Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 621–628
ISBN 978-83-60810-27-9
ISSN 1896-7094

Exploratory Programming in the Virtual Laboratory

Eryk Ciepiela, Daniel Harężlak, Joanna Kocot,
Tomasz Bartyński, Marek Kasztelnik, Piotr Nowakowski, Tomasz Gubała,
Maciej Malawski, Marian Bubak
Institute of Computer Science, AGH,
Mickiewicza 30, 30-059 Krakow, Poland
ACC CYFRONET-AGH
Nawojki 11, 30-950 Krakow, Poland
Email: {malawski,bubak}@agh.edu.pl

***Abstract*—GridSpace 2 is a novel virtual laboratory framework enabling researchers to conduct virtual experiments on Grid-based resources and other HPC infrastructures. GridSpace 2 facilitates exploratory development of experiments by means of scripts which can be written in a number of popular languages, including Ruby, Python and Perl. The framework supplies a repository of gems enabling scripts to interface low-level resources such as PBS queues, EGEE computing elements, scientific applications and other types of Grid resources. Moreover, GridSpace 2 provides a Web 2.0-based Experiment Workbench supporting development and execution of virtual experiments by groups of collaborating scientists. We present an overview of the most important features of the Experiment Workbench, which is the main user interface of the Virtual laboratory, and discuss a sample experiment from the computational chemistry domain.**

I. Introduction

MODERN life sciences, particularly simulations in biochemistry, genetics and virology, impose significant requirements on underlying IT infrastructures. Such requirements can be loosely grouped into two general domains: demand for computational resources and demand for new software tools facilitating effective, productive and collaborative exploitation of such resources by a vast range of beneficiaries. While the key goal of supporting scientific experimentation with computerized infrastructures remains the provision of large-scale computational and data storage facilities [1], it is equally important to supply scientists with tools enabling them to collaboratively develop, share, execute, publish and reuse virtual experiments.

The presented Virtual Laboratory, first conceived as part of the ViroLab project [2] and currently being extended within the scope of the PL-Grid project, aims to respond to these requirements by supplying software which permits the execution of virtual experiments. Experiments can be written in popular scripting languages and executed on the distributed resources provided by HPC institutions participating in the project. The goal of the Virtual Laboratory is to a propose a model and facilities for exploratory, incremental scripting – already omnipresent in e-scientific research – and make it reusable and actionable for entire communities. Our framework bridges the gap between the oft-inaccessible high performance computing infrastructures and the end users (i.e. domain scientists), accustomed to running calculations and collating experimental data on their desktop computers. Rather than persuade the scientists to change their daily habits, we want to provide an environment which meshes seamlessly with their style of work, yet extends their experimentation and collaboration potential with the capabilities of high-performance computing clusters.

Our experience gathered in the course of developing the ViroLab Virtual Laboratory for virologists [2], [3], [4], the APPEA runtime environment for banking and media applications in the GREDIA project [5] as well as the GridSpace environment for running in-silico experiments, has been augmented with user requirement analysis conducted during the initial phase of the PL-Grid project, involving groups of scientists from various domains such as physics, chemistry and biology. The Virtual Laboratory presented in this paper should be considered as an evolution of the approach undertaken in the ViroLab project. The new virtual laboratory is focused on interactive and exploratory programming [6], together with a Web 2.0 interaction model.

This paper is organized as follows. In section II we compare our concept with other approaches. After describing our motivation in section III we introduce the main concepts of Virtual Laboratory in section IV while its architecture is discussed in section V. In section VI we introduce GridSpace platform, which is the base technology which implements the virtual laboratory. Section VII contains an overview of the most important features of the Experiment Workbench, which is the main interface of the Virtual laboratory. Further on, we provide a description of the steps which the user undertakes while working with the Virtual Laboratory (section VIII), together with use cases. The conclusions and future work can be found in section IX.

II. Related Work

Scientific workflow systems are important tools for development and execution of e-science applications [7]. Thanks to the well-defined workflow and dataflow models in systems such as Taverna [8] or Kepler [9] it is possible to graphically design applications which can then be executed on remote infrastructures. Scientific workflows can be subject of exploratory programming, as in the case of Wings project [6], as well as of collaborative sharing, such as in the case

of myExperiment social networking website [10]. The main drawbacks of workflow systems are related to the fact that contrary to programming languages, abstract workflow models are often insufficient for describing the required application flow.

