnet.cz/en/user/genius.html

[5]http://www.egi.eu/

[6]http://www.liferay.com/products/liferay-portal/overview

[7]http://vinetoolkit.org/

[8]http://www.nice-software.com/products/enginframe

III. Toolset architecture

The toolset is divided into three layers. The bottom layer provides access to the basic data and job management services offered by the CSGrid middleware, as detailed in Subsection III-A. The middle layer extends the CSBase framework so that our instantiation of the CSGrid middleware becomes able to provide the additional features of file sharing, restricted anonymous access and data provenance tracking, as described in Subsection III-B. The top layer implements the PortEngin tool, which is responsible for building dynamic Web pages based on the settings defined by its XML configuration files, as shown in Subsection III-C. The PorEditor tool also resides in the top layer, being responsible for the edition of such configuration files and the packaging of PortEngin for its deployment and launching on typical Web application containers. The settings that the developer must do for customizing a gateway to a specific application domain are presented in more detail in Subsection III-D.

A. CSGrid basic services

The CSGrid middleware is an instantiation of the CSBase framework, which is illustrated in Figure 1. The CSBase framework is based on central server (a server farm implementation is also possible) in charge of managing all the underlying computational resources available to its users. Such server also manages a data repository that comprises a user directory (for authentication and authorization purposes), a project area (where user files are stored), and an algorithm repository. In the CSBase framework, an "algorithm" is an abstraction used for referring to scientific applications implemented as executable scripts or binaries that accept input parameters and generate outputs, but do not have any type of user interaction during its execution.

A set of execution daemons is responsible for locally managing the computational resources that are provided to the execution of algorithms. In HPC clusters, for instance, such daemons interact with local resource managers such as SLURM, SGE, and PBS, allowing the submission and monitoring of algorithm executions onto the job queues provided by such managers.[9] Likewise, a set of CSFS (CSBase File System) daemons is responsible for transferring data (*e.g.* input/output files, scripts, binaries) from and to the local filesystems of such resources.

The CSBase framework allows different technologies to be used for its user directory and for remotely accessing the local filesystems of the underlying computational resources. In our instantiation of the CSBase framework, the user directory is implemented by an LDAP server, and the filesystems of the underlying computational resources are accessed via the SSH/SCP protocol.

The CSBase framework implements a set of services, some of which are available strictly from desktop applications. One

[9]In our CSGrid instantiation of the CSBase framework all computational resources are regarded as job queues, irrespective of them being part of HPC clusters, computational grids or public/private clouds.

