PETRI NET TOOLBOX FOR MATLAB IN WEB-BASED ANALYSIS AND DESIGN OF DISCRETE-EVENT SYSTEMS

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

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

Abstract: The paper presents the results of the project ***Petri Net Web-based Laboratory*** (***PN Web-Lab***) that has been developed for the training of the Control Engineering students in *discrete-event systems* (DES) modelled by Petri nets. The main objectives envisaged by the ***PN Web-Lab*** are: the simulation, analysis and design of DES 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. The ***PN Web-Lab*** was meant to ensure full compatibility with the MATLAB software that represents a standard for the computational approaches in Control Engineering. The ***PN Web-Lab*** was implemented as a client-server application, offering a large flexibility in exploitation. An example illustrates the usage of some tools provided by the ***PN Web-Lab***. The overall conception of the ***PN Web-Lab*** can be successfully exploited to develop various Internet-based services for laboratory exercises and experiments in distance learning. *Copyright © 2005 IFAC*

Keywords: Control Engineering education, Web technologies, discrete-event systems, Petri nets, MATLAB.

1. INTRODUCTION

During the last decade, many universities offering education programs in Control Engineering (CE) included, in their curricula, courses on Discrete Event Systems (DES), as a consequence of the large impact on research and technology proved by the development of this field. The study of DES was founded on an interdisciplinary background, connected to several areas of Mathematics, among which the most consistent contributions were brought by automata and formal languages, queuing systems, Petri nets, algebraic theory of synchronization and perturbation analysis. Despite the efforts invested by several scientists for the unification of the research interests and mathematical approaches as well, the domain of discrete event systems is still far from its maturity. However, within this heterogeneous framework, the Petri net (PN) formalism proved a large flexibility in ensuring the compatibility with other theoretical instruments, in the sense that the original theory proposed by Petri in 1962 was naturally extended to incorporate timing information, description of continuous-time phenomena etc. Consequently, recent literature has reported the usage of the PN models in many CE applications, fact that motivates the growing interest of the educators for including the PN theory in the current syllabuses of DES courses.

The organization of laboratory classes relevant for DES analysis and synthesis based on the PN formalism requires specific software tools, able to cover the complexity of such problems. Although many academic or research groups developed various program packages for studying PNs (Mortensen, 2004), these software platforms are not familiar for CE students, whose regular practical training focuses on the exploitation of the generous resources provided by MATLAB and its specialized toolboxes. The idea of bridging the Petri net formalism with the widely spread usage of MATLAB represented the starting point for the design and implementation of the ***Petri Net Toolbox*** (***PN Toolbox***) at the Department of Automatic Control and Industrial Informatics of the Technical University "Gh. Asachi" of Iasi. The development of the ***PN Toolbox*** for MATLAB commenced in 2000 and, initially, it was intended as a software product of internal usage for the laboratory classes accompanying a course on DES. The first version was available in 2001 and our own tests lasted for more than one year, also involving the participation of some fifty students who prepared homeworks and semester projects (Pastravanu *et al.*, 2004). In 2003 we released version 2.0 of the ***PN Toolbox***, that, after passing the tests of The MathWorks Inc., was included in the Connections Program promoted by this software company (http://www.mathworks.com/products/connections). Thus, the ***PN Toolbox*** broadens the utilization domain of MATLAB toward the area of discrete-event systems, which is now covered only by the State-Flow package, based on the automata theory.

This paper presents the current results of the project ***Petri Net Web-based Laboratory*** (***PN Web-Lab***) that was initiated at the department of the Technical University "Gh. Asachi" of Iasi, in order to ensure the exploitation of the ***PN Toolbox*** in distance learning. The ***PN Web-Lab*** has been constructed as *a client - server application*, relying on web technologies such as http, soap, xml, jaxm, and envisaging the following objectives: (i) the simulation, analysis and design of PNs 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. At the conceptual level, the paper reflects our efforts for the modernization of the education aids in CE, as well as for enlarging the compatibility between the novelty of DES scenarios and the traditional background and programming skills of CE students.

The presentation of the ***PN Web-Lab*** is organized according to the following plan: Section 2 describes the implementation as a client-server application; Section 3 gives a brief guide of exploitation; Section 4 focuses on the role of visual information and animation; Section 5 illustrates by an example some of the facilities created for analysis and design; Section 6 formulates several concluding remarks on the importance of our work.

