A&QT-R 2004 (THETA 14)
2004 IEEE-TTTC - International Conference on Automation, Quality and Testing, Robotics
May 13 – 15, 2004, Cluj-Napoca, Romania

DISTANCE LEARNING IN CONTROL BASED ON CLIENT-SERVER APPLICATIONS - A CASE STUDY

C. Mahulea, C. Lefter, M.H. Matcovschi, O. Pastravanu

Department of Automatic Control and Industrial Informatics
Technical University "Gh. Asachi" of Iasi
Blvd. Mangeron 53A, Iasi, 700050, ROMANIA
Phone/Fax: +40-232-230751, E-mail: opastrav@delta.ac.tuiasi.ro

Abstract: The paper deals with the design, implementation and exploitation of Web-based laboratories for education in Control Engineering. The key principles and techniques are discussed for the concrete case of a software package called Petri Net Web-based Laboratory (abbreviated as PNWL), which allows the Internet training in discrete event systems modeled by Petri nets. The main objectives envisaged by the PNWL are: the simulation of Petri nets within a familiar framework (using any Java-supported Internet browser), the independence of the operation system, the integration with any http server, the convenient development of further facilities. Moreover, the PNWL was meant to ensure full compatibility with the MATLAB software that represents a standard for the computational approaches in Control Engineering. The overall conception of PNWL as a client-server application can be successfully exploited to develop various Internet-based services for practical exeercises and experiments in distamce learning.

Keywords: control education, distance learning, client - server applications, Web-based laboratory, discrete-event dynamic systems, Petri nets , MATLAB.

1. INTRODUCTION

The possibility of using Web for education generated a great deal of interest among educators throughout the world. Consequently, a large body of work has been invested to create various programming facilities to assist the progress of distance learning methodologies. The current trend in exploiting the Internet for distance learning focuses on the development of Web-based laboratories that allow students to perform experimental studies by sharing the resources available from a remote host (university department, research center etc.).

In the Romanian universities, Web-based education is drastically delayed in comparison with the continuous progress achieved by the groups of professionals working in the Western universities. Current outcomes are, in general, limited to the following two aspects: (*i*) Diversification of teaching materials (complementary information, tutorials or bibliographic references can be downloaded from various sites); (*ii*) Reduction of costs for printed texts (lecture notes or laboratory lessons are available as *pdf* or *ps* files on local servers). Although the development of tools

adequate to web-based distance learning is claimed as a priority for many universities, such approaches are in a very early stage, despite the freeness of charge for most part of know-how in this field. It is worth saying that in the long list of possible reasons explaining this slow rhythm in the Romanian universities, a crucial position is obviously hold by the poor services offered by *ro.edu.net*.

Our paper briefly presents the results of a project initiated at the Department of Automatic Control and Industrial Informatics of the Technical University "Gh. Asachi" of Iasi, for creating a Web-based laboratory integrated with the course of Discrete Event Systems, called *Petri Net Web-based Laboratory* (abbreviated as PNWL). The PNWL has been constructed so as to ensure full compatibility with the MATLAB software that represents a standard for the training in Control Engineering. The PNWL has been designed and implemented as *a client - server application*, relying on web technologies such as http, soap, xml, jaxm, and envisaging the following objectives: (i) the simulation of Petri nets within a familiar framework (using any Java-supported Internet browser), (ii) the independence of the operation system, (iii) the integration with any http server, (iv) the convenient development of further facilities.

2. PNWL AT A FIRST GLANCE

The PNWL allows the simulation of four classes of Petri net models, namely: untimed, transition-timed, place-timed and stochastic Petri nets. The dialogue with the user is ensured by a graphical interface. This was built in Java, by using the libraries **awt** and **swing** so as to permit the user's full control of all the options. In its design the authors derived a full benefit from their previous experience aquired during the implementation of the Petri Net Toolbox for MATLAB [2], [4].

