ICES/KIS Project No. 1544516

Title: **The Virtual-Lab Amsterdam Front-End**

Authors: **VLAM-G-AM Group**

Date: **18th April 2001**

Document Code: **/UvA/VLAM-G-AM/TN03**

Version: **1.1**

Status: **Proposal**

Technology & Scientific Center
WATERGRAAFSMEER

The Virtual-Lab Amsterdam Front-End (Design document)

A. Belloum, D. Groep, Z.W. Hendrikse, E.C. Kaletas

document version 18th April 2001

Note to the readers:

- This is a working document, it may be considerably changed in the future. This document aims at pointing out problems related to the design of the VLAM AM-Front-End. It is also a staring point for the design of the AM-Front-End
- This report aims also at providing enough references needed for the implementation of the VLAM AM-Front-End.
- While reading this short report the reader is invited to give his comments and remarks. We especially welcome comments from VLAM developers on how they see their developments integrated in the VLAM-AM Front-End, or which services they need from the VLAM-AM Front-End.
- Notation: if *?? sentence/word ??* is found in the text it means that this sentence/word might be not appropriate to describe the idea being discussed.
  The color used in the figures represent different components
  - Yellow: Globus toolkit components;
  - Red: packages developed for Globus;
  - Gold: Database components;
  - Cyan: shared elements, domains etc.;
  - Green: Potential software for the implementation;
- Web pages where the reader can find more in formation on the VLAM Abstract Machine developments:
  - http://www.nikhef.nl/˜davidg/vlab/
  - http://www.science.uva.nl/˜zegerh/work.html
  - http://www.science.uva.nl/˜adam/currentResearch.html
- This document is the result of the VLAM-AM design meetings which involve: E.C. Kaletas, Z.W. Hendrikse, Victor Klos, David groep and Adam Belloum.

Abstract

The VLAM-G-AM Front-End is the main interface that allows external users to access the VLAM-G hardware and software resources as well as data elements available within the VLAM-G environment. It queries other VLAM-G components on behalf of the end-users. This information is presented using a friendly GUI. In this paper both design and implementation issues are considered.

1 Introduction

The VLAM-G-AM Front-End constitutes what is known as a *science portal* since it provides an easy access to VLAM-G resources. *Science portals* generally provide easy access to problem solving environments (PSE). A PSE is defined by Gallopoulos *et al.* as “a computer system that provides all the computational facilities necessary to solve a target class of problems.” [5].

As in Abrams *et al.* [1] (who extended the definition of a PSE to include more sub-disciplines), several areas of computer science are considered here as well, such as artificial intelligence (the VLAM-G Assistant [18]) and collaborative computing (VLAM-G collaborative system [17]). Other areas include graphics and visualization, HCI (human computer interface) and networking.

Since the VLAM-G-AM Front-End is the main interface that allows external users to access the VLAM-G computing resources, it has to be easy to use and provide access to all the available features of the Virtual-Laboratory. A first division of the VLAM-G-AM Front-End has shown six main categories of components, see also Figure 1:

1. VLAM-G resource discovery
2. VLAM-G resource monitoring
3. Experiment editing and running
4. Publishing results
5. Collaborating with other users
6. Application specific services

These components constitute the building blocks of the VLAM-G-AM Front-End, and are used for all the science portals considered in VLAM. In addition to these main components, specific services may be needed for each science portal in particular. In the subsequent sections, every components is discussed in a separate section, where both design an implementation issues are elaborated upon.

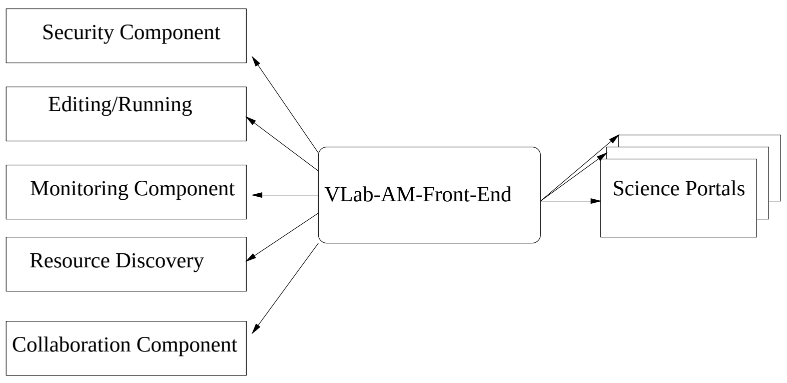


Figure 1: *VLAM-G-AM Front-End components*

The VLAM-G-AM Front-End has to allow a certain overlapping of different areas of research, sharing parts of data and processing elements where applicable, see Figure 2. A nice example of shared elements are tools for visualization, which are commonly applied in many different scientific domains.

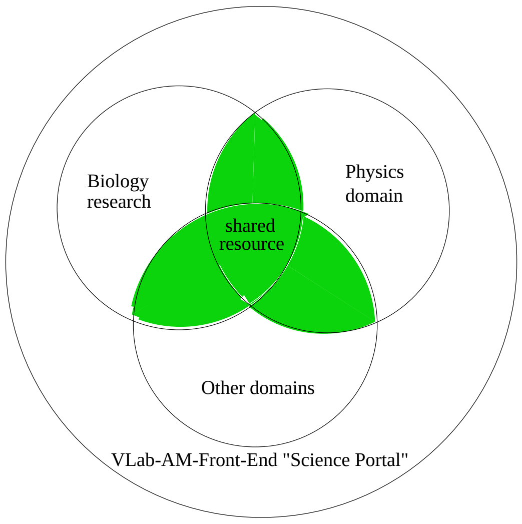


Figure 2: *The overlapping of scientific domains in the VLAM.*

1.1 HCI—human computer interface

The VLAM-G-AM Front-End is an interface between end-users and components of the PSE. An important issue when discussing the HCI is to define categories of end-users who are going to use the system: specific knowledge on PSE's of a typical end-user may vary from a novice to an experienced computer scientist. This holds true when considering a particular scientific domain as well: both junior and senior researchers may want to carry out an experiment. A simple classification of such a wide range of users has been proposed in [12], where three categories have been pointed out:

- the novice scientist,
- the senior scientist and
- the application developer.

