Towards a System-Level Science Support

Tomasz Gubała[2,3], Marek Kasztelnik[3], Maciej Malawski[1], and Marian Bubak[1,3]

[1] Institute of Computer Science AGH, al. Mickiewicza 30, 30-059 Kraków, Poland
[2] Informatics Institute, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
[3] ACC CYFRONET AGH, Kraków, ul. Nawojki 11, 30-950 Kraków, Poland
gubala@science.uva.nl, m.kasztelnik@cyfronet.pl,
{malawski,bubak}@agh.edu.pl

Abstract. Recently, there is a growing need for an information technology solution to support a new methodology of scientific investigation, called system-level science. This paper presents a new approach to development and execution of collaborative applications. These applications are built as experiment plans with a notation based on the Ruby language. The virtual laboratory, which is an integrated system of dedicated tools and servers, provides a common space for planning, building, improving and performing in-silico experiments by a group of developers. The application is built with elements called gems which are available on the distributed Web- and Grid-based infrastructure. The process of application developments and the functionality of the virtual laboratory are demonstrated with a real-life example of the drug susceptibility ranking application from the HIV treatment domain.

Keywords: System-level science, e-Science, collaborative applications, virtual laboratory, ViroLab.

1 Introduction

Nowadays we observe a new approach to scientific investigations which, besides of analyses of individual phenomena, integrates different, interdisciplinary sources of knowledge about a complex system, to acquire understanding of the system as a whole. This innovative way of conducting research has recently been called *s*ystem-level science [1]. Biomedicine is an important example of such a field, requiring this new approach, which, in turn, must be accompanied by adequate information technology solutions. The complexity of challenges in the biomedical research and the growing number of groups and institutions involved creates more demand from that part of science for new, collaborative environments. Since biomedicine experts and research groups do not work in separation, more and more attention and effort is devoted to collaborative, inter-laboratory projects involving data and computational resources. The computer science aspects of this research, which include virtual groups, virtual organizations built around complex in-silico experiments and electronic data stores are also representative for other fields.

M. Bubak et al. (Eds.): ICCS 2008, Part I, LNCS 5101, pp. 56–65, 2008.
© Springer-Verlag Berlin Heidelberg 2008

An example of such a collaborative application in the virology domain, being built and used in complex simulations by many cooperating users, is drug resistance evaluation for HIV treatment [2] [3]. As the final results of this simulation is important for everyday practice of clinical virologists, there are efforts to provide it as a service via the web [4]. The ViroLab project dedicates substantial resources to deliver a decision support system to help medical doctors issue HIV drug prescriptions [5], as it develops the Drug Ranking System (DRS) [6].

Treatment decision support systems, like DRS, are used and developed by many people. There are many groups involved in HIV research and users representing various expertise levels inside these groups work to deliver a valid, reasonably complete and efficiently working solution. In turn, this objective can be achieved only if the entire endeavor is backed by a solid, innovative and well-integrated technology that is both generic enough to support users with distinct assignments, yet sufficiently focused.

In this paper we present the ViroLab Virtual Laboratory [7]: a collaborative, modern platform for system-level science. The laboratory is a set of dedicated tools and servers that form a common space for planning, building, improving and performing in-silico experiments in the virology domain.

In subsequent sections we show how such a complex application as DRS for HIV treatment may be designed, prepared and deployed for use in a collaborative fashion by people of different expertise levels, working towards a common objective. The next section presents an overview of related initiatives, and it is followed by a detailed explanation of operation of the proposed solution. Next, we discuss the novelty and innovation of this solution. We conclude with a summary and plans for future research.

2 Background

The need for information technology solutions supporting system-level science is indicated in the Cover Features by I. Foster and C. Kesselman [1].

Problem-solving environments and virtual laboratories have been the subject of research and development for many years [8]. Most of them are built on top of workflow systems. The LEAD [9] project is an example of a virtual laboratory for weather prediction applications; its main modules include a portal with user interfaces, a set of dedicated, distributed Grid resources and a workflow system which allows for combining the present resources together, to define task-specific processing. An example of an experimentation space is the Kepler [10] system which provides a tool for composing application workflows (which could, in particular, be experiments). In the MyGrid [11] environment, the Taverna system is used to compose complex experiment processes out of smaller, atomic building blocks. A rich library of those basic elements allows for great flexibility and numerous different solutions can be developed. Collaborative extensions have been provided by the MyExperiment project [12]. A recent overview of dedicated environments supporting development and execution of complex applications in biomedicine is presented in [13].

