Environment for Collaborative Development and Execution of Virtual Laboratory Applications

Włodzimierz Funika[1], Daniel Harężlak[2], Dariusz Król[2], and Marian Bubak[1,2]

[1] Institute of Computer Science AGH, al. Mickiewicza 30, 30-059 Kraków, Poland
[2] ACC CYFRONET AGH, ul. Nawojki 11, 30-950 Kraków, Poland
{funika,bubak}@agh.edu.pl, d.harezlak@cyf-kr.edu.pl, dkrol@student.agh.edu.pl

Abstract. This paper presents the solutions for a user interface environment which have been developed within the ViroLab Virtual Laboratory to enable developers and medical researchers to develop and execute experiments. Experiments require support in form of a script editor easily extendible by a number of additional functions related to the functionality of the Virtual Laboratory, like sharing experiments, and an experiment management mechanism which enables the experiments to be executed with a number of facilities allowing, e.g. tracking the execution, logging errors. Moreover, two user groups, developers and users, need to collaborate to improve experiments and introduce more refinements to the research conducted, which imposes many requirements on the environment which becomes the main collaboration platform for the researchers. The work identifies these requirements in the domain of medical sciences and proposes solutions for efficient and convenient collaboration.

Keywords: virtual laboratory, experiment plan, user interface, collaboration environment, distributed development.

1 Introduction

Working in research projects which involve building applications in new fields of science often demands implementing reliable multi-purpose graphical user interfaces. One of the main tasks of the EU IST ViroLab project [1], [2] is to provide an easy way for different specialists – computer scientists as well as medical researchers – to access distributed resources with sharing and processing clinical data coming from medical practitioners.

The way of accessing data or using computational resources should be simplified so the end users do not need to know about the underlying infrastructure. Groups of researchers working on a project should use the facilities provided by the computer domain, as they would use laboratory equipment during conducting an experiment. ViroLab Virtual Laboratory defines *experiment* as a "process that combines together data with a set of activities that act on that data in order to yield experiment results" [3]. Furthermore, it is required that the software would constitute a virtual laboratory in which scientists from different geographical locations may collaborate and take part in an experiment. This involves preparing a presentation layer which is well-tailored for specific needs of scientific collaboration environments.

One of the primary functionalities the presentation layer must support is to provide facilities for the developer to edit an experiment plan with any enhancements which allow to share experiments (store/retrieve), access data and services (browse/fetch), test experiment plans Grid (track execution/report errors). Whenever an experiment plan is considered mature enough to be submitted for a real execution, some other presentation functionality is needed for the user, which assumes that via a standard Web browser they can launch the experiment plan and get a feedback in form of execution results, moreover, observe whether the experiment is running well or any errors occur. Challenges to be addressed are easing the development process, bridging a gap between the developer and the user, the reusability of components [4] and providing a common collaboration space, a typical issue of distributed development in various domains (e.g. [5]).

This paper presents an approach how to address the needs of two different groups of users, the ones who plan experiments and those who run them and how to meet their demands which turn into collaboration issues. A novel approach is presented to cover the development on the Grid and the interaction with the Grid problems.

The rest of the paper is organized as follows: we give an overview of related work in Section 2, then we are coming to a solution we have adopted for the research, in Section 3. The implementation status presented in Section 6 is followed by concluding remarks and future plans.

2 Related Work

A graphical user interface is built according to the specific requirements of a given project. In the area of grid programming, many efforts were undertaken to satisfy the needs of end users with respect to graphical user interfaces, especially if we are speaking of application composition. Many solutions refer to the workflow approach of representing the internal component dependencies of the applications.

In the K-WF Grid project [6], the User Interface is available through the GridSphere portal [7]. A task is defined through the Grid Workflow User Interface (GWUI) which is a Java applet. The user starts a workflow definition by providing a text sentence which describes the problem. At the next stage this sentence is analyzed and a proper context based on it is selected. GWUI is underlied by GWES – Grid Workflow Execution Service. All conflicts from within this layer are automatically delegated into GWUI – after that the user is able to see a description of the problem.

