RVLABX, A WEB-BASED INTERACTIVE LABORATORY ENVIRONMENT FOR EDUCATION AND RESEARCH

Hamadou Saliah-Hassane[1], Patrick Dumont Burnett[2], Christian Loizeau[3]

Abstract *— RVLabX stands for eXtended Remote Virtual Laboratory. It is a workplace software, that permits the creation and management of a Web-based generic, interactive laboratory with a particular emphasis on the creation of XML-based adaptable user interface for each team member involved in the process. The software can be installed on any desktop and connects to distributed virtual laboratory servers. The ultimate goal is to create and configure distributed laboratories, where users can interact with one another as if all are working both in co-operation and collaboratively on a single site. Thus, students, teachers, tutors, laboratory technicians or managers at geographically distinct locations can share their expertise and equipment.*

Index Terms — Virtual Laboratory, Remote Laboratory, Laboratory Management, User Interface, Laboratory Innovation.

INTRODUCTION

We have developed integrated software that permits us to create and configure a Web-based distributed laboratory environment and its management utilities and tools [1]. In such a remote virtual laboratory environment, users, be they students, teachers, tutors, laboratory technicians or managers, can interact with one another from geographically distinct locations.

The first component is a GUI Interface Designer, an application that presents a canvas on which controls are dropped from a list. These controls can then be moved and resized and their properties set through a window property. The binding of these controls with their data source, in this case, programs running on remote computers, is also done through this property window. The designer of the interface can switch at any time from the design mode to a test mode to verify the behavior of the interface in real-time. A user interface can then be designed by dragging and dropping the required graphical components. Then, at the end of the whole process, clicking on a button generates a corresponding XML file. This file is then sent to a Web server to be read by an ActiveX control displayed on a Web page we have called PolymorphiX, representing the second component. It allows users to utilize the GUI within a Web client. Depending on the scenario and the role of each participant within a co-operative work task, the manager can build an appropriate adaptable user interface for each team member [2]. Another important component is the User Administration Utility. This component is a user monitoring system created with an ASP (Active Server Page) technology and a database, to ensure the session control and deal with the customization issue to preserve user privacy and information integrity.

The resulting application allows rapid design of a lightweight distributed laboratory environment for collaborative learning and co-operative work through computer networks.

FUNCTIONAL DESCRIPTION OF THE SYSTEM

System Overview

The overall system (Figures 1 &2) is made up of 2 major applications. The first allows the trainer or lab manager to create a Graphical User Interface (GUI) design and the second allows users to utilize the GUI within a Web client. The GUIs are connected via a DataSocket server to a remote Virtual Instrument (VI) [3], itself connected to a Data Acquisition device (DAQ device).

GUI Design

User interfaces are designed within an application called GUI Generator (Figure 3). The application displays a canvas on which the user can slide his own controls, the latter created by using National Instruments ComponentWorks tools and those included in Visual Basic. Once placed on the canvas, controls can be moved and resized. Control parameters are set via a property window. Linking these controls to their data source (DataSocket server) is also done via property windows. At any given time, the designer can switch from design mode to execution mode in order to test his GUI. Once the GUI is properly completed, the designer can generate an XML file containing the parameters for each control existing within his GUI. This file is then sent to a Web server to be read by the polymorph Web client we call PolymorphiX.

[1] Hamadou Saliah-Hassane, Télé-université , LICEF/CIRTA Research Center, 4750 Henri-Julien, Montreal, H2T 3E4 saliah@licef.teluq.uquebec.ca
[2] Patrick Dumont Burnett, LICEF/CIRTA Research Center, 4750 Henri-Julien, Montreal H2T 3E4 pdb1@hotmail.com
[3] Christian Loizeau, Collège de Maisonneuve 3800 Sherbrooke St. East, Suite B-2250, Montreal H1X 2A2 cloizeau@videotron.ca