Scripting environments are becoming increasingly important in scientific applications and modern petascale systems. An example of this evolution is Swift Script [11] – a dedicated language designed to describe large-scale computations, involving massive data processing. An important feature of Swift is its mapping between data files and programming language variables, which facilitates input/output processing. Scripting can also be used for debugging and instrumenting parallel applications [12]. Our approach is similar in the sense that we intend to use scripting to describe the high-level workflow of the application while retaining interactivity and enabling multiple interpreters to be combined together.

The need to integrate diverse applications, data sources and technologies emerges not only in scientific applications, but also in enterprise systems [13]. Enterprise Service Bus solutions such as ServiceMix or GlassFish, aim to facilitate such integration in the Web service context [14]. The BPEL workflow language can also be applied to scientific applications [15]. However, we believe that such enterprise workflow systems are too heavyweight for simple exploratory programming, where a scripting approach seems to be more appropriate.

The GridSpace environment, which is the foundation of the ViroLab Virtual Laboratory [4], has beed developed by our team to support complex applications running on e-infrastructures such as clusters, Grids and Internet-accessible Web services. One of the goals was to support heterogeneous middleware systems using the Grid Object abstraction layer [16] and facilitate access to distributed data sources. Additional important features include support for collaborative work including tools for application development, sharing, reuse and Web-based accces. Moreover, provenance and result management components provide semantic descriptions of data and enable users to view experiment execution histories. The limitations of GridSpace include its relatively high architectural complexity and the fact that only the Ruby scripting language is supported.

III. Motivation and Goals of the Virtual Laboratory

The main features of Virtual Laboratory result from the experience gained during the ViroLab project, the requirements of external users, as well as from discussions with potential users of the PL-Grid project. The main high-level objectives are as follows:

- To provide an environment which facilitates dealing with scientific application throughout its entire lifecycle (development, deployment, operation, maintenance);
- To reflect and support the day-to-day work of scientists who need to deal with software tools – workflows, procedures (including informal ones), scripts, etc. – enhanced by modern Web 2.0 tools;
- To addresses exploratory programming and a specific type of applications called experiments.
- To support collaborative work of teams of scientists in a Web 2.0 model.

The specific goals of Virtual Laboratory are based on the analysis of e-science applications and discussions held with their authors. The main contributions come from the fields of bioinformatics and computational chemistry. Requirements include:

- Support for different scripting languages – particularly Ruby, Python, Perl and awk;
- Provisioning of tools for publishing and reusing applications/experiments;
- Support for dynamic workflows – as each step of an experiment may depend on the results of previous steps, some workflows cannot be entirely predefined. Moreover, some experiment steps may involve batch jobs;
- Support for parameter-study research, conducted either on the level of a single tool or an entire workflow;
- Support for logging experiments and ensuring their reproducibility;
- Providing easy access to scientific software packages; (the suites most commonly used in PL-Grid include Gaussian, GAMESS, TurboMole and ADF1);
- Direct access to local PBS systems without an additional grid middleware layer;
- Support for creating and using format converters for different tools, as well as adapters for different data sources (not necessarily databases);
- Secure management of user credentials and other sensitive data.

In addition to these requirements, the virtual laboratory needs to satisfy several non-functional and more technical requirements, which are described in more detail in section VI.

IV. Virtual Laboratory Concepts

The key concept associated with the Virtual Laboratory is the experiment. As defined in [3], an experiment is a process that combines data with a set of activities (programs, services) which act on that data in order to produce experiment results. it is important to distinguish the experiment plan – a specific piece of software, written using scripting languages – from the experiment run (execution of the experiment). The key feature here is that the experiment may represent a complex workflow, going beyond simple, repeatable manual execution of installed programs.

The *experiment plan* combines steps realized by a range of software environments, platforms, tools, languages etc. It is developed, shared and reused collaboratively by ad-hoc research teams. The experiment is composed of collaboratively owned libraries and services used (called *gems*) and experiment parts (called *snippets*). Gems are used to represent either program libraries, such as BioPython [17], or applications such as

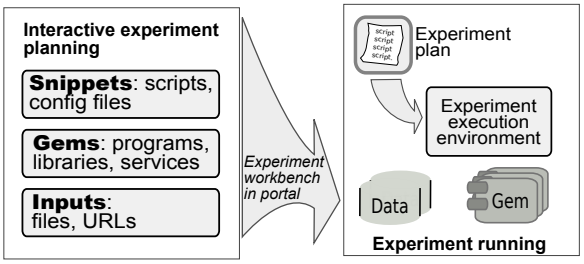

Figure 1. The GridSpace 2 experiment concept

Gaussian [18] or external services, such as the ones from EBI [19]. Snippets refer to separate pieces of code, either in scripting languages, such as Python or Bash, or in other domain specific languages or input file formats, such as used by Gnuplot or Gaussian.

