IEEE Instrumentation and Measurement
Technology Conference
Anchorage, AK, USA, 21-23 May 2002

XML Technologies to Design Didactical Distributed Measurement Laboratories

Andrea Bagnasco, Marco Chirico, Anna Marina Scapolla
DIBE, University of Genoa, Italy

Abstract - Laboratory activity is an open challenge for on-line teaching applied to scientific domains. The remote control of instrumentation and the execution of real experiments via Internet are topics of interest for many researchers. This paper introduces the model of a virtual laboratory that allows practicing via Internet in real laboratories spread on a wide area network. It is based on a main virtual laboratory server (VLS), one or more real laboratory servers (RLS) and client stations. Real laboratory servers control instrumentation and drive experiments. The network links all the distributed software components of the environment.

The paper focuses on the use of XML (Extensible Markup Language) and its related technologies to support the development of such environment. XML plays a central role in the definition of the environment (instruments, experiments and users activities) and in the maintenance process.

These technologies are very promising to address scalability and interoperability among distributed software applications and to facilitate information reuse ensuring data coherency, completeness and reliability. The proposed data organization results effective to develop, maintain and scale up the environment and facilitates the cooperation and the contribution of many partners to the virtual laboratory project.

I. Introduction

Multimedia and network interactivity are leading to new forms of teaching and learning and new roles for students who act as participants and not only spectators of their own learning process.

In scientific domains, an open challenge for on-line teaching is related to laboratory activity. Specific environments have been developed to construct virtual laboratories and to make them available though the network. Virtual laboratories can be based on simulations or alternatively, the network can provide a virtual access to personal computers that control real instrumentation and physical devices. This paper presents the results of our research in this field. It introduces the model of a virtual laboratory that allows testing theories through practice and executing real experiments that can be distributed in different laboratories, spread on a wide area network, and controlled by local computers.

Remote control of real instrumentation and laboratory experiments is realized by means of a distributed software system that results complex and requires a modular design to guarantee the possibility of extending the environment to a wide range of equipment and of applying it to many different educational domains.

Section II introduces the virtual laboratory environment, section III focuses on the system scalability and maintenance issues and explains how XML (Extensible Markup Language) and its related technologies can effectively support the development of such environment.

II. The Virtual Laboratory: An Overview

Our research, carried out during the last two years [1], led to the definition of a model to share laboratories in Internet (Internet Shared Instrumentation Laboratory - ISILAB). ISILAB is based on a distributed software environment (see Fig.1) consisting of a main virtual laboratory server (VLS), one or more real laboratory servers (RLS) and user/client stations. Internet links all these components.

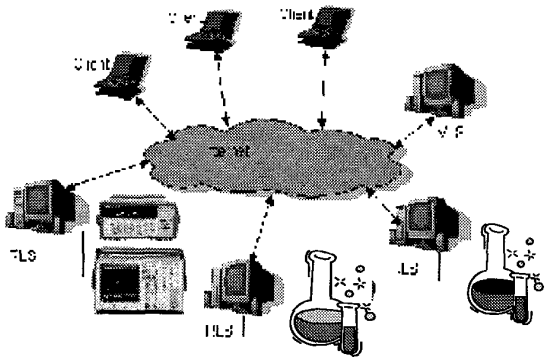

Figure 1 - ISILAB Architecture

Real laboratory servers can be spread on wide geographic area and control real experiments. No limit exists to the location of the laboratories and the only requirement is an Internet connection between client and servers controlling instrumentation. Users do not require any particular hardware and software equipment and can carry out experiments through the network. They do practice transparently to the

real location of the device under test in a multi-user concurrent way.

The environment offers the chance to make experiments in supervised or in concurrent mode. System administrators, who are in charge to maintain the database of experiments, set the execution mode of each experiment. In supervised mode there is a privileged user who is the only one able to modify interactively the operational conditions, acting on the instrumentation controls, and the other users are only able to see the response of the system on their computer screens. This mode can be very effective in a context of distance education as teachers can show real laboratory experiments via Internet.

