Collaborative Virtual Laboratory for e-Health

Marian BUBAK[1,2], Tomasz GUBAŁA[2,3], Marek KASZTELNIK[3], Maciej MALAWSKI[1], Piotr NOWAKOWSKI[2], Peter M. A. SLOOT[3]
[1]*Institute of Computer Science AGH, al. Mickiewicza 30, 30-059 Kraków, Poland*
[2]*ACC CYFRONET AGH, ul. Nawojki 11, 30-059 Kraków, Poland*
[3]*Informatics Institute, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands*

Abstract: This paper describes the Virtual Laboratory for e-Health system which is currently being developed in the EU IST ViroLab project. The Virtual Laboratory is an environment which enables clinical researchers to prepare and execute computing experiments using a distributed Grid infrastructure, while not requiring in-depth knowledge on Grid computing technologies. By virtualizing the hardware, computing infrastructure and databases, the Virtual Laboratory is a user friendly environment, with tailored workflow templates to harness and automate such diverse tasks as data archiving, data integration, data mining and analysis, modeling and simulation.

1. Introduction

Important scientific breakthroughs of tomorrow will most probably be delivered by large teams of multidisciplinary researchers gathered in different institutes and cooperating on defined issues. An example of such an endeavor is the field of virology research, especially the part of it tackling anti-HIV treatment problems [1]. For this kind of challenges there is a strong need for modern, distributed and robust virtual laboratories with capabilities, allowing many scientists from distinct domains and institutes to collaborate.

The main objective of this research is to carefully design and deliver a person-centric environment that could combine efforts of computer scientists, virology and epidemiology experts and experienced physicians. The virtual laboratory should provide means for the first group of users to develop in-silico experiments and for the later ones to use previously prepared experiment plans to perform complex research. Therefore a set of dedicated tools need to be provided in the integrated fashion as different types of users need specific approach yet they altogether should be used in the same research endeavor. Moreover, the laboratory needs to be robust enough to overcome infrastructure failures, enabling adjusting and extensions, supporting multiple technological solutions.

Many solutions have been proposed for the construction of application for deployment on the Grid. The Grid environment is diverse in its nature, hence elements of application may be deployed on various computing resources within the Grid and may utilize various implementation and access technologies. In order to maintain flexibility and adaptability of the proposed solution, it is imperative to provide a way in which to access and make use of all those types of resources, whenever a workflow is to be constructed.

We believe that the solution to this problem is the introduction of an abstract layer which would hide the technological diversity of Grid computing services and allow us to construct applications in a uniform way. The application specified by the user would then be executed within the virtual laboratory and the system would take care of calling individual elements in an appropriate manner. This would potentially span the following technologies: Grid jobs (specified and executed through JDL files) [2], Web Services, WSRF [3] and the CCA component framework [4].

To access this range of middleware technologies a mechanism should be developed that enables this in an easy and uniform way. What is more, a uniform description (e.g. semantic description) of Grid resources is very important for building such components as the Collaborative Virtual Laboratory for e-Health, hereafter called the ViroLab Virtual Laboratory.

2. Objectives

The paper provides a discussion of challenges and solutions related to development of a virtual laboratory for virology and epidemiology. The description is centered around a e-science experiment case study, involving different types of users, with focus on a experiment development and execution on the Grid infrastructure. The paper presents an environment for experiment developer, underlying runtime system, and a provenance tracking tool.

The main goal of the paper is to show how these tools can be integrated to provide a complete and robust virtual laboratory, assisting the users in conducting modern in-silico experiments.

For development of the ViroLab platform, we have identified a number of objectives which will determine the use of selected components and existing systems. The most important of these are:

- performance and technological dependencies (whether the component in question performs adequately in a Grid environment and whether it bases on well-established technologies),
- take-up of the component (whether the component is being actively developed and whether there is a sizeable user community dedicated to supporting this component),
- consistency of component functionality and the requirements of ViroLab users (whether the component provides functionality which can be easily adapted to the needs of medical practitioners and other types of ViroLab users),
- open-source licence (since ViroLab is a scientific project, open-source and free software components are naturally preferable to commercial ones).