Figure 1 presents the main window that appears whenever PNWL is started by an http browser. This figure permits a quick visualization of the following six elements whose role is detailed below: (1) Menu Bar, (2) Toolbar, (3) Drawing and simulation panel, (4) Sim/Draw button, (5) Drawing area, (6) Status Bar.

The **Menu bar** (1) is placed horizontally, in the upper side of the window and displays a set of seven menus by means of which the user can select all the facilities available in PNWL. These menus are briefly described in the following: The **File** menu (including the sub-menus **New**, **Open**, **Save** and **Exit**) provides functions for handling files. The **Options** menu allows the user to personalize the visualization and simulation conditions, by choosing the graphical theme of the entire application ("Look and feel" principle), the initial color(s) of the Petri net nodes, the animation speed when the simulation is run in the slow mode. The **Draw** menu offers tools for graphical editing (places, transitions, arcs, markings and labels) in the drawing area. The **Petri Net Type** menu permits assigning the net type to a new model, or changing the net type of an already existing model. The **Simulation** menu gives the user the possibility to control the simulation progress, to record the simulation results and to specify the stop condition when the simulation is run in the fast mode. The **Analysis** menu provides tools for studying dynamical properties of the net. The **Help** menu offers on-line information on the exploitation of PNWL.

The **Toolbar** (2) is equipped with a number of buttons mapping the most frequently used facilities available in the menus displayed by the **Menu Bar**. The **Drawing and Simulation Panel** (3) maps either the drawing or the simulation commands in accordance with the user's option selected via the **Sim/Draw Button** (4).

Within the **Draw** mode, the **Edit** button enables the access to each graphical element of the net displayed in the **Drawing Area** (5), by means of specific dialogue boxes. Within the **Sim**ulation mode, the **Drawing Area** (5) accommodates the performance of the simulation, for which the user can choose the running type: step-by step (accompanied by animation), slow (accompanied by animation) and fast (without animation). The **Status Bar** (6) displays information typical to the current mode of the PNWL exploitation, selected by the **Sim/Draw Button** (4). The information corresponding to the **Sim**ulation mode comprises the simulation time elapsed and the number of events occurred since the beginning of the simulation.