The key paradigm of the enhanced version of virtual laboratory, namely exploratory programming, involves experimentation – step-by-step programming where steps are not known in advance but rather defined on an ad-hoc basis, depending on the results of previous steps. The experiment may need to be re-enacted numerous times, with some ad-hoc customization introduced dynamically, once workflow execution has already commenced. Experiment execution cannot be fully automated and requires continuous supervision, validation or even intervention. This implies a dynamic nature of experiment plan – certain decisions need to be taken at runtime (e.g. code provided from input data). Nevertheless, experiment execution has to remain traceable, verifiable and repeatable.

Due to its focus on exploratory programming, the virtual laboratory is best suited for a try-evaluate-decide process, without losing the context of the experimentation, as opposed to applications with well-known implementations. The virtual laboratory aims at composing and automating higher-level, time-consuming workflows that combine existing software. This property contrasts with applications which perform "atomic" processing. Another interesting feature involves support for novel combinations of existing software modules, data sources and computational capabilities which may result in valuable utilities, as opposed to well-defined workflows which are already addressed by existing software.

V. Architecture of Virtual Laboratory

Fig. 2 presents the architectural overview of the Virtual Laboratory. It is divided into four layers: the Experiment Wokbench layer, available to the user as a set of Web applications, the Experiment Execution Layer which forms the runtime environment of the platform, the Gem Layer which includes all generic and application-specific libraries accessible to the experiments, and the Grid Fabric layer, with all the resources and middleware needed to access them.

The topmost layer is formed by a Web portal which constitutes an entry point for the whole Virtual Laboratory. This gives users access to the Virtual Laboratory from any workstation equipped with a web browser. This layer exposes a portal which is intended as a common, tool-rich workbench for all Virtual Laboratory researchers where they can perform their daily experimentation, collaborate, communicate and share resources (including reusable code). This layer is accessible to end-user browsers via the HTTPS protocol. File Manager is responsible for easy access to data files, Experiment Console allows editing and running experiment snippets, Credential Manager helps handle passwords, certificates and other secrets required by some parts of experiments, and Graphical Experiment Builder is intended to construct more complex experiments graphically.

Further down lies the Experiment Execution Layer where consecutive parts of experiments, provided by the users through the Portal, are evaluated in the context of a particular user account on an experiment host machine. A single experiment can invoke multiple interpreters, such as Python or Ruby. The key concept here is that an experiment can be developed in a piecewise fashion. Individual experiment parts are executed by an experiment interpreter which preserves state (namespace, runtime values) between evaluations. Consequently, experiment development and execution may overlap, and the activities of writing and executing code are intertwined. Such an approach admits introspective, interactive, explorative and dynamic experimentation recorded as experiment code. Moreover, interpreters are executed in separate processes, which provides better isolation and fault tolerance (i.e. a crash o a single interpreter does not influence the others). The Experiment Workbench Layer and the Experiment Execution Layer are in constant communication via the SSH protocol family. We also intend to allow direct user access to the Experiment Execution Layer though bare SSH and SCP.

The next layer consists of Gems which are libraries/modules/utilities invoked by experiments at runtime. The Gem Layer provides APIs for experiment developers, enabling programmatic access to underlying resources (Grid, clusters) and functionality exposed in the form of libraries or services. There are gems which are generic, such as the one which provides access to PBS batch system, or application specific ones, such as Gaussian.

The Grid Fabric Layer forms the lowest level of the infrastructure. It consists of grid resources available to Virtual Laboratory users, including clusters accessible through PBS, grids available through their dedicated middleware packages (e.g. gLite), external services (e.g. Web Services) and data sources (e.g. RDBMSs) which the users may explit of in the course of their research activities.

The entire Virtual Laboratory operates in a Single Sign On (SSO) mode, according to the accounts defined in the external authentication system and incorporating security policies involved in accessing Grid middleware. In the case of PL-Grid installation, it uses PL-Grid LDAP directory so that the virtual laboratory is automatically accessible to all registered PL-Grid users.

