A METHODOLOGY FOR INTEGRATING NEW SCIENTIFIC DOMAINS AND APPLICATIONS IN A VIRTUAL LABORATORY ENVIRONMENT

E. C. Kaletas, H. Afsarmanesh, L. O. Hertzberger
University of Amsterdam, Informatics Institute
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Keywords: Virtual laboratory, scientific domain integration, scientific application integration, integration methodology

Abstract: Emergence of advanced complex experiments in experimental sciences resulted in a change in the way of experimentation. Several solutions have been proposed to support scientists with their complex experimentations, ranging from simple data portals to virtual laboratories. These solutions offer a variety of facilities to scientists, such as management of experiments and experiment-related information, and management of resources. However, issues related to adding new types of experiments to proposed support environments still remain untouched, causing inefficient utilization of efforts and inadequate transfer of expertise. The main topic of this paper is to present *a methodology for integrating new scientific domains and applications in a multi-disciplinary virtual laboratory environment*. In order to complement the methodology with the right context, the paper also presents an experiment model that uniformly represents scientific experiments, data models for modelling experiment-related information, and mechanisms for the management of this information.

1 INTRODUCTION

In recent years, technological advances and systematic growth of research efforts in experimental science domains, such as life sciences, have achieved breakthroughs that go beyond the imagination of mankind even a decade ago. Advances in the laboratory techniques and the introduction of highly automated laboratory instruments have caused both the automation of many steps in scientific experiments and the generation of very large amounts of data, leading to a change in the way that scientists perform their scientific research. However, new ways of experimentation pose many requirements and challenges to both scientists and developers of supporting tools and infrastructures. The following can be mentioned among these challenges, which create a burden on the shoulders of scientists and developers of supporting tools:

- Increasing number and complexity of laboratory instruments and procedures require significant effort to perform an experiment. For example, clone preparation phase of a microarray experiment contains up to 51 steps.
- Large amounts of data are generated during experiments. For instance, material analysis experiments for complex surfaces typically generate 20 images per day where data size goes up to 100 MB per image (Frenkel et al., 2001).
- Heterogeneity causes difficulties in modelling and storage of experimental information. For instance, 281 biological database resources in 18 different categories were listed in 2001 (Baxevanis, 2001).
- Lack of standards for modelling and representation of information causes wasted efforts for developing *specific solutions* to overcome *common problems* and difficulties in comparing the results obtained in multiple experiments.

New solution methods and problem solving techniques are needed to address the challenges. However, addressing these challenges require long-term multi-disciplinary efforts between scientists from both experimental science domains and computer science. In experimental science domains, scientists involved in these experimentations must both advance their models and mechanisms supporting the analysis of resulted information, and design innovative investigation approaches leading to discoveries. From the computer science point of view, however, the complexity of emerging experimentations necessitates that the support environment be *generic* and constitute a *horizontal base infrastructure*, on top which

C. Kaletas E., Afsarmanesh H. and O. Hertzberger L. (2004).
A METHODOLOGY FOR INTEGRATING NEW SCIENTIFIC DOMAINS AND APPLICATIONS IN A VIRTUAL LABORATORY ENVIRONMENT.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 265-272
DOI: 10.5220/0002634802650272

vertical software tools can be developed, each providing a specific service for the experimentations.

Several solutions have been proposed to support scientific experimentations, ranging from simple Web-based query interfaces for scientific databases to complex problem solving environments and virtual laboratories. Some of these environments target specific applications while others are offering more generic support. However, issues such as extension of support environments with facilities for emerging scientific experiments (i.e. integration of *vertical software tools* in the *horizontal support environments*) still remain untouched. More precisely, a **methodology** for integrating new scientific domains and applications in a support environment is still lacking; causing inefficient utilization of efforts and inadequate transfer of expertise. Yet, a further challenge would be to provide a methodology for integrating diverse heterogeneous scientific applications in a *multidisciplinary* support environment.