0-7803-6669-7/01/$10.00 © 2001 IEEE

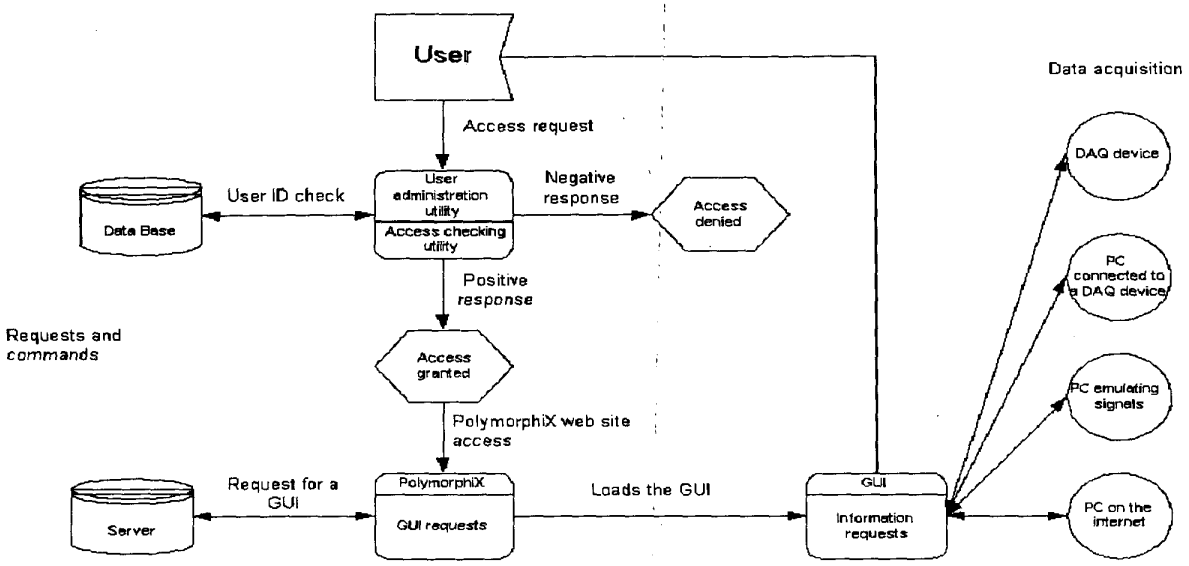

Figure 1 Overall functionality

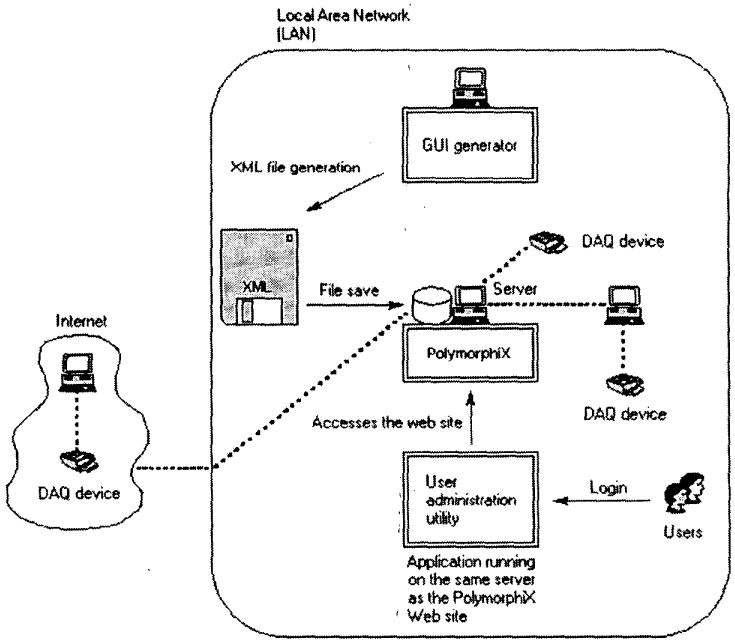

Figure 2 System Overview

Utilizing the PolymorphiX Web Client

A user wishing to utilize PolymorphiX, which is an ActiveX control, must first hook up to the Web server to have access to the GUI. Following a successful user ID check, the user is directed to the Web page containing the control. The user must download the desired GUI if it is his first visit. The ActiveX control is saved on the user's PC, thus there will be no need to download it again on future visits.

The PolymorphiX control is a GUI composed of two zones (Figure 4). The top half is empty until the desired GUI is loaded. The bottom half can display two different zones. The first allows the user to type in the address of the desired interface and also displays data concerning other users who are currently using the same interface elsewhere on the network. The second zone allows users to communicate in written form with the administrator or with other interface users via a chat client.

After entering the address of the desired interface, the user activates the download function. The PolymorphiX control reads the XML file located at that address and displays the GUI defined by the file.