The four layers of the Virtual Laboratory are distributed and spread over physical computational nodes. The Experiment Workbench Layer operates on the so-called Portal Host. The

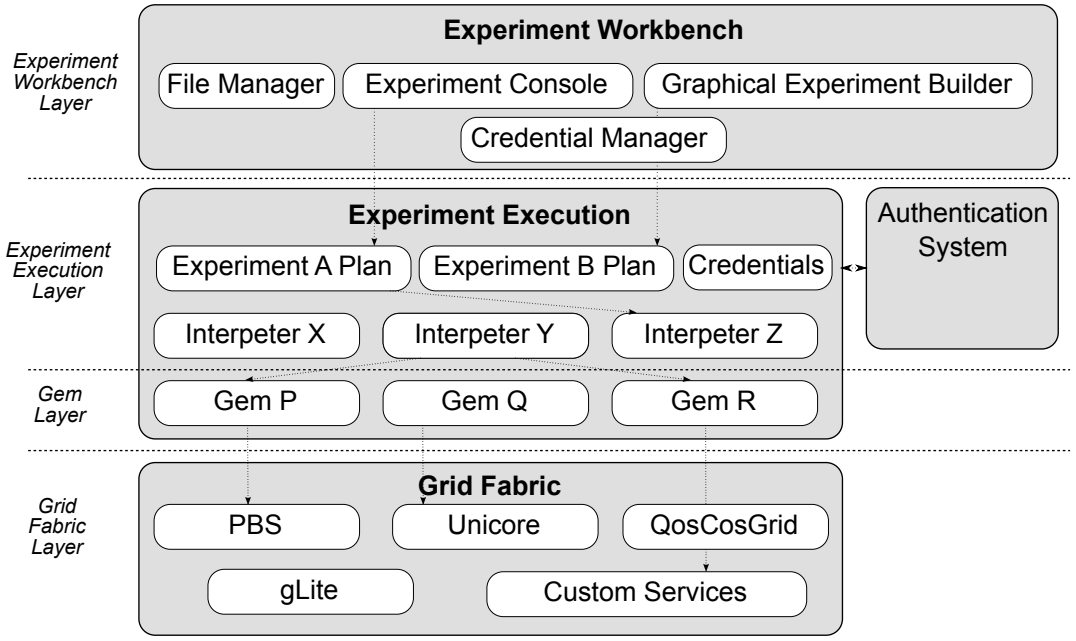

Figure 2. Architecture of the Virtual Laboratory including logical layers (captions on the left-hand side), main modules and their dependencies.

Experiment Execution Layer and the Gem Layer reside in one or more Experiment Hosts, which can be one of the front-end machines (user interfaces) of the clusters. The Grid Fabric Layer spans many distributed computational resources. The Portal Host and the Experiment Host may reside on distinct machines (in order to improve scalability); however they may also operate on a single host when system compactness is more important.

VI. GridSpace – Implementation

The presented Virtual Laboratory, is based on the core technology, called GridSpace, first conceived as part of the ViroLab project [2] and currently being extended within the scope of the PL-Grid. While still supporting the earlier version of the GridSpace framework, in GridSpace 2 we implement the new functionalities which focus on interactive and exploratory programming [6], together with a fully Web-based user interaction model.

In addition to the requirements specified in section III, there are some technical goals, which were not satisfied by earlier versions of the Virtual Laboratory. They include support for multiple scripting languages and more interactive and ad-hoc scripting capabilities. Non-functional requirements such as performance, maintability and ease of use have also been taken into account, motivated by the need to deliver production-quality software to be deployed on the PL-Grid national infrastructure.

It is important to note that GridSpace is a generic environment rather than a specific application. This means that it can be used to set up a specific instance of the Virtual Laboratory in support of a specific application domain. It can be applied whenever existing software modules, interpreters etc. need to be combined with other components. The learning curve involved in porting an application to the platform should be minimized.

Another feature of GridSpace is that it exploits Web 2.0 mechanisms by facilitating application development, operation and provisioning. This means that the entire experiment development and execution cycle is accessible from within a web browser, and moreover, experiments and their results can be shared among community members.

VII. Current Features of the Experiment Workbench

The main features of the current version of the Experiment Workbench include:

- File management
- Dedicated visualization openers
- Interactive interpretation of experiment script snippets
- Storing and sharing experiments using XML format
- Secure management of user credentials (passwords, certificates etc.)

The main exeriment workbench window is shown in Fig. 3. The file management tab is seen on the left, while the right-hand part of the application screen is taken up by the experiment console consisting of several experiment snippets. Above the console, in a separate window, a plugin for displaying graphical data is shown.