P-Grade project [8] uses a special designed graphical language – *Grapnel*. It is used for defining relations between resources. This language hides calls to low level communication functions; it uses predefined topology templates for communication between resources (pipe, ring, etc.). A workflow is defined graphically. The use of the Grapnel language resulted in the emergence of a Grapnel language-oriented Editor (GRED) which features automated positioning of ports and flow controls elements and support of grid portals. While this approach is very user-friendly for beginners, experienced users want to have more control on what and how is designed, e.g., with text editors.

Taverna project [9] represents another approach to building application workflows problem, mainly used in bioinformatic domain. User Interface is represented by a standalone application (called Taverna Workbench) within which the user composes a workflow from prepared blocks, e.g. services. No technical knowledge about workflows is required, thus it can be used by the scientist without programmistic skills.

The `myExperiment` project [10] is closely connected to Taverna. Its main goal is to provide a so called collaborative space for the Taverna project community via web portal. It can be used to present (as a screenshot exported from Taverna Workbench), share or collect feedback from users about a workflow.

However, these solutions do not decouple the processes of building and executing the applications, which very often induces additional overhead for end-users who also need to be experts in workflow construction and constrains the environment just to executing workflows without the capabilities that are required by scientific communities (e.g. sharing knowledge, managing results, etc.).

Recently, the area of web development has gained a lot of attention thanks to the new technologies often related to as *Web 2.0 technologies*. There are many examples of applying the techniques that come with the new trend of web development in e-Science [11]. The capabilities of the new approach allow to create collaboration environments based on well-known web standards which are already successfully used by social communities all over the world, such as MySpace [12] or Facebook [13]. The content in such environments is provided and assessed by the users themselves, which makes it very dynamic and rich. An important goal would be to create such environments applicable to the science domain, where knowledge is provided and shared in communities of scientists, using standard web tools (web browsers, web services).

3 Requirements for UI and a Solution in Virolab

Our approach to providing user interfaces reflects the adopted assumptions on classifying the ViroLab users' community into two groups with respect to user interfaces.

The first user group of the ViroLab Virtual Laboratory are *experiment developers* who combine their domain knowledge and technical skills to plan and develop new experiments. Therefore providing tools that support experiment development process is crucial. Another aspect that should be kept in mind is collaboration. Mechanisms for sharing knowledge and experience should be as important as development tools. These features are provided by a dedicated

Integrated Development Environment (IDE), we call ViroLab *Experiment Planning Environment* (EPE). The idea behind EPE is to gather the facilities supporting the above activities into a single integrated environment.

A challenge is to prepare and integrate a set of tools that will be on the one hand powerful enough to satisfy an advanced user and on the other hand it should be user friendly, so that users who are not familiar with applied technologies could use it. In either situation EPE should decrease the complexity level of the experiment management and publishing process.

The functionality of the environment is concentrated around two main parts that address the above mentioned issues:

– a dedicated editor for experiment development that provides syntax highlighting and code auto-completion,
– the Experiment Repository client that enables sharing experiments between developers and publish experiments to the scientific community.

Moreover, due to the distributed nature of computation in the Virolab Virtual Laboratory a tool that provides information about available computational resources, which are stored in the *Grid Resource Registry*, is needed. Another feature that facilitates experiment planning is access to the data model which is available in an ontology form. Also an *Outline* view that shows each object from the source code, e.g., variables or methods definitions, is available.

¿From the collaboration point of view, EPE provides a mechanism for exchanging information between experiment developers and scientists. It is also possible to share a single experiment among a group of developers who are geographically distributed. It is especially important when developers from different institutes want to collaborate on a single project.

The second user group of the Virtual Laboratory involves the *experiment users* who need to manipulate the queue of run experiments and trace the execution of experiment plans. This group concentrates mostly around clinicians and doctors of specific medical domains, who are not assumed to have complex knowledge about scientific computing. Therefore developing a generic interface for experiment management is a challenge. Our analysis of user requirements has resulted in the identification of four main purposes regarding the functionality of the *Experiment Management Interface* (EMI).