Most of problem solving environments and virtual laboratories are built on top of scientific workflow systems. The work on extension of the expressiveness of their programming models, interoperability, and on enabling access to different computing resources is still a subject of research [14]. In this paper, basing on the experience from workflow systems, we present an alternative approach to building systems supporting system-level science.

3 Drug Ranking Experiment in Virtual Laboratory

3.1 Experiment Pipeline

The process of experiment preparation in the collaborative ViroLab virtual laboratory is composed of well-defined steps (Fig. 1). At the beginning, the medical expert defines requirements for the experiment: what are its objectives, what kind of data and computation is required. Subsequently, the experiment developer, by analyzing these requirements, identifies the functional blocks that constitute the application. These computational elements of the ViroLab virtual laboratory are called *gems* and, in most cases, are available in the distributed Web- and Grid-based infrastructure. Otherwise, they have to be created, published and registered in the virtual laboratory, thus becoming available for other developers who may reuse them in their own experiments.

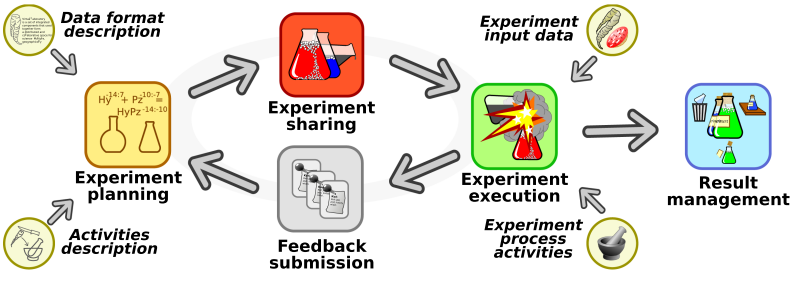

Fig. 1. Experiment pipeline: consecutive steps of an experiment in the virtual laboratory

Once all required computational activities are available, an experiment plan may be created. This purposed virtual laboratory provides an expressive, easy way to use a notation based on a high-level scripting language called Ruby [15]. The experiment plan is a Ruby script. The Ruby language provides a clear syntax, a full set of control structures and, as a result, it enables expressing experiments of arbitrary complexity levels in the form of scripts.

After the script is created and it fulfills (according to the developer) all the experiment requirements, it is stored in a dedicated repository and becomes available to other members of a given virtual organization. As a result, the scientist does not need to become familiar with scripting details, and may access the

virtual laboratory through a portal as well as browse and execute the available experiments using dedicated tools [16]. During application execution, provenance data is created and stored in dedicated provenance storage. This information is used by the scientist to search for interesting data and its origins [17].

The experiment script, as released by a developer, may not be optimal or may lack some functionalities. The virtual laboratory enables the scientist to easily communicate with the developer using a dedicated tool to submit user feedback, which is then used by the developer to produce a better version of the application.

The Drug Ranking System was created as a result of the experiment pipeline described above. Interpretation of the susceptibility of the HIV virus to particular drugs involves several steps. Some of these steps have to be performed manually (a blood sample has to be taken from the patient, the genetic material from the virus has to be isolated and sequenced). Once these steps are complete, a set of valid information is placed into a database. This material provides the required input for the DRS system. Knowing the nature of the experiment, a medical expert defines its structure (Fig. 2). A set of nucleotide sequences of the HIV virus has to be obtained. These sequences are then the subject of subtype detection algorithms and alignment processes, which create a list of mutations. This list is passed to the drug resistance expert system which returns virus-to-drug susceptibility values. When the experiment plan is defined, the developer can start searching for required gems or create them if they are not available, and implement the experiment plan.