The main topic of this paper is to present a methodology for integrating new scientific domains and applications in a multi-disciplinary virtual laboratory environment. Nevertheless, integration of new domains and applications in a complex environment like virtual laboratory requires a common understanding of scientific experiments, availability of both well-defined data models for modelling experiment-related information and mechanisms for the management of this information. The remaining of this paper is structured as follows: First an overview of existing and emerging support environments is provided in Section 2. In Section 3, general structure of scientific experiments is presented, followed by a description of an experiment model. Then 'process data flows' are introduced in this section as a tool for modelling experiments. Section 4 focuses on a specific support environment, namely the VLAM-G virtual laboratory environment. Section 5 further focuses on VIMCO, the information management platform of VLAM-G, and addresses issues related to modelling and management of information related to scientific experiments. The proposed methodology is described in Section 6. Finally, Section 7 concludes.

2 SUPPORT ENVIRONMENTS

This section describes different types of support environments for scientific experimentations. The remaining of this paper, however, focuses on virtual laboratories.

Science Portals. *Science Portal*s (Ashby et al., 2001), (Pierce et al., 2002) are emerging as convenient mechanisms for providing a single point of access with familiar and simplified interfaces to a specific set of resources that are of importance to a specific scientific community. Resources can be, for instance, computational, storage, networking resources, electronic whiteboards, or a digital library. These resources are usually made available to users as services. Users can make use of these services either in a stand-alone manner using only one service at a time, or in a collective manner using a number of services simultaneously. Science portals only provide a uniform means for accessing resources. Users themselves are responsible for the correct and efficient usage of the available resources.

Problem Solving Environments. A *Problem Solving Environment* (PSE) (Allen et al., 2001), (Schuchardt et al., 2002) is a computer system that provides all the computational facilities needed to solve a target class of problems (Gallopoulos et al., 1994). Common characteristics of PSEs include among others a target class of science or engineering problems, natural appearance and ease of use, provision of multiple solution paths or algorithms in the target areas, and availability of parameterized algorithms. Users can be assisted through automatic and/or semiautomatic selection of a proper solution method from the available set of solution methods, or by providing ways to easily incorporate new solution methods. Although some examples of generic PSEs do exist such as Cactus (Allen et al., 2001), most of the current PSEs target a specific class of problems.

Virtual Laboratories. A *Virtual Laboratory* (VL) (Afsarmanesh et al., 2002), (Messina, 2003) provides an electronic workspace for distributed collaboration and experimentation in research, to generate and deliver results using distributed information and communication technologies (Vary, 2000). It supports an aggregation of people who pursue a related set of research activities and share resources, where the resources including the people may be geographically distributed and associated with different institutions. Therefore, virtual laboratories bring together best combination of skills, expertise, and tools to carry out the same type of research that is done in a single real laboratory (Messina, 2003).

At present, the concept of 'virtual laboratory' has been associated with several different meanings, depending on the wide variety of application interests and fields (e.g. education, games, science, engineering, manufacturing, etc.). On the other hand, there are some recent VL projects that target the provision of generic solutions and offer a wide variety of functionality, and that are applicable to a wider set of problems (Afsarmanesh et al., 2002), (Catlin et al., 2000). One such virtual laboratory that offers generic solutions to

scientists is the Grid-based Virtual Laboratory Amsterdam (VLAM-G), which is described in Section 4.

3 MODELLING EXPERIMENTS

In a VL environment, it is natural that all activities are centered around *experiment*s. The experiment model underlying a VL outlines the overall approach that the VL follows for supporting scientific experiments. Furthermore, the experiment model allows a methodological definition of complex experiments in a problem domain. Thus, the experiment model lies in the core of a VL. This section focuses on the modelling of scientific experiments and describes an experiment model, by first presenting high-level general structure of scientific experiments, then providing a description of the experiment model. This section then introduces 'process data flows' as a tool for modelling experiments.

3.1 General Structure of Experiments

A scientific experiment consists of a number of *experiment component*s. Components of an experiment correspond to activities, physical entities, or data elements involved in that experiment. Every experiment is part of a *project*, which may contain one or more experiments that are related to each other (Figure 1).

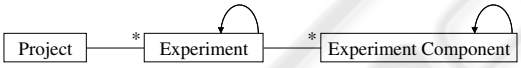

Figure 1: General structure of scientific experiments

A **project** is a grouping of related experiments. Experiments in a project may follow an order (e.g. time course experiments), or may be unordered (e.g. comparison experiments).