When an experiment is carried out in concurrent mode all users are able to interact with instruments and see the results of their own commands. The coherency of each experimental session is guaranteed by the RLS.

A prototype of the virtual laboratory has been developed with the main goal of validating design and implementation choices and setting up a framework to evaluate performance and didactical usefulness of the proposed approach. Students, remotely accessing the laboratory, are introduced to a set of experiments/lessons leading to gain knowledge of electronic instruments, measurement procedures and circuits under test. Each experiment is presented as a lecture with a didactical target and the proposal of a collection of exercises to be conducted via web. A set of hints, tips and instrument handbooks are available on-line and provide a continuous support to conduct experiments. Figure 2 is a snapshot of the prototype. It is possible to have both simplified and very realistic virtual panels, in order to hit different didactical target.

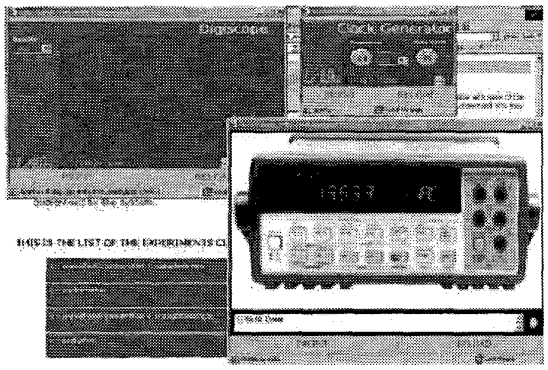

Figure 2 - The learning environment.

I.1. The VLS

The VLS is a network node hosting a web server (in the prototype it is a G web server from National Instruments) that introduces users into the virtual laboratory, implements the access control policy, and logs users activities.

The access is controlled on the basis of login/password credentials and only authorized users can get into the laboratory.

The VLS presents the list of available experiments to users and gives information about each experiment, such as a brief description, the number of active users and the location of the experimental set up. The knowledge of the real location of an experimental set-up is not required but it can be useful for establishing connections with the laboratory technicians. When a user asks for an experiment, the VLS verifies the availability of the required experiment; then it initialises the communication between the user and the RLS that controls the selected experiment. After that, the instruments' interfaces (Java Applets) are delivered to the user. This interaction is summarized in figure 3.

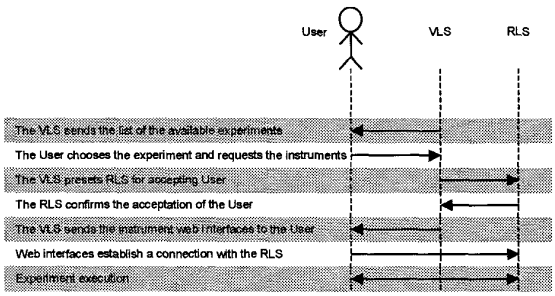

Figure 3 – Interactions between the user and the system.

I.2. The RLS

Real laboratory servers manage the interactions between users and experimental set-up. Each RLS receives the inputs from users via TCP, applies them to the instruments, which are connected via local bus (i.e., IEEE488), retrieves the results and sends them back to users. RLSs run a specific server-application (written in LabVIEW from National Instruments) that takes care of separating the different user data spaces (Contexts). The Context is the data structure that caches information about the current configuration of each instrument involved in a specific experiment, carried out by a specific user. When a user sends a command (by acting on the virtual instrument panel), his/her Context is updated and applied to the appropriate instrumentation set. Instruments' responses are recorded in the Context and changes are sent back to the user. A Context identifier is assigned to each user when an experiment is selected; this assures the coherency among context data, virtual instrumentation and user identification during the experiment execution. In order to communicate with the RLS, Applets must send an identifier

that contains both the Context and the specific instrument IDs.