The first task is to *locate a desired experiment* by browsing through various experiment repositories and then through experiment versions. The repositories are populated by developers in the process of creating experiments with EPE. Different versions deliver different functionalities with respect to the same experiment plan (e.g. enhancements like mail notification or use of a service with different algorithm). Another task would be to *manage the experiment execution* and to *monitor its status*. This part also includes interacting with the experiment by providing data when needed. The data requests are planned by the experiment developer during the implementation phase. The third task is to *gather*, *store* and *share results*. This involves building a collaboration space for the users executing the experiments, which enables sharing the data returned by an experiment and the knowledge which comes with the results. Finally, it is desirable to enable *providing feedback on the quality* of an experiment to its developers. And last but not least is the user-friendliness of the interface, which refers to both accessibility and use.

While the described solution delivers two separate graphical user interfaces, specific for each user group, there are possible common fields of cooperation between these two user groups. It can comprise reporting problems of the correctness of experiment plans, identifying new requirements, enhancements, etc. This issue imposes a need in a *collaboration space* between EMI and EPE users. This role is played by an experiment repository, which is used by EPE to store and manage versions of the developed experiments and by EMI to list and present details of the available experiments. The solution under discussion allows to use more than a single repository, one per specific field of interest.

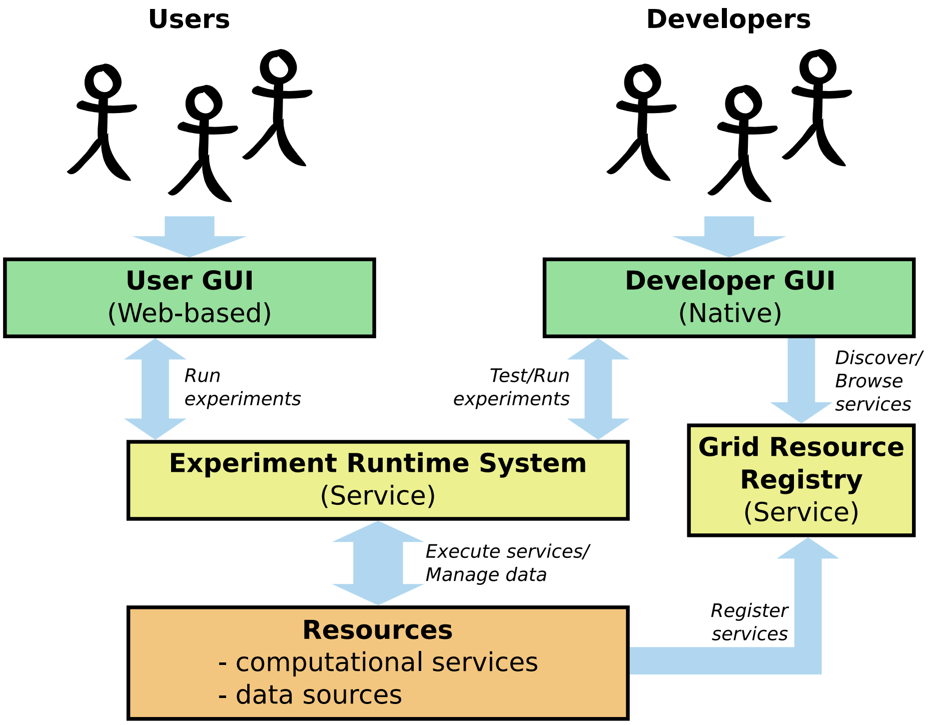

Fig. 1. General concept of user interfaces in ViroLab

In Fig. 1 an overview of User Interfaces with the underlying system structure is presented. On the left, experiment-user side, an easily accessible solution using well-known standards is required. To accomplish this, the interface is implemented as a lightweight web application. The concept of thin client enables users to manage their experiments through even mobile devices.