1.2 Implementation issues

The VLAM-G-AM Front-End will combine *commodity components* with existing *Grid technologies*, thereby promoting reuse of existing code. Moreover, it should be readily accessible from a desktop environment, providing effective operation in large-scale, multi-institutional, wide area environment.

Commodity components focus on issues such as scalability, component composition and desktop presentation [12]. Grid technology focuses on high performance and high throughput computing on a global scale. A challenge is to find an appropriate match between similar concepts in these technologies. Research groups have already developed some tools in order to meet this challenge, *c.f.* the Java CoG Kit [10]. Using the Java CoG Kit, both project and computational resources are handled via a procedural API and call-backs respectively (Job object and Java events).

Due to the wide range of VLAM-G end-users and the wide range of scientific domains, it is essential that the VLAM-G-AM Front-End is designed in such a way that it can easily be adapted to include either a new scientific domain or a new group of users. A modular architecture is essential to establish such a flexibility.

Figure 3 shows the main interaction of the VLAM-G-AM Front-End with other modules developed within the VLAM-G project. The aim of this report is to discuss in detail the design and the implementation issues of both of the VLAM-G-AM Front-End and its interactions with other VLAM components.

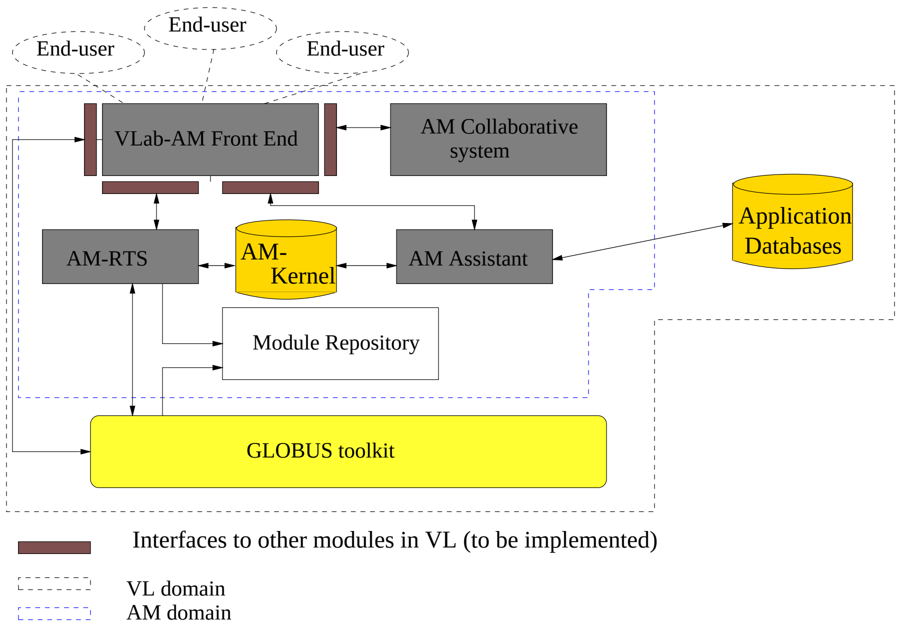


Figure 3: *Position of the Front-End within global architecture of the VLAM. Its interaction with various sub-systems has been drawn: the Abstract Machine Run Time System (VLAM-G-AM RTS), the Meta-data computing directory (Globus MDS), the VLAM-G collaboration system, and the VLAM-G data handling system (AM-Kernel DB).*

1.3 Some definitions

In this report, we will be using terms which may be ambiguous when not properly defined. Therefore this section will be concluded with some of the most frequently used terms.

Software resources: data processing modules specifically developed for the VLAM, as well as the tools and software packages accessible via VLAM-G environment. Each software resource will have a number of attributes associated to it such as location of the executable, execution platform, permissions and accounting information.

Hardware resources: include both computational and network resources. Users should be able to browse these resources and be able to extract all the necessary information on demand.

Availability: is meant to indicate whether a resource is present (a part of the network or a node might be down) and whether a particular user has the appropriate permissions to either monitor or use this resource.

2 VLAM-G resource discovery

The VLAM-G resource discovery component enables users to have seamless access to all the available software and hardware resources of the Virtual-Laboratory. The information needed to provide such a service will be offered by a collection of meta-data stored either in a database (AM-Kernel Database) or in the Globus Meta-Computing Directory (MDS). A connecting layer contains interfaces to the subsystems managing the meta-data directories:

- Interface querying the database
- Interface transmitting user requests to the global computing resources

2.1 Design

Every accessible resource must be uniquely identifiable. Therefore it has to have a number of attributes. The MDS service provided by the Globus toolkit allows a hierarchical characterization of a grid resources using openLDAP [*] The Grid Object Specification [21].