2. IMPLEMENTATION OF THE PN WEB-LAB

The ***PN Web-Lab*** is implemented as a client-server application, whose general organization can be synthetically presented by means of the diagram in fig. 1. This diagram emphasizes the interaction between the main software modules.

2.1. The Client Application

The key functionality of ***PN Web-Lab*** can be visualized only from the client side, with the help of a

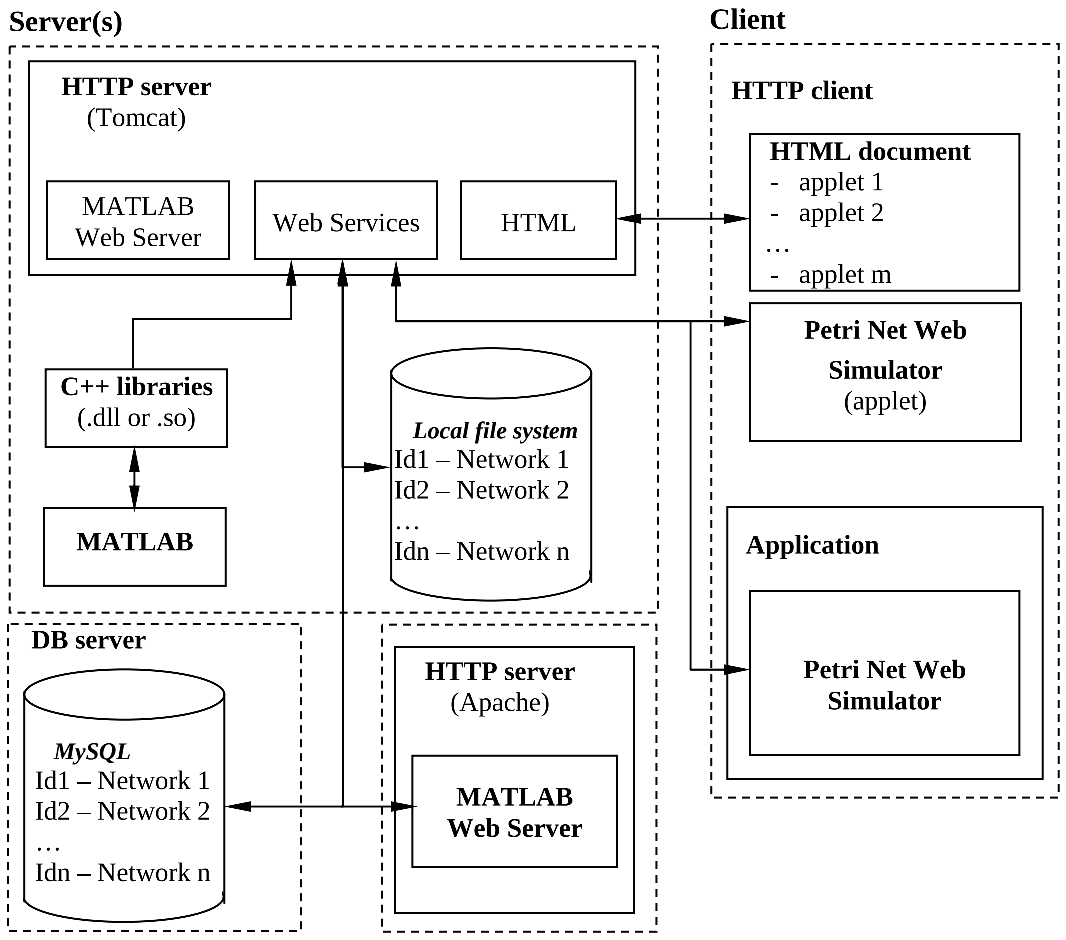


Fig. 1. Schematic presentation of the interactions between the modules of the client - server application.

Java frame. The choice of Java (Sun Microsystems, Inc., 2002) 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 stand-alone 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 ***PN Web-Lab*** 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 (see the details given by the next section) is activated only when such operations become possible. It is worth mentioning that javapolicy can be used to personalize the users' rights for file handling.

2.2. 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 ***PN Web-Lab*** 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** button appears on the screen only after all application files are loaded in the cache.