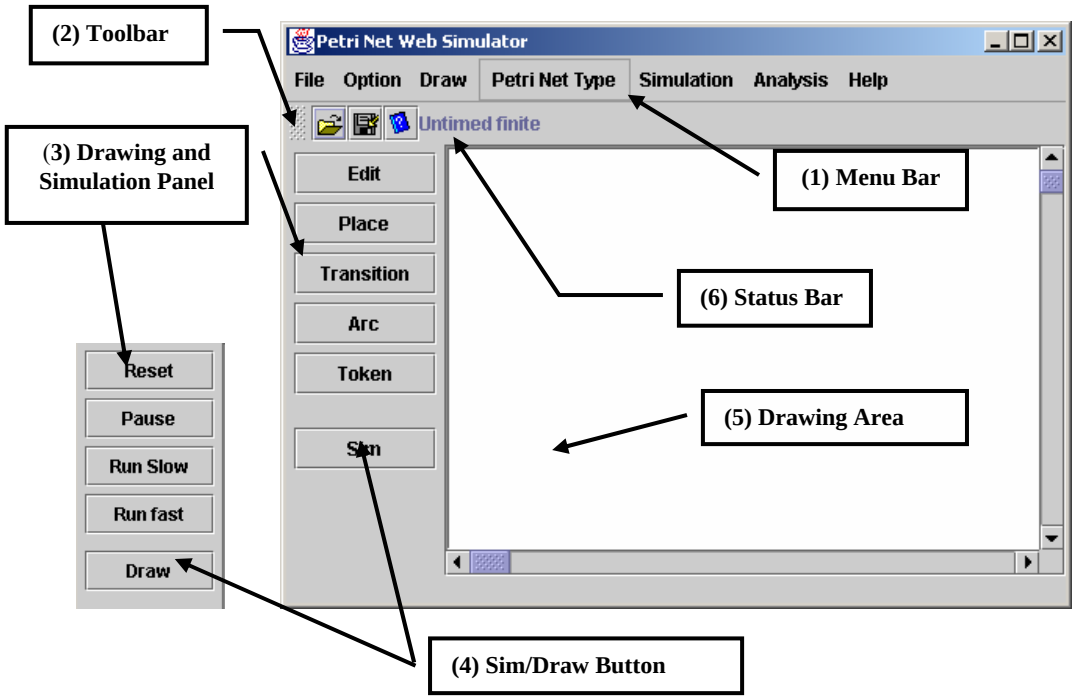

Figure 1. The main window of the PNWL opened by an http browser.

3. DESIGN AND IMPLEMENTATION OF PNWL AS A CLIENT - SERVER APPLICATION

The general organization of the client-server application can be synthetically presented by means of the diagram in figure 2 that points out the interaction between the main software modules.

3.1. The Client Application

The key functionality of PNWL can be visualized only from the client side, by the help of a Java frame. The choice of Java [6] as the programming language for developing this application resulted in two remarkable advantages: (i) the independence of the platform, meaning that the application runs identically under different operating systems; (ii) the usage of the same code by a Java applet, a web page, or a standalone application.

The application can be integrated with an html page and can be started by the Internet browser as a regular applet, which, thus, imposes its security requirements for running. In the definition of the applet, the identifier of a Petri net model can be

introduced as a parameter, and, consequently, the PNWL is initialized with the net corresponding to that identifier. Since the implicit configuration prevents the Java code from accessing the local file system, the **Load/Save** option of the **File** menu is activated only when such operations become possible. It is worth mentioning that javapolicy [5] can be used to personalize the users' rights for file handling.