The GOS defines a formal syntax for the definition of objects that form the core of the grid Information Services (GIS) [20]. According to this model, a computational resource may be represented by an object of the following form (following is an explicit example for database designers):

```
GRID::ComputeResources IBJECT-CLASS ::= {
    DUBCLASS OF Grid::PhysicalResource
    RDN = hn (hostName)
    CHILD OF {
        Grid::organizationalUnit,
        Grid::organization,
    }
    MUST CONTAIN {
        canonicalSystemNames  :: cis,
        manufacturer :: cis, single,
        model :: cis, single,
        serialNumber :: ces, single,
    }
    MAY CONTAIN {
        diskDrive :: dn, multiple,
    }
}
```

2.2 Implementation issues

To implement the resource discovery component (see Figure 1), most of the information needed is provided by directory services: MDS and the VLab AM-Kernel Database. Consequently, two interfaces are needed as shown in Figure 4. The *the Globus interface* allows one to query for Grid related information (coming from Globus), the *VLAM-G Assistant interface* allows one to query the VLAM-G-AM kernel database. The latter stores information related to the VLAM-G administration and applications.

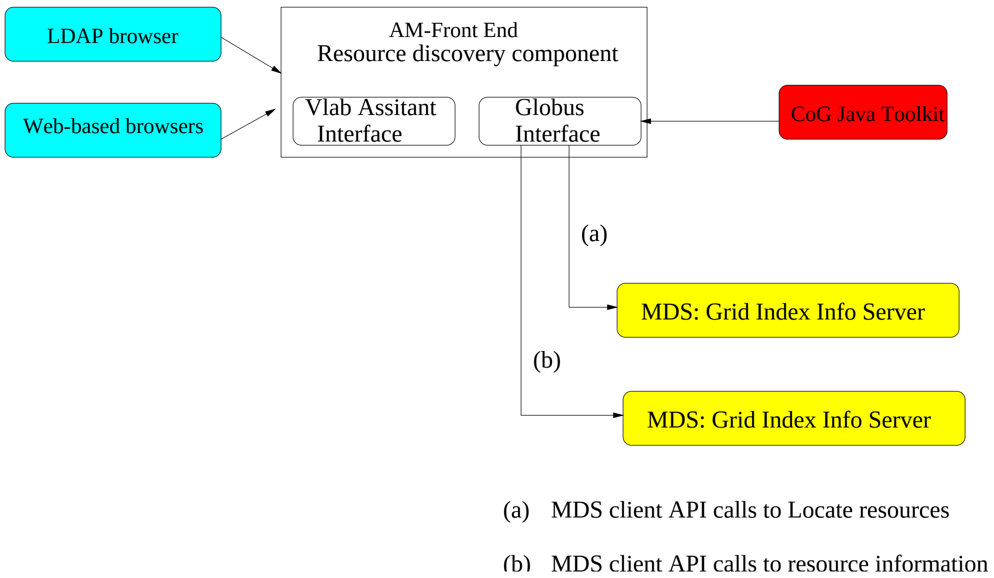
Globus Interface

The Globus interface issues two types of calls

- white-page look-ups: query a particular resource, *e.g.* operating system, processor, memory, etc.

[*] LDAP RFC2251: http://www.ietf.org/rfc/rfc2251.txt

- yellow-page look-ups: query the grid for a resource matching a certain specification, *e.g.* all available resources running Intel Linux 2.2.18, with at least 256Mb RAM.



LDAP browser
Web-based browsers
AM-Front End
Resource discovery component
Vlab Assitant Interface
Globus Interface
CoG Java Toolkit
(a)
MDS: Grid Index Info Server
(b)
MDS: Grid Index Info Server
(a) MDS client API calls to Locate resources
(b) MDS client API calls to resource information

Figure 4: *VLAM-G-AM Front-End resource discovery component architecture*

A Java implementation of these interfaces seems to be most appropriate in this case. The CoG toolkit may be used to provide easy access to Globus services: the "*org.globus.mds*" package simplifies access to the MDS services. This package features:

- establishing a connection to the MDS Server,
- querying the MDS content,
- printing and
- disconnecting from the MDS server.

Following are some references to open source code tools developed within the Globus project that could be used in conjunction to the development of this service.

LDAP browser [†]**:** a visualization tool capable of displaying information stored in a directory service using the LDAP protocol Features: browsing, searching and editing of the Directory Information Tree (DIT), support for the LDIF (LDAP Date Interchange Format).

Web-based browser: (MDS Browser and the MDS Object Browser) a web interface that allows one to view and browse information from MDS servers. Demonstration interfaces can be found at:

- http://www-unix.globus.org/cgi/mds/local_host_lookup_3.pl
- http://www-unix.globus.org/cgi/mds/select_host.pl

NetSolve: offers the ability to look for computational resources on a network, chooses the most suitable, and supports fault tolerance [2].

[†]LDAP Browser http://developer.novell.com/contest/spotlight.htm

Computing portals: Portals are the Web-based front ends for the for the grid. Portals are developed using the Globus Commodity Grid toolkit (COG) [10, 11, 12, 13], myproxy authentication model (recently added to the GIS), and the Grid Portal Kit (GridPort) [15].

GridPort is designed to aid development of science portals on computational grids. Examples are user portals, applications interfaces and education portals. GridPort leverages standard, portable technologies to provide information services that other portals may access and incorporate.

GridPort is based on advanced web technology, including security (PKI) and meta-computing. Web pages and data are built from server-side Perl/face (scripts which render the information from the database) and simple HTML/JavaScript on the client side. This guarantees easy access from any browser.

