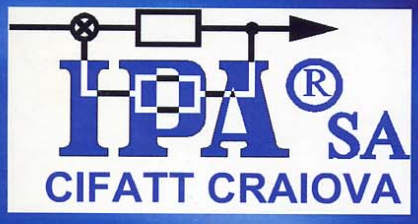

UNIVERSITY OF CRAIOVA
Automation, Computers & Electronics Faculty
INSTITUTE OF RESEARCH AND DESIGN IN AUTOMATION
ROMANIAN SOCIETY OF AUTOMATION & TECHNICAL INFORMATICS
IEEE - ROMANIAN SECTION

INTERNATIONAL SYMPOSIUM ON SYSTEM THEORY

SINTES 11

- XIth Edition -

1 VOL.

AUTOMATION MECHATRONIC SYSTEMS

Editura Universitaria Craiova

Craiova - ROMANIA
October 23-24, 2003

ROLE OF ANIMATION IN TEACHWARE FOR CONTROL ENGINEERING – A CASE STUDY

Cristian Mahulea, Mihaela-Hanako Matcovschi, Octavian Pastravanu
Department of Automatic Control and Industrial Informatics
Technical University "Gh. Asachi" of Iasi, Blvd. Mangeron 53A, 6600 Iasi, Romania
Phone/Fax: +40-232-230.751, E-Mail: {cmahulea, mhanako, opastrav}@ac.tuiasi.ro

Abstract: The paper identifies three main objectives supported by animation in developing teachware for Control Engineering: complementary information for simulation experiments, substitute for expensive hardware and demos for help on-line. Brief details are given with respect to the software technologies that allow approaching each of the three objectives. Concrete aspects pertaining to these technologies are illustrated by means of the numerous facilities available in the recently created Petri Net Toolbox for MATLAB.

Keywords: animation, online help, Petri nets, MATLAB software.

1. INTRODUCTION

In conventional Control Engineering education, the combination of theoretical knowledge and practical experience is ensured through lectures and exercises together with highly resource-demanding laboratory courses (Schmid, 2003). The new student-oriented type of teaching aims to develop critical thinking skills, the students being challenged to teach themselves by examples. This trend represents a difficult task, not only for the students, but also for those who educate. New media technology can help teachers to overcome the complexity of their new mission. Within this context, the techniques based on animation are frequently encountered because the recent digital equipment allows running such applications that support the intuitive understanding in the following directions: (i) arid numerical simulations based on abstract mathematical models, (ii) interactive demonstrations of the usage of expensive laboratory hardware or (iii) of specific software tools.

Hence, the organization of the first part of this paper follows the three directions (i)–(iii). Section 2 points out the importance of animation as a companion for simulation experiments. The possible employment of animation as a substitute for costly equipment in the practical training is discussed in Section 3. Some key points on the role of animation in the demonstration of the exploitation of scientific software by inexperienced users are delivered in Section 4. The second part of the paper is devoted to a case-study (Section 5) that illustrates the implementation of the three directions (i)–(iii) for the concrete situation of the software ***Petri Net Toolbox*** developed by the authors to run under MATLAB. Some concluding remarks are formulated in Section 6.

2. SIMULATION EXPERIMENTS ACCOMPANIED BY ANIMATION

The intuitive support provided by simulation in understanding the essence of different types of dynamics, can be enriched through adequate animation. Thus, the time-evolution of the variables used by the mathematical model (which, eventually, remain abstract entities) get concrete meaning by their association with the motion of graphical objects. Such objects should be chosen to express the physical reality and the animation should be carefully synchronized with the progress of the numerical simulation.

Nowadays, the scientific computation offer includes many environments ensuring highly accurate simulation experiments. However, the development of animation ingredients is far from a simple task when the basic tools are missing. Therefore, the initiative of creating animated companions for the pure numerical simulation should take into account the availability of such built-in facilities in the used software. In this respect, the exploitation of MATLAB (The MathWorks Inc. 2003), LabView (National Instruments Corporation 2003), Dymola (Dinasim AB 2003) are typical examples because, besides numerous computational procedures, a set of basic functions allows the user to construct demonstrative animations.