The Context data structure has been designed having in mind that the status of a virtual instrument can be represented by a set of heterogeneous parameters: the Components. A cluster of three parameters (identifier, type, value) characterizes each Component. In this way, the Context of an experiment consists of a bi-dimensional array: the set of components that are necessary to describe the status of the instruments involved in the experiment. A collection of Contexts (a three-dimensional array) represents the data area of the whole environment.

This model lets to control concurrency and facilitates the development of different GUIs and driver adapters for the same instrument driver. We can insert or remove instrument drivers without changing the RLS core software, but simply, copying files in a specific directory.

Figure 4 shows how instruments are managed by the RLS.

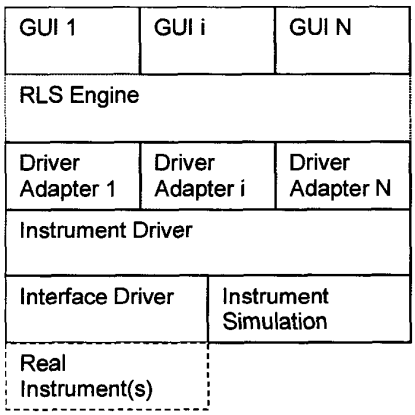

Figure 4 – Instruments Management.

Driver Adapters convert the user controls (knobs, switches and so on) to Instrument Driver inputs/outputs. They are software modules that manage the communication between the GUI and the instrument driver. The status of the instrument, set by the user, is elaborated and translated to a specific call to the instrument driver.

Instrument drivers are general-purpose drivers that are usually provided by vendors. They transform numeric values into instruments' specific commands. Instrument drivers should be able to exchange data with both real and simulated instrumentation. There are categories of Instrument Drivers, called IVI drivers, which have this characteristic in native way. [2]

I.3. User Interfaces

The laboratory experiences can be carried out via the most popular web browsers. HTML pages, simple and portable carrier of information, are used for introducing and explaining experiments. Java Applets are the natural choice for GUIs, because of the flexibility in design, facilities in network programming and platform independence [3]. In the prototype, Applets has been developed following two different approaches. We have used both a common JDK environment and a commercial tool, AppletVIEW™ from Nacimiento™ [4], which is specific to the development of instrument control panels. The communication between RLS and Applet GUIs uses a TCP-based protocol from Nacimiento™, the Virtual Instrument Transfer Protocol (VITP).

III. Scalability, Reusability and Maintenance Issues

The virtual laboratory must be regularly updated in order to meet students and teacher needs: new experiments are added and new instruments controlled. The success and the effectiveness of the environment are strongly conditioned by the way we can add new components and reuse existing instrumentation to deliver new experiments.

To achieve these goals our model provides a modular and scaling structure, so that different developers can contribute to increase the number of experiments. The insertion of a new experiment implies to build a certain number of objects, such as web pages for the experiment explanation, applet GUIs, instrument driver adapters and so on. A large set of heterogeneous components are involved in the system maintenance/upgrade. For instance, an experiment has to be described in terms of identifier, title, description, author, RLS location, didactical target, activities, instruments and so on. Each experiment can be associated to a set of exercises and each exercise is composed of a list of tasks, hints and expected results. Again, each instrument has a default configuration that is specific to a particular hardware device. The same real instrument can be managed by multiple software user interfaces that are characterized by a different set of functionalities.

It is clear that some degree of automation is mandatory for the upgrading process; we need to reduce maintenance effort and to guarantee coherency.

I.4. The XML Solution

We propose the use of XML to describe the environment data concerning instruments, experiments and users activities.