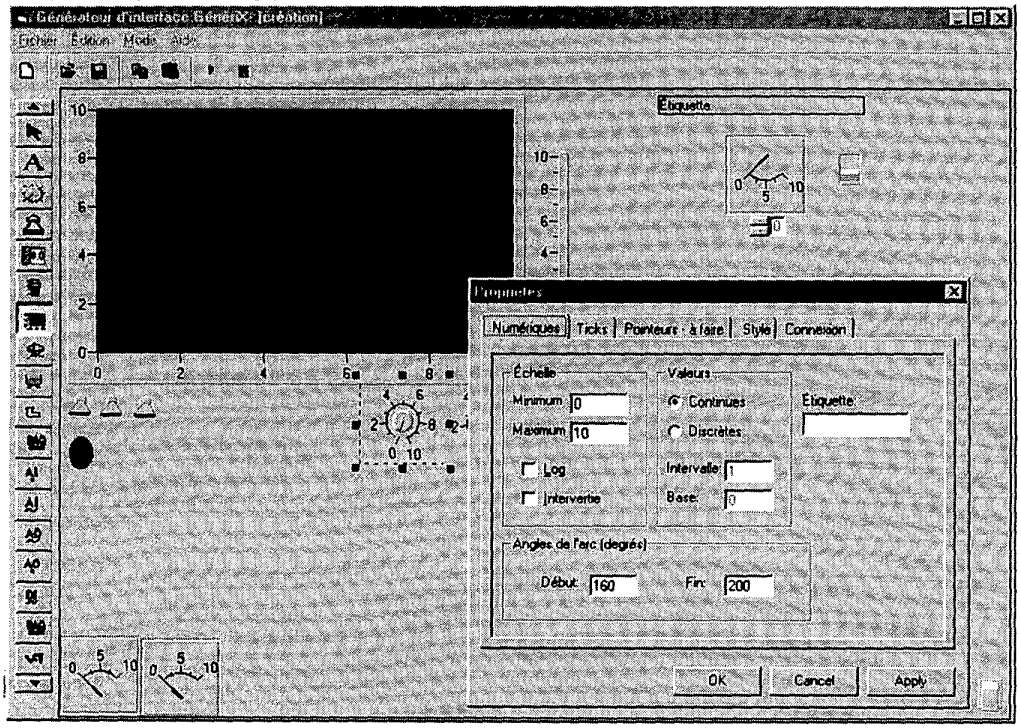

Figure 3 GUI Generator Application

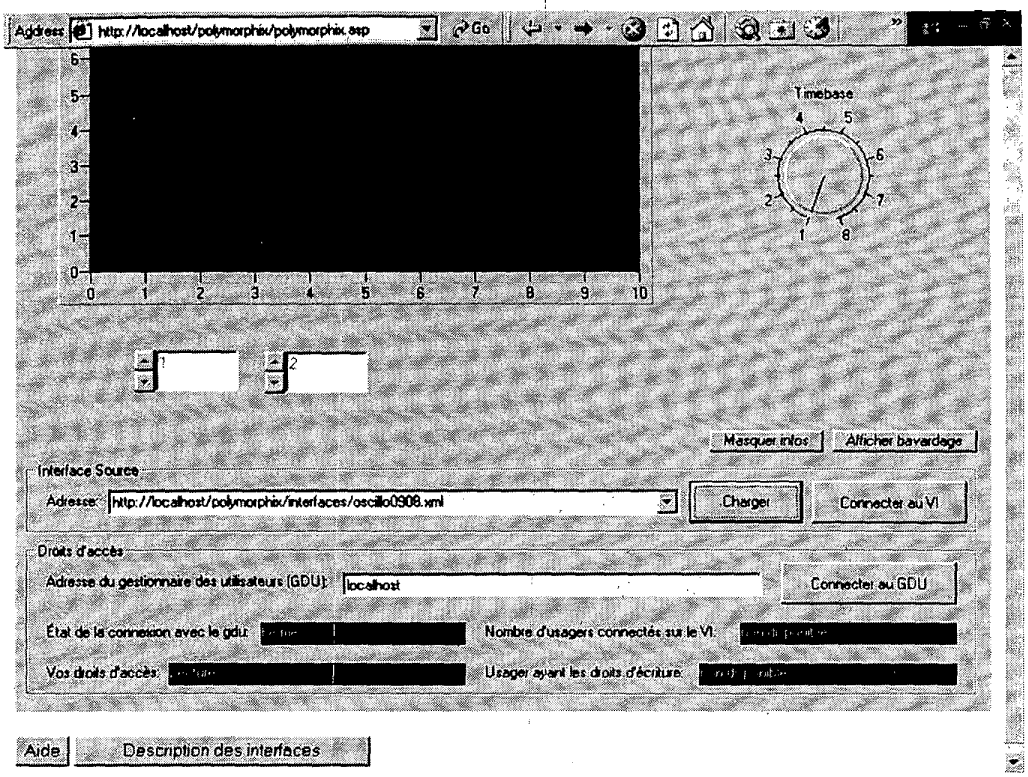

Figure 4 PolymorphiX Web client connected to a remote Virtual Instrument via Internet

User Administration Utility

Thanks to ASP (Active Server Page) technology, a user administration tool has been developed to restrict access to those users having obtained permis sion to do so, following registration. The application is called User Administration Utility. Information concerning authorized users is kept in the server's database.

Once a user loads an interface onto the PolymorphiX Web site, s/he can only receive information displayed on his screen from a remote VI by default. S/He can, however, send instructions to the VI once s/he has been authorized to do so by the administrator via the user management application called User Administration Utility. This application receives requests from various clients connected to the VI. The User Administration Utility has a list of all users connected to the VI and wishing to interact with the said VI and can thus control access. The application relays to the PolymorphiX clients information related to each user's access level (read only or read and write), the number of users connected to the same VI and the user having interaction right-of-way (one at a time) with the VI. The administration application also acts as a chat server between users.

The GUI Generator application and PolymorphiX Web client/control are created with Visual Basic 6 and ComponentWorks 3.0 [4] ActiveX control along with DataSocket 3.0. The User Administration Utility application is also created with Visual Basic 6. The GUI Generator and User Administration Utility applications are independent applications that run on Windows 9x and NT. The PolymorphiX client is an ActiveX control.

A Web server capable of executing ASP scripts, such as the Personal Web Server (a streamlined version of Microsoft Internet Information Server) running on Windows 98 is used to host the PolymorphiX Web site. The different VIs used as data sources are created with LabVIEW 6i and are hooked up to a NI-DAQ server, a National Instrument Data Acquisition Server.

The database utilized for user ID validation is drawn up with Microsoft Access 2000 and is stored on the application Web site.

