APPROACHES TO PROGRAMMING FOR TELE-MEASUREMENT

Eleftherios Kayafas [(1)], Florin Sandu [(2)], Ioannis Patiniotakis [(1)], Paul Nicolae Borza [(2)]

[(1)] Division for Communications, Electronics & Information Engineering, Department of Electrical & Computer Engineering, National Technical University of Athens, 157 73 - Zographou, Athens, Greece
Phone +30-1-772-2544 Fax +30-1-772-2538 e-mail: kayafas@cs.ntua.gr , ipatini@softlab.ece.ntua.gr
[(2)] Department of Electronics & Computers, Faculty of Electrical Engineering and Computer Science, "Transilvania" University of Brasov, Bulevardul Eroilor, nr. 29, Brasov – 2200, Romania
Phone +40-68-478705 Fax +40-68-475751 e-mail: sandu@vega.unitbv.ro , bozapn@vega.unitbv.ro

Abstract - *Introducing and solving some specific problems of remote access to experiments, the present paper has not only a generic (architectural) point of view, but also a practical one. Based on their previous experience in tele-measurement (SMTP / POP3 and LabView implementations), [1] and [4], the authors built an integrated solution for the "publishing of the test and measurement capabilities" based on Java / Data Socket (or WinSock) transfer for remote access and LabView / Component Works / Visual Basic for local management of the automated measurement centers. A very friendly wyswyg ("what-you-see-is-what-you -get") real time duplex control / monitoring was practically implemented in VB emulated panels of the instruments that can be tele-accessed via a web-page of the remote virtual laboratory, without the requirement of any special Test & Measurement skills at users' level (allowing them to concentrate on the specific experiment that can belong to any field of engineering).*

Keywords - open & distance learning ("ODL"), automated test & measurement ("T&M"), virtual instruments ("VI").

1. INTRODUCTION

The present paper is centered on authors' approach to a Virtual Laboratory ("VL").

The VL is able to "publish its experimental resources", offering services over an access network or the Internet by using well-known and widely available technologies and products; this way the unhindered access to the VL services is guaranteed and compatibility problems are avoided. There is a variety of specific issues and problems like interoperability, accessibility, resource sharing, security, equipment protection, availability, maintenance and administration. A generic solution was designed as an integrated approach, based on the Hewlett-Packard "MSA" (Measurement Subsystems Architecture) principles: co-design of hardware ("HW") configuration and T&M SW, in the Standard Commands for Programmable Instruments (SCPI) [3] and IVI (Interchangeable VI-s) context. Its 3-tier architecture implements the VL SW frame and IT platform. Besides our approach, we are discussing also alternative implementations and anticipate solutions to different and more complicated problems.

2. THE GENERIC ARCHITECTURE

Fig. 1 presents the VL architecture from the Information and Communication Technology ("ICT") point of view.