VLAM-G Assistant Interface

The VLAM-G project extends grid discovery features provided by the Globus Toolkit. It allows retrieval of information related to previous experiments. It offers VLab-specific data, see (Figure 4. Such a feature may be built using the Experiment Environment data model (EE data model) proposed in [6]. In the EE data model, elements from which the experiment is built up may either be processes or data elements and can be randomly ordered allowing for different data flows. The retrieval of information related to data elements of previous experiments is performed by the *VLAM assistant*, which forwards this information to the VLAM-G-AM Front-End, using either a user profile or a user request (Figure 5).

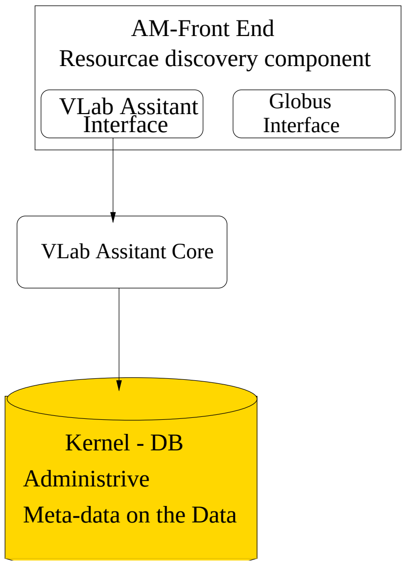


Figure 5: *VLAM-G-AM Front-End Data discovery architecture*

The above mentioned EE model enables users to keep track of all information they judge to be important for their experiments and contains the core data needed for VLAM-specific resource discovery. In addition, a (small) extension is needed to maintain information related to VLAM-G-AM administration, such as the topology of experiments, user sessions, module descriptions etc. [9]. This will henceforth be referred to as the EE-extended model.

In the EE-extended data model, experiments are characterized by a number of features, such as the topology of an experiment and the modules composing a particular experiment.

A module may either be an elementary (*i.e.* doing simple processing) or a composite modules, also known as super-modules. The latter may recursively be built up from elementary and composite modules. Both atomic and super-modules are presented to the users as black boxes with a number of I/O ports. The VLAM-G-AM interperter decomposes each super-module into its constituent components. The EE-extended data model allows for such an approach by defining an appropriate hierarchy between data elements (topology, super-modules, modules, ports, and connections) [7].

3 VLAM-G resource monitoring

3.1 Design

VLAM-G users should able to monitor their experiments. There are two ways of monitoring parameters: the parameters may be monitored using the VLAM-G-AM Front-End (using application probes), or using the Globus monitoring tools. The monitoring process is carried out via the VLAM-G AM or even directly via the Globus monitoring toolkits. This is shown in Figure 6.

3.2 Implementation

The Globus HBM (HeartBeat Monitoring) service may be easily accessed Using the package *org.globus.hbm* from the CoG toolkit. However, monitoring of application related parameters is done via the VLAM-G AM, as is indicated in [16].

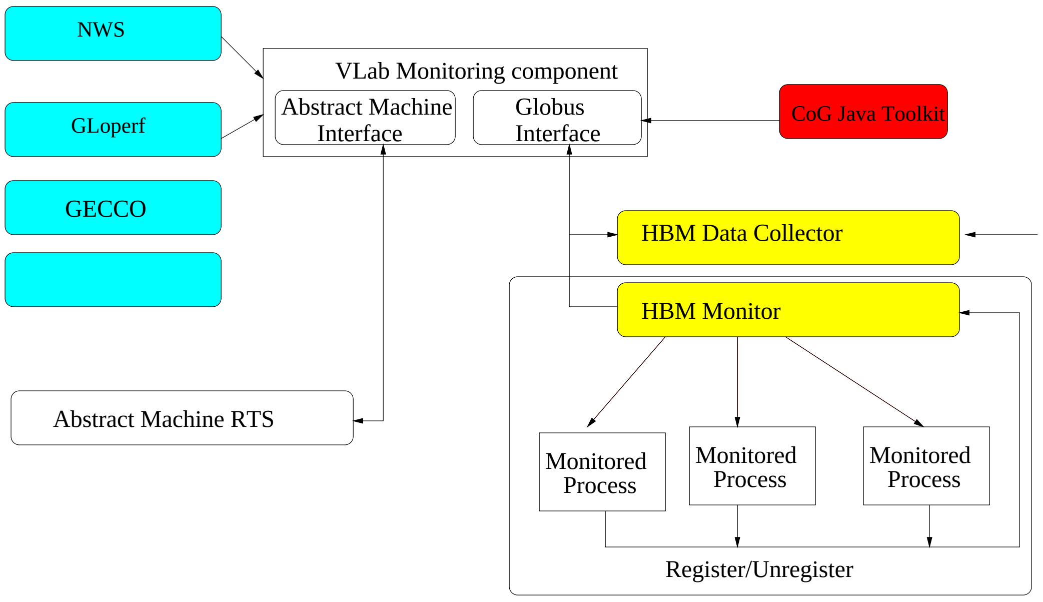


Figure 6: *VLAM-G-AM Front-End monitoring architecture.*

Following are some references to the grid monitoring toolkits:

NetSolve : described in the previous section 2.

Network Weather Service (NWS): provides accurate forecasting of the dynamic change in performance of distributed computing resources [22].