In addition to the basic features available directly in the Experiment Wokbench there are additional mechanisms which can be used in more advanced experiments. The first one is WebGUI, which allows experiments to expose dedicated Web interfaces. Another – semantic integration – enables the construction of domain-specific models and data exchange

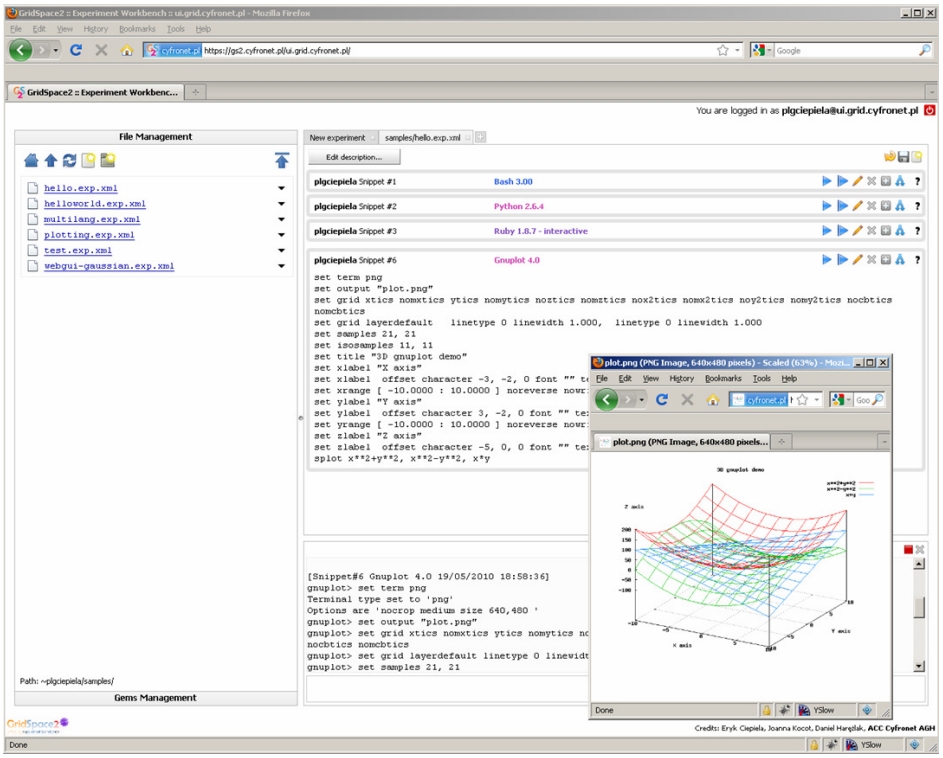

Figure 3. Experiment Workbench screen, including experiment snippets and a sample result plot.

formats to facilitate colaboration and data sharing between experiments and users.

A. File management

At the present stage of development, the GridSpace 2 platform provides basic but stable functionallity which presents a sound basis for further expansion. The most fundamental feature is to enable users to manage their files on an experiment host through an HTTPS interface, enabling basic file system operations as well as uploading, downloading and accessing files via URLs. Data elements and experiment files are assigned globally unique URLs, which facilitate access, annotations and linking resources in a weblike manner. The URLs are given as: https://experiment.workbench.name/experiment.host.name/files/username/path/to/a/file. The Experiment Workbench handles these URLs by accessing the remote experiment host in the scope of the authenticated session. An SSH connection is established with the host, using the provided login/password pair. Following authentication, the workbench performs all file system operations via SCP/SFTP on behalf of the end user. Therefore, authorization relies on file access policies of the operating system residing on the experiment host.

B. Visualization openers

Files served by the Experiment Workbech can be downloaded to the end-user's desktop or viewed and edited within a web browser using so-called openers. The mechanism follows and extends the idea URL-accessible files by introducing the HTTPS GET param "opener", yielding URLs such as https://experiment.workbench.name/experiment.host.name/files/username/path/to/a/file?opener=openerName. The Experiment Workbench handles such requests by serving a page with an embedded opener applet instead of the file itself. The opener applet is configured on the fly, using the input URL, so that it is able to get and put a file using the above mentioned HTTPS interface. As the opener applet launches within the web browser, the authenticated session context (implemented using cookies) covers its function as well.

C. Interactive interpretation of experiment script snippets

The Experiment Workbench implements the exploratory programming paradigm through interactive interpretation of experiment parts called snippets. A snippets is an atomic planning and execution unit of an experiment. Exploratory development involves experiment planning and execution, both of which start at the same time and proceed in parallel. During the exploration process snippets can be incremetally added to the experiment plan, which, in turn, can be incrementally executed in a snippet-by-snippet manner. If the experiment plan calls for a change in a snippet which has alrady been evaluated in a run, the experiment needs to be reexecuted or rolled back (the latter functionality is, however, a challanging issue which will be addressed in the scope of further research).

