ICES/KIS Project No. 1544516

Title: **The Virtual Lab-AM Collaborative System**

Authors: **Adam Belloum**

Date: **November 30**

Document Code: **/UvA/VLAM-G-AM/TN02**

Version: **1.1**

Status: **Proposal**

Technology & Scientific Center
WATERGRAAFSMEER

The Virtual Lab-AM Collaborative System (Design document)

Adam Belloum

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 VLab Collaborative system. The aim is to do an analysis of what have been done so far by some research groups active in this area.
- This report aims also at providing enough references for the implementation of the VLab collaborative system and provides a starting point for its design.
- While reading this short report the reader is invited to give his comments and remarks. We especially welcome comments from VLab developer on how they see their developments interacting with the VLab Collaborative system.
- 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.
- This document is the result of the VLab-AM design meetings which involve: .... and Adam Belloum.

1 Introduction

The dramatic progress made in the communication technology has drastically eased the collaboration among scientists all around a world. The virtual-Laboratory is just one of many projects that aims to facilitate work within a distributed and heterogeneous environment. It is thus of a major importance for the Virtual-Laboratory project to tackle the problem of the end-users collaboration. Indeed, it is common in scientific environment that researches collaborate world wide to do some experiments. In a recent past the scientists have to gather in a specific location to do these experiments. Since in the Virtual-Laboratory project the scientific community is the main target. It would be wise to provide this community with a powerful and efficient collaborative mechanisms. In this report, we try to summarize some of the well known mechanics of collaboration.

2 Mechanics of Collaboration

It is often considered in the literature that collaborative systems (groupware) are difficult to build. The researches and developers often link this difficulty to the fact that groupware are heavily impacted by the social factors (Grundin 1990). And thus the often simplified laboratory models that strip the system from its context of use may lead to develop inefficient systems. In a scientific environment more factors can impact the groupware such as the type of the experiments, the scientific domain, and the type of experiment. The basic activities of collaboration can be quite different from one experiment to another and from one scientific domain to another. These activities, called also mechanics of collaboration, are defined by Gutwin as: "the small-scale actions and interactions that group members must carry out in order to get a shared task done" [8].

The mechanics of collaboration help in a great deal in defining the mean of collaboration. A There are many classes of groupware systems depending on the mean collaboration being used, the literature reports a variety of means such as the shared-work space, tele-data, tele-presence, or the futuristic cyberspace [7]. The tasks that happen in a collaborative system are of few types:

- creating of new artifacts.
- organization of existing artifacts.
- exploration of the space or of a set of artifacts.
- construction of larger objects from components or pieces.
- the management of an autonomous systems represented in the workspace.

A number of group activities have been identified as comprising the mechanisms of collaborations

- Explicit Communication: is the most basic mechanism of collaboration it involves intentional exchange of information among the group members.
- Consequential communication: is a subtle mechanisms of collaboration it unintentional give information as one go about his activities.
- Coordination of actions: an important mechanism of collaboration used to avoid conflict among the group members.
- Planning: can also be used as a mechanism of collaboration especially for periodic tasks and shared resource allocation.

- Monitoring: By monitoring the work-space the group member can get information on the evolution of the teamwork.
- Assistance: group members provide help to one another. It requires that understand what others are doing and where they are in their task.
- Protection: prevent others to intentionally or inadvertently destroy or modify ones work.

In Johnson's work the usability of the collaborative software depends upon the efficiency with which the collaborative software *answers* a set of general classes of questions such as *who are the other users?, What are the artifact involved?, What has changed since yesterday?, Who should be informed of the change?, Is the information in the system uptodate? etc.*. To answer these class of question the application has to collect, represent, store, modify, analyze, and distribute *state of information* [10]. The state information is used by Johnson as an organizing principle for CSCW architectures. Thus in order to answer these type of questions, a system has to represent information about the users, artifacts, types of changes, the context of collaboration, and the communication mechanisms.