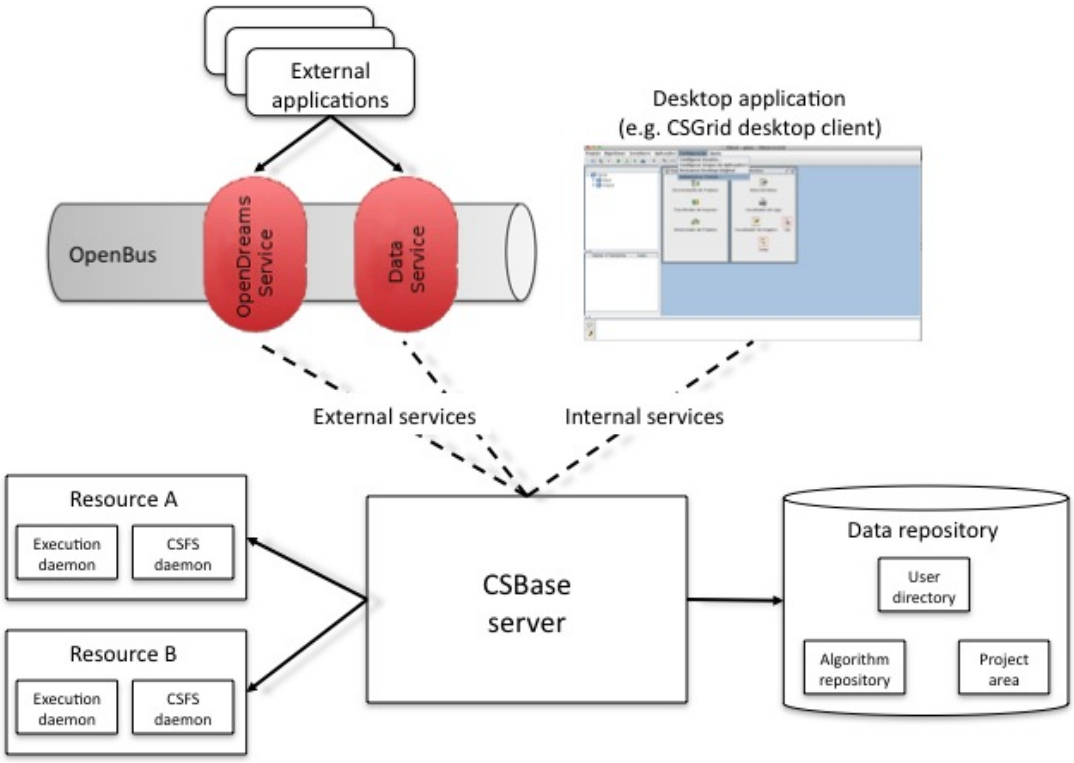

Fig. 1. CSBase Architecture (based on: [1]).

example of such application is the CSGrid desktop client, which offers a set of administrative tools for the CSGrid middleware (*e.g.* for resource management, user management, and algorithm management). Other CSBase services are also exported to external applications through a CORBA-based service bus called OpenBus. Such bus is responsible for the registration and lookup of both external CSBase services and applications that consume such services. Any service or application that intends to interact with the OpenBus service bus must have an X.509 digital certificate properly configured in its access control service.

Two main basic services exported by the CSBase framework in the OpenBus service bus are of interest for the present work: OpenDreams (*OpenBus Distributed Resource and Algorithms Management Service*) and ProjectService.

OpenDreams offers a set of operations for the submission, monitoring and control of jobs (algorithm executions in the CSBase jargon) on remote computational resources. Its service interface is based on OGF's DRMAA 1.0 specifications [5]. The OpenDreams service explores the algorithm abstraction in the CSBase framework to provide a very flexible approach to mitigating resource heterogeneity, which is key feature for the SINAPAD network. The algorithm repository of the CSBase framework allows the same algorithm to have multiple versions, each one being described by a specific set of input and output parameters. For each algorithm version, multiple executable scripts or programs may be provided to reify the algorithm abstraction in different computational resources. This feature allows our CSGrid instantiation of the CSBase framework, and consequently the PortEngin tool, to make it completely transparent to the researcher not only in which computational resource, but also in which *type* of computational resource—whether it be an HPC cluster, a computational grid, or a public/private cloud—his/her job is running.

ProjectService offers a set of operations for uploading, accessing and manipulating files in the project area of the user. User files are organized by projects, and users can share their files in a specific project with other users. Other features offered by the CSBase framework, which are not crucial to the presentation of our approach, may be found in [1].

B. Features added to the CSGrid middleware

The CSBase framework combines characteristics of typical gateway development tools (*e.g.* by abstracting away the details of the computational resources) with those of content management tools (*e.g.* by allowing file sharing between users). Nevertheless, some features not offered by such framework have been incorporated into our CSGrid instantiation. These features are presented below. They have been implemented in the Java programming language and its documentation is available at http://www.lncc.br/sinapad/csgrid-api/.

1) File sharing: The CSBase framework already allows file sharing between users. Nevertheless, the configuration of access permission categories is somewhat complex and restricted to desktop applications that access the internal CSBase services (such as the CSGrid desktop client). To allow the researcher him-/herself to share files in a common area within a science gateway, as well as publish them for open access from the

Internet, we have implemented an additional functionality in our CSGrid instantiation. Such functionality is based on the definition of a special CSBase user called `shared`. This user cannot execute jobs and just have access to its own project area.