XML and related technologies are very promising to pursue scalability and reusability, as they provide interoperability among distributed software applications and facilitate information reuse ensuring data coherency, completeness and reliability. XML allows to represent data in a self-explanatory, data-centric way. Each element of a XML file is embedded into a tagged structure. This feature

guarantees the portability of data through different hardware/software platforms, that means portability through space and time [5]. Beyond XML, "the XML family" is a growing set of modules that offer useful services to accomplish important and frequently demanded tasks like a standard way to add hyperlinks to an XML file *(Xlink)* and syntax constructions (*XPointer* and *Xfragments*) needed to navigate inside it. An Xpointer, for instance, is a bit like a URL (Unified Resource Locator), but instead of pointing to documents on the Web, it points to pieces of data inside an XML file. Moreover, language features like *XSL (Extensible Stylesheet Language), DOM (Document Object Model)* and Schema exploit the power in terms of flexibility and expressiveness of XML. *XSL* is the advanced language for expressing style sheets. It is based on *XSLT*, a transformation language used for rearranging, adding and deleting tags and attributes. The *DOM* is a standard set of function calls for manipulating XML (and HTML) files from a programming language. *XML Schema* lets developers to precisely define the structures of their own XML-based formats [6]. The existence of these standards avoids the user to bind to a particular vendor. The large diffusion of XML has made available a lot of powerful COTS (commercially-off-the-shelves) parsers for every kind of programming language (C++, Java, Visual Basic, LabVIEW, TCL, and so on). The most of these are available for free and they are also very reliable, because they have been tested and validated in very extensive way by a large community of developers.

Dialects of XML, applied to describe and control instrumentation are under development [7] or already submitted to the community [8], as in the case of Virtual Instruments Markup Language (VIML) [9]. VIML was introduced to describe the user's interface of virtual instruments. A specific XML Schema defines rules and names to describe commands and data that are transferred to and from instruments.

I.5. XML and ISILAB

The Internet shared laboratory makes a well-built usage of VIML and defines an appropriate XML Schema for the description of the whole information required by the system.

Figure 5 is a graphical representation of the system data model.

Experiments are the focus of the system. Each experiment is defined through many elements, such as the author, the location (name of the RLS), procedural tasks and so on. A certain number of exercises or activities can be associated to an experiment. For example, we can think to a set of tests that students are invited to execute in order to practice. Each exercise has a textual description and one or more pictures associated to it. The experiment execution requires the control, and thus the configuration, of a certain number of instruments. The configuration of each instrument is described using XML according to the model introduced in section II.2.

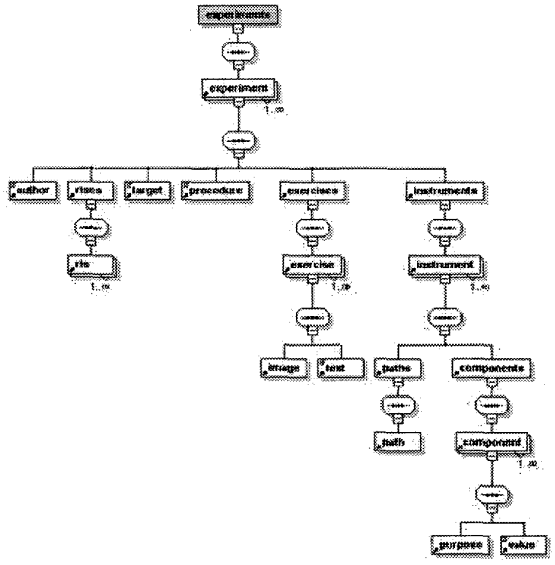

Figure 5: The System Data Model

In this way, each system component (instrument, experiment, exercise, and so on) has a proper XML definition that can be managed using a commercially-off-the-shelf parser. For the development of the prototype, we have chosen to use the MSXML [10] parser that can be easily called by a LabVIEW™ application. On the VLS side, the parser dynamically generates web pages containing the list of the available experiments, tasks and exercises. The RLS configures itself extracting data from the XML structure: addresses of real instruments, instrument drivers and Driver Adapters locations, components values and so on. The XML structure acts as both a database and a configuration file. It is easily readable and manageable. The portability of XML lets us imagine a network of virtual laboratories where contents, both hardware and software, are provided from a community of distributed institutions. In this scenario, teachers can easily set-up experiments using a network of laboratories.