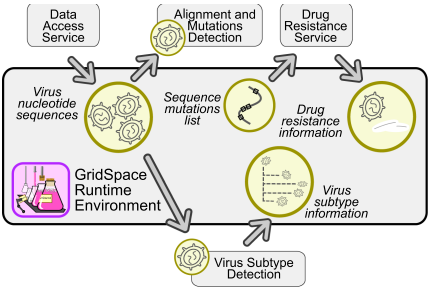

Fig. 2. "From Genotype to Drug Resistance" experiment

3.2 Development and Publication of Gems

As already hinted in Section 3.1, the basic computational building blocks of experiments are called *experiment gems*, which follows the name introduced for Ruby libraries (Ruby gems [15]). Although in the experiment script all such gems are represented with a uniform API based on the *Grid Object* abstraction [18], the gems themselves may be implemented using various technologies. Such an approach to integration of multiple technologies was motivated by the very vivid diversity of existing Grid- and Web-based middleware systems which may be used to provide access to computation. There are standard Web services, WS-RF, distributed component frameworks such as MOCCA [19] or ProActive [20], as well as large-scale job-processing systems such as EGEE LCG/gLite [21]. The goal of the Virtual Laboratory is to support gems using all these technologies.

Before a gem can be used in Virtual laboratory experiments, it has to be prepared by a gem developer. Fig. 3 shows schematically the required steps. After the interface of the gem is defined, it must be implemented using a selected technology. For simple, stateless interaction a standard Web service is the preferred solution. If a gem requires stateful (conversational) interaction and may benefit from dynamic deployment on remote resources, then implementing it as MOCCA component may be a good choice. Otherwise, if running a gem is a CPU-intensive and time-consuming task, it may be reasonable to implement it as a standalone program, which may be submitted as a job to such Grid infrastructures as EGEE or DEISA.

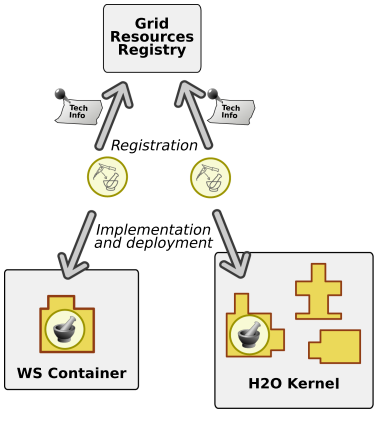

Fig. 3. Gem development: following interface definition, a gem has to be implemented, deployed in a specific technology and registered in GRR

Once the gem is developed, it has to be registered in the Grid Resource Registry (GRR), which is a central service of the Virtual Laboratory. GRR stores a technical description (*techinfo*) of each gem, including all information about the interface, implementation and details required to deploy or invoke the gem. It is possible to register gems which are published by third parties on the Web in the form of Web services: in that case it is enough to provide the WSDL file, describing the given service. Before actual registration takes place, the gem developer may write testing and debugging scripts which operate directly on the gem *techinfo*. Following registration in the GRR, the gem becomes visible to all experiment developers and can be shared throughout the Virtual Laboratory.

In the Drug Ranking experiment described in this paper, the gems include the Drug Resistance Service [5] and the RegaDB HIV sequence alignment and subtyping tools [22].

3.3 Experiment Planning, Scripting and Publishing

After the requirements of the experiment are defined and the missing *gems* developed, installed and registered in the GRR, the developer can start creating the experiment plan. The plan links data and computation into a working application. As presented in Section 3.2, the *gems* can be implemented using different technologies and, consequently, the creation of an experiment that connects these technologies, becomes complicated. To hide the complexity of the underlying middleware, a high-level object-oriented API called the Grid Operation Invoker – GOI [18] has been introduced. Uniform access to computations is enabled by providing three level of resource description (Fig. 4) – *Grid Object*, *Grid Object Implementation* and *Grid Object Instance*. During creation of the experiment plan only the highest level is used, although, if necessary, the

developer can define all the resource's technical details using one of the lower layers. The next problem that occurs while creating the experiment plan is access to the medical data. The virtual laboratory provides a high-level, secure API that enables querying different data sources with the Data Access Client – DAC (a client of the ViroLab Data Access Service [23]).