In our approach when the researcher shares a file within a science gateway, he/she is actually copying such file into the project area of the user `shared`. A shared file cannot be modified, and only the researcher who shared the file can remove it from the project area of the user `shared`. Moreover, any researcher can directly copy a shared file into his/her own project area (without needing to download and subsequently upload the file), which he/she can then modify or use for performing a job.

2) Data provenance tracking: In its original structure, the CSBase framework keeps a history of job submissions, but does not help with preserving the original files a user provides as input as well as the output files generated as part of a specific submission. The user is then solely responsible for keeping track of such files. Nevertheless, automatic data provenance tracking [6] is very important in e-Science, because it guarantees that a certain *in-silico* experiment be reproducible in the future.

To support automatic provenance tracking in our CSGrid instantiation, we have implemented an additional functionality in it that maintains a history of each job submission, including the arguments and files used as input and the output files generated by the algorithm execution. The researcher may then download the data stored in such a history by using the job monitoring functionality that the PortEngin tool provides to the science gateways. Importantly, such data is immutable and kept separated from the project area of the user in the CSBase framework.

3) Restricted anonymous access: The restricted anonymous access feature allows an anonymous researcher to use, in a limited way, a science gateway without needing to provide user credentials or to be registered in the CSBase's user directory. We have implemented such feature in our CSGrid instantiation of the CSBase framework through the definition of another special CSBase user called `guest`. For each invocation (session) of a science gateway that has this feature enabled, the PortEngin tool creates a temporary folder in the project area of user `guest`. Such folder is associated with a unique identifier for that session and can only be accessed during that session. This folder is removed some time (configurable in the PortEngin tool) after the execution of the job.

To submit a job the anonymous researcher shall respond to a visual challenge (a *captcha*) and enter a valid email address, which will be used for informing the researcher about the completion of the job execution and for providing a URL (associated with the unique identifier of the corresponding gateway session) from where its results can be accessed. Importantly, these results may be accessible by anyone that possesses the unique identifier of the gateway session from which the job was submitted.

An anonymous researcher has restrictions that are not configurable in the PortEngin tool, such as not being able to share files or to keep track of submissions on the science gateway after closing his/her session. Other restrictions are configurable in the PortEngin tool, such as limitations on the input arguments for the algorithms or on the allowed algorithm versions and types of computational resources that the anonymous researcher may invoke. Details about such configurations are presented in the following section.

C. The PortEngin tool

The architecture of the PortEngin tool is illustrated in Figure 2. From the point of view of a scientific application developer, the PortEngin tool is a simple Web application that needs an ordinary Java Servlet container for its deployment. From the point of view of an administrator of the CSGrid middleware, the PortEngin tool is an external application that consumes the OpenDreams and ProjectService services published in the OpenBus service bus. The PortEngin tool creates instances of sciences gateways that offer Web interfaces for: (i) uploading, downloading, viewing and editing files in the project area of the users, (ii) submitting jobs, including validation tips on how to fill out the required algorithm arguments, and (iii) monitoring jobs, indicating their unique identifiers, in which computational resources such jobs are running and when they were submitted, among other information.

The XML configuration files allow the gateway engine to be customized for each instance of science gateway according to the needs of its researchers. These files are:

- `openbus.xml` - defines the digital certificates used by the science gateway for authentication in the OpenBus service bus;
- `config.xml` - describes the input and output arguments of the algorithms associated with the science gateway. Such file is used both by the gateway engine for dynamically assembling Web pages, and by the CSBase framework for assembling the invocation command for the specific scripts or binaries that reify the algorithm in the underlying computational resources;
- `portal.xml` - links the science gateway to a project area and defines which algorithms may run through this gateway. Such file also defines whether the Web interface will be generated on every re-deploy of the gateway or not (this is particularly important when the scientific application developer intends to change the Web layout that the PortEngin tool generates as default).
- `modules.xml` - enables/disables additional features (seen in Section III-B) to be provided by the science gateway;
- `authentication.xml` - configures restricted anonymous access to the science gateway, when such feature is enabled (*e.g.* if an input argument must have a more stringent set of allowed values).