The state management can be described by the eight dimensions pointed out by Johnson's work namely: producer, consumer, trigger, representation, location, distribution, and time. This work has pointed out that support for state representation does not lead always to the most generic tools, and often a "CSCW architecture may consciously and correctly sacrifice certain state facilities to increase its overall utility".

Almost every collaborative system is accessed through a *Multiuser Interface.* The main characteristic of multiuser applications is their ability to automate a number of collaborative tasks. Three categories of tasks have been pointed out in [3] *computation tasks* and *interaction tasks* , and *collaborative tasks.* In this study the focus will be put on the collaborative aspect of the multiuser applications. To allow multiuser to collaborate in real time (synchronous collaboration), multiuser applications have to deal with problems such as dynamically connection/disconnecting with remote users, multiplexing/demultiplexing input/output, informing users, providing concurrency and access control. The current state of the technology allow the development of systems with such features without starting from scratch. Prasun Dewan has classified the tools that support the implementation of multiuser interface in eight categories, namely: Database systems, Distributed systems, Message servers, shared object systems, shared window systems, multiuser toolkits, multiuser UIMSs, and multiuser user-interface generators. Each of these tools have its advantages and its disadvantages, more important is the fact the seems to have complementary features, even better it appear in Prasun Dewan study that some of these tools are based on concepts that could be combined to develop a multi-user application. Figure 1 shown how multi-tools have been combined to implement a simple multi-user interface (the tools used are discussed in Section 3.

Because of the availability of such tools the designer of groupware application can spend more time in the functionality of the applications. However, the tools supporting the development of groupware applications differ in the architectural abstractions that they present to the developers [13]. Low-level toolkits expose the underlying distributed system to the developers, an example of such toolkits is the GroupKit toolkit [12]. The high-level toolkits abstract all the gory details of the networking, distribution, and concurrency; examples of such system are Rendezvous [5], and Weasel []. A combined category of tools allow both low and high level abstractions, these provide more possibilities to the developers; toolkits allowing such a behavior are: the Suite system [2], The GEN system, Prospero system [4], and AMF-C [1]. Tores and Graham show in [13] annotation can be used to separate the functional design of the groupware from the design