The Experiment Planning Environment (EPE [16]) supports creation of experiment plans. EPE is an RPC application based on the Eclipse platform which offers an integrated set of tools and a dedicated editor for writing experiment plans. The Domain Ontology Store (DOS) plug-in is a graphical browser that enables discovery of semantic information about the data and computational services. The Grid Resource Registry browser (GRR-browser) plug-in allows browsing registered services, their operations, input, output parameters and the attached documentation. These two plug-ins are integrated with the EPE experiment plan editor and between them provide a powerful mechanism for data and service discovery.

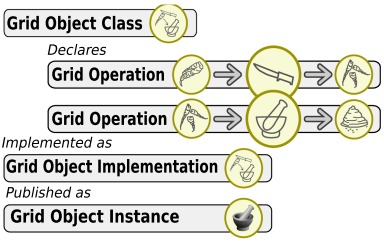

Fig. 4. Grid Object abstraction

The DRS experiment plan (see Fig. 5) was created using this set of tools. The developer knows that three computational services (responsible for subtyping, aligning and drug ranking) are required. Using the DOS plug-in all computational parts that return subtyped, aligned and drug-ranking results are found. Afterwards, by switching from DOS to the GRR-browser plug-in, the developer is able to see the details of the *gems* operations. The statements which result in the creation of selected resources, are added to the experiment plan directly from the browser plug-in. EPE is also integrated with the Experiment Repository version control system (based on Subversion), which facilitates collaboration between developers. As a result, many developers can work on single experiment plan, sharing it with other members of a virtual organization.

The last step in experiment plan development is to make it available to the medical expert who is the application end user. The release plug-in, integrated with EPE, simplifies the experiment plan release process. During this process a new branch in the SVN repository is created and the experiment plan is copied with a unique version number and licence file.

3.4 Execution of Experiment

Both GOI and DAC are elements of the GridSpace engine (GSEngine [24]) which provides runtime support. It allows executing experiment plans locally on the developer's machine, or remotely, on the server (Fig. 6). EPE is integrated with the runtime, thus making experiment plan creation and testing easy. For the medical expert who is the end user of the created experiments, a dedicated Web based application (Experiment Management Environment – EMI [16]) is created,

```
patientID = DataRequester.new.getData("Provide patient\'s ID")
region = DataRequester.new.getData("Region (\"rt\" or \"pro\")")

nucleoDB = DACConnector.new("das",
   "angelina.hlrs.de:8080/wsrf/services/DataAccessService","","","")
sequences = nucleoDB.executeDistributedQuery(
   "select nucleotides from nt_sequence where
    patient_ii=#{patientID.to_s};")

subtypesTool = GObj.create("RegaDBSubtypesTyool")
subtypes = subtypesTool.subtype(sequences)
puts "Subtypes: #{subtypes}"

mutationsTool = GObj.create("RegaDBMutationsTool")
mutationsTool.align(sequences, region)
mutations = regaDBMutationsTool.getResult

drs = GObj.create("DrugResistanceService")
puts drs.drs("retrogram", region, 100, mutatations)
```

Fig. 5. Listing of the decision support system experiment plan

hiding the complexity of the technology layer. It allows browsing information about the released experiment plans' versions (their names, descriptions, licences) and executes them. Thanks to close integration with the GSEngine, interaction between users and experiment plans is realized. This mechanism allows receiving additional information from the user during script execution. For example, the DRS experiment (Fig. 5) requires two pieces of input data from the user: patientId – necessary to receive patient sequences from the medical database, and the region – required by the Drug Resistance Service.

4 Innovation

The ViroLab virtual laboratory provides an environment to collaboratively plan, develop and use biomedical applications. The main innovation of the presented platform is dedication to multi-expertise task-oriented groups. Tools are provided for technical personnel, developers and administrators whose task is to maintain and enrich the experiment space. Additionally, there are tools that help virologists and healthcare providers perform their treatment-related tasks. The respective objectives and actions of these user groups are combined together with a set of remote services, information stores and other integration techniques. In this way the laboratory helps entire research teams (both traditional and virtual, Internet-wide ones) reach their scientific and professional goals more effectively.