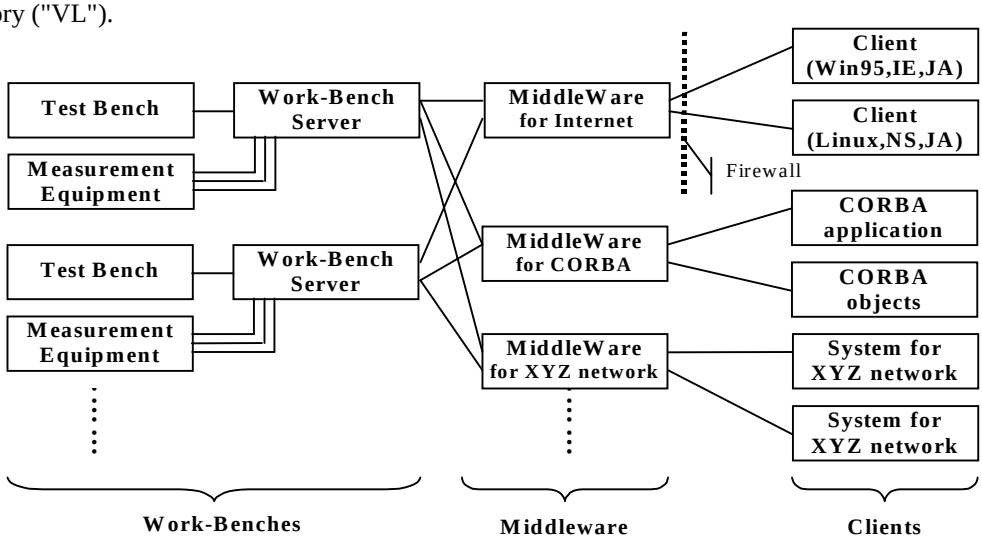

Fig. 1 - The generic architecture of a Virtual Laboratory system

2.1 The Virtual Laboratory architecture

The levels of this 3-tier architecture are*:*

a) The Work-Bench ("WB") Servers ("WBS") controlling (operation and access of only one user at a time) the Test-Benches ("TB") of automated Test & Measurement Equipment ("TME") - Power Supplies, Function Generators, RF Synthesized Generators, Digital Multi-Meters ("DMM"-s) and Digital Oscilloscopes - the Units-Under-Test ("UUT-s") and Relay Benches ("RB") implementing the IviSwtch function (for automated signal routing and sub-systems reconfiguration, test point multiplexing and load adjustment), connected to the Server through a Centronics interface. WBS host Data Acquisition ("DAQ") boards and are connected to their TB-s: IEEE488.2 (GPIB) interfacing to the automated T&M instrumentation, parallel Centronics interfacing to relay benches ("RB"-s); serial RS232C interfacing to micro-controllers ("µC"-s) development systems (part of them could manage more complex routing as well). The results can be sent to the users at the same time as they become available. These local configurations can be modeled according to the modern concept of "colonies" [5].

b) The Web-Server ("WS") Level ("WSL"); WSL is connected to WBS-s - integrated in a Local Area Network ("LAN") by a Gateway ("GW"). WS-s can optionally provide extra services such as messaging - in order to notify users who are waiting for the release of a TB, queuing - in order to implement First-Come-First-Served schemes or even automated guidance of experiments - where users can simply give a description of their experiment and the server will conduct it for them. The role of these Middleware ("MW") nodes is to mediate between the WBS-s and the clients (at the 3rd level). Obviously, there are many types of public networks that can be used to access a VL. Therefore, it is necessary to have a dedicated MW node for each type of access network that we want to integrate with our VL. Typical examples are MW for the Internet, for Novel networks or even for CORBA applications or Jini networks. MW is responsible for propagating the users' interactions to the proper server and vice-versa. In addition, it should be able to identify and authenticate the eligible users, protect their privacy and security and eventually provide them with some extra service such as a list of currently available WS-s. The MW can be extended to provide billing information and usage statistics that are very useful to Application Service Providers (ASP-s). ASP-s are companies that setup applications on their computers and hire them to interested individuals or organizations.

c) The Client Level ("CL"), groups the remote users of the VL. At CL, users' computers or devices are accessing the VL services through a network. They provide friendly and intuitive user interfaces that enable users to work with the remote TB-s in an almost realistic manner. What should be stressed is that clients can vary, from PC-s running MS Windows, to Mainframes running UNIX, to Jini enabled devices or even mobile phones. In addition, clients can even be SW components, such as CORBA or RMI objects [2] that act like virtual users or real user proxies. It is reasonable to expect that, in most cases, the users will be real life persons and their clients will be PCs running MS Windows or Linux. However, in the near future, Application Service Providers ("ASP") might offer VL services.

2.2 The software (SW) structure

VL SW implements databases ("DB"-s) - oriented users' management according to fig. 2.