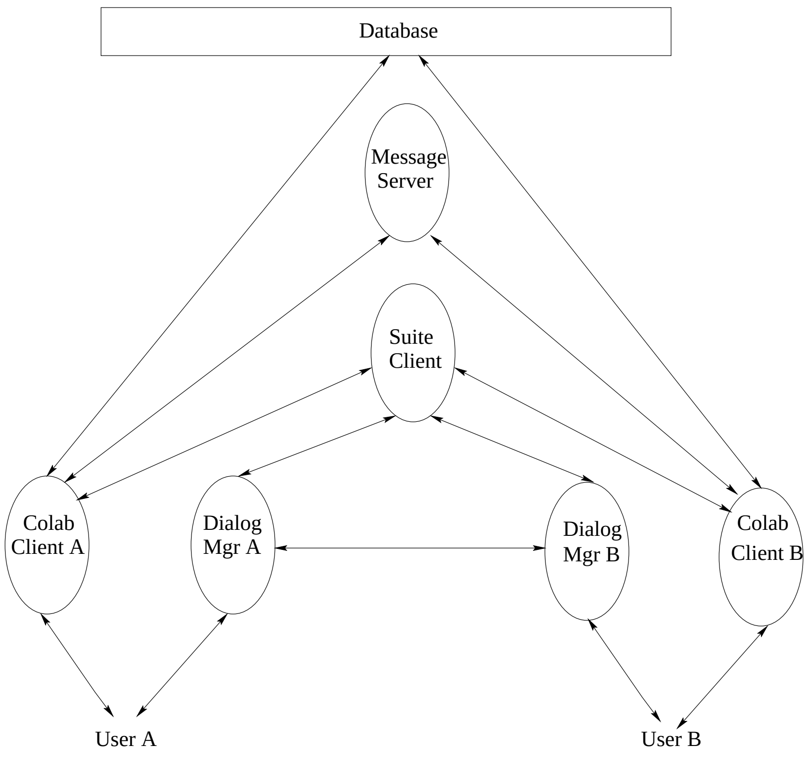

Figure 1: *Combining several tools to implement a multi-user interface*

of its distributed system. When developing the functionality of the application, annotations are used to guide its implementation as distributed system. The annotations allows to select between a variety of distributed styles using what it called *semantic-preserving annotations* (Annotation can affect the performance but not the A functionality of the application). Annotations give some hints to the runtime system as to how the architecture should be implemented as a distributed system. In the Clock environment the annotations are represented with icons allowing user to select between a number of styles: caching algorithm, replication strategies, concurrency control etc.

Since the World Wide Web is becoming by far the most used way to access geographically distributed information, more and more developments are performed in order to allow synchronous groupware application over the WWW. GroupScape is one of these project, in this work techniques has been developed to couple the WWW and synchronous applications [6]. The techniques are based on the model-view-controller architecture (MVC) and add two new HTML tags bring synchronous functionality to web documents. The GroupScape browser allow among other things: group web browsing, application based browsing and embedded synchronous views, and embedded application events.

3 Groupware Technology

The technology used in groupware systems is categorized along two primary dimensions: the asynchronous/synchronous aspect of the collaboration and the location of the group members. In the context of virtual-Lab, in order to allow scientists to collaborate and work together on a experiments, it is necessary to provide them with real time cooperative system. In this section we focus thus more on the technology used in synchronous groupware systems.

Synchronous or real-time collaboration has been classified in four different style namely: Face-to-face meetings, remote conferencing, casual real time application, and multi-uses applications. The work on computer support for real time collaborative work

When a groupware has been designed a number of method have been proposed for their evaluation, among others"

- Heuristic Evaluation
- walkthrougs
- Through observations
- Through user questionnaires

3.1 Example of groupware toolkits and environment

An environment for developing groupware

The Virtual Network Computing (VNC) is free of use environment, developed by the AT&T Cambridge. Using the VNC entire desktop can be accessed from any Internet connected machine [11]. VNC has been implemented using the Client-Server paradigm, the client part of the VNC, called also viewer, is an ultra-thin client application that supports a number of network display devices. The NVC viewer includes: an X-based viewer (which runs on Solaris, Linux, and Digital Unix workstations), a Win32 viewer that runs on Windows NT and 95, and a Java applet tha runs on any Java-capable browser.

The VNC server provides the pixel data in the format requested by the viewer. Two servers have been implemented one for platform using X, and one for Windows NT/95 platforms. The VNC allows a single desktop to be accessed by several places simultaneously, thus supporting application sharing in the style of computer-supported cooperative work.

GrooupKit From University of Calgary

GroupKit is a freely available toolkit developed by Mark Roseman and Saul Greenberg at the University of Calgary. It is based on the Tcl/Tk scripting language (running under Unix/X), and designed to support a wide range of real-time groupware applications. Facilities include remote procedure calls, shared data structures, flexible session management and concurrency control, and novel multi-user widgets. Well-documented, lots of examples and quick to learn [12]. See the GroupKit Home Page:

http://www.cpsc.ucalgary.ca/grouplab/groupkit/

GroupKit is an open source code, all code in the GroupKit library conforms to the coding conventions found in the Tcl/Tk Engineering Manual (C code) and the Tcl/Tk Style Guide (Tcl scripts). The GroupKit code is relatively both consistent and documented, making it easier to understand and modify. If you have code that you are planning to contribute to GroupKit, please ensure that it follows these conventions.