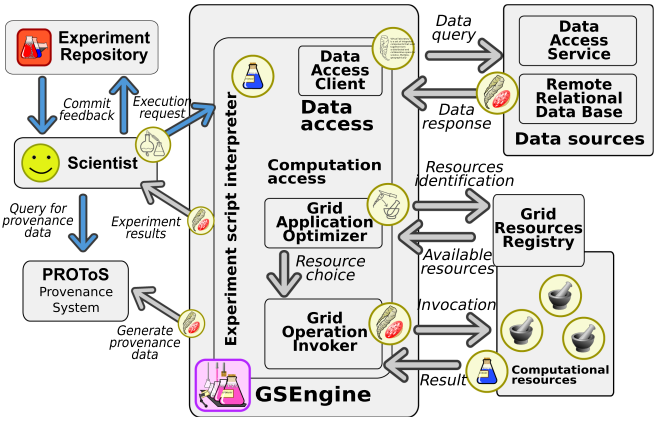

Fig. 6. GSEngine - collaborative environment for experiment plan execution

Another innovative feature of the presented solution is stress on the generality of provided solutions in the middleware layer. The GridSpace runtime components are designed to support various remote computation technologies, programming models and paradigms. Together with this generic and multi-purpose solution, the environment provides a set of user-oriented tools that allow customizing, arranging and populating the virtual laboratory space with content and solutions specific to certain application domains. It is a method of harvesting the end users' high creativity to help them co-create their environment rather than tailoring ready-to-use solutions. Since the e-Science domain is evolving very quickly, we argue that this model of a generic platform with specific content is best suited for technically knowledgeable teams of scientists. The described concept of independent analysis gems and data sources as well as the scripting glue used to combine them in desired experiments, ensures easy reconfigurability, extensibility and enables ad-hoc recomposition of laboratory content and applications.

The presented platform facilitates fast, close cooperation of developers and users on experiments. Since an in-silico experiment is subject to frequent changes, modifications and enhancements, the traditional software model of releases, downloads, deployments and bug reports is not effective enough. Instead, the ViroLab experiment planning and publishing model encourages quick, agile software releasing and a corresponding versioning scheme. In this model, enhancement reports can be provided right away in the experiment execution tool and they are immediately visible to all interested programmers, who may publish new experiment versions which are also immediately ready to use by all interested scientists in the group. The additional licensing and terms-of-use information, always attached to experiments, saves the end users time that would otherwise be spent on finding out whether and how the results of experiments may be used and published.

5 Summary

The applicability and suitability of the virtual laboratory was demonstrated with the real-life example of the drug susceptibility ranking application from the HIV treatment domain. The main innovation of this work is the novel design of the virtual laboratory that allows for truly collaborative planning, development, preparation and execution of complex data acquisition and analysis applications, so crucial for the biomedicine field.

In the proposed environment people of different occupations, both advanced script developers and scientists can effectively and collaboratively conduct their respective tasks, contributing to a common goal. The current version of the presented platform, rich documentation and tutorials are available from the ViroLab virtual laboratory site [7].

The laboratory is under continuous development. One of the most important features to be added is a module for management of results produced by experiments. Effort is being invested in semantic descriptions of data and computations. Consequently, finding interesting information will become easier and the corresponding middleware will be able to track the provenance of results in an application-specific way. This, in turn, will lead to future experiment repeatability. The listed functionality aspects are of great importance for system-level science.

Acknowledgements. This work was partially funded by the European Commission under the ViroLab IST-027446, the IST-2002-004265 Network of Excellence CoreGRID projects, the related Polish SPUB-M grant and the Foundation for Polish Science. The Authors are grateful to Piotr Nowakowski for his comments and suggestions.