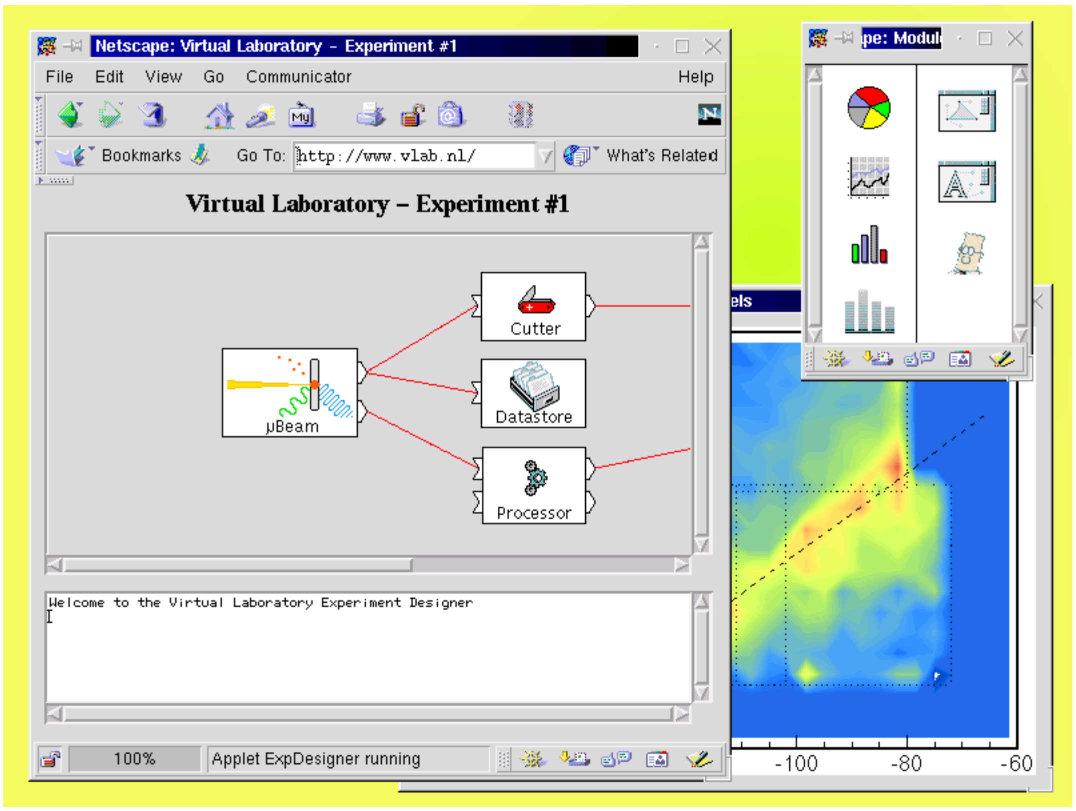
Graph Enabled Console COmponent (GECOO): a graphical tool for specifying and monitoring execution of sets of tasks (with mutual dependencies).

Globus Hart Beat Monitoring (HBM): provides a simple, highly reliable mechanism for monitoring the state of processes. The HBM is designed to detect and report failure of processes which have registered themselves to the HBM. Originally designed for monitoring Globus system processes exclusively, the HBM design has been extended to allow simultaneous monitoring of both Globus system and application processes.

Globus performance (Gloperf): GloPerf performs periodic network performance tests between pairs of IP addresses using a "librarized" version of the netperf utility. Using a netperf library permits a single process to act as both a netperf client and server [‡].

4 Experiment editing and running

Editing of experiments is one of the most important topics. Editing will be performed using an appropriate science portal, but basically boils down to a drag-and-drop GUI (Figure 7). Processing elements, selected from a predefined list appear on an editing sheet as a box with a number of I/O ports. Connections are established by drawing lines between input and output ports. A user may also include his own modules. The experiment editing interface automatically generates a module skeleton appropriate to his needs. Thereafter a user can add his own code to this skeleton.



Welcome to the Virtual Laboratory Experiment Designer

Applet ExpDesigner running

Figure 7: *VLAM-G-AM Front-End editing "ergonomics".*

The editing component of the VLAM-G-AM Front-End may be considered as a simple *visual programming environment*. Therefore it must fulfill some basic requirements with respect to GUI design. This is not to be underestimated,

[‡]http://www.globus.org/details/gloperf.html

see [4]. For example, a common encountered pitfall (which used to give GUI designers serious problems) is information overload. In many designs this problem has been solved by spreading questionnaires to users, and by interviewing the programmers.

A number of interesting conclusions been drawn from the analysis performed at York university, when designing the CLock Visual programming Language [4]. At an early stage of ClockWorks, application development consisted of two sharply separated steps: architecture design and implementation. The design suffered from a lack of information on architecture. At a later stage, users started following a more progressive approach. It became clear that the design of a visual programming environment is a continuing process of interaction between the programmers and environment developers. The following steps turned out to play an important role

- performing task analysis by Visual programmers to determine the requirements in the visual programming environment.
- evaluating the visual programming environment using heuristics, and task-oriented specifications.
- collecting feedback from users using complementary methods. Each method has its pro's and con's. For example, when questionnaires and surveys were of limited use, the Gomoll's ten steps [4, 14] has proven to be very successful.
- involving both expert and naive visual programmers in the analysis; an expert may not want to use such an environment at all!

In most cases, Visual programming environments will be applied to real-world problems. It is thus of extreme importance that issues such as scalability have to be considered during early stages of development already:

- Allowing designers to perform a step wise approach, in other words support for a continuing refinement process in the design of visual applications.
- Providing support for information hiding .