GroupKit 5.0 is a substantial rewrite, loosely based on the design found in the unreleased GroupKit 4.0. The GroupKit API has been completely changed, and now relies on the namespace mechanism introduced in Tcl 8.0. Internally, the code has also been cleaned up to conform with Tcl coding conventions. GroupKit 5.0 supports Unix, Windows and Macintosh

Egret

Egret is an Emacs-based freely available toolkit developed by Philip Johnson at the University of Hawaii. EGRET is an environment for building domain-specific, collaborative hypertext applications [9]. The design, implementation, and evolution of EGRET has been used to explore many issues in high performance, multi-user, client-server hypertext database systems. For more information on Egret and related projects, see the CSDL Home Page:

http://csdl.ics.hawaii.edu/

ClockWorks

The ClockWorks programming environment is a graphical editor for building, browsing, and executing software architectures of interactive systems. A software architecture is a set of components organized according to a tree structure. Sophisticated component grouping mechanisms allow large sofware architectures to be presented compactly and intuitively (and makes doing groupware a snap!). A declarative view language is used to specify user interfaces. For more information, see The Clock Home Page:

http://www.qucis.queensu.ca/ graham/clock.html

Rendezvous

Rendezvous, out of Bellcore, was one of the first generation of groupware toolkits, and features like its Abstraction-Link-View paradigm have inspired a lot of the later work. Unfortunately, Rendezvous has (as they say) ceased to be. If you're interested, you may want to stop by the excellent CSCW Pointers page that Tom Brinck has collected.

Suite

Suite is a Unix/X groupware toolkit that tries to address issues such as sharing abstractions, coupling, access control, undo/redo, concurrency control, merging, inheritance, and distributed architectures. Suite is best known for its use in investigating issues of flexible coupling. For more information....[2]

http://www.cs.unc.edu/ dewan/

Como/Promondia

Como (renamed Promondia) is a Java-based system that can be used to develop groupware applets, called "commlets" in the system. The goal of the project is to standardize interactive communication on the internet. A prototype is currently available, and includes a shared whiteboard, scheduler, chat and a game. See the Promondia Page for more information. Also check out a review of Como. For more information, see the Promondia Home Page:

http://www4.informatik.uni-erlangen.de/Projects/promondia/

Promondia is by far not the only Chat System in the Web. But it is unique in that it combines the following features:

- Anyone with a Java enabled browser can use the applets simply by looking at a web page. No software has to be installed on the user's systems. Software updates do not have to be distributed to the users. As soon as they are on the server everybody uses them automatically.
- There is no need for clicking "Reload" or having the browser auto-reload. New information is automatically transmitted to all recipients in real time.
- Promondia is more than just Chat. Promondia is an open system that supports voting, conferencing with shared whiteboards, games, and customized applications for both the Internet and intranets.

Sur ng delays are minimal; the smallest Promondia Applets are about 6 kb- less than a typical image.

Administration is easy and powerful. Access control, foreign languages, and application speci c options can be set using a nice java interface (or, by editing ASCII les) without restarting the server.

HTML-formatted logging output and statistics can be generated on the y.

Currently supported communication methods include Chat moderation (even for the IRC connectivity), voting and surveys, a shared whiteboard and a game.

Habanero

Habanero is another Java based system, this one brought to you by the good folks at NCSA. Habanero programs are built by sharing common state information, by wrapping program objects with collaboration-aware facilities .For more information, see the Mushroom Home Page:

http://havefun.ncsa.uiuc.edu/habanero/

Mushroom

Mushroom is a Java-based framework being developed by Tim Kindberg at the Queen Mary & West eld College, University of London. It supports distributed collaborative working, group formation and interaction, shared resource management and privacy. Mushroom provides Mrooms: working spaces for groups of collaborating users. They are units of focus on related tasks, and units of resource management. For more information, see the Mushroom Home Page:

http://www.dcs.qmw.ac.uk/research/distrib/Mushroom/

The Java Collaborator Toolset

This is a package that helps to make any Java application collaborative. It does this by providing a replacement for the Java AWT. This replacement intercepts all events, and sends them to all copies of the application via an event distributor. The project lead for this was involved in the earlier XTV project. For more info, see the Java Collaborator Toolset page:

http://www.cs.odu.edu/ kvande/Projects/Collaborator/

COAST

The COAST framework is an add-on to VisualWorks Smalltalk that supports the development of a wide range of synchronous groupware applications. The framework o ers the advantages of a fully distributed and replicated architecture without burdening the application developer with the complexity of such architecture. As a consequence, the key features such as transparent replication of data objects, a distributed optimistic concurrency control, and the automatic update of views when shared data objects change are kept mostly transparent to the application developer. The COAST framework is available for non-pro t use (academic and evaluation purposes). More detailed information about the framework can be found on the COAST home page at GMD-IPSI:

color blue http://www.darmstadt.gmd.de/publish/ocean/activities/internal/coast.html

4 Collaboration in VLAM-G

In VLAM-G both of the asynchronous and synchronous collaborative systems are needed. For certain type of experiments VLAM-G users require real-time collaborative system that allow them to exchange point of views, share working space, and have the same or different views of the on going experiment this is the case of the data-flow like of experiments. For experiments that last for a long period of time, the VLAM-G collaboration system has to support dynamic number of experimenters i.e. users can join or leave an experiment while it is running. For these kind of experiments, VLAM-G collaborative system should use asynchronous collaboration techniques to contact the disconnected members of the ongoing experiments to deliver urgent messages.

Figure 2 shows some interactions of the VLAM-G collaboration system and the rest of the VLAM-G parts. Two main interaction have been outlined: the first with the VLAM-G Gui, and the second with the VLAM-G-Abstract-Machine RTS.

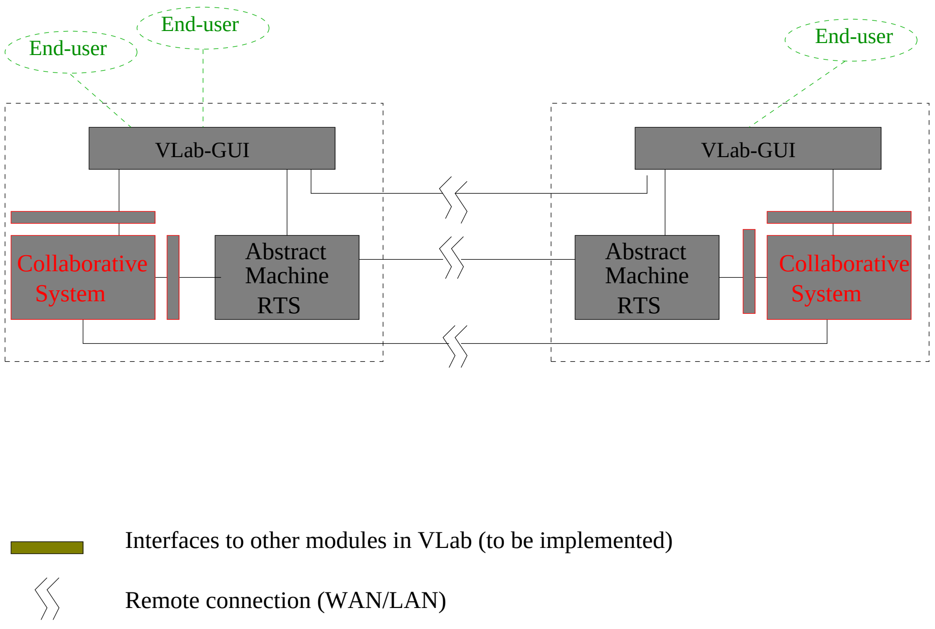

Figure 2: *VLAM-G collaborative system integration*

The VLAM-G Collaborative system should include a *session manager* which must start and initiate all the software and hardware available/needed for the session. The session manager will allow the user by joining a previously started session or to create a new one. A directory services will contain all the needed information for both of the join and create session. The managers of the video, sound, and whiteboard when started by the session manager initiate themselves using information stored in the directory service. Finally, a registration module within the session manager collects all the information provided by the VLAM-G GUI when an end-user wants to join/create a new session.