References

1. Foster, I., Kesselman, C.: Scaling system-level science: Scientific exploration and it implications. Computer 39(11), 31–39 (2006)
2. Vandamme, A.M., et al.: Updated european recommendations for the clinical use of hiv drug resistance testing. Antiviral Therapy 9(6), 829–848 (2004)
3. Rhee, S., et al.: Genotypic predictors of human immunodeficiency virus type 1 drug resistance. In: Proceedings of National Academy of Sciences of the United States of America. National Academy of Sciences, vol. 103 (2006)
4. Rhee, S., et al.: Human immunodeficiency virus reverse transcriptase and protease sequence database. Nucleic Acids Research 31(1), 298–303 (2003)
5. Sloot, P.M.A., 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)
6. Sloot, P.M.A., Tirado-Ramos, A., Bubak, M.: Multi-science decision support for hiv drug resistance treatment. In: Cunningham, P., Cunningham, M. (eds.) Expanding the Knowledge Economy: Issues, Applications, Case Studies, eChallenges 2007, pp. 597–606. IOS Press, Amsterdam (2007)
7. ViroLab Virtual Laboratory, `http://virolab.cyfronet.pl`

8. Rycerz, K., Bubak, M., Sloot, P., Getov, V.: Problem solving environment for distributed interactive simulations. In: Gorlatch, S., Bubak, M., Priol, T. (eds.) Achievements in European Reseach on Grid Systems. CoreGRID Integration Workshop 2006 (Selected Papers), pp. 55–66. Springer, Heidelberg (2008)
9. Droegemeier, K., et al.: Service-oriented environments in research and education for dynamically interacting with mesoscale weather. IEEE Computing in Science and Engineering (November-December 2005)
10. Altintas, I., Jaeger, E., Lin, K., Ludaescher, B., Memon, A.: A web service composition and deployment framework for scientific workflows. ICWS 0, 814–815 (2004)
11. Stevens, R.D., et al.: Exploring williams-beuren syndrome using mygrid. Bioinformatics 1(20), 303–310 (2004)
12. MyExperiment: myexperiment website (2007), `http://myexperiment.org`
13. Aloisio, G., Breton, V., Mirto, M., Murli, A., Solomonides, T.: Special section: Life science grids for biomedicine and bioinformatics. Future Generation Computer Systems 23(3), 367–370 (2007)
14. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the Challenges of Scientific Workflows. IEEE Computer 40(12), 24–32 (2007)
15. Thomas, D., Fowler, C., Hunt, A.: Programming Ruby – The Pragmatic Programmer's Guide, Second Edition. The Pragmatic Programmers (2004)
16. Funika, W., Harężlak, D., Król, D., Pęgiel, P., Bubak, M.: User interfaces of the virolab virtual laboratory. In: Proceedings of Cracow Grid Workshop 2007, ACC CYFRONET AGH, pp. 47–52 (2008)
17. Baliś, B., Bubak, M., Pelczar, M., Wach, J.: Provenance tracking and querying in virolab. In: Proceedings of Cracow Grid Workshop 2007, ACC CYFRONET AGH, pp. 71–76 (2008)
18. Bartynski, T., Malawski, M., Gubala, T., Bubak, M.: Universal grid client: Grid operation invoker. In: Proceedings of the 7th Int. Conf. on Parallel Processing and Applied Mathematics PPAM 2007. LNCS. Springer, Heidelberg (to appear, 2008)
19. Malawski, M., Bubak, M., Placek, M., Kurzyniec, D., Sunderam, V.: Experiments with distributed component computing across grid boundaries. In: Proceedings of the HPC-GECO/CompFrame workshop in conjunction with HPDC 2006, Paris, France (2006)
20. Baduel, L., Baude, F., Caromel, D., Contes, A., Huet, F., Morel, M., Quilici, R.: Programming, Deploying, Composing, for the Grid. In: Grid Computing: Software Environments and Tools. Springer, Heidelberg (2006)
21. EGEE Project: Lightweight middleware for grid computing (2007), `http://glite.web.cern.ch/glite`
22. de Oliveira, T., et al.: An automated genotyping system for analysis of HIV-1 and other microbial sequences. Bioinformatics (2005)
23. Assel, M., Krammer, B., Loehden, A.: Management and access of biomedical data in a grid environment. In: Proceedings of Cracow Grid Workshop 2006, pp. 263–270 (2007)
24. Ciepiela, E., Kocot, J., Gubała, T., Malawski, M., Kasztelnik, M., Bubak, M.: Gridspace engine of the virolab virtual laboratory. In: Proceedings of Cracow Grid Workshop 2007, ACC CYFRONET AGH, pp. 53–58 (2008)