In order to test the ISILAB system, a set of experiments on digital electronics has been set up. The main didactical target of these experiments is to test basic digital circuits. Once the test bench has been prepared and instruments connected to the RLS via GPIB, we need to adapt the software environment and to give visibility to the experiments. This task is extremely facilitated by the data model described in the previous section. We don't need to create specific HTML files. GUIs and drivers of the instruments can be reused if they already exist.

To add a new experiment it requires filling new elements of the XML structure. Commercial tools like XML Spy™

[11] are helpful in accomplishing this task. Figure 6 reports a view of the XML document..

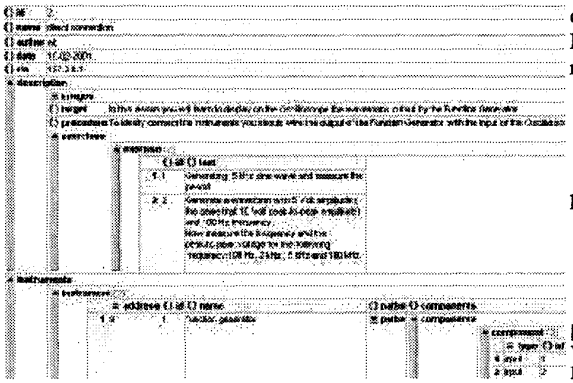

Figure 6: A view of the XML document

Using this kind of editor, the insertion of new data into the XML document is very simple. If compared with the manual creation of HTML pages, this approach is absolutely time effective and lets to maintain a stylistic coherency by separating data from presentation. The same approach is used to introduce automation in building the software that drives instruments. Java Applets are automatically generated starting from an XML description of the instruments' GUIs. The development of the driver adapters is simplified by the use of a unique model that is valid for all the instruments. The use of a template speed up the rate of production.

IV. Conclusions and Future Work

In this paper we have presented an Internet shared laboratory that allows executing real laboratory experiments, controlled by computers and spread on a wide area network.

The focus is on the model and the methodologies used to describe the virtual laboratory and to structure system data.

Data organization results effective to develop, maintain and scale up the environment and facilitates the cooperation and the contribution of many partners.

The laboratory components are easily reusable. Instruments can be operated with different levels of complexity and functionalities according to different requirements. The commands and the data exchange protocol are unaltered and reusable; different user interfaces can be defined by means of simple specifications.

XML technologies play a central role in the definition of the environment and in the maintenance process. They guarantee portability and interoperability.

The experiment description will be the subject of further investigation in order to integrate those elements that can transform each single experiment into a pedagogical unit to be shared and reused for different educational targets. We look at experiments as learning objects that must match the users' learning goals.

Classification of experiments, according to a standard classification scheme, such as the IEEE-Learning Object Metadata, will provide an open and flexible way for management and reuse.

Acknowledgement

The authors would like to thank Corrado Franchi for his precious contribution in the prototype development.

V. References

[1] A. Bagnasco, M. Chirico, G. Parodi, A. Sappia, A.M. Scapolla, "A Virtual Laboratory for Remote Electronic Engineering Education", in International Perspective on Tele-education and Tele-learning, Ashgate Book, 2000, pp. 1-14.

[2] www.ivifoundation.org

[3] Chung Ko, Chi et Al., "A Web-Based Virtual Laboratory on a Frequency Modulation Experiment", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 31, No. 3, August 2001.

[4] Travis, Jeffrey, "Internet Application in LabVIEW", Prentice Hall, 2000

[5] Goldfarb, Charles F., and Prescod, Paul, "The XML Handbook", Prentice Hall PTR 2001.

[6] www.w3c.org

[7] Instrument Markup Language, http://pioneer.gsfc.nasa.gov/public/iml/

[8] http://www.w3.org/XML/

[9] http://www.nacimiento.com/viml

[10] www.microsoft.com

[11] www.xmlspy.com