The location of the application code on the http Server means that the user needs no specific configuration before launching the ***PN Web-Lab*** 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 ***PN Web-Lab***, specific procedures are started to communicate either with the database of the application or with the MATLAB via C++ libraries or the MATLAB Web Server (The MathWorks Inc. 2001). 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 ***PN Web-Lab*** 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 third 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 when 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 when 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 are used. This technique allows the exploitation of the soap protocol for the client-server communication (operating as an exchange of structured information within a decentralized and distributed environment).

3. DESCRIPTION OF THE PN WEB-LAB

3.1. Facilities

In the current version of the ***PN Web-Lab***, five types of classic PN models are accepted, namely: untimed, transition-timed, place-timed, stochastic and generalized stochastic. The timed nets can be deterministic or stochastic, and the stochastic case allows using appropriate functions to generate random sequences corresponding to probability distributions with positive support. The default type of an arc is regular, but the user is allowed to change it into double or inhibitor, if necessary.

The ***PN Web-Lab*** has an easy to exploit *Graphical User Interface* (GUI) with a twofold purpose. First, it gives the user the possibility to draw PNs in a natural fashion, to store, retrieve and resize (by Zoom-In and Zoom-Out features) such drawings. Second, it permits the simulation, analysis and design of the PNs, by exploiting all the computational resources of the environment. Unlike other PN software, where places are meant as having finite capacity, our toolbox is able to operate with infinite-capacity places. In addition, the ***PN Web-Lab*** allows the assignment of priorities and/or probabilities to conflicting transitions.

After drawing a PN model, the user can: (i) visualize the *Incidence Matrix*, which is automatically built from the net topology; (ii) explore the *Behavioral Properties* (such as liveness, boundedness, reversibility etc.) by consulting the *Coverability Tree*, which is automatically built from the net topology and initial marking; (iii) explore the *Structural Properties* (such as structural boundedness, repetitiveness, conservativeness and consistency); (iv) calculate *P-Invariants* and *T-Invariants*; (v) run a *Simulation* experiment; (vi) display current results of the simulation using the *Scope* and *Diary* facilities; (vii) evaluate the global *Performance Indices* (such as average marking of places, average firing delay of transitions etc.); (viii) perform a *Max-Plus Analysis* (restricted to event-graphs); (ix) *Design* a configuration with suitable dynamics (via automated iterative simulations).

3.2 Exploitation

There are two modes in which the ***PN Web-Lab*** may be exploited, namely the *Draw Mode*, that allows the user to build a new PN model or modify the properties of an existing one, and the *Explore Mode* that enables the user's access to simulation, analysis and design tools. The GUI exhibits eight control panels (see fig. 3 in the next section): *Menu Bar* (1), *Quick Access Toolbar* (2), *Drawing Area* (3), *Drawing Panel* (4), *Draw/Explore Switch* (5), *Simulation Panel* (6), *Status Panel* (7) and a *Message Box* (8). Further on, all these panels are briefly described.

The *Menu Bar* (1) displays a set of nine drop-down menus, from which the user can access all the facilities available in the ***PN Web-Lab***. These menus are enabled in accordance with the exploitation mode of the ***PN Web-Lab***.

The *File* menu offers facilities for file-handling operations. This is the only menu available when the ***PN Web-Lab*** GUI is started. The *Modeling* menu provides tools for graphical editing (graph nodes, arcs, tokens, labels) a model in the *Drawing Area*. The *View* menu allows choosing specific conditions for visualization of the current model. The *Properties* menu provides computational tools for the analysis of the behavioral and structural properties of the current PN model. Through the *Simulation* menu the user may control the simulation progress and record the results. At the end of a simulation experiment, the *Performance* menu allows the visualization of the global performance indices that are stored in an HTML format. These indices are separately recorded for transitions and for places. The *Max-Plus* menu allows performing the simulation and analysis of an event graph (marked graph) based on its max-plus state-space model. A new figure is opened and all the facilities available for max-plus analysis and simulation are accessible. The *Design* menu is used for the synthesis of timed PN models; this allows simulations for several types of parameterizations considered in the PN architecture. The *Help* menu provides information for the exploitation of the ***PN Web-Lab*** and allows visualization of four Flash demo-movies initiating the user in the exploitation of the ***PN Web-Lab***.