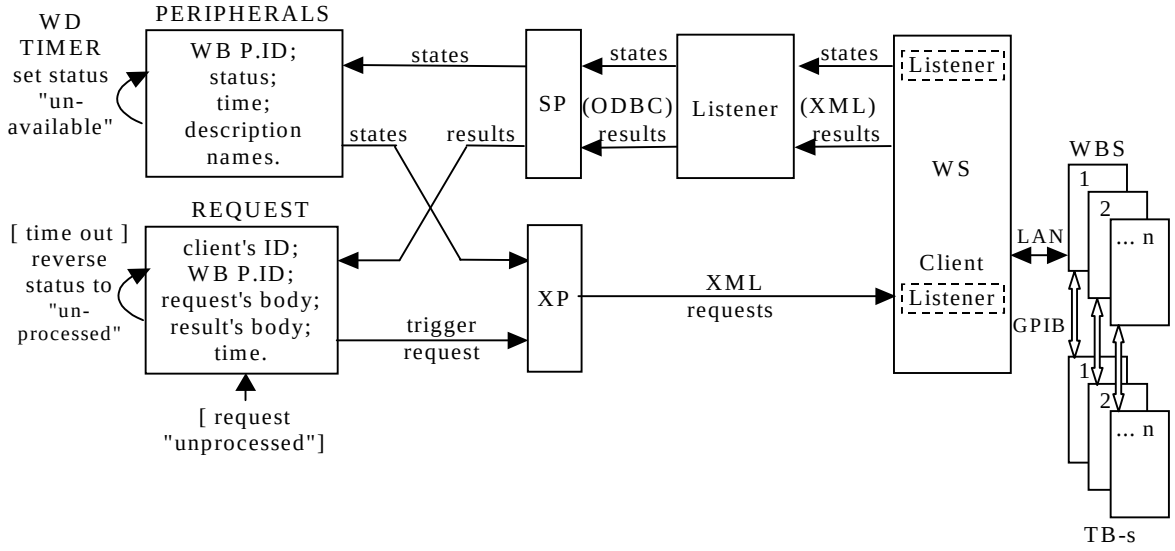

Fig. 2 - Information flow-chart of the DB-oriented users' management

The main form of the VL's Web page ("WP") redirects towards the login / new user registration form. The management is oriented on databases ("DB"-s) of users that, according to their rights, can enable or not the access to the VL. Users are defined by their "name", "password" and granted "rights" (e.g., self-registered new users can start only by simulation). Upon completion of their login form, registered users can start their work-session accessing the remote VL.

Users' resource allocation enables access to:

- A specific WP, for a configured TB;
- A dynamic configuration form that can match available colonies to user's need for a new T&M system. These new TB-s are given a Peripheral ID ("P.ID") that indicate the WBS and the required instruments.

Once a work-session was completed, each request is recorded in a MS SQL7.0 - managed DB. The specific records include the user's IP, the P.ID of targeted resources and the main body of the request that specify the members of instruments' colony involved in the experiment, their topology and mandatory capabilities, as well as the experimental variables (with the allocated memory for the results). The record is "time-stamped" with the moment of the request; a "time-out" management will return the message "unavailable WB" if results don't arrive to the DB.

The requests are triggering the SQL server (hosted by the WS) to translate the http- or SMTP-POP3 form of the message to XML, before GW sends it, to the WBS. This transfer can be done either by the C-programmed translation application assisting the DB or, directly, by network traffic through specific sockets - WinSock object in Visual Basic ("VB") or Data Socket Transfer Protocol ("DSTP"), very suitable for packages of experimental data.
The GW, listener for XML messages is building a queue analyzed by the WS according to the VL's DB, in order to decide when to send requests to WBS-s.

The DB of "Peripherals' States" is recording the P.ID-s, time-stamps, available resources (their amount, type and capabilities, as well as possible routing and connectivity), triggering events for the start/stop of experimental phases. This DB can record the "unavailable WB" signal from a Watch-Dog ("WD") after this one was waiting a predetermined time for an experimental result that didn't arrive.

WBS-s are watching the LAN in order to "pick-up" the messages addressed to them. At this 3rd level, these messages are translated in sequences of specific commands distributed via the multiple local interfacing (GPIB, Centronics or RS232C) to the instruments and RB-s. WS is periodically informed (via GW) with results and states of the experiment. Completion of the experiment triggers deletion of its corresponding request from the (WS hosted-) DB. This deletion can be done as well at the "time-out" signal sent by the WD in any abnormal circumstances.

3. PRACTICAL IMPLEMENTATION

In our implementation of the VL, we simplified the generic architecture and tailored it to the specific needs of a testing environment. We have integrated it with the main access network, the Internet. In fig. 3 they are also depicted our design choices (*cheapest general-purpose SW* that can take advantage of *all the capabilities* of our available instrumentation - see http://vega.unitbv.ro/~vlab) and the communication protocols.