4 Experiment Planning Environment

As mentioned above, the main goal of EPE is to simplify the experiment development process by providing an extendible collection of dedicated tools. In order to fulfill this requirement, EPE is based on the Eclipse Rich Client Platform (Eclipse RCP) [14]. The core of the platform includes a component called Equinox which is an implementation of the OSGi core framework specification [15]. From the user's viewpoint this means that each part of the environment is treated separately and can be loaded during runtime on demand, which is known as the *lazy initialization* technique.

As mentioned, we do not want to limit the environment to a fixed set of tools. Therefore another feature of the EPE architecture should be extendibility. From the very beginning Eclipse RCP was assumed to be a platform that should be extended easily by new features. Therefore, a special kind of component, called *plugin*, that provides some functionality along with information about itself which is processed by the platform, is used. Thus it is possible to develop new plugins and add them to the environment or publish them and allow other experiment developers to make use of them.

However, plugins make the environment too fine grained, thus it becomes difficult to manage EPE that may contain dozens of plugins. To overcome this problem, the *feature* was defined as a group of plugins that are logically linked and address a certain developer activity. This approach facilitates organizing the internal structure of EPE and extending its functionality.

The current version of the EPE contains the following elements (see Fig. 2):

– *Workbench* that is the central point of the application that manages the layout of each part of the EPE. From the workbench one can choose a *view* or an *editor* to open.
– *Script editor* which supports the experiment development process. It aims to provide such functions as: syntax coloring and code assistance.
– *Experiment Repository client* that supports experiment sharing operations. Dedicated wizards enable to export experiments to or import them from the Experiment Repository.
– *Connector* to the execution service. After setting up the experiment interpreter (e.g. GSEngine) one can start running an experiment with one click. Different versions of the interpreter can be applied along with different experiments.

5 Experiment Management Interface

To support interactions with the user EMI provides a set of related visual components that allow to perform dedicated actions (e.g. run experiment) via a web browser. In Fig. 3 several dependencies between visual components and server-side client libraries are presented. The user is operating in the client layer which is available through a web browser. This allows for platform-independent interactions between the users and the experiment. Each of the visual components is

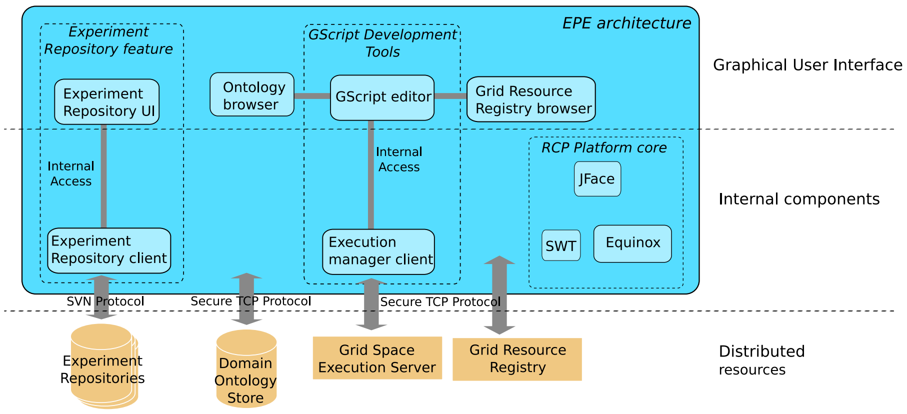

Fig. 2. EPE components and their dependencies

supported by a relevant client library which communicates with the underlying resource. The basic functionality of the EMI interface includes three components:

– *Repository Browser Component* – This component is responsible for presenting available experiments to the user. It is possible to connect to several repositories at a time and browse their contents. The repositories are implemented as SVN code stores. To provide coherence between many repositories a uniform directory structure of the experiments had been established. Access to such repositories is possible only through dedicated clients which keep the content according to the agreed standard.
– *Execution Manager Component* – To execute experiments and monitor their execution status, including handling user data requests, this component was implemented. It may execute more than one experiment at a time and present the actual state which can currently be either *running*, *input awaiting* or *finished*. On the server-side, a client library is responsible to forward experiment execution requests to the Grid Space Server [16]. Between the client and the server an independent communication protocol is used based on the secure TCP layer.
– *Result Management Component* – This component manages experiment results. They may be assessed and either removed or saved in an external Result Store for future use. One of the most important features of this component is the ability to share the results with other researchers. This follows the idea of delivering a collaboration space for groups of scientists.