D. Storing and sharing of experiments using the XML format

The experiment plan is modeled as a sequence of snippets. Each snippet is associated with an interpreter required for its

execution. Interpreter specification includes a command, a set of environment variables and a prompt character sequence which is required in order to enable identification of script line evaluation completeness. Interpreters can be intaractive (capable of evaluating snippets line by line) or batch-oriented (the whole snippet is sent to the interpreter followed by an EOT character, upon which the environment waits for output). Any executable capable of operating in one of the above modes may be configured as a GridSpace 2 interpreter. The idea of an experiment file is that it presents a complete and standalone artifact sufficient for performing the experiment run. As long as it is contained in a single file, it can be associated with a global URL, which makes the experiment an addressable, shareable and linkable resource within the Web 2.0 space. Along with snippet code and interpreter specifications, the experiment XML file contains metadata including the name of the experiment, its description, its creation date, author names and comments. In addition, each snippet may specify a list of so-called secrets, i.e. data elements (such as passwords) which represent user credentials. This data should not be stored in experiment files but instead retrieved from the local user's wallet each time a given experiment is run. The Workbench provides a user-friendly way to manage secrets and use them in experiments while the GridSpace platform facilitates secure storage of credentials and other sensitive data.

A sample experiment XML file in a simplified notation is shown in Fig. 4. It includes a metadata tag, where user comments can be added during experiment evolution. An interpreter definition is presented in line 26 where it is possible to specify the command to initiate an interactive session using PBS. In lines 31–36 of this "hello world" example we can see the sample code in the specified languages.

```
1  experiment:
2    metadata:
3      expname: Hello Experiment
4      author: plgciepiela
5      description:
6         Demo example that prints out hello messages.
7      comments:
8        comment:
9          author: plgciepiela
10         payload: Very simple example, just demo.
11   interpreters:
12     interpreter:
13        cmd="bash --noprofile --norc"
14        interactive="true"
15        name="Bash 3.00"
16        prompt1="$
17        " prompt2="&gt; "
18       envvar: name="PS1" value="$ "
19     interpreter:
20        cmd="/software/local/bin/python"
21        interactive="true"
22        name="Python 2.6.4"
23        prompt1="&gt;&gt;&gt; "
24        prompt2="... "
25     interpreter:
26        cmd="qsub -I -q plgrid -S /bin/bash -v PS1=$"
27        interactive="true"
28        name="PBS - Bash 3.00"
29        prompt1="$"
30        prompt2="&gt; "
31   snippet: id="1" interpreterName="Bash 3.00"
32     code: echo "Hello, Bash"
33   snippet: id="2" interpreterName="Python 2.6.4"
34     code: print("Hello, Python")
35   snippet id="3" interpreterName="PBS - Bash 3.00"
36     code: echo "Hello, Bash via PBS"
```

Figure 4. Sample "Hello world" experiment XML file (tags are represented as bold text for clarity).

E. WebGUI

In order to enrich user experience a WebGUI integration tool is available. It provides well-defined frames for incorporating external web applications in the experiment execution flow. The tool enables experiment creators to plan interactions with the end user and either retrieve additional data or present intermediate experiment results. For integration with external web applications simple JSON communication is used, which makes the process of creating new or modifying existing web applications straightforward. For less demanding experiment creators who do not wish to create or reuse external applications, a generic web UI implementation is available. Through a simple JSON-based definition, available from the experiment code, a graphical interface can be spawned and presented during experiment execution. This implementation allows for building web forms using standard controls (e.g. text fields, text areas, radio boxes, etc.) In addition, a rich text editor control is available.

F. Semantic integration

Among the goals of the Virtual Laboratory, as stated in Section III, is the provision of a generic technology, supporting scientists from specific application domains. Since each of the *in-silico* experiments supported by our platform comes from a specific field of science, the need for domain-specific data models is clear. The semantic integration concept [20] is a method of building application-specific data models and cross-combining them with protocols and tools developed for the application environment. In other words, semantic integration helps scientific developers support structures and taxonomic characteristic for a given field of science via generic storage mechanisms and generic information exchange protocols.

The incorporation of semantic integration in the presented Virtual Laboratory provides a means of storing data and metadata for bioinformatics applications (such as protein pocket finding), as well as for computational chemistry applications running Gaussian and GAMESS. In the former case it is used to store and publish several gigabytes of data produced in high-throughput computations involving numerous proteins. In the later case it helps store and exchange metadata for the output files generated by various chemistry packages. Since the Virtual Laboratory application pool is still being extended, we can expect that the semantic integration solution will eventually cater to other scientific domains as well.