Experiments are composed of three kinds of **experiment components** (see Figure 2 for experiment components involved in a typical microarray experiment). *Physical entities* are used during laboratory activities or during instrumentation. *Activities* may represent laboratory activities, instrumentations, or computational processes. *Data elements* in turn correspond to both raw data generated by instruments and input data to computational processes. The generic descriptive elements represent the components that are common to all experiments and provide descriptive information about them, such as *organism*, *gene*, or information about hardware and software tools (e.g. *scanner*). Such descriptive information does not change from one experiment to another.

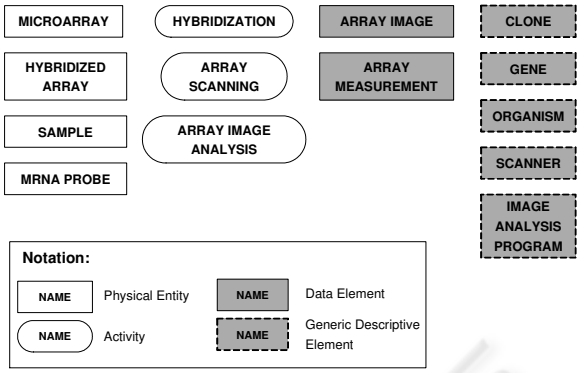

Figure 2: Components for microarray experiments

3.2 The Experiment Model

An experiment model is developed to uniformly represent scientific experiments from different domains. A brief summary of the experiment model is provided below. For more information on the experiment model, please refer to (Kaletas et al., 2003).

The experiment model consists of two main components, namely *experiment procedure* and *experiment context* (Figure 3).

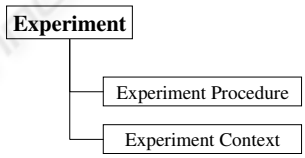

Figure 3: The experiment model

An **experiment procedure** defines the approach taken to solve a particular scientific problem, by defining the experiment components that are typically involved in the experiments of the same type. Thus, an experiment procedure standardizes the experimental approach for experiments of the same type.

An **experiment context**, on the other hand, describes the solution. An experiment context is an instantiation of an experiment procedure. It describes the accomplishment of a particular experiment, by providing descriptions of each experiment component involved in the experiment. Thereby, it provides the context for that particular experiment.

Processing and analysis of large data sets constitute an important part of scientific experiments, and hence require special attention. In an experiment, during its processing and analysis, data flows from one computational process to another. This data flow is represented by a directed graph. Nodes in the graph are computational processes, while the connecting arcs are the data flowing through the processes. This

data flow graph is called as *computational processing*. An experiment procedure may contain computational processes. Consequently, particular experiments that are instantiating this procedure contain descriptions of these processes in their experiment contexts.

3.3 Process Data Flows

Given a scientific domain, the first step in modelling the experiment-related information is to perform a detailed analysis of the experiment components for each experiment in that domain. Result of the analysis is a conceptual model, which includes a comprehensive component-by-component definition of experiments in the domain.

In this conceptual model, every component involved in an experiment, the properties of each component, and the relationships among the components are defined. Such a detailed and comprehensive definition of an experiment is referred to as the *Process Data Flow* (PDF). Figure 4 presents the microarray PDF as an example using the same notation as Figure 2 for experiment components. This example microarray PDF depicts the order of components in the experiment (i.e. the experiment flow) and also the relationships between components and generic descriptive elements in microarray experiments.

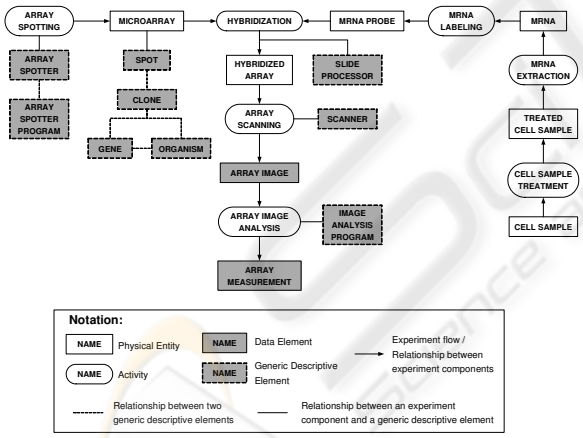

Figure 4: Process-Data Flow for microarray experiments

4 VLAM-G

The Dutch VLAM-G (Grid-based Virtual Laboratory Amsterdam) project (Afsarmanesh et al., 2002), (Afsarmanesh et al., 2001) provides the main context for the work presented in this paper. VLAM-G is a multi-disciplinary virtual laboratory environment that provides the required generic environment for multi-disciplinary research in experimental science domains.