3.1 Description of the sub-systems

The WBS are computers equipped with the appropriate HW (NI PCI-GPIB cards and AT-MIO16E10 DAQ cards) and SW - MS Win95 and VB, National Instruments ("NI") LabView and Component Works ("CW") elements - that enable them to control the TB-s. They receive instructions through the network, via the WS - MW. The specific SW that controls the TB-s was written in VB and contains CW elements. NI CW is grouping 32 bit controls that can be "plugged into" any ActiveX compatible software development packages (Visual C++, Delphi etc.) - in our case MS VB.

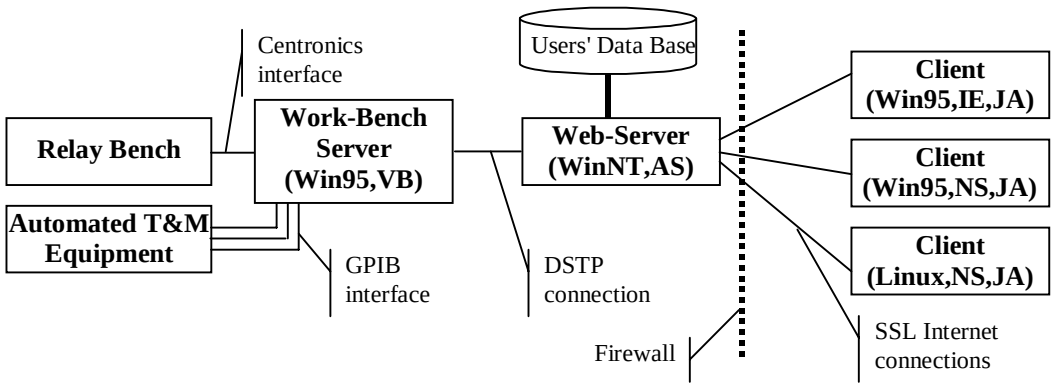

Fig. 3 - The architecture of our VL system

These specific controls are dedicated to D-AQ, processing and display, GPIB, VISA serial (and USB) controls (for the data flow between the controller PC and automated T&M equipment connected to GPIB) and data-socket controls, to exchange and distribute data between different “targets” as there are unified by DSTP (files, applications or web-servers of different kinds).

The WS (MW) is the coordinator of the whole system, monitoring the status of all WBS-s. It receives instructions from the CL and routes them to the appropriate WBS. It also returns the corresponding results as well as any system messages and logs all the activity.. It protects the system from hackers and bad-mannered users, allowing access only to eligible registered ones. The MW includes Application Server ("AS") SW, such as the MS Internet Information Server, able to carry out the aforementioned tasks. The necessary SW has been developed using VB as well. Communication of the WS with the WBS-s is accomplished via their common LAN, by DSTP.

The *Firewall* ("FW") is an optional component in the architecture, as it can significantly increase the system's security. The FW can be a SW package installed on the WBS computer or a separate computer or device, which filter the network activity and protect the LAN from hackers.

The user's computer at CL, employed for remote access the to the VL system is, usually, a PC running a Web browser (e.g. a PC Pentium III running Windows95 and Internet Explorer 5). However, other configurations can be used. In order to retain interoperability among various platforms and environments a Java Applet ("JA") - including TB simulation and a "hyper-schematic" (re-drawn as it can be reconfigured by mouse-clicks that, at the same time, cause *the appropriate remote switch in the RB*) - will be presented to the user to enable him/her to choose and conduct an experiment.
The connection between the CL and the WBS (MW) can be protected with the Secure Socket Layer (SSL) protocol, which enables encrypted and secure communication over the Internet.

3.2 Java Applet (JA) development

JA is a piece of SW that runs inside a Web browser. The Web browser should be Java-enabled and the Java Runtime Engine should be activated! The JA implements the front-end of the VL system to the user. The user should provide his/her login and password to the JA and the JA will log him/her in the system using SSL. Then the JA provides a list of available experiments (unallocated WBS) to the user in order to choose/reserve one for interactive experimentation. It also allows access to other services, mainly automated experiments as well as messaging and notification when a currently reserved WBS is released. The results will be automatically mailed back to the user.