D. The PortEditor tool

In spite of the fact that the XML files described in Section III-C have only a few tags to be configured, such configuration may be tricky at times specially with regard to the parameters that set up the communication with the CSGrid middleware. Besides, the packaging of such files together with the binaries that implement the various functionalities of the PortEngin tool must follow an strict organization so that it can be properly deployed in a Web application container.

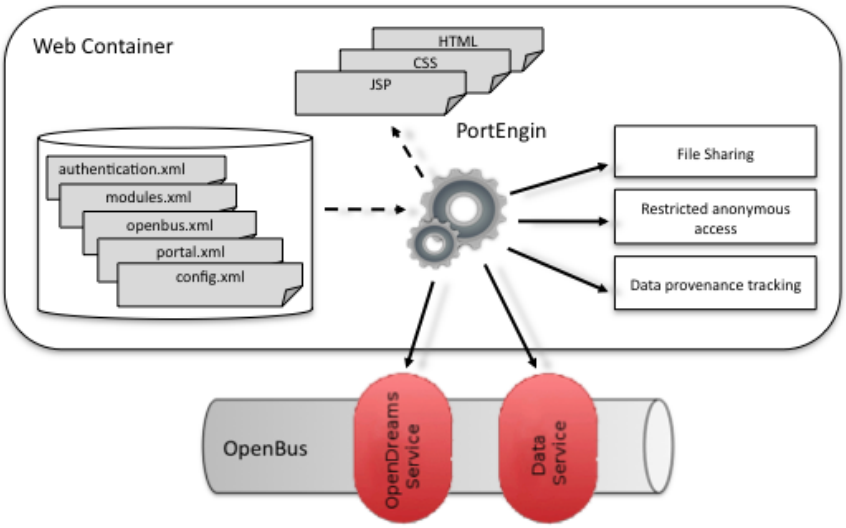

Fig. 2. The PortEngin tool.

Finally, any science gateway prototyped with the PortEngin tool must have a valid digital certificate registered in the OpenBus service bus so that it can consume the OpenDreams and ProjectService services.

Aiming at aiding the scientific application developer in the aforementioned issues, we have developed the PortEditor tool, whose interface is presented in Figure 3. The topmost part of the figure illustrates the functionality of the PortEditor tool that allows a registered developer to edit previously created gateways, whereas the bottommost part shows how a new gateway may be configured. Such tool uses the same LDAP service that the CSGrid middleware employs for user authentication and authorization, so that only registered developers can access the tool to rapidly prototype new science gateways. Once a new science gateway is configured in the PortEditor tool, a deployment package is provided to the developer (see the topmost part of Figure 3), so that he/she can deploy it on a Web application container. Importantly, however, only after the digital certificate of the newly configured science gateway has been registered in the access control service of the OpenBus service bus, will the developer be able to effectively launch the science gateway for the researchers to make use of it. To partially automate such process, when a new science gateway is prototyped in the PortEditor tool, the CSGrid administrators receive an email message informing them about such configuration so as for them to evaluate it and proceed with the certificate registration accordingly.

IV. Examples

Figures 4 and 5 show screenshots of two science gateways currently on production in the SINAPAD network that have been prototyped with the PortEngin tool.

The first example (DANCE – http://www.lncc.br/sinapad/DANCE) provides a service for the efficient evaluation of different kinds of network centralities in complex networks [7]. Figure 4 shows, from top to bottom, the Web pages for job submission and monitoring and file sharing that have been automatically generated by the gateway engine. The algorithm associated with this gateway requires very simple input and output arguments (one input file, one number, one selection option, and one directory for output files), and the layout of

Fig. 3. The PortEditor tool.

this website is the default one generated by the gateway engine.