The *Drawing Area* (3) is provided with a grid, where the nodes of the PN graph are to be placed, and with two scrollbars (on the right and bottom sides) for moving the desired parts of the graph into view. The *Drawing Area* is organized as a matrix of cells with 50 rows and 50 columns. In one cell the user can draw a single node (place or transition). The *Drawing Panel* (4) presents five image buttons that facilitate user access to *Edit Objects*, *Add Place*, *Add Transition*, *Add Arc* and *Add Token* commands. Similarly, the *Simulation Panel* (6) presents buttons for *Reset*, *Step*, *Run Slow* and *Run Fast* commands. It also provides two instruments for visualizing the progress of the simulation: *Diary* and *Scope*.

The simulation is driven by an asynchronous clock corresponding to the occurrence of events. In the untimed case, the sequencing of the events is reduced to simply ordering their occurrence, without any temporal significance, unlike the timed case when simulation requires a continuous correlation with physical time. Three modes of simulation are implemented in the ***PN Web-Lab***, namely: *Step*, *Run Slow* and *Run Fast*. The *Step* and *Run Slow* simulation modes are accompanied by animation; the user can record the progress of the simulation in a log file with HTML format. After ending a simulation (run in any of the three modes) a number of *Performance Indices* are available to globally characterize the simulated dynamics.

For timed or (generalized) stochastic PNs, while in the *Step* and *Run Slow* simulation modes, the *Scope* facility opens a new MATLAB window that displays (dynamically) the evolution of a selected performance index versus time.

For untimed PN models, the *behavioral properties* (e.g. boundedness, liveness, reversibility etc.) may be studied based on the *coverability tree* of the net. The *structural properties* (Murata, 1989) are approached as integer programming problems; the minimal-support P- and T-invariants (Martinez and Silva, 1982), (David and Alla, 1992) are displayed, on request, in separate windows.

A facility for the synthesis of timed or (generalized) stochastic PN models is *Design*, which allows exploring the dependence of a *Design Index* on one or two *Design Parameters* that vary within intervals defined by the user. For each test-point belonging to this (these) interval(s) a simulation-experiment is performed in the *Run Fast* mode. The results of all these simulation-experiments yield a graphical plot (2-D or 3-D, respectively) defining the dependence of the selected *Design Index* on the *Design Parameter*(*s*); the extreme values of the *Design Index* are numerically displayed (see fig. 3 in the next section).

The ***PN Web-Lab*** is able to derive, directly from the topology and initial marking of a place-timed event graph, the max-plus state-space representation (Bacelli *et al.*, 1992). The following facilities are available for the max-plus analysis: (i) displaying the matrix-form of the equations; (ii) max-plus simulation; (iii) graphical plots of the simulation results.

4. VISUAL INFORMATION AND ANIMATION

To enlarge the addressability of the ***PN Web-Lab***, it includes a series of animation facilities aiming both to support the intuitive understanding and to guide the users in the exploitation of the software.

In the simulation of the PN models (*Step* and *Run Slow* modes), numerical computation is accompanied by animation whose role consists in providing the user with visual information (current token contents of the places, currently firing transition), complementary to the numerical data available at the end of a simulation experiment.

At the same time, the ***PN Web-Lab*** was meant to illustrate, by short movies, behaviors that are typical for discrete event systems, for example sequential/parallel sharing of resources, routing policies, services in queuing networks etc. The implementation combines, by means of the ActionScript Toolbox for Macromedia Flash (Macromedia, 2003), various techniques such as 2D and 3D graphics developed in Adobe Photoshop 7 (Adobe Systems Inc., 2003) and Maya 4.5 (Alias|Wavefront Inc., 2003), respectively. Each movie shows the physical motion of a real-life system synchronized with the token dynamics in the associated PN model.

The online help of our software provides some animated demos whose purpose is to present specific sequences of operations in handling the GUI and the interpretation of numerical results. By watching these demos, the user learns how to handle the key problems of DES within a PN framework: usage of adequate PN type in model construction, study of behavioral/structural properties, analysis of max-plus representation, simulation and interpretation of the results, parameterized design etc.

5. EXAMPLE

Consider a flexible manufacturing system (FMS) that consists of two different machines (a lathe (M1) and a drilling machine (M2)), a robot (R) and a buffer (D) with two slots between the two machines, as described in (Lewis *et al.*, 1995). Every input part must be processed by M1 first and then by M2 in order to get the final product. Both machines are automatically loaded and are unloaded by the robot. A variable number of pallets can be used to fix on the processed parts. The functioning of the FMS is illustrated by a short movie available in a collection of demo programs developed for ***PN Web-Lab***; a frame of this movie is reproduced in fig. 2. The processing times on M1 and M2 are uniformly distributed in the intervals [35, 45] (time units) and, respectively [80, 90] (time units), while the unloading operations take 20 time units. The design purpose is to find the optimal number of pallets and the optimal duration (considered constant but unknown) for releasing and recycling a pallet so as to ensure the best value for the mean production cycle time.