The two initial application cases of VLAM-G were the DNA microarray application from the life sciences domain and the material analysis for complex surfaces application (MACS) (Frenkel et al., 2001) from the physics domain. The work described in this paper has been initiated and motivated by the need to support the requirements of both these initial application cases and any future applications within the VLAM-G.

VLAM-G allows its users to:

- perform multi-disciplinary, collaborative experiments in a uniform, integrated environment,
- complement their in-vitro experiments with in-silico experiments,
- define customized experimental procedures and analysis flows,
- reuse generic software components, and
- share hardware, software, storage, networking resources as well as knowledge and experience.

The architecture of the VLAM-G and interaction among its components are shown in Figure 5. Main components of the architecture are described below.

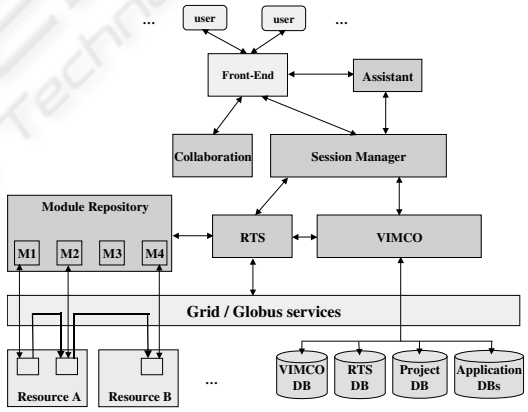

Figure 5: VLAM-G architecture

Front-End is the user environment of the VLAM-G. Users interact with the VLAM-G only through the Front-End. The Front-End consists of a number of graphical user interfaces (GUIs), which present the VLAM-G functionality to its users in a uniform way.

Session Manager acts as a gateway to the VLAM-G server side. It manages the active user sessions, and is responsible for coordinating the interactions among the VLAM-G components.

The distributed computing and networking resources on the Grid are made available to VLAM-G users through the **Run Time System**. Generic

VLAM-G processing functionality and application-specific problem solvers are presented to its users as self-contained *software entities*. RTS provides an API which encapsulates the Grid computing code within a simple interface. Users compose their *computational processings* by attaching a number of software entities to each other, which are then executed by the RTS on the Grid environment.

Collaboration component of the VLAM-G enables simultaneous collaborative design and execution of experiments. The **Assistant** assists users during the design of an experiment, for instance, by suggesting the most efficient software to perform a specific task. Implementation of these two components is planned for future work.

Module Repository is a persistent storage for binaries of software entities.

The *Virtual Laboratory Information Management for Cooperation* (**VIMCO**) is the information management platform of VLAM-G. VIMCO is described in details in the following section.

5 VIMCO

This section describes the VIMCO facilities for modelling and manipulation of information related to scientific experiments in the VLAM-G environment.

5.1 Modelling Information Related to Scientific Experiments

Modelling experiment-related information handled in a VL is important, since the employed data models have a direct influence on the architectural components of the VL and their functionalities. Information related to scientific experiments include *experiment information* (e.g. experiment procedures, experiment contexts, computational processings), *user information* (e.g. information about users and user roles, access rights definitions), and *VL-related information* (e.g. session information, information about the services provided by the VL). This section builds on the experiment model presented earlier in this paper, and describes the modelling of information related to scientific experiments in VL.

5.1.1 Experiment Information

In this subsection, data models developed for the modelling of different types of information about scientific experiments are described. Note that all data models described below are object-oriented. Due to space limitations, however, UML diagrams and detailed descriptions of the data models are omitted. Details of the data models presented here can be found in (Kaletas et al., 2003).