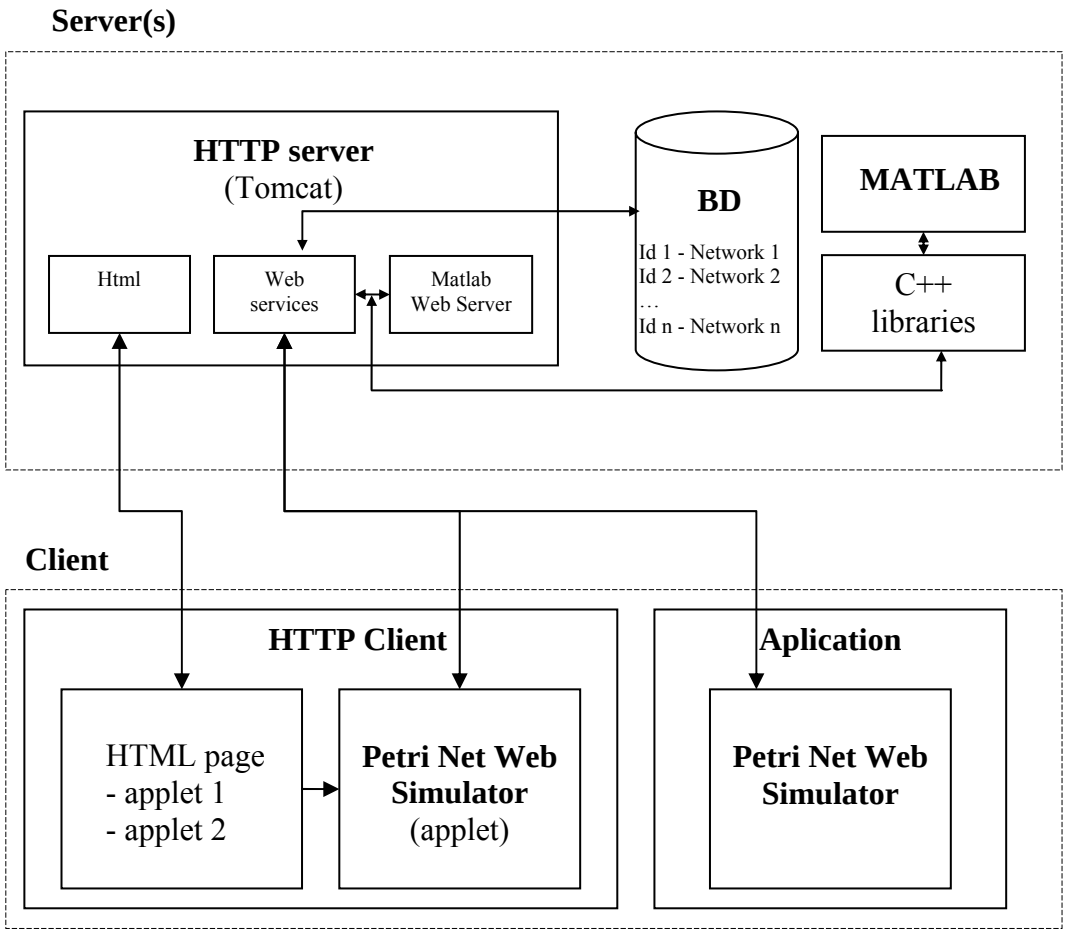

Figure 2. Schematic presentation of the interactions between the modules of the client - server application

3.2. The Server Application

The server application is organized modularly. It contains three independent components (modules) that can operate on the same computer, or in different locations (case when the http protocol is used for communication). The only advantage resulting from the usage of a single machine for hosting the three components is the increased speed in responding to the PNWL requests.

The *http Server* represents the first component of the server application and it can be any http service provider ensuring compatibility with the specifications of the Servlet 2.3 technology. It stores the application code in a compressed form that is sent to the client when the client loads an html page containing one or several applets of the application. Once an adequate html page is accessed, the **Launch Simulation Button** appears on the screen only after all application files are loaded in the cache.

The location of the application code on the hhtp Server means that the user needs no specific configuration before launching the PNWL and, moreover, he/she has always access to the latest version of the application. When the http Server receives, from a client, a request for the web services typical to PNWL, specific procedures are started to communicate either with the database of the application or with the MATLAB via C++ libraries [7] or the MATLAB Web Server [8]. Eventually, the results are returned to the corresponding client. The modular organization of the application allows data to be processed on any machine that is able to communicate with the http Server. The locations of the PNWL database, C++ libraries and MATLAB Web Server are given only in the configuration files of the http Server (i.e. ensuring total transparency from the client's point of view).

MySQL represents the second component of the server application and it administrates the following types of stored information about the Petri net models: (i) predefined models, (ii) intermediate steps requested by the simulation of the predefined models so as to reduce the number of MATLAB calls; (iii) a local cache of the simulated models (other than the predefined ones); (iv) models saved by the users.

The *MATLAB connecting module* represents the second component of the server application and its role consists in controlling the MATLAB calls. This module has been implemented by two different strategies: (i) the integration of the MATLAB Web Server with the http Server by means of cgi scripts; (ii) the usage of the C++ libraries via api interfaces.

In strategy (i), the MATLAB is started by the MATLAB Web Server and keeps running in the background no matter if there exist requests from clients. This strategy is preferred whenever the MATLAB and the http Server run on different machines. The server includes the called MATLAB functions and their input parameters into a post-type request, and the results are available from the response page of the MATLAB Web Server. In strategy (ii), the C++ libraries operate as a gateway between Java and MATLAB, which, in the background, opens a MATLAB session for the called functions, and their results are returned to the Java programs. This strategy is preferred whenever the MATLAB and the http Server run on the same machine.

Requests and replies within the framework of the application are conveyed on the world wide web. For storing the information about the Petri nets, xml files [10], are used. This technique allows the exploitation of the soap protocol [9] for the client-server communication (operating as an exchange of structured information within a decentralized and distributed environment).

4. EXPLOITATION OF PNWL - AN EXAMPLE

Consider a sending - receiving system, adapted from [3] and [1], whose Petri net model is introduced in the **Drawing Area** of PNWL as shown by the screen capture given in figure 3.