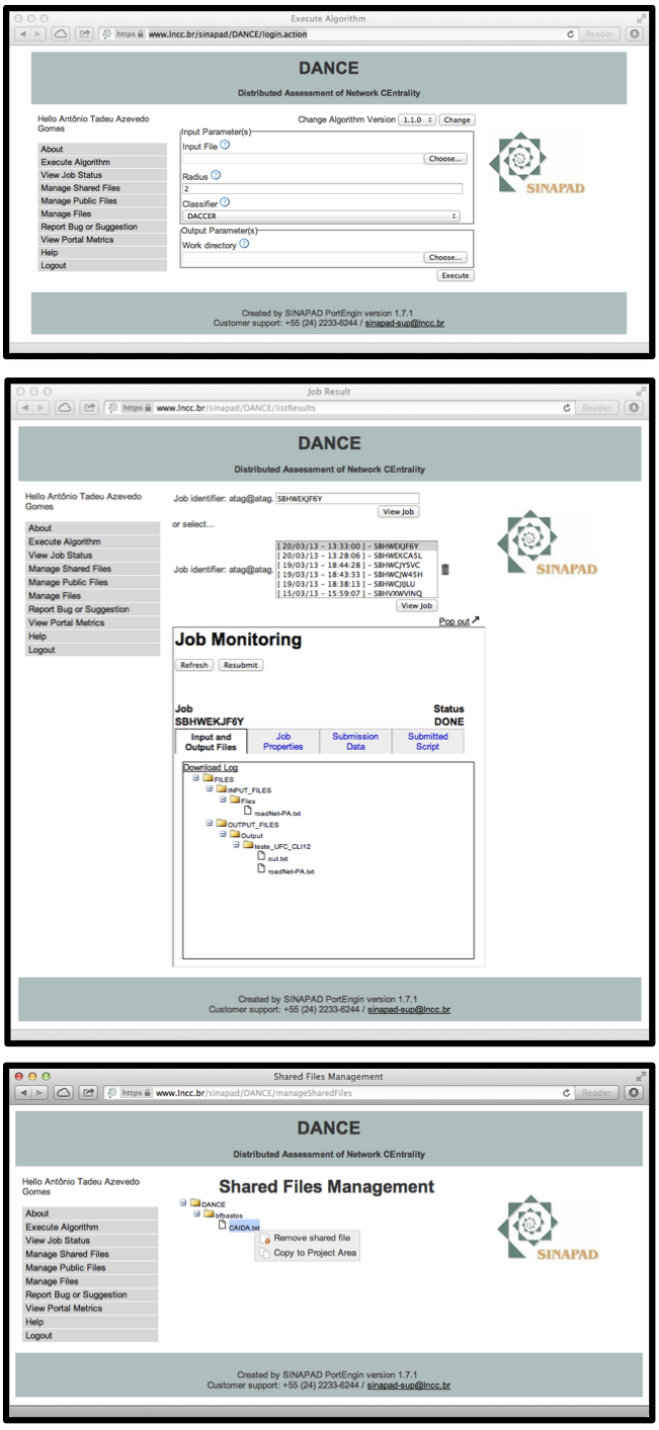

Fig. 4. The DANCE science gateway.

The second example (ProFraGer – http://www.lncc.br/sinapad/Profrager) provides a service for generating libraries of protein fragments. Figure 5 shows the Web page for submitting ProFraGer jobs that has been generated by the gateway engine. The algorithm associated with this gateway, unlike DANCE, requires a much larger and more varied set of input arguments (files, values and ranges, selections), which are reflected in the Web page dynamically assembled by the gateway engine. This gateway implements the restricted anonymous access functionality, and had its Web layout modified by the developer of the ProFraGer software. It is important to notice that the layout of the Web interface provided by the science gateway is unrelated to the operation of the gateway engine. Therefore, the Web layout may be modified by editing the HTML, CSS and JSP files that define such interface, without any change in the implementation of the gateway engine itself.

Fig. 5. The ProFraGer science gateway.