Modelling Experiment Procedures. The *Procedure Data Model* facilitates the representation and storage of experiment procedures, and consists of the following classes: `Procedure`, `ProcedureElement`, `ProcedureConnection`, and `ProcedureGUI`. An instance of the `Procedure` class contains the general information about a procedure, including its version. Multiple versions of the same procedure can exist. Every `Procedure` object consists of a set of `ProcedureElement`s, which correspond to experiment components. For each procedure element (and hence experiment component), there is a corresponding data type in the application database schema to hold the information about that experiment component. The corresponding data type in the database schema for a procedure element is represented by the `className` attribute. `isProcessing` attribute of a procedure element indicates whether the element actually corresponds to a *processing*. Instances of `ProcedureConnection` class correspond to the relationships between classes in the database schema, indicated by the `relName` attribute. Finally, `ProcedureGUI` objects keep information about the graphical properties of a procedure for displaying it to users. Procedure definitions are stored in the application databases.

Modelling Experiment Contexts. A context is an instance of a procedure, and the procedure elements correspond to data types in an application database. This means, a context actually consists of a number of objects in the application database, hence, modelling the application database schema for the domain of an experiment is sufficient for modelling the contexts of that experiment.

Using the general structure of scientific experiments, a base data model for experimental information called **Experimentation Environment Data Model** (EEDM) is designed. EEDM covers the common aspects of experiments; namely *scientist* owning the experiment, *input*, *activities* applied on the input, *output* obtained from an activity, *hardware* and *software* used during the experiment, *conditions* and *parameters* for the activities, devices and software, and a ***recursive flow of processes and data*** where a specific order is followed during an experiment.

EEDM provides generic constructs for these common aspects of scientific experiments, which are extended (sub-typed) in application databases for modelling experiment-specific information.

Modelling Software Entities and Computational Processings. A computational processing covers both descriptions of the executables used (i.e. software entities) and run-time information for the processing. *Software entities* represent self-contained executable programs. They are mainly characterized by the tasks that they perform, their input and output data types, parameters, and run-time requirements. For every executable, its developer specifies the run-time requirements to help a user to decide which software entities to use in his/her processing. A user designs a *computational processing* by attaching a number of software entities to each other through their input and output ports. This design is stored in RTS DB for future references and for possible re-runs.

5.1.2 User Information

User information is used in a VL for security and access control, and for administration purposes. In addition, application databases contain user information to capture, for instance, the owner of a physical entity or data element (e.g. microarray or array image), or operator of an instrument (e.g. array scanner). As such, information about VL users is defined and stored in all VL databases.

Every VL user has a unique *username*, and assumes a *role*. Roles are used to define and enforce access rights for groups of users. Access to information in the VL is controlled through a set of *restrictions*. It is considered that by default every user has all privileges on all available information. Depending on the role that a user assumes, these unlimited privileges are restricted by a set of 'restrictions'. Below, the specific roles defined for VL users are summarized.

- **Application users** are the actual users of the VL. An application user is typically associated with a scientific domain (e.g. molecular biology), and can only access the application database for this domain. S/he has full control on her/his contexts and processings but only read access to procedures.
- **Domain experts** are application users who have extensive knowledge and experience in a given domain and on the experiments being performed in that domain. They are responsible for modelling application databases (together with Information Management Admin), designing experiment procedures, defining laboratory protocols, and registering descriptions of software entities.
- **Tool developers** are the developers of software entities (i.e. vertical software tools).
- **Administrators** are responsible for the proper management and operation of the VL. *Information Management Admin* is responsible for modelling application database schemas and for maintaining these databases. *VL Admin* is responsible for resource management, infrastructure maintenance, and user management.

5.1.3 VL-Related Information

During its operation, the VL itself manages and manipulates certain types of information, such as information about active sessions and information about the services it provides.

Session management is an important functionality in a VL for controlling and coordinating user activities. All user activities occur within a session. For recovery reasons, VIMCO provides persistence to session information (in VIMCO DB). A `Session` class is defined to store the session information, which contains owner of the session, observers in case of a collaborative session, and the procedures, contexts, and processings being used in the session.

The *services* that are made available by the VL to its users are highly dependent on the existing experiment-related information in the VL. For instance, availability of a 'microarray experiment procedure' allows a VL to provide 'making microarray experiments' as a service to its users. In order to provide a list of services to users, VL makes use of the experiment-related information stored and managed by VIMCO. The list of available services is generated by VIMCO by extracting the necessary information from what is available in the databases.

5.2 Managing Information Related to Scientific Experiments

VIMCO provides the necessary functionality and mechanisms for the manipulation of different types of information that it manages. Very briefly, VIMCO supports the following functionality:

- managing experiment procedures, contexts, and computational processings;
- defining and enforcing usage policies for experiment contexts;
- managing user information;
- defining and enforcing access rights;
- managing VL-related information;
- uniformly accessing multiple, heterogeneous application databases; and
- sharing and exchanging information among collaborating partners.