4.1 Design

While editing experiments, VLAM-G users have access to the services described in Section 2. Access to resource related information should be strait-forward. Let us give some requirements needed for editing and running experiments:

- The characteristics (attributes) of a modules should be accessible by a mouse click. Example attributes: data type of the I/O ports, the module parameters (if defined by the module writer), execution platform and usage history.
- When an end-user wants to use his own code to the experiment, he must be able to generate a skeleton by simply filling in a form (in a pop-up window) when the appropriate menu item is activated. This newly created module should be made accessible just like the resident ones: the VLAM-G-AM Front-End has to generate an additional box representing this module.
- The VLAM-G-AM Front-End has to perform a minimum check during the editing process, for example type checking (connection between inputs and an outputs of different types should not be allowed).
- The VLAM-G-AM Front-End should support collaborative editing of an experiment, *i.e.* geographically distributed users must be able to edit an experiment. The collaboration means are addressed in a separate technical report [17]

4.2 Implementation

The development is totally dependent on the interface to the VLAM-G AM, The VLAM-G collaborative system [17] and the VLAM-G resource discovery module Figure 10.

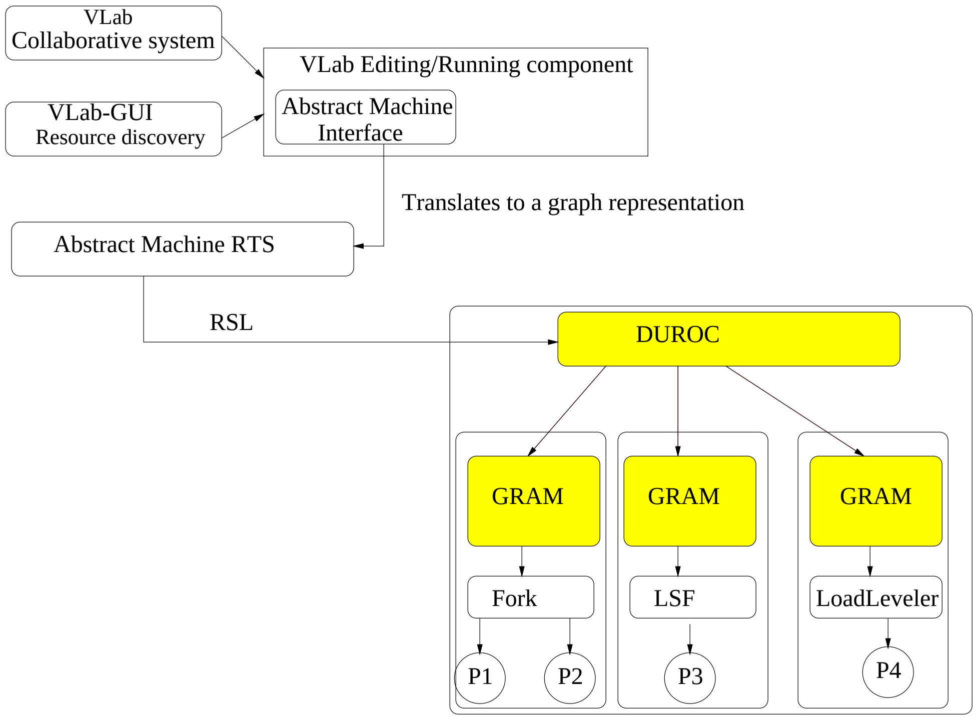


Figure 8: *VLAM-G-AM Front-End editing component architecture*

Experiments are usually part of a larger study that aims to answer a particular question. For example, a PIXE study of a surface sample (including acquisition of data, analysis and visualization), is part of an experimental process flow to determine the decomposition into its elements. In this respect, the biogenetic experiments are similar, although their proper experimental steps are generally smaller and less automated. These processes (topics) have previously been identified as application domains, and database-supported process flow-schemes have been designed (MACS [3], Expressive [8]).

These case studies provide an excellent starting point for the 'AM-Assistant': they supply the scientist with a flow-template which he can use during his experiment (Figure 9 shows the process flow defined in the MACS data-model). These flow-templates provide the Virtual Lab information repositories with the relevant context for the data that are going to be generated. In addition, these templates make sure that all the necessary information is provided by the users running an experiment. By adding new process-flow schemes, new application domains can be added to the Virtual Lab.