There are a few other application-specific science gateways that are currently on production in the SINAPAD network. Such gateways are listed in htttp://www.lncc.br/sinapad/portais.php.

A. *Discussion*

During our prospecting for new developers and researchers interested in using our proposed toolset, some of them reported two main limitations in it: the lack of support for command-line interfaces, and the poor integratability of PortEngin with widely used content management tools such as Liferay. Our approach to tackle such limitations was to split the PortEngin tool into two sublayers: a lower core sublayer and a higher user interface layer.

The core sublayer implements an API for the higher layer to programmatically access, in a convenient way, the various services offered by the PortEngin tool. The documentation for such API is available at http://www.lncc.br/sinapad/core-api/.

The higher user interface layer allows different interface **personalities** to be implemented over the lower sublayer. One

```
get    - downloads a file in the project area of the user onto the local filesystem
         Usage: get    --project <project name>
                       --file <remote file path>
                       --dest <local destination>

list   - lists the files in the project area of the user (optionally from a specific path in such area)
         Usage: list   --project <project name>
                      [--dir <dir path>]

put    - uploads a file or directory in the local filesystem onto the project area of the user.
         If uploading a directory, provide a ZIP file of the desired directory and use the switch --directory
         Usage: put    --project <project name>
                       --file <local file path (directories must be ZIP files)>
                       --dest <remote destination>
                      [--directory]

remove - removes a file in the project area of the user
         Usage: remove --project <project name>
                       --file <remote file path>

queues - lists the computational resources available for an specific algorithm (optionally for a specific version).
         Usage: queues --algorithm <algorithm name>
                      [--version <version>]

run    - runs an algorithm (optionally using an specific version, or a specific computational resource).
         Other options and flags are algorithm-specific.
         Returns the job id of this algorithm execution.
         Usage: run    --algorithm <project name>
                      [--email <email>]
                      [--version <version>]
                      [--queue <queue name>]
                      <algorithm parameters in the format -KEY=value>

stats  - verifies the status of algorithm executions (optionally in a specific state, or during a specific time slot,
         or with a specific job id).
         Usage: stats --algorithm <project name>
                     [--status (DONE | FAILED | RUNNING | WAITING),..]
                     [--begin <yyyyMMddhhmmss>]
                     [--end <yyyyMMddhhmmss>]
                     [--job <job id>]
```

Fig. 6. CLI commands.

such personality is the Web application that implements our current science gateways (such as DANCE and ProFraGer, presented in Section IV) and is targeted by our PortEditor tool. Another personality is a command-line application that allows the researcher to have access to the underlying computational and data resources by means of a terminal shell interface. Such personality is further described below. Other personalities may be developed, allowing the lower core sublayer to be easily integrated, for instance, with Liferay. Such approach has been adopted in the implementation of a Liferay-based web portal for climatology applications, which is available at http://cenapadportal.cptec.inpe.br/.

The command-line interface (CLI) personality offers researchers the most flexible way for them to manage job submissions and project areas in the underlying resources that comprise the SINAPAD network. Using such interface the researcher may, for instance, automate the submission and monitoring of a (possibly huge) batch of jobs in a single script. Such approach is particularly useful in scientific experiments based on parameter sweeping or Monte Carlo methods.

Figure 6 presents the main commands which are offered by the CLI personality to the researcher. Crucially, all such commands have correspondence with operations available in the Web-based science gateways that employ the PortEngin tool. Moreover, all algorithms and projects that are accessible from the Web-based science gateways may be also accessible through the CLI personality, provided that the researcher has the necessary access rights for them. Since the algorithm abstraction and the project areas employed in both personalities are the same, such duality does not incur in an additional cognitive load on the researchers [8].

A single command-line application is responsible for implementing all available commands in the CLI personality. Such application is registered in the OpenBus service bus so as to be able to consume the OpenDreams and ProjectService services. Such application is accessible through an SSH server available in the SINAPAD network.