For instance, in MATLAB the kernel of the software allows the simultaneous update of the numerical and the imaging contexts, such that various types of animation applications can be programmed whose degrees of complexity are depending on the purposes of the instructional tasks. Consequently, a wide range of applications can be envisaged as training aids for studying phenomena in different fields of engineering, starting from the simple illustration of the basic laws in physics and ending with the operation of sophisticatedly automated systems.

In the direction discussed by this section, we can say that LabVIEW had a revolutionary role due to the flexibility

of the programming language, disposing of the complexity of traditional development tools. Rather than focusing on the numerical interpretation of the data provided by simulation, students can concentrate on the global approach of the simulation experiments whose results are delivered in a meaningful form by the easy to handle visualizing instruments.

Another way to address simulation is offered by Dymola which ensures the automatic manipulation of formulas describing dynamics from many engineering domains. The numerous ready to use model libraries, which can be exploited by graphical model composition, combined with 3-D animation facilities, permit much faster simulation than the traditional usage of equations and block diagrams.

3. ANIMATION AS A SUBSTITUTE FOR EXPENSIVE HARDWARE

For laboratory classes, the experiments performed on costly equipment can be replaced by relevant animation illustrating the main features of the real life operation. Although such a solution cannot involve students in the physical handling of the studied systems, the animated framework brings noticeable information that, otherwise, could be revealed only by mental experiments. An increasing class of beneficiaries comprises persons enrolled in distance learning programs, by bringing them closer to the practical aspects of the laboratory training.

The complexity of the animation depends on the envisaged purpose of training and contributes to the correct intuition of phenomena accordingly. This type of training can be exploited even in an interactive manner, aiming to develop certain skills in using some equipment, not just to simply illustrate its functioning. Basically, the implementation should result in a simulator of the hardware that can be reproduced in multiple copies and distributed to a large number of beneficiaries. In these cases, the simulation mechanism remains concealed during the regular training, the emphasis being placed on the usage of the simulated equipment.

A classical example of software that allows creating such a complex simulator and may be regarded as a standard for the training in Control Engineering is LabView. As a software that pushed the virtual instrumentation forward, LabVIEW offers a powerful graphical development environment specialized for signal acquisition, measurement analysis, and data presentation. The way students learn can be substantially improved since the laboratory classes are no longer focused on building the simulation experiments, but on running them and interpreting their results by means of dedicated tools for data visualization, management and analysis.

The behavior exhibited by large, complex multi-domain models can be explored in Dymola, whose new modeling methodology based on object orientation supports hierarchical structuring and enables the usage of drag and drop techniques for handling components from libraries. These components are defined as model classes with inheritance properties that facilitate the reuse of modeling knowledge. Connections between modules are conveniently described by defining connectors that model physical couplings. In the case of mechanical systems, predefined parameters can be set for their visual appearance.

4. ANIMATED DEMOS FOR HELP ON-LINE

Modern software packages, in addition to detailed manuals, ensure help on-line for the current exploitation. Less experienced users can find an extremely valuable guidance in the animated demos meant to illustrate the main capabilities of the software. Such demos are of a crucial importance for exhibiting the correct sequencing of commands accessible via graphical user interfaces (GUIs). In particular, for scientific software the demonstrations should be able to cover different types of problems, significant for the area addressed by the package, pointing up both the organization of the input data and the interpretation of the results. Generally speaking, the usage of animation by demonstrative programs is approached as the design of short movies, whose implementation requires specialized tools.

Frequent solutions to build movies are technically assisted by the various Macromedia products (Macromedia 2003), which concomitantly ensure the Internet access. For instance, Macromedia Flash (including ActionScript as a powerful scripting language) allows the development of web applications for detailing the usage of complex software. The purpose of such a guide is to assist the training step-by-step, starting from the primary formulation of an engineering problem, followed by the construction of a mathematical model and ending with the adequate manipulation of the software.

