computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

METHODOLOGY FOR DISTRIBUTED DESIGNING OF DISTRIBUTED VIRTUAL INSTRUMENTS

Wieslaw Winiecki

Institute of Radioelectronics, Warsaw University of Technology
Nowowiejska 15/19, Warsaw, Poland, E-mail w.winiecki@ire.pw.edu.pl

Abstract: *A generic architecture of a networked Distributed Virtual Measurement Instrument (DVI) is presented. General assumptions for distributed design of networked DVI's are described. The generic architecture of a distributed, networked and platform-independent environment for DVI designing, together with the generic methodology for distributed designing of networked DVI is proposed. Currently employed DVI design methodologies are presented and assessed.*

Keywords: - *Networked Distributed Virtual Instrument, Internet, design method*

1. INTRODUCTION

The development of telecommunication and computer technologies inspires creation of new device and system design methodologies. The traditional model based on a fixed infrastructure is being replaced by a new, virtual and flexible model based on the distributed information infrastructure. The new teleengineering methods are based primarily on the global infrastructure supplied by the dynamically developing Internet. The new designing paradigm, aided by the access to the wide-band Internet, can be called the distributed engineering paradigm.

We'll refer to the network-aware design aiding environment for distributed virtual measurement instruments (DVI) as the networked DVI design environment. It contains a programming environment together with the access mechanisms for the DVI designer. A compound environment, elements of which might be installed on different computers on a common network will be called a distributed networked design environment.

There are two primary groups of DVI design methodologies:

- Using integrated programming environments (e.g. LabVIEW, LabWindows/CVI, VEE),
- Using object-oriented languages (e.g. C++, Java),
- Other.

The most widely employed DVI design methodology is the one proposed by National Instruments [1]. Scarce non-commercial integrated design solutions are usually based on commercial products, e.g. [2], [3]. DVI projects written in C++ language are currently seldom encountered; on the other hand, the number of Java language realized projects rises owing to the language's operating system independence (platform independence) [3]-[8].

Among other DVI design methodology proposals, a method using state graphs [9] should be mentioned, as well as an interesting method of Internet-based simulator design, using the LabVIEW environment core [10], enhanced in [11] at [12] with elements allowing for designing not only simulators, but real, distributed virtual measurement instruments as well.

The synthetic comparison of the employed design environments and the generic environment, as well as the methodology of designing of the DVI's using these environments is presented in the tabular form in Table.1. The following notation has been assumed:

Y - denotes the presence of the option,

N - denotes the lack of the option,

[T] - denotes that the option might be present, but it is not mandatory,

[N] -denotes the lack of the option while requiring additional conditions to be true for the application to work properly (usually using non-standard solutions).

In the X/Y notation, the first symbol regards the design environment used while designing the DVI, while the second regards the completed DVI (the stage of using the DVI)

The analysis of the gathered information leads to a conclusion that none of the currently employed environments does not conform to the requirements for the generic environment, either concerning the architecture and the design methodology.

The current concept of DVI design assumed – in the case of three-layer structure of the design environment – the creation of the client application on the server machine and provisioning it to the client using a computer network, a WWW server and a web browser. In the case of the classic client-server environment architecture – designing the client application on the client machine, which requires the environment to be installed on the client computer, or designing the application on the server computer and installing it permanently on the client machine after creating an executable version.

The majority of the available commercial integrated design environments which can be used to design DVI's, has a few fundamental drawbacks: they are operating system dependent on both server- and client side, they require specialized runtime environments to be installed on the client machine. The advantage of these environments is a relatively simple and rapid design process using visual tools and the large number of specialized libraries.

Using general-purpose programming languages such as C/C++ to design DVI's significantly extends the design time and therefore makes the project more expensive. Furthermore, programs written in C/C++ are still operating system dependent, so preparing a multi-platform DVI software requires re-compiling (and possibly modifying) the source code for each of the target operating systems.