The virtual laboratory is a tool for the viral disease experts to plan and conduct in-silico experiments. These experiments yield results that can be shared, discussed and stored within the virtual laboratory. The tools that are at the user's disposal within the laboratory help to plan and run experiments using distributed Grid resources, including unified data sources and the previously prepared computational units. The chosen unified naming schema and common terminology helps the various users inside the ViroLab virtual laboratory understand one another and exchange ideas more easily. The design of a virtual laboratory serving such a purpose is the subject of this document.

In order to keep the following text easier to understand, introduction of certain definitions is needed. An experiment is any kind of processing being developed and executed within the ViroLab virtual laboratory. It may, and in fact usually does, involve acquiring experiment input data from distributed resources, running remote processing on this data using distant computing units and storing the obtained experiment results in dedicated storage space. A result of an experiment is any data being produced by the experiment that is meaningful for the performed research (this excludes for instance any control flow variables, temporary variables etc.). A ViroLab experiment result does not need but usually is a subject of analysis, sharing and storing. We use the experiment plan concept to refer to a definition of experiment execution flow of control – such a plan is provided in a form described by the experiment planning notation. The notation defines what constructs may be used in an experiment plan and explains their semantics. An experiment plan written down in such a form is sometimes called a script. The script will utilize operations (called Grid Operations) executed on a set of entities called Grid Objects which will provide a virtualization of the resources present in the underlying Grid infrastructure.

3. Methodology

One of the most important challenges facing the virtual laboratory is the diversity of users (from programmers to medical doctors) and the heterogeneity of underlying resources and technologies available for scientists (from local databases to Grid infrastructures as EGEE). Therefore our approach is based on the idea of multiple abstraction levels of provided resources which may be offered to different types of users and hide the technological differences. The high-level view will be provided by the portal available for end users such as physicians, whereas the experiment development and execution process will be assisted by tools allowing full control over the experiment as needed, including the programming phase.

A crucial decision to support required flexibility while preserving the ease of use and high level of abstraction, was to use a modern scripting language approach. This assumption, supported by the analysis of the state-of-the art and discussions with the users, has led to the design of tools presented in the following section.

The concept of our virtual laboratory is inspired by the state of the art in systems such as virtual laboratories and scientific workflow engines. Several leading technologies should be mentioned in this context.

Geodise [5] is a Grid-enabled optimization and design search tool for engineering which includes the environment for creating new applications (workflows). It is based on Matlab [6] or Jython [7]. Different approach is represented by Kepler [9], Taverna [9] and Triana [10], which are the systems for building scientific workflows. Worth mentioning are also VL-e [11] which is a virtual laboratory for e-Science project funded by the Dutch government which also exploits a scientific workflow concept, whereas projects such as NESSgrid [12] and Polish Virtual Laboratory developed by PSNC [13] are focused on providing remote access to scientific instruments to perform real experiments.

4. Technology Description and Development

An overview of the ViroLab virtual laboratory structure is shown in Figure 1. This figure shows the interaction and role sharing of the ViroLab subparts of the architecture and indicates the main interface channels between these modules. The diagram also proposes the main flow of information in the channels and the nature of the data being pushed through them. Identified parts of the architecture are subject to research and development activities in specific Work Packages and tasks.

The first tool, the Experiment Development Environment, is designed to support domain-related applications developers. It is an integrated development platform for a programmer planning a future experiment. The central idea is to use modern, easy-to-learn scripting language (Ruby) and extend it with a set of dedicated capabilities. This additional functionality provides an interface to the Grid infrastructure used for processing experiments. As the Grid resources tend to be numerous and volatile, the environment helps to use them on a higher level of abstraction - the details regarding execution mechanism, communication protocols and instance location are hidden from the developer who uses their abstractions to plan an experiment. Ontological vocabularies and taxonomies are provided for easy searches among those abstractions of resources. Once an experiment is prepared, it is shared with interested scientists for the purposes of execution.