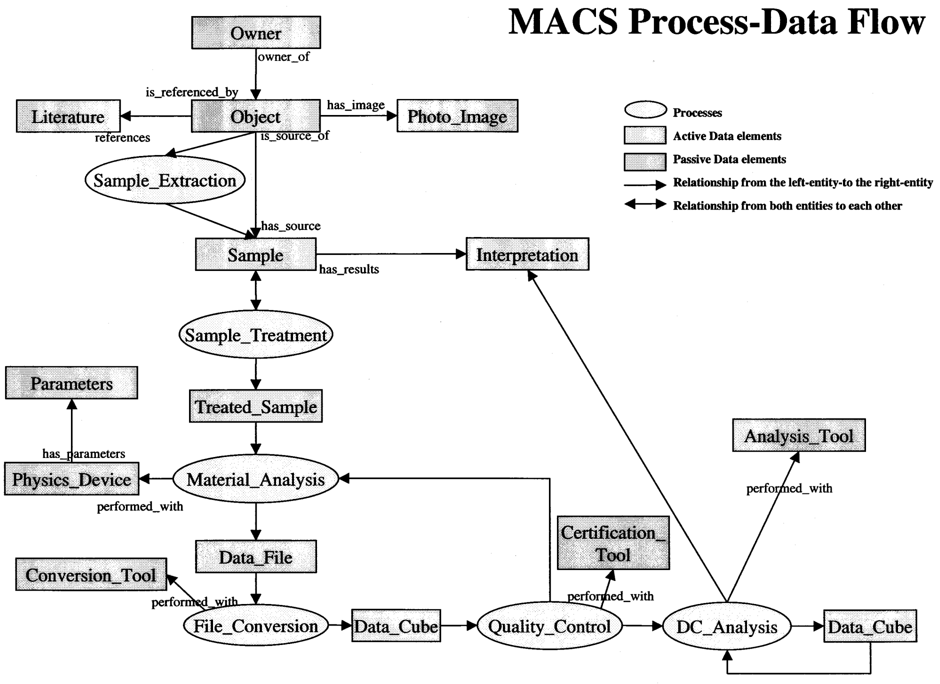


Figure 9: *An example experiment, the MACS process-flow.*

The process-flow chars should be envisioned on a higher level than the DFG (data-flow graphs) executed the AM Run-Time System (RTS). Given their visual nature, they provide a good starting point for a specific scientific experiment: depending on its context (physicist interested in surface analysis, genomics expert, physician), an appropriate process flow template is presented. The user is required to provide the relevant context to his experiment — in the case of MACS: what object, where was the sample taken —, then possibly perform a detailed experiment — Material_Analysis in MACS —. Data generated by this experiment will be stored on bulk media, where the meta-data and context will be an integral part of the process flow context.

Once all the meta-data has been entered, the experiment can be executed. Because the meta-data play such an important role during execution, it is also known as active meta-data: they do not only describe their associated data, they also tell how certain operations are to be performed on it.

As a practical side-note, one may think of a GUI based on this process scheme, Figure 9. A new experiment starts with a fresh template, where the user — by clicking — selects the steps (but cannot skip any step before starting his experiment). Process stages that only provide context will usually result in a fill-in form. Steps associated with experiments will pop-up a AM experiment designer (describing 'data-flow' AM-RTS experiments) as well as filling the boxes associated with the resulting data file (to describe information content and meta-data). In this way, the 'Assistant' can also supply context to the AM-RTS and the modules running there (*e.g.* modules writing data to a database).

5 Publishing results

Users may publish information obtained on a particular experiment using the VLAM-G-AM Front-End. They should also be able to submit new processing elements (modules) to the VLAM-G standard module library.

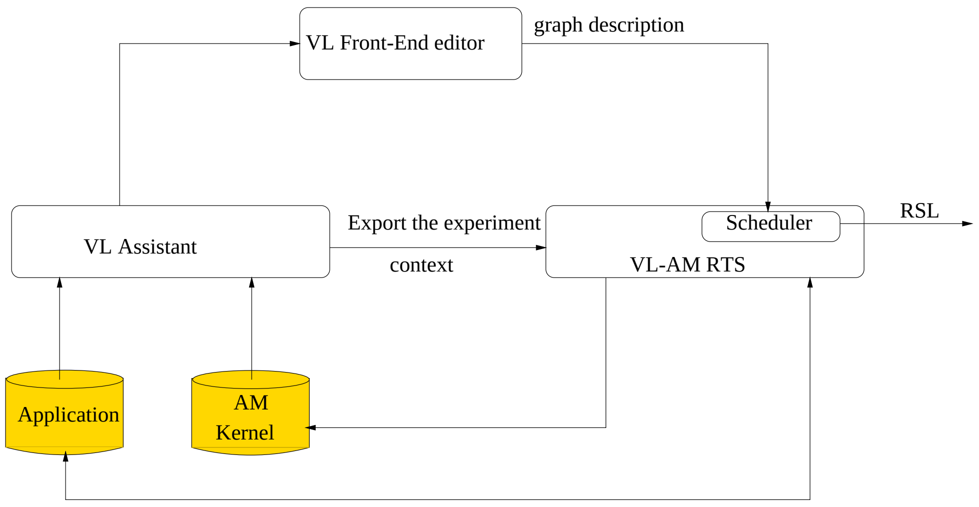


Figure 10: *VLAM-G-AM Front-End editor interactions*

5.1 Design

The publishing of results has to be done via a form generated by the VLAM-G-AM Front-End to guarantee uniformity. Depending on the domain of the experiment, a user will be provided with standard form to fill in. Not only results can be published this way, but also a topology of an experiment, as well as the components being used. This kind of publishing is performed automatically without interaction of the VLAM-G system manager(s).

On the other hand, submission of new code for the shared library, needs the authorization of VLAM-G system manager(s). VLAM-G end-users need to have access to a specific form for these kind of submissions. Its format has to be determined yet, in cooperation with the future VLAM-G system manager(s).

5.2 Implementation

The implementation of the publisher component is more or less strait forward. It has to interact with the Database and the Globus MDS. To publish an experiment topology, the publisher component has to translate the graph representing the experiment topology into an XML description.