6 THE PROPOSED METHODOLOGY

Lack of a methodology for supporting new scientific domains and applications in a virtual laboratory causes inadequate transfer of expertise and hence inefficient utilization of efforts. In this paper, several different aspects of a virtual laboratory environment were addressed with a focus on its information management component. In specific, an experiment model is presented for uniformly representing diverse scientific experiments. Furthermore, several data models for modelling different types of information related to scientific experiments, and functionality for the actual management of this information are provided. The results obtained and the experience gained during the VLAM-G project, and especially during the two application cases, are used for defining a *methodology* for integrating new scientific domains and applications in VL. The methodology provides step-by-step guidance to domain experts, tool developers and administrators during the process of adding new domains, applications and resources to VL.

The methodology provided in this section is generic and reusable; it can be uniformly applied to different scientific domains and to different scientific applications. This is partially due to the genericness and reusability of the models described here.

Following subsections describe the methodology.

6.1 Integration of a New Domain

Integration of a new domain in VL covers the development of a domain-specific application database, integrating and registering this database into the VL and its information management platform, and defining and registering the domain users (application users). Steps to take when integrating a new domain in VL are enumerated below.

1. *Study different types of experiments* being performed in the domain, *to the level of detail of single components* involved in the experiments and the related information for each component. This step is realized by the experts in the domain, in close collaboration with an Information Management Admin (IM Admin).
2. *Generate process data flows* (PDFs) for each experiment type. Discuss the PDFs with the experts from the domain, make any necessary modifications, and eventually confirm the PDFs. This step is performed by a Domain Expert in close collaboration with an IM Admin.
3. For each PDF, *map the PDF elements to the base data types in the EEDM*, and model each PDF element by sub-typing one of the base EEDM data types. The result is the process data flows with elements that are defined as sub-types of base EEDM types. These PDFs form the base for the application database schema for the domain. This step is performed by an IM Admin together with a Domain Expert.
4. *Cross-study different experiment types* defined in terms of extended EEDM constructs, and identify the elements that are common to different experiments (e.g. organism info). Identify the relationships between components in different experiments and these common elements. Confirm the result with experts in the domain. The result is the *core schema for the application database*. This step is performed by a Domain Expert and an IM Admin.
5. *Map the core database schema to a data definition language* (e.g. Object Definition Language). Add any other data models required by the information management platform (e.g. procedure data model). This step is performed by an IM Admin.
6. *Create the application database* by loading the schema defined in the data definition language. This step is performed by an IM Admin.
7. *Register the database to the information management platform* by providing the required information about the new data source. This step is performed by an IM Admin.
8. *Define and register the domain users* using the user management functionality. This step is performed by a VL Admin.

6.2 Integration of a New Application

Once the new domain is integrated, new applications from that domain can be integrated. Steps to take when integrating new applications are given below.

1. *Define experiment procedures for different types of experiments* that will be offered to users of this application. Graphical editors provided by the VL can be used for procedure definition. This activity is performed by a Domain Expert.
2. *Develop software entities for application specific functionality* (e.g. analysis tool for microarray data). This step is performed by a Tool Developer.
3. After the quality control applied to software entities, *store their definitions* in the information management platform, and *register them to VL*. These activities are performed by a Domain Expert.
4. Using the access rights management functionalities, *define the user roles* and *restrictions* for each role on the data types of this domain. Assign one role to each domain user. These activities are performed by a Domain Expert.

7 CONCLUSIONS

The main subject of this paper was to provide a methodology for integrating emerging scientific domains and applications in a virtual laboratory environment. Integration of new domains and applications in a complex virtual laboratory environment requires a common understanding of scientific experiments, availability of both well-defined data models for modelling experiment-related information and functionality / mechanisms for the management of this information. Therefore, in this paper:

1. an experiment model is introduced that is capable of uniformly representing different aspects of heterogeneous scientific experiments;
2. several data models are provided for representing the experiment-related information, which are uniform and reusable (to model information related to heterogeneous experiments), and open, flexible and extendible (to improve the developed models in the future when needed and to model new types of experimental information);
3. mechanisms for managing the experiment-related information in a VL are mentioned, that are generic and reusable (to support uniform management of heterogenous experimental information); and
4. a methodology is defined for integration of new domains and applications in VL based on the results obtained and experience gained during the VLAM-G project presented here, which provides a step-by-step guidance to domain experts, tool developers and administrators during the process of adding new applications / resources to the VL.