The lower layer of virtual laboratory runtime controls the experiment execution. It consists of Grid computation access and distributed data access. The first component is responsible for searching specific computational resources, detecting the appropriate protocols to be used and contacting these resources to perform given steps of experiments. The tool is able to interface RPC-type of services (suitable for low-communication short tasks such as the virus sequence genotyping tool [14]) as well as batch job submission systems like gLite (for longer and computationally-demanding tasks such as binding affinity calculation using molecular dynamics techniques [15]).

A scheduling module uses current information about the state of the computing infrastructure to choose proper resource instances to call.

Data integration, a crucial part of many eScience activities, is used to combine multiple (possibly heterogeneous) data resources and to present them to the virtual laboratory user as a single, virtual data source. This involves dynamic translation of database schemas at runtime and delivering them in a uniform format. The use of both those techniques to combine various remote data with dedicated computations into a single experiment, together with the rich standard library of the scripting language, creates a powerful tool both for developers of domain-specific experiments and scientists who use experiments to obtain results.

The scientific results that are obtained using the virtual laboratory may be subject to sharing and further analysis, therefore a dedicated provenance tracking subsystem is provided for recording and storing the result derivation paths. Using both a set of sensors to gather runtime events and annotations provided manually by the users, the system is able to re-create on demand the set of processing steps that led to a certain result of experiment. This capability helps other researchers trust the shared results. Furthermore, a dedicated repository of developed experiment scripts is provided to share the experiments themselves so other scientists may test the repeatability of results on their own.

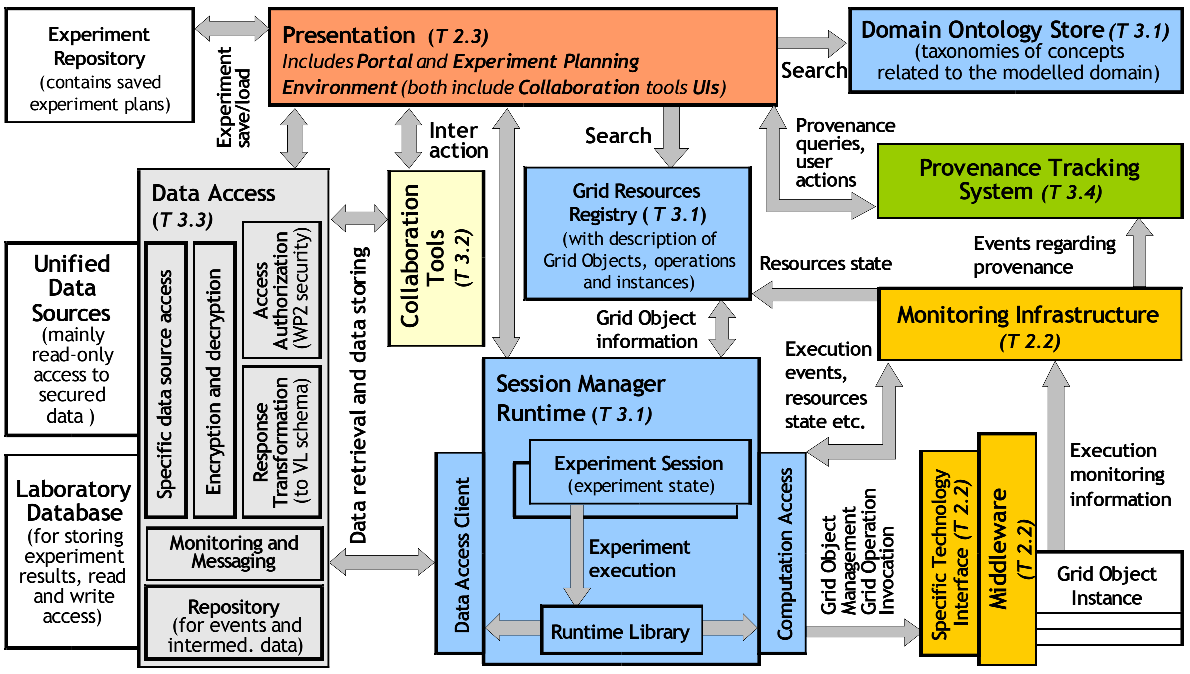
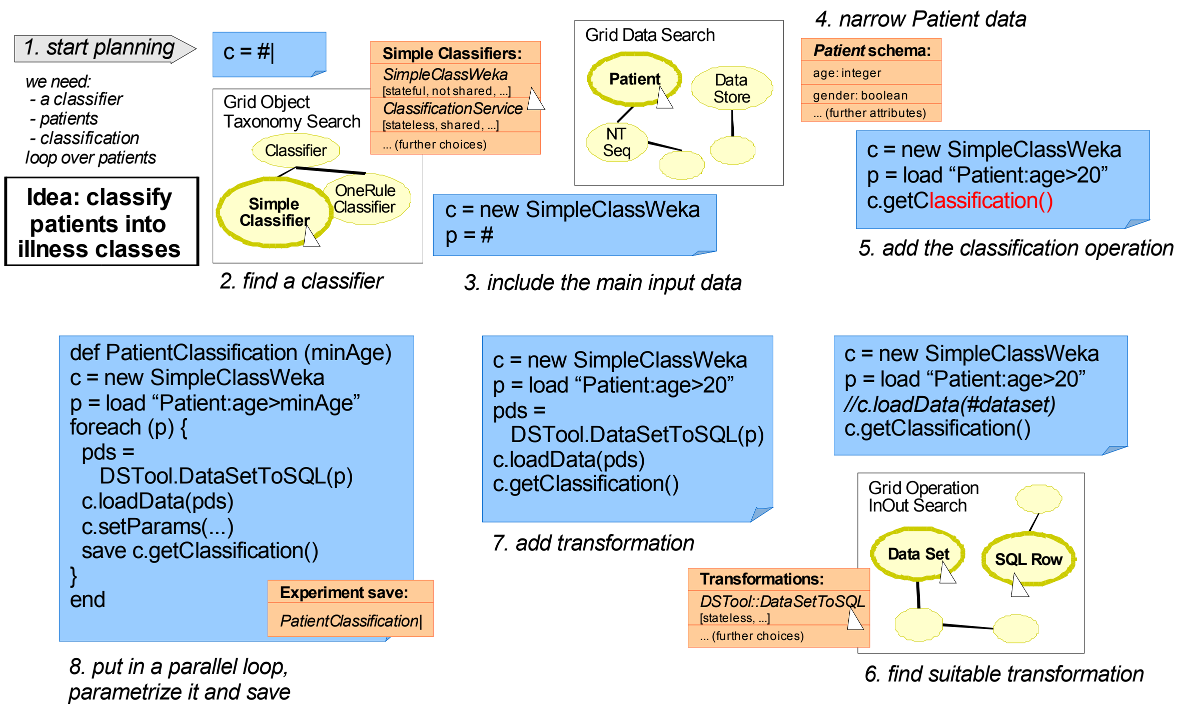