Table 1. The synthetic comparison of programming environments for DVI designing

Architecture elements	**LabVIEW, LabWindows/CVI (with web browser)**		**LabVIEW, LabWindows/CVI (without web browser)**		**VEE (with web browser)**		**VEE (without web browser)**	
Designing/using the DVI	client	server	client	server	client	server	client	server
Web browser	N/Y	Y/[Y]	N/N	N/N	N/Y	Y/[Y]	N/N	N/N
Web server	N/N	[Y]/Y	N/N	N/N	N/N	[Y]/Y	N/N	N/N
Programming environment fixed on the machine	N/N	Y/[Y]	Y/N	Y/[Y]	N/N	Y/Y	Y/Y	Y/Y
Database	N/N	[Y]/[Y]	N/N	[Y]/[Y]	N/N	[Y]/[Y]	N/N	[Y]/[Y]
Client application fixed on the client machine	-/N	-/-	-/Y	-/-	-/N	-/-	-/Y	-/-
Environment engine fixed on the machine	N/Y	Y/Y	Y/Y	Y/Y	-/-	-/-	-/-	-/-
Measurement application features								
Measurement control	-/Y	-/[Y]	-/Y	-/[Y]	-/N	-/[Y]	-/Y	-/[Y]
Result visualisation	-/Y	-/[Y]	-/Y	-/[Y]	-/Y	-/[Y]	-/Y	-/[Y]
Application compilation (creating executable version)	N/-	Y/-	[Y]/-	Y/-	N/-	N/-	N/-	N/-
Remote application compilation on the server	-	N/-	-	N/-	-	N/-	-	N/-
Environment/ Generated code operating system dependence	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y

Architecture elements	**Other environments RAD**		**Java environment with VJM**		**Environment [10-12]**		**Generic environment**	
Designing/using the DVI	client	server	client	server	client	server	client	server
Web browser	N/[Y]	N/[Y]	N/[Y]	[Y]/[Y]	Y/Y	[Y]/[Y]	Y/Y	[Y]/[Y]
Web server	N/N	N/[Y]	N/N	[Y]/[Y]	N/N	Y/Y	N/N	Y/Y
Programming environment fixed on the machine	Y/[N]	Y/N	N/N	Y/N	N/N	Y/Y	N/N	Y/N
Database	N/N	[Y]/[Y]	N/N	[Y]/[Y]	N/N	Y/Y	N/N	Y/Y

Client application fixed on the client machine	-/Y	-/-	-/[N]	-	N/N	-	N/N	-
Environment engine fixed on the machine	Y/Y	Y/Y	N/Y	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y
Measurement application features								
Measurement control	-/Y	-/[Y]	-/Y	-/[Y]	-/Y	-/[Y]	-/Y	-/[Y]
Result visualisation	-/Y	-/[Y]	-/Y	-/[Y]	-/Y	-/[Y]	-/Y	-/[Y]
Application compilation (creating executable version)	[Y]/-	Y/-	N/-	Y/-	Y/-	Y/-	N/-	Y/-
Remote application compilation on the server	-	N/-	-	N/-	-	N/-	-	Y/-
Environment/ Generated code operating system dependence	Y/Y	Y/Y	N/[N]	N/[N]	Y/Y	Y/Y	N/N	N/N

One of the operating-system independent languages is the Java language. Software written in this language requires a runtime environment to be installed on the computer, yet it is supplied free of charge by Sun Microsystems. Unfortunately, for the purpose of DVI design the language has a significant disadvantage of long application design time, stemming from the lack of generally available, complete, specialized libraries supporting the measurement system and virtual instrument design. The existing libraries (JavaBeans) contain a strongly limited set of "measurement' graphic components. There are no complete pure Java design environments allowing for remote designing of the DVI software from any client machine without installing a design environment on the computer. Existing environments, such as NetAcquire [13], allow only to supply server-side designed Java applets to the client machines. The proposal of the Java-based design environment presented in [10]-[12] uses the LabVIEW environment downloaded from the server to the client machine, which makes the client application operating system dependent. Furthermore, the distribution of linking and compiling processes between the server machine and the client machine, and particularly the necessity to compile the client application on the Clint machine, makes the design environment less universal and flexible.