VIII. Working with GridSpace and Examples of Usage

Virtual Laboratory defines a specific model of interacting with applications. The main procedure for preparing an experiment is as follows:

1) The user identifies a procedure (process, workflow) that involves manual use of a number of software pieces which could benefit from (semi-)automation, making them easy to use for the user and for the research team.
2) The user writes this procedure in a stepwise fashion where each step is decided upon by viewing the outcome of previous steps.
3) At each step the user takes advantage of one of a number of programming languages, platforms or programs which are the most suitable for the current purpose.
4) Following each step the user may open the retrieved files with the available tools (e.g. display a graph, visualize molecules, show text content etc.)
5) The user can retrace his/her steps in order to find the best path to the solution.
6) The user can save the current sequence of steps (i.e. the experiment) and open it later for further development.
7) Having discovered the right sequence of steps, the user can save the experiment again and specify the group of users who will be allowed to run or further modify that experiment.
8) The user can send a link to the experiment web application to his/her group.
9) The link leads to a page where, following successful login, other users can run the web application or open it in an experiment editor.
10) The user can enrich experiments with custom graphical user interfaces which collect input and display results.

As an example of use we can present an application from the chemistry domain involving the study of aqueous aminoacid solutions. The analysis process is a workflow which involves multiple steps realized using many tools, languages and libraries. First, Packmol [21] is used to perform molecular dynamics simulations for animoacid aggregation in the presence of water. The resulting solution is visualized with Jmol [22] and can be manually checked prior to further processing, i.e. computing a spectrum using the Gaussian [18] tool. In order to extract spectrum-related information from the Gaussian output file we need to use the CCLIB [23] library written in Python. Finally, spectrum information can be visualized as a plot using GnuPlot.

In addition to the above, many other programming languages and tools can be configured in the Experiment Workbench and thus made available for experiment developers.

Our sample installation of the Experiment Workbench supports a set of interpreters and tools including Bash 3.00, Python 2.6.4, Perl 5.8.5, Ruby 1.8.7, Packmol, Gaussian 09 and Gnuplot 4.0. All interpreters can be launched directly on the experiment host or through the Torque Portable Batch System.

IX. Conclusions and Future Work

In this paper we have presented the main requirements, concepts and current status of the Virtual Laboratory based on GridSpace platform. Although based on experience gained in the course of the ViroLab project, GridSpace 2 constitutes a novel framework. Its main advantage is support for exploratory programming, where each experiment consists of snippets programmed interactively in multiple programming languages using a web console. The Experiment Workbench allows interactive experiment development, file manegement and experiment sharing. The Virtual Laboratory also supports web-based graphical user interfaces and semantic integration.

GridSpace 2 has been made available for the users of the Pl-Grid project for beta testing. Preliminary feedback from bioinformatics and computational chemistry applicaiton domains shows promising results. The final release and integration with the PL-Grid infrastructure is planned for the end of 2010. The continuously updated beta installation of the Virtual Laboratory has been made available to the PL-Grid users and is accessible at the Virtual Laboratory website [24]. More information about the GridSpace 2 technology, including demos and presentations can be found at [25].

Future work will focus on enhancing the usability and security features. One of the planned enhancements involves development of a graphical tool for constructing experiments with hierarchical snippet trees. Another will provide handling of multiple security credentials to facilitate access to heterogeneous middleware systems and data sources. We are also adding support for more application-specific gems, interpreters and visualization tools to extend the range of supported application domains.

Acknowledgments

The research presented in this paper has been partially supported by the European Union within the European Regional Development Fund program no. POIG.02.03.00-00-007/08-00 as part of the PL-Grid project (www.plgrid.pl) and ACC Cyfronet AGH grant 500-08. Maciej Malawski acknowledges support from the UDA–POKL.04.01.01-00-367/08-00 AGH grant.

References

[1] U. Schwiegelshohn, R. M. Badia, M. Bubak, M. Danelutto, S. Dustdar, F. Gagliardi, A. Geiger, L. Hluchy, D. Kranzlmueller, E. Laure, T. Priol, A. Reinefeld, M. Resch, A. Reuter, O. Rienhoff, T. Rueter, P. Sloot, D. Talia, K. Ullmann, R. Yahyapour, and G. von Voigt, "Perspectives on grid computing," *Future Generation Computer Systems*, vol. In Press, Corrected Proof, pp. 1104–1115, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6V06-5046M26-9/2/48b1a0c4be91df6f89554f74e94ae792

[2] ViroLab team at CYFRONET, "The ViroLab Virtual Laboratory Website," 2009, http://virolab.cyfronet.pl.