Experiment Repository (contains saved experiment plans)
Experiment save/load
Presentation (T 2.3)
Includes Portal and Experiment Planning Environment (both include Collaboration tools UIs)
Search
Domain Ontology Store (T 3.1) (taxonomies of concepts related to the modelled domain)
Inter action
Search
Provenance queries, user actions
Provenance Tracking System (T 3.4)
Data Access (T 3.3)
Specific data source access
Encryption and decryption
Access Authorization (WP2 security)
Response Transformation (to VL schema)
Monitoring and Messaging
Repository (for events and intermed. data)
Collaboration Tools (T 3.2)
Grid Resources Registry (T 3.1) (with description of Grid Objects, operations and instances)
Resources state
Events regarding provenance
Monitoring Infrastructure (T 2.2)
Unified Data Sources (mainly read-only access to secured data)
Laboratory Database (for storing experiment results, read and write access)
Data retrieval and data storing
Grid Object information
Execution events, resources state etc.
Session Manager Runtime (T 3.1)
Experiment Session (experiment state)
Experiment execution
Runtime Library
Data Access Client
Computation Access
Grid Object Management Grid Operation Invocation
Specific Technology Interface (T 2.2)
Middleware (T 2.2)
Execution monitoring information
Grid Object Instance

Figure 1: Virtual Laboratory structure

5. Results

Figure 2 outlines, in a rather detailed diagram, the steps required to plan a simple experiment that runs on the ViroLab virtual laboratory infrastructure. The example considers a simple program to load a set of data (particularly, an array of patient-related records), perform complex processing on it (processing that is in fact both parallel and distributed) and, finally, store the acquired results in a data store. While the specific details regarding the syntax of the language of the behavior of the accompanying tools are not firmly set and may well be subject to changes, the overall picture is what is important in this example. The step-by-step description below assumes an experiment developer as the active user.



```
c = #|
```

```
c = new SimpleClassWeka
p = #
```

```
c = new SimpleClassWeka
p = load "Patient:age>20"
c.getClassification()
```

```
c = new SimpleClassWeka
p = load "Patient:age>20"
//c.loadData(#dataset)
c.getClassification()
```

```
c = new SimpleClassWeka
p = load "Patient:age>20"
pds =
  DSTool.DataSetToSQL(p)
c.loadData(pds)
c.getClassification()
```

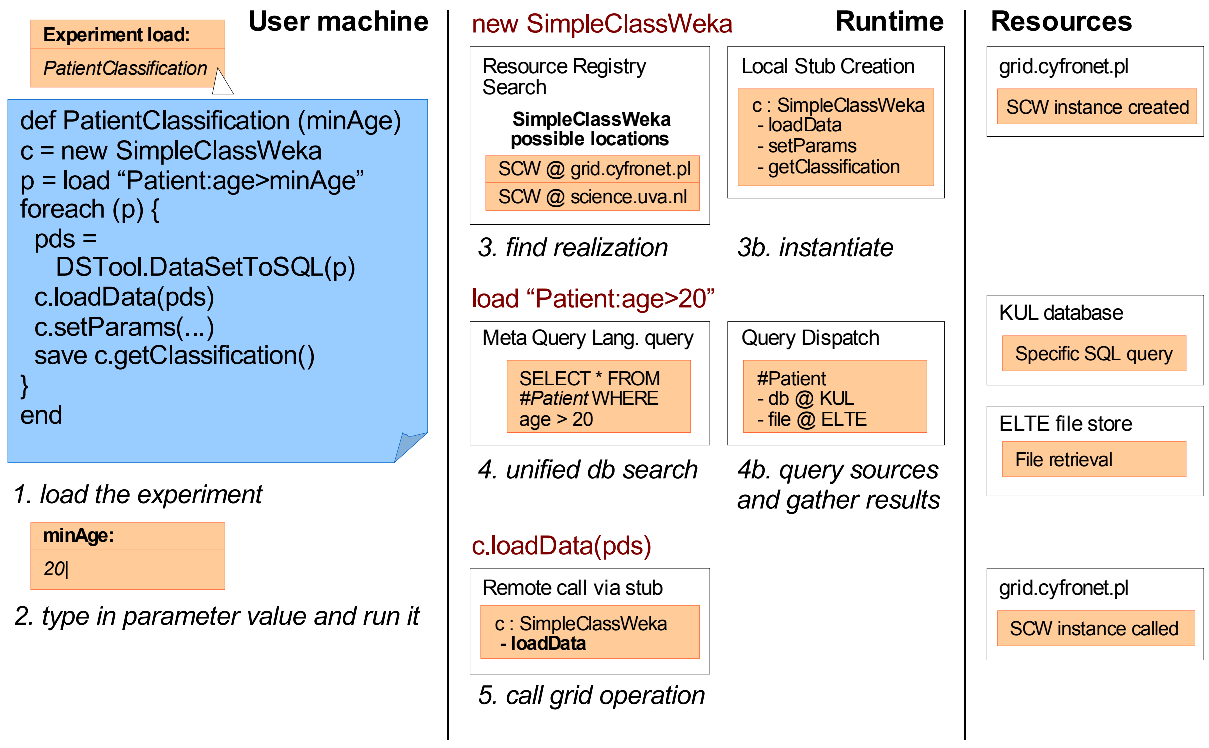
```
def PatientClassification (minAge)
c = new SimpleClassWeka
p = load "Patient:age>minAge"
foreach (p) {
  pds =
    DSTool.DataSetToSQL(p)
  c.loadData(pds)
  c.setParams(...)
  save c.getClassification()
}
end
```