Given the situation it is advised to create a new DVI design methodology, which would combine the advantages of the commercial design environments and the Java-based architecture.

2. THE PROPOSED GENERIC ARCHITECTURE OF NETWORKED DVI

The proposed generic architecture of a networked DVI is shown in Fig. 1.

The architecture blocks should function as follows:

Client application:
- Communication with the measurement server,

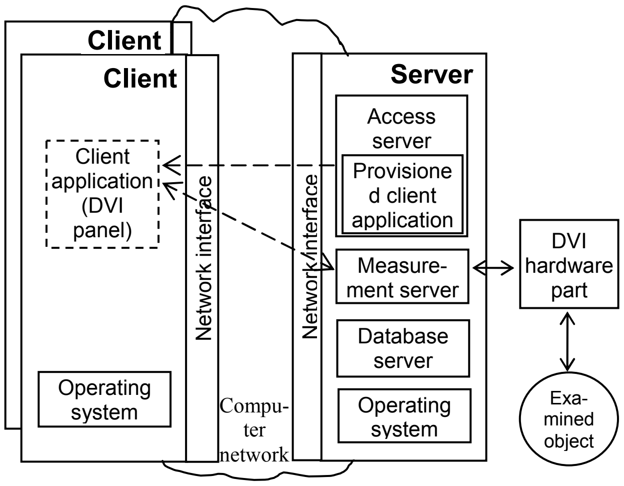

Fig. 1 - Generic architecture of networked DVI.

- Communication with the user, allowing for:
  - ♦ Programming the hardware DVI part controls,
  - ♦ Launching the measurement procedure,
  - ♦ Displaying the measurement results,
  - ♦ Interrupting the measurement process.

Measurement server application:
- Communication with the hardware DVI part,
- Communication with the client application,
- Communication with the database,
- Performing the measurement procedure (controlling the hardware part and the result acquisition).

Access server:
- Exposing the client application to the user through a computer network, for the purpose of temporarily setting up on the client computer

Database server:
- Storing the measurement data,
- Exposing the archived measurement data to the users.

Network interfaces:
- Allowing client and server computers to communicate through a computer network.

Operating system:
- Client and server computer management.

In a general case, the creation of DVI software requires the designing of:
- The communication between the server and the DVI hardware part (local bus communication protocol),
- The communication between the client and the server,
- Server-side measurement procedures (together with the software drivers for the DVI hardware part),
- Communication between the client application and the user (graphical user interface),
- Queuing or blocking different user originated tasks,
- Data storage (optionally) together with the mechanisms of exposing the data to the measurement server and the user,
- User authorization.

The requirements for the generic networked DVI might be expressed as:

Remote measurement execution and result visualization form any designer (user) computer connected to the network,

Communication between the client and the server using standard, open communication protocols.

Lack of necessity to permanently install the client software on the client computer (provisioning from the server machine),

Operating system independence of the client application (application portability),

Operating system independence of the server application,

Possibility of concurrent server accesses for numerous users.

3. THE PROPOSED GENERIC ARCHITECTURE OF A DISTRIBUTED NETWORKED ENVIRONMENT FOR DVI DESIGNING

The generic networked DVI distributed design methodology should allow for designing all the specific software blocks building the DVI software, using any client computer with any operating system, with no need to permanently install the client-side design environment, and additionally:

- the design environment might be set up on any client computer with any operating system,

- the process of designing may be interrupted at any moment, on any stage of the design, and resumed at any time using any client computer.

The proposed generic architecture of a distributed networked environment for DVI designing is proposed in Fig. 2.

The fundamental elements of the architecture are the two parts of the design environment:

I. Client part, provisioned to the client by the access server through the computer network for the purpose of temporarily installing on the client computer

II. Server part, containing, among others, procedures (software drivers) allowing for the communication with the hardware part as well as a compiler allowing for the remote compilation of the client application on the server.