5 To be discussed/done

1. The collaborative requirements needed for VLAM-G.
2. The selection of appropriate toolkit to develop the VLAM-G collaborative system
3. Finaly some implementation issues

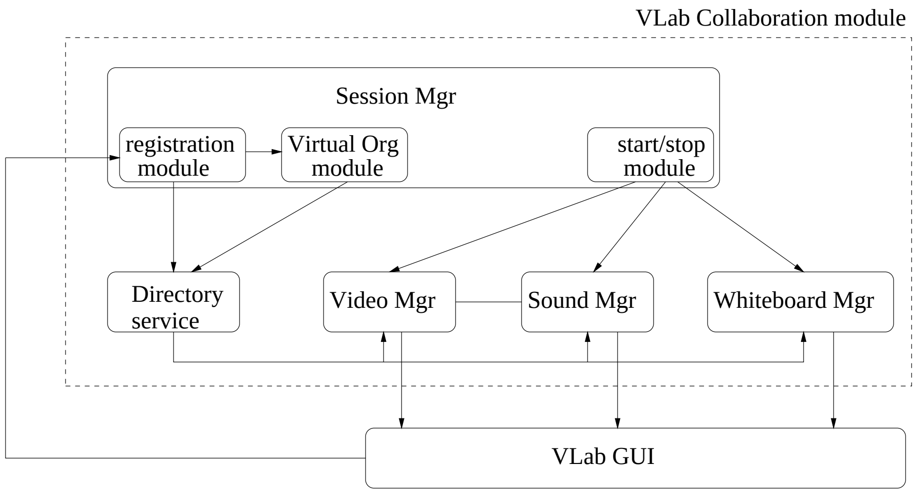

Figure 3: *VLAM-G collaborative system Architecture*

- The strategies to be used for the video, sound, and whiteboard.
- Programming models to be used for the implementation
- integration with VLAM-G GUI.
- other issues (suggestions are welcome)

4. other issues (suggestions are welcome)

References

[1] F. T. Bernard, David, and P.Primet. Framework and patterns for synchronous groupware: Am-c approach. In *Proceedings of Engineering for HCI (EHCI98)*, 1998.

[2] P. Dewan. A guide to suite. Technical Report XXXX, Perdue University, 1993.

[3] P. Dewan. Tools for implementing multiuser user interfaces. Book: User Interface Software -Chapter 8-, John Wiley & Sons Ltd, 1993.

[4] P. Dourish. Consistency guarantees: Exploiting application semantics in a collaborative toolkit. In *Proceedings of ACM CSCW*, 1996.

[5] R. Gill, T. Brinck, S. Rohall, and J. Patterson. The rendezvous language and architecture for constructing multi-users applications. In *Proceedings of ACM TICHI Conference*, pages 1(2): 81–125, 1994.

[6] T. N. Graham. Groupscape: Integrating synchronous groupware and the world wide web. In *Proceedings of Engineering for HCI (XXXX)*, 19XX.

[7] S. Greenberg and E. Chang. Computer support for real time collaborative work. ...., ..., ...

[8] C. Gutwin and S. Greenberg. The mechanics of collaboration: Developing low cost usability evaluation methods for shared workspaces. ...., ..., ...

[9] P. Johnson. The egret primer: A tutorial guide to coordination and control in interactive client-server-agent applications. Technical Report ICS/CSDL-TR-95-10, University of Hawaii, 1995.

[10] P. Johnson. State as an organizing principle for cscw architectures. Technical Report ICS-TR-96-05, University of Hawaii, 1996.

[11] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, Jan/Feb 1996.

[12] M. Roseman. Online documentation web page. http://www.cpsc.ucalgary.ca/grouplab/groupkit/gk5doc/ Last updated June 28, 1998, University of Calgary, 1998.

[13] T. Urnes and T. N. Graham. Flexibly mapping synchronous groupware architectures to distributed implementations. In *Proceedings of Design, Specification and Verification of Interactive Systems (DSV-IS'99)*, 1999.