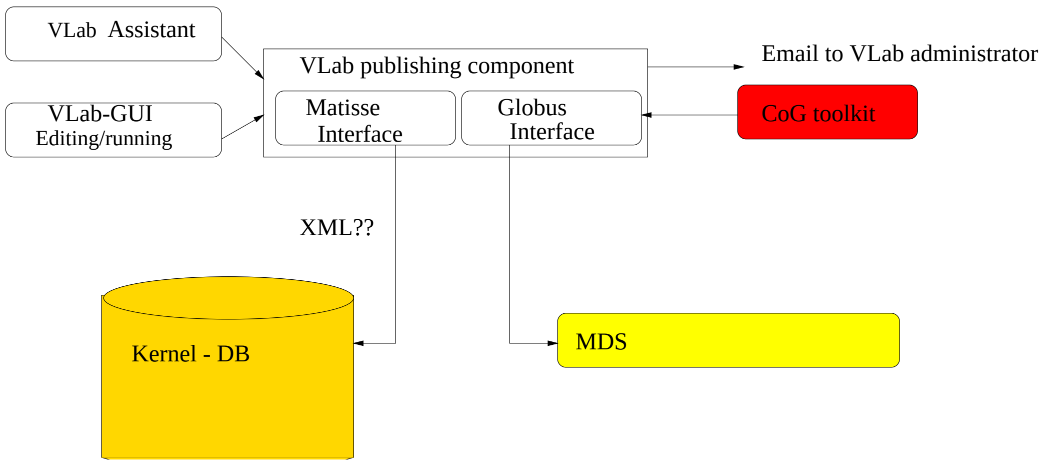


Figure 11: *VLAM-G-AM Front-End publishing component*

6 Conclusions

This report focuses on the design of the VLAM-G-AM Front-End. Six components have been indicated. For each of these components, we have presented basic requirements and proposed possible ways of implementing them.

The VLAM-G-AM Front-End combines the information stored in the Globus MDS with the information stored in the VLAM-G database *AM-Kernel*. This guarantees a comprehensive access to all the available resources in the VLAM-G environment.

The design of the VLAM-G-AM Front-End presented in this report, promotes the reuse of Globus components. Most of the VLAM-G-AM Front-End components are based on existing Globus tools, which makes this design open to all the developments currently being investigated within the Globus project. So far we have not studied performance of any of these components. It seems useful to wait for a first prototype, before addressing any probable performance bottlenecks.

This document also addressed topics related to the definition of the graphical environment needed to set up a typical VLAM-G experiment. An extensive meta-data driven system (active meta-data) based on application process-flows has been proposed.

Data manipulation and analysis have been addressed in a separate document [19].

References

[1] M. Abrams, D. Allison, D. Kafura, C. Ribbens, M. B. Rosson, C. Shaffer, and L. Watson. Pse research at virginia tech: An overview. white paper http://vtopus.cs.vt.edu/~pse/intro.html, Department of Computer Science Virginia Tech.

[2] H. Casanova and J. Dongarra. Netsolve: A network-enable sever for solving comutational science problems. Proposal Draft, Data Access Grid Forum working group, 2000.

[3] A. Frenkel, G. Eijkel, H. Afsarmanash, and L. Hertzberger. Information management for physics applications in the vl environment. Technical Report CS-2001-03, University of Amsterdam, 2000.

[4] T. N. Graham, C. A. Morton, and T. Urnes. Clockworks: Visual programming of component-based software architectures. *Journal of Visual Languages and Computing*, pages 175–196, July 1996.

[5] E. Houstis, E. Gallapoulos, and J. Rice. Problem-solving environment for computational science. *IEEE Computational Science & Engineering*, 3(4):18–22, July-September 1997.

[6] E. Kaletas and H. Afsarmanesh. Virtual laboratory experimentation environment data model. Technical Report CS-2001-01, University of Amsterdam, 2001.

[7] E. Kaletas, A. Belloum, D. Group, and Z. Hendrikse. Vl-am kernel db: Database model. Technical Report UvA/VL-AM/TN07, University of Amsterdam, 2000.

[8] E. Kaletas and et al. Expressive - a database for gene expression experiments. Technical Report CS-2001-02, University of Amsterdam, 2000.

[9] V. Klos. Database model design for the abstract machine. Technical Report -Draft Version-, Nikhef, 2000.

[10] G. V. Laszewski, et, and al. Cog high-level components. White paper, Argonne National Laboratory, 2000.

[11] G. V. Laszewski, et, and al. A java commodity grid kit. White paper to be published in Experience and Practice 2001, Argonne National Laboratory, 2001.

[12] G. V. Laszewski, I. Foster, J. Gawor, P. Lane, and M. Russell. Designing grid-based problem solving environments and portals. white paper, Argonne National Laboratory, 2000.

[13] G. V. Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke. Cog kits: A bridge between commodity distributed-computing and high-performance grids. In *Proceedings of the ACM Java Grande 2000*, San Francisco, June 2000.

[14] J. Nielsen. *Usability Engineering.* AP Processional, Cambridge, MA, 1993.

[15] M. Thomas, S. Mock, and J. Boisseau. Development of web toolkits for computational science portals: The npaci hotpage. In *Proceedings of High Performance Distributed Computing Conference 2000*, Pittsburgh, Pennsylvania, August 1-4 2000.

[16] A. van Halderen. User guide for vlab developers. Technical Report UvA/VLab-AM/TN05, University of Amsterdam, 2000.

[17] VLab-AM-Group. Vlab am collaborative system. Technical Report UvA/VLab-AM/TN03, University of Amsterdam, 2000.

[18] VLab-AM-Group. The vlab assistant. Technical Report UvA/VLab-AM/TN0??, University of Amsterdam, 2000.

[19] VLab-AM-Group. Vlab data handling system. Technical Report UvA/VLab-AM/TN04, University of Amsterdam, 2000.

[20] G. von Laszewski. Defining schemas for the grid information services. Proposal for the Grid forum Draft, Grid Forum, 2000.

[21] G. von Laszewski and P. Lane. Mdsml: An xml binding to the grid object specificaction. Proposal Draft, Grid Forum, 2000.

[22] R. Wolski, N. T. Sprin, and J. Hayes. The network weather service: A distributed resources preformance forecasting services for the metacomputing. Technical Report TR-CS98-599, UCSD, 1998.