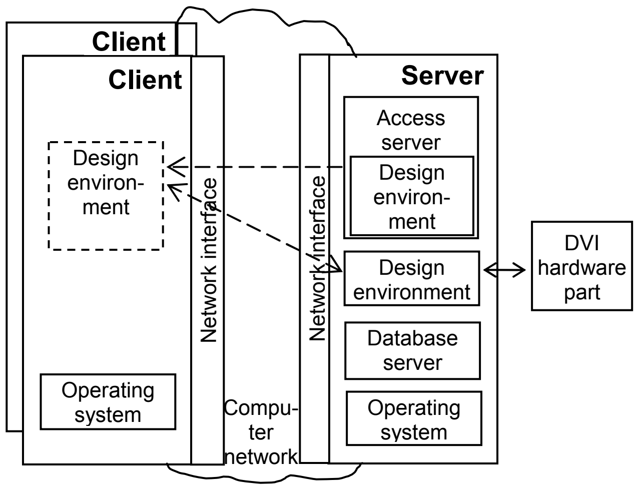

Fig. 2 - The proposal of a generic distributed environment architecture for DVI designing.

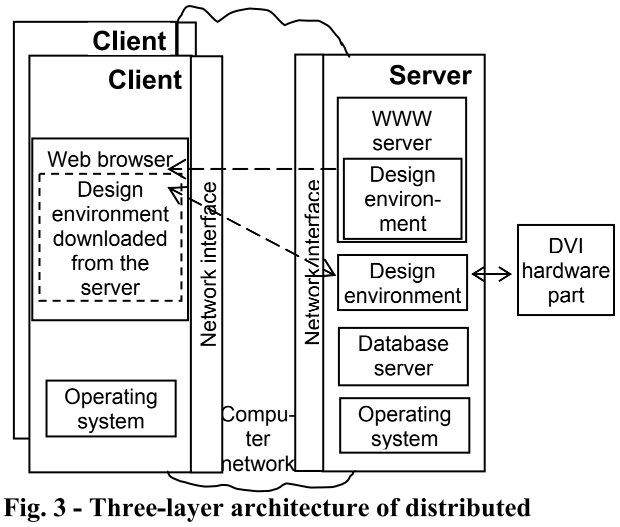

Fig. 3 - Three-layer architecture of distributed environment with the use of a web browser for DVI designing.

The mentioned architecture might be realized in two versions: as a classic client-server architecture (without a web browser), or as a three-layer architecture (with a web browser). The three-layer architecture has significantly richer capabilities, conforming to the assumptions for the generic network DVI distributed design methodology.

The three-layer architecture of a distributed DVI design environment, with a web browser installed on the client machine and a WWW server as the access server is shown in Fig. 3.

The generic architecture of a distributed DVI design environment should allow for realizing all of the assumptions for the generic distributed DVI designing methodology.

4. THE PROPOSED GENERIC METHODOLOGY FOR DISTRIBUTED DESIGNING OF NETWORKED DVI'S

Given the generic distributed DVI design environment architecture and the server-side installed environment, it is possible to form the following methodology for designing networked DVI's:

- Downloading the design environment to the client computer from the server through a computer network and running it on the client machine.
- Authorized project opening (either new one or previously partially designed) by the designer on the client machine.
- Designing a new graphical user interface (or modifying previously designed one) on the client machine, using objects available in the client design environment.
- Designing a new measurement algorithm (or modifying previously designed one) (meaning a handling program for events generated by the user using the DVI graphical panel, by the hardware part or by the computer) on the client machine using objects available in the client design environment.

Sending the RVMI design to the server through the computer network and storing the project in the database.

Sending the server-side compilation command to the server through the computer network.

Downloading the compiled project to the client computer and running the DVI project in the test mode.

All of the mentioned design activities should be strongly supported by the design environment. Furthermore, there should be a possibility to interrupt the design, to store it in the server-side database and to resume the designing process on another client machine.