Another feature supported by the Repository Browser Component is the feedback mechanism. This lets users executing an experiment contact a developer of this particular expeirment and report on its quality. This ensures a user-developer loop to be a part of the solution's lifecycle and improves the cooperation between the two groups.

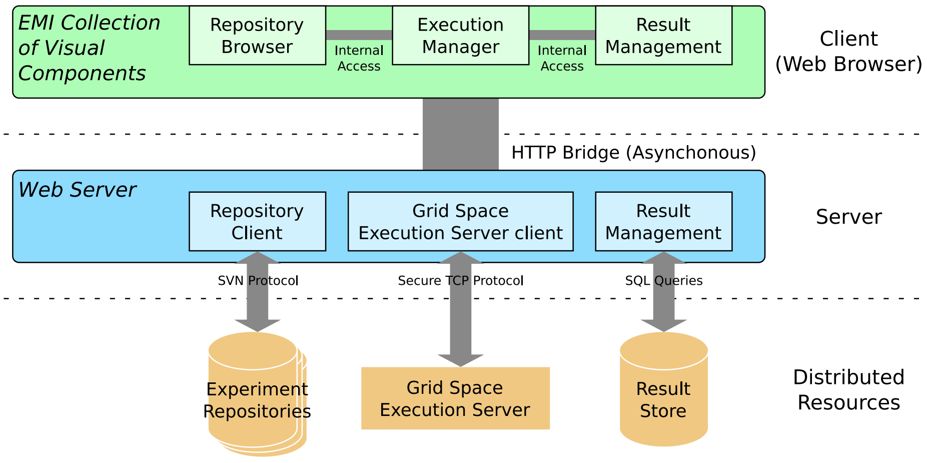

Fig. 3. EMI visual components and their dependencies to server-side clients and underlying resources

The communication model between the client and the server layers is asynchronous. This is a consequence of using Web 2.0 mechanisms such as AJAX. While in the standard *HTTP request-response* model all the contents of a web page including each of the components would have to be refreshed, using new mechanisms allows for much lower communication overhead between the client browser and the server, because only the content that needs updating is refreshed. This also decouples the requests coming from individual visual components and allows for convenient and less centralized web programming.

Another advantage of using the new web programming techniques, namely GWT [17] is the possibility of communication between individual visual components on the client side without the server's participation. This allows for fast content reloads in the client's web browser (as far as interactions with the server are not needed).

6 Current Usage of the Environment

The prototypes of both Experiment Planning Environment and Experiment Management Interface are integrated with other components of the ViroLab Virtual Laboratory [3]. Now they are being improved according to the reported end-user requirements. Both user groups successfully exploit and test the interfaces, thus cooperating on new scientific experiments. It is also freely available from [1], where one may provide feedback or notify about a useful feature that could be included. The EMI web interface is also available and accessible through the project page [3].

Both environments have already been used to implement and run a series of experiments involving simple tests, data access and virology, namely *Nucleotide sequence*, *HIV subtype* or *Sequence alignment* experiments. A full list of experiments can be found at [1]. The implementation already includes using different middleware technologies and infrastructures as well as interaction with the end users running the experiments.

7 Summary and Future Work

In the paper an approach to delivering a presentation layer for Grid users and developers is presented. The layer aims at fulfilling the collaboration requirements of those groups by providing on the one hand an integrated Grid development environment (EPE) and on the other hand a web-based management interface (EMI). The environments enable tight collaboration among and between those groups which is seen as a novelty in bridging the gap between Grid developers and users. One of main goals of this solution is to reach out beyond the scope of the ViroLab and provide a generic approach to using the Grid. Present prototypes already work and provide most of the functionality described in this paper. Many experiments have already been developed and successfully executed which acknowledged the feasibility of the solution. Subsequent improvements are applied to the system and include the following.