The architecture of the JA running in the browser at CL consists of four distinct modules that carry out different types of tasks. These modules are connected in a tiered manner: User Interface ("UI") / Applet Logic ("AL") / Multiplexer-Dispatcher ("MD") / Communication ("CM"), enabling an easier maintenance of the applets. Each one uses the services of the module in the tier bellow.

The 1st module, *User Interface* ("UI"), provides the user with a familiar and easy to use way to communicate and interact with the VL. More specifically, this module provides the *Login* screen of the system, as well as a graphic representation of the TB (a virtual panel of the T&ME and the above-mentioned "hyper-schematic" of the UUT), which is to be remotely controlled. UI does most of the function calls that are usual in local operation.

The 2nd module, *Applet Logic* ("AL"), is responsible for reading and interpreting the user input, simulating the functionality of T&ME and UUT. For example, this module would allow only V (Volt) and mV as measurement units when setting voltage values. AL enforces a *behaviour* to the applet, whereas the UI merely provides the *visual part* of the applet. AL and UI modules communicate via direct function calls. Both of them can effectively simulate the way the real equipment would operate.

The 3rd module, *Multiplexer/Dispatcher* ("M/D") is responsible for conveying the instructions and settings collected by the UI, and verified/normalized by the AL modules, to the VL server, in order to deliver them to the actual equipment. This module communicates with the AL module using the M/D Application Programming Interface ("API"), which is a set of standard function calls for delivering instructions/settings using a predefined messaging format. This module of the applet is capable to accept messages (in the form of M/D API calls) from *multiple* UI-AL pairs. This is very important because it enables the *split* of the UI and AL modules to simpler and more manageable ones. Considering the complexity of the UI, this feature provides extra flexibility.

The M/D module is also responsible for receiving messages from the VL server and dispatching them to the appropriate AL module (as stated before, there might be more than one AL modules. This is also achieved via the callbacks mechanism of the M/D API. The callbacks mechanism enables the various AL modules to introduce themselves to the M/D module so that it can be able to route messages to them.

Eventually, the M/D module transforms the various M/D API calls to a multiplexed text message, in a predefined format that is comprehensible by the VL server. This format can be a proprietary one, like in our case, or an XML based format. In addition, the M/D module can convert the multiplexed text messages received from the server in to M/D API callbacks to the appropriate AL modules.

The 4th module, *Communication* ("CM"), is responsible for establishing a persistent connection to the VL server, and then sending multiplexed text messages from the applet to the server and vice versa. Furthermore, the CM module encrypts the messages sent and decrypts the messages received from the server, based on the password provided by the user when he/she logged in the system.

3.3 The multiplexed text message format ("MTMF")

MTMF is as defined below, in BNF form:

```
<MESSAGE_SET> ::=  <MESSAGE> '\n'
                                -- \n means line break
<MESSAGE>     ::= [<SERVICE> ':' <SUBSERVICE> ':'
                   <PARAMETERS>]*
<SERVICE>     ::=  <TEXT>
<SUBSERVICE>  ::=  <TEXT>
<PARAMETERS>  ::=  <PARAMETER> [',' <PARAMETER>]*
<PARAMETER>   ::=  <NAME> '=' <VALUE>
<NAME>        ::=  <TEXT>
<VALUE>       ::=  <TEXT>
<TEXT>        ::= [<LEGAL_CHARS>]*
<LEGAL_CHARS> ::=   any ASCII character
                    but ',' '=' and '\n'
```

In other words, a MTM set consists of 0 or more simple, single-line messages, of the form

```
Service1:subservice1:param1=value1,param2=value2,…
```

An example of such a message set is given bellow:

```
FUNCTION_GENERATOR:FREQUENCY:Freq=2.5, Unit=kHz
MEASUREMENT:VOLTAGE:
```

The meaning of the above message is:

a) The Function Generator should be set to generate a signal of frequency 2.5kHz.
b) The DMM is requested to send the current level of voltage.

The applets have been implemented using the Java programming language (the Java 2 Enterprise Edition - Java). The tool used for the creation of the applet, is the Inprise JBuilder 4 foundation edition. Furthermore, the M/D and CM modules were designed and developed manually in order to maximize control and flexibility of these modules.