5. VERIFYING THE PROPOSED METHODOLOGY OF THE INTERNET-BASED DVI DESIGN

To verify the validity of the proposed methodology, a programming environment called IBDT (Internet-Based Design Tool) has been developed, which allows for designing and executing distributed measurement applications.

The following assumptions were made for the proposed Internet-based DVI design methodology, inferred from the requirements for the generic network DVI and the generic design methodology:

Client application operating system independence,

Server application operating system independence,

HTTP protocol based communication,

Remote DVI software programming from any client machine,

Access to the uses projects from any client machine,

Possibility to store the design state on the server side,

Remote client application compilation on the server,

An easy method to add new measurement device drivers to the environment (lack of need to recompile the environment after adding a new hardware to the measurement server).

The possibility to meet the proposes assumptions were checked. The client- and server application platform independence can be supplied by using the Java language. The remote DVI designing from any client machine might be obtained by realizing the measurement server software as a web application accessible from typical web browsers, e.g. using Java servlets. Usually using the software drivers for the hardware DVI part is an obstacle to the operating system independence; preparing wrapper libraries, acquiring parameters from the input stream, placing results in the output stream and error messages in the error stream gives the possibility to overcome this constraint. In the case of changing the operating system, e.g. from MS Windows to UNIX, it is sufficient to replace the wrapper program without any modifications made to the DVI software. Furthermore, this solution allows for easy addition of the new instrumentation device drivers. After adding a new device to the measurement server it is sufficient to declare a wrapper name and its parameters. The possibility to store the DVI design state of the project works conducted on any client machine on the server side, might be provided by a database which would gather information concerning measurement devices, their functions, the users and their projects. The additional advantage of this solution is the possibility to define access rights to the particular measuring equipment.

The environment for designing of the networked DVI's has been created using Java language, which allows the environment itself, as well as the applications created with it, to be operating system independent. The only constraint will be the necessity to install the JDK (Java Development Kit) or the JRE (Java Runtime Environment) environment, supplied free-of-charge by Sun Corporation, on the client machine. The client application of the network DVI created using an environment conforming to the described methodology is a Java applet. Designing the client application involves designing the graphic DVI panel and the DVI program, building complete DVI software. The concept of the new approach to DVI design has been presented in [7] and [8]. The advanced version of the realized environment, named IBDT, has been presented in [13].

The IBDT environment consists of two parts: server-side environment and client-side environment. The server-side environment is responsible for communicating with the measurement devices, the database containing the information of the devices as well as for compiling the projects. The client-side environment is used for designing the panels of the virtual instruments and generating the code responsible for the measurement logic. The IBDT environment has to be installed on the measurement server, where the DVI hardware part of the DVI is connected. The client-side environment is downloaded to the client computer using web browser.

The scheme of the distributed design methodology for the networked DVI is presented in Fig.4.

The DVI control panel is created in a visual way, by placing components representing graphical control objects and graphical visualizing objects on the client application's main panel. The set of graphical control and visualizing objects, known as controls, includes: horizontal and vertical sliders of integer and floating values, knobs of integer and floating values, single- and double state switches, light emitting diodes, numeric displays, graphic displays etc.

The graphic environment supports creating the DVI software by supplying the list of devices currently attached to the server as well as their functions. Inserting the function calls to the DVI program is realized by wizards leading the user "by hand" from the device and function selection, through setting the parameters, up to the point of inserting a complete call into the code.

The complete source code of the client application is sent to the server and then compiled and placed in the web server folder, from where it can be downloaded and executed on the client machine.