The main goal of the development layer is to provide the user with facilities that support experiment development process. Due to the fact that experiment is planned with a special script language (GScript) it is crucial to provide an extendible and helpful *script code editor*. This layer can also involve an integrated preprocessor, semantic code inspection, code autocompletion and an ontology browser. Within this layer the user can access the ViroLab Grid Environment and execute a script code (created in the script code editor) and trace the execution.

The prototypes of EPE and EMI are being enhanced to meet further requirements requested by target users. Specifically, for the EPE environment, additional administrative plugins are required and a more advanced autocompletion mechanism for convenient experiment planning. As for the EMI, a more user-friendly interface with a better feedback support and result management system is going to be implemented. The interactivity issues of the web client are also addressed, for this purpose Web 2.0 tools such as AJAX are used in the current implementation course. Further plans include implementing an advanced mechanism for requesting user data that supports building complex and dynamic forms. Another extension will let different formats of results to be presented to the users for convenient data analysis.

Acknowledgments. This work is partly funded by the European Commission under the ViroLab IST-027446 and the IST-2002-004265 Network of Excellence CoreGRID projects, as well as, the related Polish grant SPUB-M.

References

1. Virolab Virtual Laboratory Home Page, `http://www.virolab.org`
2. Sloot, P.M., Tirado-Ramos, A., Altintas, I., Bubak, M., Boucher, C.: From Molecule to Man: Decision Support in Individualized E-Health. Computer 39(11), 40–46 (2006)
3. Gubala, T., Balis, B., Malawski, M., Kasztelnik, M., Nowakowski, P., Assel, M., Harezlak, D., Bartynski, T., Kocot, J., Ciepiela, E., Krol, D., Wach, J., Pelczar, M., Funika, W., Bubak, M.: Virolab virtual laboratory. In: Proceedings of Cracow Grid Workshop 2007, pp. 35–40. ACC CYFRONET AGH (2008)
4. Goble, C., Roure, D.D.: Grid 3.0: Services, semantics and society. In: Proceedings of Cracow Grid Workshop 2007, pp. 10–11. ACC CYFRONET AGH (2008)
5. Anthes, C., Volkert, J.: A toolbox supporting collaboration in networked virtual environments. In: Sunderam, V.S., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2005. LNCS, vol. 3516, pp. 383–390. Springer, Heidelberg (2005)
6. Knowledge-based Workflow System for Grid Applications, `http://www.kwfgrid.net`
7. Gridsphere Project Home Page, `http://www.gridsphere.org`
8. P-grade Project Home Page, `http://www.lpds.sztaki.hu/pgrade/main.php?m=1`
9. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock6, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20(17), 3045–3054 (2004)
10. myExperiment Project Home Page, `http://www.myexperiment.org`
11. Fox, G.C., Guha, R., McMullen, D.F., Mustacoglu, A.F., Pierce, M.E., Topcu, A.E., Wild, D.J.: Web 2.0 for grids and e-science. In: INGRID 2007 - Instrumenting the Grid, 2nd International Workshop on Distributed Cooperative Laboratories - S.Margherita Ligure Portofino. (2007)
12. MySpace.com: An international site that offers email, a forum, communities, videos and blog space (2008)
13. Facebook Utility Home Page, `http://www.facebook.com`
14. Eclipsepedia RCP Description, `http://wiki.eclipse.org/Rich_Client_Platform`
15. OSGI Specifications Home Page, `http://www2.osgi.org/Specifications/HomePage`
16. Ciepiela, E., Kocot, J., Gubala, T., Malawski, M., Kasztelnik, M., Bubak, M.: Gridspace engine of the virolab virtual laboratory. In: Proceedings of Cracow Grid Workshop 2007, pp. 53–58. ACC CYFRONET AGH (2008)
17. Google Web Toolkit – GWT, `http://code.google.com/webtoolkit`