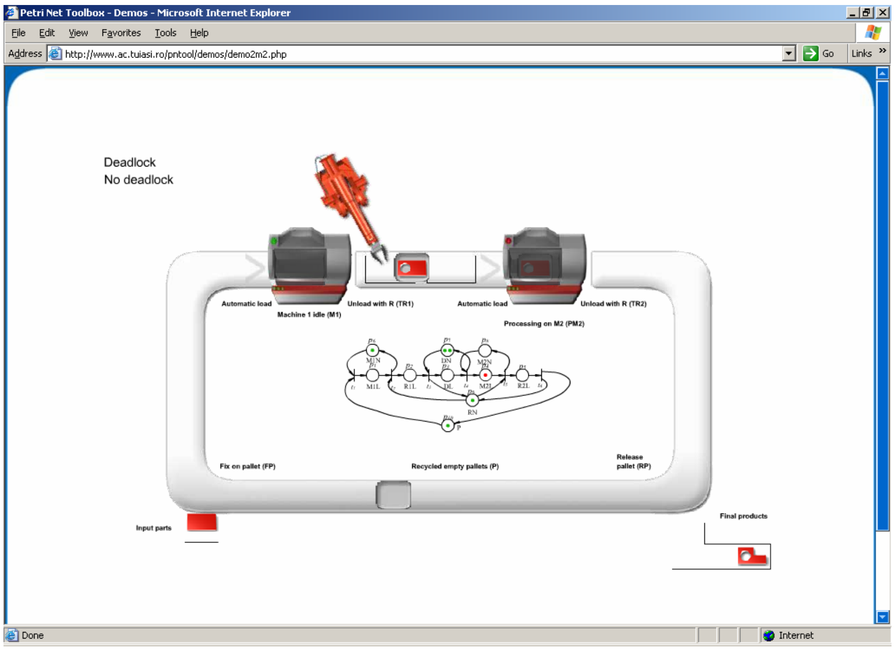
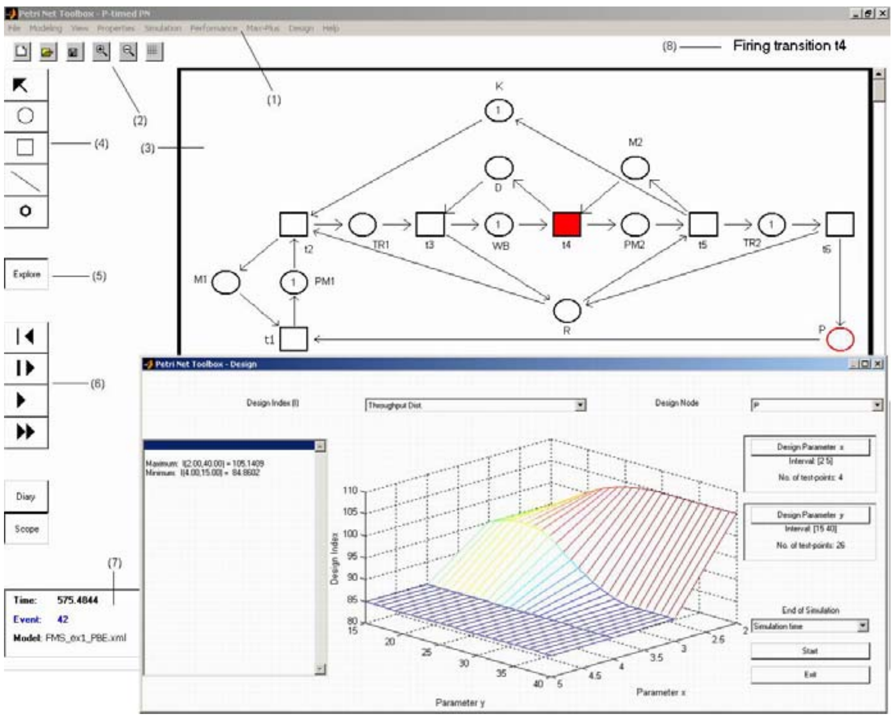


Fig. 2. Screen capture of a frame in a movie that illustrates the functioning of the FMS used in Example, concomitantly with the dynamics of the associated PN model.