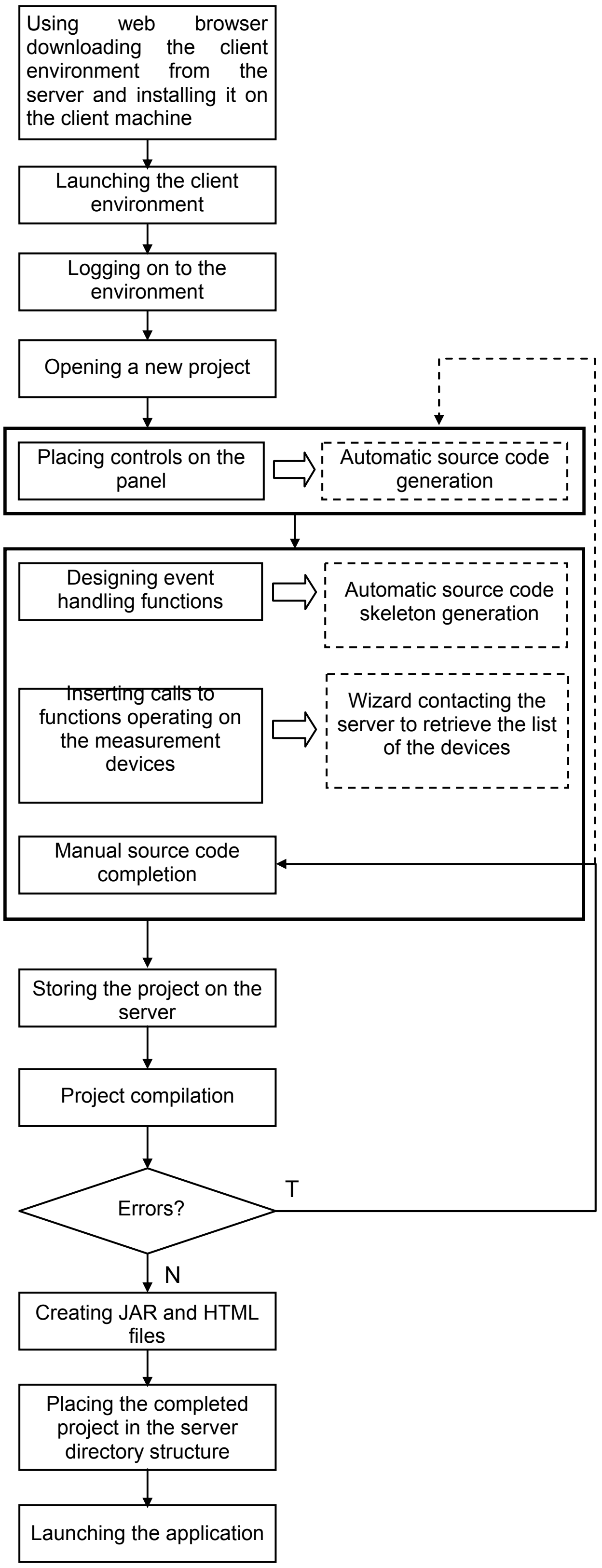
Using web browser downloading the client environment from the server and installing it on the client machine
Launching the client environment
Logging on to the environment
Opening a new project
Placing controls on the panel
Automatic source code generation
Designing event handling functions
Automatic source code skeleton generation
Inserting calls to functions operating on the measurement devices
Wizard contacting the server to retrieve the list of the devices
Manual source code completion
Storing the project on the server
Project compilation
Errors?
T
N
Creating JAR and HTML files
Placing the completed project in the server directory structure
Launching the application

Fig. 4 - The methodology of internet-based DVI designing with the use of the IBDT environment.

6. CONCLUSIONS

A new, generic architecture of networked DVI design environment together with a generic methodology of networked DVI designing is proposed. The new approach to DVI designing enable one to remotely design DVI via the Internet, giving an access to the user's project from any client machine, possibility to remotely compile the project together with storing its state on the server side and simple adding new measurement device drivers to the environment without the necessity to recompile the programming environment. None of existing programming environments is compatible with the proposed design methodology. Possibilities of creating such an environment, which is compatible with the proposed design methodology for DVI designing, were shown.