3.4 VB-CW implementation of the hyper-schematic

Here is an example of VB programming for a switch of the hyper-schematic (displayed on user's computer as his/her browser accessed the Web page of the VL), associated with a boolean Com1.Value that remotely controls the 1st relay of the TB, corresponding to the most significant bit ("MSB") at the output of the Centronics parallel port of the lab-server. In order to avoid unwanted routing (e.g. short-circuits of one test-point to others, at *the single input* of a digital multi-meter) *exclusion* of other switching (associated with Com5.Value and Com6.Value in this example) must be provided. When a click of user's mouse dictates local flopping of the specific *icon of the switch* (CWButton1, from the NI CW set), the MSB of the byte "byt" will toggle (by boolean adding $128 = 2^7$ to the old value) and this new value is sent by DSTP (CWDataSocket control from the CW set) to the WS (according to the specification CWDataSocket1.ConnectTo "dstp://vlab.unitbv.ro/relay") wherefrom the new value is sent, by DSTP as well, to the parallel port of the WBS (with port_addr = $888_{(10)} = 378_{(16)}$).

```
Private Sub Com1_ValueChanged(ByVal Value As Boolean)
    If Com1.Value = True Then
        port_addr = 888
        byt = byt + 128
        written_byte = byt
        Call write_port(port_addr, written_byte)
        CWDataSocket1.Data.Value = CVar(byt)
        Com5.Value = False
        Com6.Value = False
        CWButton1.Value = True
    Else
        port_addr = 888
        byt = byt - 128
        written_byte = byt
        Call write_port(port_addr, written_byte)
        CWDataSocket1.Data.Value = CVar(byt)
        CWButton1.Value = False
    End If
End Sub
```

The DSTP protocol is very efficient for secure, non-interlocked, high-capacity experimental data transfer but invoking WS-hosted VB applications at CL requires digital signatures, creating security problems - at least warnings but, more often, running restrictions that could be overthrown, for instance, by packing a complete deployment of the VB application (with all .OCX needed) that could be downloaded by the user and entirely run on his PC. However, another problem could arise: the need of opening the specific port 3015 (dedicated to DSTP) by the user's web administrator. This problem was solved by the authors replacing the DSTP transfers (that are, almost all, *step-by-step* in the hyper-schematic approach, not taking a real advantage from the productivity of this protocol) with a simple WinSock transfer (command `CWDataSocket1.Data.Value = CVar(byt)` will be replaced by `tcpClient.SendData byt` and so on). Although JA does not have so many security problems, they are deficient in HW control capabilities and, as they have to be compiled on user's machine, they are *slower*. So, the CL-WS communication was implemented by Java but the WS-WSB communication was still implemented by MS VB + the aforementioned NI specific CW plug-ins.

4. CONCLUSION

There were discussed architectures for tele-measurement, with HW-SW layers starting with the core of automated T&ME (that requires specific solutions) up to general-purpose Internet programming,. Judging productivity versus ease of use, the authors preferred a "what you see is what you get" - WYSWYG simpler approach, very "visual" and yet able to be completed with video-conferencing support.

REFERENCES

[1] F. Sandu, W. Szabo and P.N. Borza, "Automated Measurement Laboratory Accessed by Internet", in *Proc. of the XVI IMEKO World Congress 2000*, Vienna, Austria, 25-28 September, 2000, vol. II, pp 111-116.

[2] A. Michalas, V. Zoi, N. Sotiropoulos, N. Mitrou, V. Loumos, E. Kayafas, "A Comparison of Multimedia Application Development Platforms towards the Object Web", *Computer Standards & Interfaces* 22 (2000) pp 13-26, Elsevier, Netherlands.

[3] J. M. Pieper, "Standard Commands for Programmable Instruments", SCPI Consortium, ACEA, Wierden, The Neteherlands, 1998.

[4] F. Sandu, D. Nicula, P. Ogrutan, "Implementations for a Virtual Electronics Laboratory", *Proc. of the 7th International Conference - "Electronics '98"*, Sozopol, Bulgaria, 23-25 October 1998, book 1, pp 133-138.

[5] Y.U. Cao, T.W. Chen, M.D. Harris, A.B. Kahng, M.A. Lewis and A.D. Stechert, "A Remote Robotics Laboratory on the Internet", Published by Commotion Lab, CS Dept, University of California, Los Angeles, at http://inet.nttam.com