The methodology defined in this paper is generic and can be implemented by different support environments in different ways. During the course of the VLAM-G project, the methodology has been applied to the two initial application cases (the DNA microarray and MACS applications), and the received feedback was used to further improve/refine the methodology. Currently, two databases for these applications together with experiment procedures and some analysis tools are provided to VLAM-G users. Furthermore, development of a new application for gene sequence analysis studies using this methodology is ongoing.

REFERENCES

Afsarmanesh, H., Belleman, R. G., Belloum, A. S. Z., Benabdelkader, A., van den Brand, J. F. J., Eijkel, G. B., Frenkel, A., Garita, C., Groep, D. L., Heeren, R. M. A., Hendrikse, Z. W., Hertzberger, L. O., Kaandorp, J. A., Kaletas, E. C., Korkhov, V., de Laat, C. T. A. M., Sloot, P. M. A., Vasunin, D., Visser, A., and Yakali, H. H. (2002). Vlam-g: A grid-based virtual laboratory. *Scientific Programming*, 10(2):173–181.

Afsarmanesh, H., Kaletas, E. C., Benabdelkader, A., Garita, C., and Hertzberger, L. O. (2001). A reference architecture for scientific virtual laboratories. *Future Generation Computer Systems*, 17(8):999–1008.

Allen, G., Benger, W., Dramlitsch, T., Goodale, T., Hege, H. C., Lanfermann, G., Merzky, A., Radke, T., Seidel, E., and Shalf, J. (2001). Cactus tools for grid applications. *Cluster Computing*, 4(3):179–188.

Ashby, J. V., Bicarregui, J. C., Boyd, D. R. S., van Dam, K. K., Lambert, S. C., Matthews, B. M., and O'Neill, K. D. (2001). A multidisicplinary scientific data portal. In *Proceedings of the 9th International Conference and Exhibition on High-Performance Computing and Networking (HPCN Europe 2001)*, pages 13–22.

Baxevanis, A. D. (2001). The molecular biology database collection: An updated compilation of biological database resources. *Nucleic Acids Research*, 29(1):1–10.

Catlin, A. C., Gaitatzes, M., Houstis, E., Ma, Z., Markus, S., Rice, J. R., Wang, N.-H., and Weerwarana, S. (2000). *Enabling Technologies for Computational Science - Frameworks, Middleware and Environments*, chapter 24, pages 301–313. Kluwer Academic Publishers.

Frenkel, A., Afsarmanesh, H., Eijkel, G. B., and Hertzberger, L. O. (2001). Information management for material science applications in a virtual laboratory. In *Proceedings of the 12th International Conference on Database and Expert Systems Applications (DEXA 2001)*, pages 165–174.

Gallopoulos, E., Houstis, E., and Rice, J. R. (1994). Computer as thinker/doer: Problem-solving environments for computational science. *Computing in Science and Engineering*, 1(2):11–23.

Kaletas, E. C., Afsarmanesh, H., and Hertzberger, L. O. (2003). Modelling multi-disciplinary scientific experiments and information. In *Proceedings of the Eighteenth International Symposium on Computer and Information Sciences (ISCIS03)*, pages 75–82.

Messina, P. (2003). The emergence of virtual laboratories for science and engineering - igrid2002 presentation. Electronically available at: `http://www.igrid2002.org/ppt/Paul_Messina.ppt`.

Pierce, M., Youn, C., and Fox, G. C. (2002). The gateway computational web portal. *Concurrency and Computation: Practice and Experience*, 14(13–15):1411–1426.

Schuchardt, K. L., Myers, J. D., and Stephan, E. G. (2002). A web-based data architecture for problem solving environments: Application of distributed authoring and versioning to the extensible computational chemistry environment. *Cluster Computing*, 5(3):287–296.

Vary, J. P. (2000). Report of the expert meeting on virtual laboratories. Technical Report CII-00/WS/01, United Nations Educational, Scientific and Cultural Organization.