A platform independent, Java-based programming environment, called IBDT (Internet-Based Design Tool), has been developed to verify the validity of the proposed methodology. The three-layer architecture of the distributed, networked environment IBDT for DVI designing, with a web browser installed on the client machine and a WWW server as the access server, enables many users to design independently DVIs from their client computers in the same time. This feature is especially useful for education purposes.

7. REFERENCES

[1] LabVIEW - User Manual, ver.6.1. National Instruments Corp., Austin, 2002

[2] R. Wang, L. Wang , C. Geng, H. Zhou. The design of VPP Software Development Environment. Proc. IEEE IMTC 2002 Conf., Anchorage, AK, USA, 21-23 May 2002, pp. 403-408.

[3] M. Bertocco, M. Parvis. Platform Independent Architecture for Distributed Measurement Systems. Proc. IEEE IMTC 2000 Conf., Baltimore, USA, May 2000, CDROM.

[4] D. Grimaldi, L. Nigro, F. Pupo. Java-Based Distributed Measurement Systems, IEEE Transactions. on Instrumentation and Measurement, Vol.47, No 1, February 1998, pp. 100-103.

[5] D. Grimaldi, M. Marinov. Distributed Measurement Systems, Measurement, Vol.30, No 4, Dec. 2001, pp. 279-287.

[6] P. Bobinski, R. Jukaszewski, W. Winiecki. Designing Distributed DAQ Systems Using JAVA. Proc. of XVI IMEKO World Congress, Vienna, September 25-28, 2000, Vol. IV, pp. 45-48.

[7] M. Karkowski, W. Winiecki. A New Java-based Software Environment for Measuring Systems Designing. Proc. IEEE IMTC'99 Conf., Budapest, Hungary, May 21-23, 2001, pp.397-402.

[8] W. Winiecki, M. Karkowski. A New Java-based Software Environment for Measuring Systems Designing. Proc. IEEE Trans. on I&M, Vol. 52, No 6, Dec. 2002, pp. 1340-1346.

[9] F. Zubillaga Elorza, C. Allen, I. Leggett. A Design Automation Toolkit for Virtual instrumentation. IEEE IMTC'98 Conf., Minnesota, USA, May 18-21, 1998, pp. 338-341.

[10] A. Ferrero, V. Piuri. A Simulation Tool for Virtual Laboratory Experiments in a WWW Environment. IEEE Trans. on I&M, Vol. 48, No 3, June 1999, pp. 742-746.

[11] L. Benetazzo, M. Bertocco, F. Ferraris, A. Ferrero, C. Offelli, M. Parvis, V. Piuri. A Web-Based Distributed Virtual Educational Laboratory, IEEE Trans. on I&M, Vol. 49, No 2, April 2000, pp. 349-355.

[12] L. Benetazzo, M. Bertocco, F. Ferraris, A. Ferrero, A. Offelli, M. Parvis, V. Piuri. A Web-Based Distributed Virtual Education Laboratory. Proc. IEEE IMTC 2000 Conf., Baltimore, USA, May 2000. CDROM.

[12] http://netacquire.com

[13] W. Winiecki, T. Mielcarz. Internet-based Methodology for Distributed Virtual Instrument Designing. Proc. IEEE IMTC'2003 Conf., Vail, Colorado, USA, May 20-22, 2003, pp. 760-765.

__Wieslaw Winiecki__ received his MSc and PhD degrees from the Faculty of Electronics, Warsaw University of Technology, in 1975 and 1986, respectively. Since 1975 he has been with the Institute of Radioelectronics of that university. Since the beginning of his professional carrier he has been involved in the activities of the Group on Computer-aided Measurements concerning the hardware and software for measuring systems. Since 1987 he has worked as Assistant Professor on the problems of measurement automation and interface systems. He is the author of two books and over 100 scientific publications. Head of the Computer-Aided Measurement Laboratory; member of the Measuring Systems Section of the Metrology and Instrumentation Committee, Polish Academy of Science; vice-president of the Measurement Committee of the Polish Society for Measurement, Automatic Control and Robotics POLSPAR; and member of IEEE.