Figure 2: Preparation of experiment plan

The developer starts with the main idea of a new experiment such as "load all patients who are older than 20 and classify them with respect to the types of illnesses". From this high level, flow-based viewpoint the basic blocks that are common to most of such analytical applications need to be selected and configured, such as for example input requisition, processing part execution and output management. In order to commence the main processing part the developer starts the functional ontological search widget that allows him/her to choose the desired functional block. This step is immediately followed by the resource registry interface that allows to choose a specific class of resources from those annotated as Simple Classifiers. This part is labeled as step 2 in the diagram in Figure 2. This also triggers the acquisition of possible operation signatures and their transfer into the developer's workspace. This is a pre-condition for the autocompletion functionality presented in step 5. In addition to simple display of possibilities, the end user is guided through the knowledge technology supported hinting systems such as the what-to-do information provided by the workspace in step 6.

The input data for the experiment is acquired by means of the unified ViroLab data access schema and involves several steps: invocation of the domain ontology browser to choose a proper entity to be loaded and preparing the use of the ViroLab autogenerated database schema tool simplifying the query creation process. Both those actions are represented by steps 3 and 4 in the diagram. By using the ontology schema search the developer will also be able to define the type of data transformation that is required (see step 6 in the figure).

Finally, execution of the processes in parallel and ensuring that the execution is performing well and that the results are saved properly concludes the design process. An optional additional step can be added to make the developed experiment more general and parameterize each step. As a result, the experiment may be reused with different input values (in the chosen example this could be patient age). An experiment plan prepared in this way may now be stored in the experiment repository for other developers as baseline or for the medical scientists who can use it directly.



```
def PatientClassification (minAge)
c = new SimpleClassWeka
p = load "Patient:age>minAge"
foreach (p) {
  pds =
    DSTool.DataSetToSQL(p)
  c.loadData(pds)
  c.setParams(...)
  save c.getClassification()
}
end
```

```
SELECT * FROM
#Patient WHERE
age > 20
```

Figure 3: Execution of experiment plan

Once an experiment has been prepared, it can be executed by the end user – namely, the medical scientist. The pre-assumption is that at least one experiment has been prepared as described in the previous section. In this specific case a medical scientist wants to use the newly prepared patient-by-illness classification software. The procedure is presented in Figure 3, and, as previously, the overall picture is what matters now (the specific details of syntax and presentation may easily be altered in the further development process).

First of all, the user loads the experiment plan e.g. by using a known name. Assuming that the experiment plan has been prepared in the parameterized form, the user is forced to provide input for the minAge parameter. What happens afterwards is presented in steps from 3 to 5 in the picture. These parts depicts in turn the act of remote object instantiation (step 3), contacting the data access server to obtain a response to the data retrieval query (step 4) and the actual remote operation invocation with the acquired data as input (step 5). In the first case the virtual laboratory runtime enacts the Grid object invocation mechanism and uses the built-in scheduler to list available resource instances that are conformant with the developer-chosen class of resources (in this case, every option identifies a possible location where an instance of the SimpleClassWeka may be created). The decision which place to choose could be either given explicitly by the developer in the experiment plan or left to the artificial Scheduler. The Scheduler takes into consideration the available information sources and chooses optimal solutions according to a user-selected policy. Once the remote object has been contacted (or, in this case, previously instantiated on a suitable machine and then contacted) the remote operation call performed in step 5 may occur.

In the meantime, the experiment executes the predefined query on behalf of the user as in the example of patient data categorization for patients with the user-defined parameter of a min age of 20, utilizing the ViroLab unified data access server. This server is able to determine which real data sources have to be contacted in relation to this query and it is also able to perform the appropriate data retrieval afterwards. The specific set of user access rights may influence the available choice of data sources. Data from different data sources are merged and integrated before being delivered to the experiment execution host. The runtime mechanism within the execution host controls proper data import into the experiment interpreter making it accessible to the application (see step 4 in the diagram).