CONCLUSION

One of the main advantages of the GUI Generator application is the ease and simplicity with which users can create and distribute Graphical User Interfaces. GUI

distribution has become much more streamlined, because instead of distributing a different ActiveX control for each interface, all one now needs is a simple XML file. The PolymorphiX control simply has to read this file in order to be properly configured.

LabVIEW 6i now allows the user to connect a VI interface component to a DataSocket server via a dialogue box. This process is less time-consuming and no longer requires inserting DataSocket controls and links to various controls in the diagram window. Thus, with these two applications, the user can connect VI controls to ActiveX Web controls even if s/he only has a rudimentary knowledge of Visual Basic or only a minimum grasp of LabVIEW.

With the GUI Generator application, one can add controls within an interface, modify its parameters and link controls to a data source (DataSocket server). However, it does not allow the add-on of a new code so as to modify data received from or sent to the DataSocket server. Furthermore, the PolymorphiX Web client can be modified to read VBScript. However, in this case, the user would have to be familiar with Visual Basic. Obviously, a developer familiar with Visual Basic or with Visual Basic 5.0 Control Creation Edition (VB CEE), a streamlined version available at no charge, that allows for designing ActiveX controls, will enjoy working with this application. In both cases, the developer must also use ComponentWorks and DataSocket controls.

REFERENCES

[1] H. H. Saliah, L. Villardier, B. Assogba, C. Kedowide, T. Wong, *"Resource Management Strategies for Remote Virtual Laboratory Experimentation", "2000 Frontier in Education Conference:* Building a Century of Progress in Engineering Education", Kansas City, USA, October 18-21, 2000.

[2] I. De la Teja, A. Longpré, G. Paquette, *"Designing Adaptable Learning Environments for the Web: A Case Study"*. In Ed-Media 2000 Proceedings. June 26 - July 1st, 2000. Montreal, Quebec, Canada.

[3] *LabVIEW Basics 1 Hands-on Course*, National Instruments, Austin, 1998.

[4] J. TRAVIS, *Internet Applications in LabVIEW*, Prentice Hall, 2000.