V. Conclusion

Our experience with the provisioning of HPC services to the Brazilian scientific community through the SINAPAD network clearly demonstrates that much of the effort employed by scientific application developers (and often researchers alike) is either on learning the idiosyncrasies of the highly-heterogeneous computational resources that comprise the SINAPAD network (*e.g.* available compilers, local resource managers, hardware architectures), or on the *ad-hoc* development of customized, Web-based science gateways that provide transparent access to such resources. The effort presented herein aims at simplifying the development of science gateways through a "*zero programming*" strategy. In such a strategy, the configuration of a small set of XML files is sufficient to enable any supported functionality in the gateway engine that underlies the application-specific science gateways

offered in the SINAPAD network. Importantly, the inclusion of a new CLI personality in our gateway engine copes with the specific needs of some researchers as regards the automation of submission and monitoring of batches of jobs.

It is worth considering that this paper has not mentioned an important trend in e-Science: the use of scientific workflow management systems (SWMSs). Nevertheless, taking as the sample space the researchers that make use of the computational resources provided by the SINAPAD network, it seems that the vast majority of the Brazilian scientific community is either: (i) still pretty much unaware of the facilities that the SWMSs may provide or (ii) does not see value in such facilities due to the deployment effort that such systems might demand from the scientific application developers, when one considers the integration of such systems with the highly-heterogeneous computational resources of the SINAPAD network. In this sense, we have progressed work on the integration of some worldwide-known SWMSs such as Galaxy[10] and Taverna[11] in our toolset by turning them onto other personalities that make use of the lower core sublayer described in Section IV-A.

ACKNOWLEDGMENT

This work was partially supported by the Brazilian Ministry of Science, Technology and Innovation (MCTI), and by the Brazilian National Research and Education Network (RNP). The authors thank Klaus Wehmuth and Artur Ziviani for their involvement in the configuration of the DANCE science gateway. The authors also thank the Group for Molecular Modeling of Biological Systems at LNCC for their involvement in the configuration of the ProFraGer science gateway, and also for their valuable suggestions that considerably improved our toolset.

REFERENCES

[1] M. Julia de Lima, C. Ururahy, A. Lucia de Moura, T. Melcop, C. Cassino, M. N. dos Santos, B. Silvestre, V. Reis, and R. Cerqueira, "CSBase: A framework for building customized grid environments," in *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, ser. WETICE '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 187–194. [Online]. Available: http://dx.doi.org/10.1109/WETICE.2006.26

[2] N. Wilkins-Diehr, D. Gannon, G. Klimeck, S. Oster, and S. Pamidighantam, "TeraGrid science gateways and their impact on science," *Computer*, vol. 41, pp. 32–41, November 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1477047.1477119

[3] TeraGrid Forum, "Liferay: recommendations from selected users," 2012, http://teragridforum.org/mediawiki/index.php?title=Liferay.

[4] R. Barbera, G. La Rocca, R. Rotondo, A. Falzone, P. Maggi, and N. Venuti, "Conjugating science gateways and grid portals into e-collaboration environments: the Liferay and GENIUS/EnginFrame use case," in *Proceedings of the 2010 TeraGrid Conference*. New York, EUA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1838574.1838575

[5] Open Grid Forum, "Distributed resource management application API specification 1.0," 2008, http://www.ogf.org/documents/GFD.133.pdf.

[6] Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *SIGMOD Rec.*, vol. 34, pp. 31–36, September 2005. [Online]. Available: http://doi.acm.org/10.1145/1084805.1084812

[7] K. Wehmuth and A. Ziviani, "Distributed assessment of network centrality," *CoRR*, vol. abs/1108.1067, 2011.

[10] http://galaxyproject.org

[11] http://www.taverna.org.uk

[8] F. Paas, A. Renkl, and J. Sweller, "Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture," *Instructional Science*, vol. 32, no. 1-2, pp. 1–8, 2004. [Online]. Available: http://dx.doi.org/10.1023/B%3ATRUC.0000021806.17516.d0