While not depicted in Figure 3, during each of the steps 1, 3 and 4 a process of authorization may take place. This depends on the access restrictions of a particular resource. The components of the virtual laboratory that access the resources have to be integrated with the security mechanism. This ensures that all secured interactions at runtime are done by properly authenticated and authorized users or components.

6. Benefits

The Virtual Laboratory carries with itself several important advantages, particularly in comparison to existing workflow construction tools. The following benefits should be mentioned:

- compatibility with a wide spectrum of computing resources (Web Services, WSRF, component architectures),
- integration of various types of data resources (plain databases, OGSA-DAI database aggregations, custom data access solutions),
- concealment of the underlying implementation details, enabling experiment developers to prepare experiments with little knowledge on Grid computing, and experiment users to execute these experiments without any such knowledge,
- powerful and open-ended nature of the development environment, which does not constrain developers in the same way as current “drag and drop” workflow construction tools do.

It should also be noted that while the presented solution is used in support of a scientific endeavor, namely improving the quality of HIV treatment through exhaustive studies of HIV virus mutations, this in no way restricts the Virtual Laboratory from being applied to other areas of science and business where high-performance computing is deemed necessary and Grid solutions may be applied.

7. Conclusions

In this paper we describe an approach to development of a virtual laboratory for modern e-science experiments in the framework of eHealth. These experiments involve different types of users and harness heterogeneous resources. The integrated use of several levels of abstraction and a set of dedicated, task-oriented tools enables a high-level view on the system for its end users (such as medical doctors) while at the same time giving full control over experiment to advanced users such as scientists assisted by scientific programmers.

This virtual laboratory will be integrated into the ViroLab system [16] which aims at facilitating medical knowledge discovery and decision support for infectious diseases. Initial prototypes have been developed and are now under testing with further applications expected towards the end of 2007.

Acknowledgements

The authors wish to express their thanks to E. Ciepiela, J. Kocot, P. Pęgiel and I. Ryszka for fruitful discussions. The work described in this paper is supported by the European Union through the IST-FP6-027446 project "ViroLab".

References

[1] P.M.A. Sloot; A. Tirado Ramos; I. Altintas; M.T. Bubak and C.A. Boucher: *From Molecule to Man: Decision Support in Individualized E-Health*, IEEE Computer, (Cover feature) vol. 39, nr 11 pp. 40-46. November 2006

[2] Ian Foster el al., "Modeling Stateful Resources with Web Services", Globus Alliance, Argonne National Laboratory, IBM, USC ISI, Hewlett-Packard, Jan, 2004. http://www.globus.org/wsrf/ModelingState.pdf.

[3] The gLite project: http://glite.web.cern.ch/

[4] Maciej Malawski, Marian Bubak, Michał Placek, Dawid Kurzyniec, Vaidy Sunderam: *Experiments with Distributed Component Computing Across Grid Boundaries*, Proc. of HPC-GECO/COMPFRAME Workshop in Conjunction with HPDC'06, Paris, June 2006

[5] Geodise homepage; http://www.geodise.org

[6] Matlab homepage; http://www.mathworks.com

[7] Jython project homepage; http://www.jython.org

[8] Kepler project homepage; http://kepler-project.org

[9] Taverna homepage; http://taverna.sourceforge.net

[10] Triana homepage; http://www.trianacode.org

[11] VL-e homepage; http://www.vl-e.nl

[12] NESSGrid homepage; http://www.neesgrid.org/

[13] PSNC Virtual Laboratory homepage; http://vlab.psnc.pl

[14] T. de Oliveira, et al: An Automated Genotyping System for Analysis of HIV-1 and other Microbial Sequences. Bioinformatics 21(19): 3797-3800 (2005)

[15] S.K. Sadiq, S.J. Zasada, P.V. Coveney, Grid *Assisted Ensemble Molecular Dynamics Simulations of HIV-1 Proteases Reveal Novel Conformations of the Inhibitor Saquinavir*, Computational Life Sciences II, Springer 2006, LNCS vol 4216, pp. 150-161

[16] ViroLab: Virtual Laboratory for Decision Support in Viral Disease Treatment; EU IST Project FP6-027446; www.ViroLab.org