The simulation of the untimed Petri net provides qualitative information, proving the cyclic operation induced by the send - acknowledge mechanism. If a T-timed Petri net model is used, the simulation also supplies quantitative information that permits the analysis of the dynamics in terms of the efficient exploitation of the resources. Assume that to the six transitions of the Petri net model we assign exponential distributions with the following parameters: 3 ms for t1, 4 ms for t2, 2 ms for t3, 3 ms for t4, 5 ms for t5, 1 ms for t6. The steady-state functioning is characterized

by the performance indices: a period of 15.6 ms, a utilization degree of over 40% for the communication channel; a utilization degree of 30% for the equipment preparing data to be sent; a utilization degree of 70% for the equipment processing the received data.

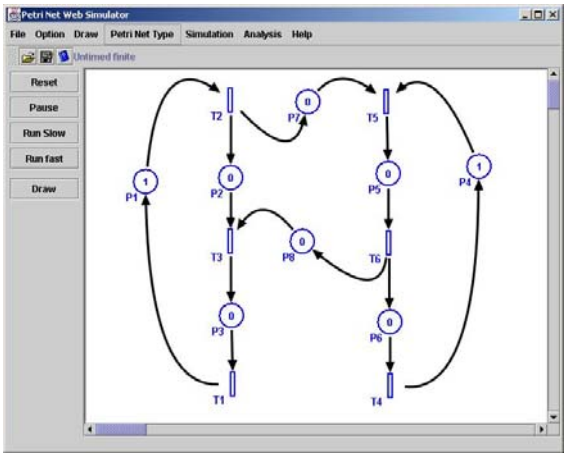

Figure 3. Screen capture of the PNWL Drawing Area containing the Petri net model discussed by Example 1

5. CONCLUSIONS

We have presented the design and implementation of the PNWL - an Internet-based laboratoty for training in discrete-event systems modeled by Petri nets. The client – server philosophy used for creating the PNWL ensures a large flexibility of the application and permits its further development relying on the same implementation techniques. The whole paper proves the high potential offered by the client - server approach to the construction of modern instruments for distance learning, focusing on the extensive exploitation of the Internet resources.

6. REFERENCES

1. Desrocheres, A.A. and R.Y. Al-Jaar (1993). *Modeling and control of automated manufacturing systems*. IEEE Computer Society Press, Rensselaer, Troy, New-York.
2. Matcovschi, M., C. Mahulea, and O. Păstrăvanu (2003). Petri Net Toolbox for MATLAB, *Proc. of the 11th IEEE Mediterranean Conference on Control and Automation MED'03*, Rhodes, Greece, 18-20 June, 2003, Abstracts pp. 71, CD-ROM.
3. Murata, T. (1989). Petri nets: properties, analysis and application. In: *Proc. of the IEEE*, **77**, pp.541-580.
4. Pastravanu, O., M. Matcovschi and C. Mahulea (2004). Petri Net Toolbox – teaching discrete event systems under Matlab. In (Voicu, M., Ed.), *Advances in Automatic Control*, pp. 247-256, 2004, Kluwer Academic Publishers, Boston.
5. Sun Microsystems, Inc. (2002a). *Java 2 Platform Security* (http://java.sun.com/ j2se/1.4/docs/guide/security/index.html)
6. Sun Microsystems, Inc. (2002b). *Java Technology and Web Services* (http://java.sun.com/webservices/docs.html)
7. The MathWorks Inc. (2001a). *MATLAB C/C++ Math Library*. Natick, Massachusets.
8. The MathWorks Inc. (2001b). *MATLAB Web Server*. Natick, Massachusets.
9. World Wide Web Consortium (2001a). *SOAP Version 1.2 Part 1: Messaging Framework* (http://www.w3.org/TR/2001/WD-soap12-part1-20011217/)
10. World Wide Web Consortium (2001b). *XML Fragment Interchange*. (http://www.w3.org/TR/ xml-fragment)