[3] M. Bubak *et al.*, "Virtual laboratory for development and execution of biomedical collaborative applications," in *Proceedings of the 21st IEEE CBMS, June 17-19, 2008, Jyväskylä, Finland*. IEEE Computer Society, 2008, pp. 373–378.

[4] M. Bubak, M. Malawski, T. Gubala, M. Kasztelnik, P. Nowakowski, D. Harezlak, T. Bartynski, J. Kocot, E. Ciepiela, W. Funika, D. Krol, B. Balis, M. Assel, and A. T. Ramos, "Virtual laboratory for collaborative applications," in *Handbook of Research on Computational GridTechnologies for Life Sciences, Biomedicine and Healthcare*, M. Cannataro, Ed. IGI Global, 2009, ch. XXVII, pp. 531–551.

[5] P. Nowakowski, D. Harezlak, and M. Bubak, "A new approach to development and execution of interactive applications on the grid," in *8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008), 19-22 May 2008, Lyon, France*. IEEE Computer Society, 2008, pp. 681–686.

[6] Y. Gil, V. Ratnakar, E. Deelman, G. Mehta, and J. Kim, "Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*. AAAI Press, 2007, pp. 1767–1774.

[7] Z. Zhao, A. Belloum, and M. Bubak, "Special section on workflow systems and applications in e-science," *Future Generation Comp. Syst.*, vol. 25, no. 5, pp. 525–527, 2009.

[8] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services," *Nucl. Acids Res.*, vol. 34, no. suppl_2, pp. W729–732, July 2006. [Online]. Available: http://dx.doi.org/10.1093/nar/gkl320

[9] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. B. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, 2006.

[10] D. De Roure, C. Goble, and R. Stevens, "The design and realisation of the myexperiment virtual research environment for social sharing of workflows," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 561–567, May 2009. [Online]. Available: http://eprints.ecs.soton.ac.uk/15709/

[11] M. Wilde, I. T. Foster, K. Iskra, P. H. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford, and I. Raicu, "Parallel scripting for applications at the petascale and beyond," *IEEE Computer*, vol. 42, no. 11, pp. 50–60, 2009.

[12] F. Gioachin and L. V. Kale, "Dynamic high-level scripting in parallel applications," in *Parallel and Distributed Processing Symposium, International*. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 1–11.

[13] G. Hohpe and B. Woolf, *Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2004.

[14] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall PTR, March 2005.

[15] W. Tan, P. Missier, R. Madduri, and I. Foster, "Building scientific workflow with taverna and bpel: A comparative study in cagrid," in *Service-Oriented Computing âĂŞ ICSOC 2008 Workshops*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 118–129. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01247-1_11

[16] M. Malawski, T. Bartyński, and M. Bubak, "Invocation of operations from script-based grid applications," *Future Gener. Comput. Syst.*, vol. 26, no. 1, pp. 138–146, 2010.

[17] P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. L. de Hoon, "Biopython: freely available python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, June 2009. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btp163

[18] Gaussian, Inc., "Gaussian," 2010, http://www.gaussian.com.

[19] H. McWilliam, F. Valentin, M. Goujon, W. Li, M. Narayanasamy, J. Martin, T. Miyar, and R. Lopez, "Web services at the european bioinformatics institute-2009." *Nucleic acids research*, vol. 37, no. Web Server issue, pp. W6–10, July 2009. [Online]. Available: http://dx.doi.org/10.1093/nar/gkp302

[20] T. Gubala, M. Bubak, and P. M. Sloot, "Semantic integration of collaborative research environments," in *Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine and Healthcare*, M. Cannataro, Ed. IGI Global, 2009, ch. XXVI, pp. 514–530.

[21] L. Martínez, R. Andrade, E. G. Birgin, and J. M. Martínez, "Packmol: a package for building initial configurations for molecular dynamics simulations." *Journal of computational chemistry*, vol. 30, no. 13, pp. 2157–2164, October 2009. [Online]. Available: http://dx.doi.org/10.1002/jcc.21224

[22] A. Herraez, "Jmol: an open-source Java viewer for chemical structures in 3d," 2010, http://www.jmol.org/.

[23] N. M. O'Boyle, A. L. Tenderholt, and K. M. Langner, "cclib: a library for package-independent computational chemistry algorithms." *Journal of computational chemistry*, vol. 29, no. 5, pp. 839–845, April 2008. [Online]. Available: http://dx.doi.org/10.1002/jcc.20823

[24] Virtual Laboratory Team at CYFRONET, "The PL-Grid Virtual Laboratory Website," 2010, http://wl.plgrid.pl.

[25] DICE Team at CYFRONET, "GridSpace 2 Website," 2010, http://gs2.cyfronet.pl.