Time: 575.4844
Event: 42
Model: FMS_ex1_PBE.xml

Design Index (I) Throughput Dist.
Design Node P

Maximum: I(2.00,40.00) = 105.1409
Minimum: I(4.00,15.00) = 84.8602

Design Parameter x
Interval: [2 5]
No. of test-points: 4

Design Parameter y
Interval: [15 40]
No. of test-points: 26

End of Simulation
Simulation time

Fig. 3. Screen capture presenting the main window of the PN Web-Lab (it displays the PN model of the FMS used in Example) and the Design window.

Since the robot R constitutes a sequentially shared resource in the system, a Kanban controller must be used so as to limit the number of parts in the critical subnet serviced by R and avoid the deadlock.

The PN model is presented in the upper part of the screen capture given in fig. 3. The design experiment performed in the ***PN Web-Lab*** considered two parameters: the number x of pallets in the system, varying from 2 to 5, and the time y for releasing a pallet, varying from 15 to 40 time units. The *Design* window is shown in the lower part of fig. 3 and presents the dependence on the two parameters for the mean production cycle time. The conclusion of the design experiment is that it suffices to use 3 pallets and increase the time for releasing a pallet up to 35 time units in order to minimize the mean production cycle time. This conclusion is not surprising, because the mean production cycle time cannot be reduced below the value of 85 time units which represents the mean processing time on the bottleneck machine.

6. CONCLUSIONS

Laboratory classes are equally important as the courses for the training of the CE students in DES. Unlike the rapid access via Internet to many sites providing lecture notes, the Web-based practical demonstrations require specialized technologies for sharing the resources available from a remote host (university department, research center etc.).

By developing the ***PN Web-Lab*** as a client-server application, we intended to facilitate the access to the analysis and design instruments provided by our ***PN Toolbox*** exploitable under MATLAB. These instruments allow a unified approach to different types of PNs, which ensures the premises for an efficient instruction at different levels of complexity. Guided by the online help and the demonstrative programs, the user needs a short time to learn how to handle the tools, and his major intellectual effort can focus on the construction and careful work with the PN models. In addition, the user can run several short movies that illustrate behaviors typical for DES, bringing a noticeable contribution to the intuitive understanding of the analysis and design problems.

REFERENCES

Adobe Systems Inc. (2003). *Home Page*, http://www.adobe.com.

Alias|Wavefront Inc. (2003). *Home Page*, http://www.aliaswavefront.com.

Bacelli, F., G. Cohen, G.J. Olsder and J.P. Quadrat (1992). *Synchronization and Linearity, An Algebra for Discrete Event Systems*, Wiley, New York.

David, R. and H. Alla (1992). *Du Grafcet aux réseaux de Petri* (2e édition), Hermes, Paris.

Macromedia Inc. (2003). *Home Page*, http://www.macromedia.com.

Martinez, J. and M. Silva (1982). A simple and fast algorithm to obtain all invariants of a generalized Petri net, In: C. Girault and W. Reisig (Eds), *Application and Theory of Petri Nets*, Informatik Fachberichte 52, Springer, pp. 301-310.

Lewis, F.L., H.H. Huang, O. Pastravanu and A. Gurel (1995). Control systems design for flexible manufacturing systems, In: A. Raouf, and M. Ben-Daya (Eds.), *Flexible Manufacturing Systems: Recent Developments*, Elsevier Science, pp. 259-290.

Mortensen, K.H. (2004). *Petri Nets Tools and Software*, http://www.daimi.au.dk/PetriNets/tools,

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications, *Proc. of the IEEE*, vol. 77, pp. 541-580.

Pastravanu**,** O., M. Matcovschi, C. Mahulea (2004). Petri Net Toolbox – teaching discrete event systems under Matlab, In Voicu,M. (Ed.), *Advances in Automatic Control*, pp.257-270, Kluwer Academic Publishers

Sun Microsystems, Inc. (2002). *Java Technology and Web Services* (http://java.sun.com/webservices/docs.html)

The MathWorks Inc. (2001). *MATLAB Web Server*. Natick, Massachusets.