ActionScript provides elements, such as actions, operators, and objects that are put together in scripts. The movie is set up so that events trigger these scripts, for example, the user can stop/play the movie, switch between different types of simulation. In addition, ActionScript includes built-in objects and functions, allows the creation of objects and functions and permits the movies to run as long as it is not stopped by the user (every event is implemented as a function and every resource is modeled by an object). The other feature used in the elaboration of the movies is the opportunity to import videos and images from external files, extremely useful in this particular case because the functionality of real systems requires high quality captures.

Unlike the traditional help facilities, where a primary problem is described just by a simple text (with static figures), the applications implemented in Flash can give multiple visual details on the dynamical aspects. As regards the manipulation of the software, once the model is available, the utilization of the software is emulated by a movie (showing how to handle the windows, the dialogue boxes, the menu options etc.). Moreover, the physical interpretation of the results can also be

illustrated, giving the students a deeper insight into the meaning of the solved problem.

Generally speaking, the Macromedia technology permits to create: (1) frame-by-frame animation (in which is possible to make a separate image for each frame) and (2) tweened animation (creating frames that link together keyframes). Tweened animation is an effective way to create movement and changes over time while minimizing file size. Therefore, another advantage offered by the usage of Macromedia Flash in creating animated demos is represented the size of the resulting movie (smaller than the one of conventional movies) which makes such demos more attractive to download through the Internet and, consequently, very appropriate for distance learning.

5. ANIMATION FACILITIES IMPLEMENTED IN PN TOOLBOX 2.0 – A CASE STUDY

The ***Petri Net Toolbox*** (***PN Toolbox***) for MATLAB (Mahulea 2002; Matcovschi *et al.* 2003a and 2003b; Matcovschi and Pastravanu 2002) was designed and implemented at the Department of Automatic Control and Industrial Informatics of the Technical University of Iasi. Its development was motivated by the fact that little has been done towards developing adequate MATLAB instruments for dealing with PNs.

As an educational aid, this software is suitable for applications illustrating the theoretical concepts provided by courses on PNs with different levels of difficulty, e.g. (Pastravanu 1997; Pastravanu *et al.* 2002), allowing relevant experiments for studying the event-driven dynamics of physical systems encountered in many technical fields (such as flexible manufacturing systems (FMSs), computer systems, communication protocols, power plants, power electronics).

An easy to exploit GUI (The MathWorks 2001a) 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. Moreover, it permits the simulation, analysis and design of the PNs, by exploiting all the computational resources of the environment.

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

5.1. Animated Support for Petri Net Dynamics

The simulation mechanism is based on the rule for enabling and firing of transitions specific to the type of the current PN model. In the simulation modes *Step* and *Run Slow*, numerical computation is accompanied by animation whose role consists in feeding 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. The animation technique is based on the general philosophy of the *object-oriented graphics system*, called **Handle Graphics** (The MathWorks Inc. 2001b). The nodes and arcs of a model are uniquely identified as MATLAB objects whose properties define (i) the characteristics of the PN, (ii) the graphical representation of the objects in the special area reserved for model drawing and (iii) the simulation state. The animation effects are obtained by automatically calling the **set** function for the properties referring to the appropriate instance of an object.

MATLAB benefits of an easy-to-use interface to the Java programming language that facilitates the access to the huge collection of functions provided by its built-in class libraries. MATLAB's Java interface enables to create and manipulate Java components from the MATLAB command line and integrate them into the MATLAB environment. In this sense, we consider figure 1 as very relevant as presenting a screen capture of a Java applet, which shows the values of global performance indices obtained subsequent to the simulation of a model in the ***PN Toolbox***.

5.2. Movies to Illustrate the Functioning of Real Systems

Most of the real-life systems whose functioning is driven by events present a certain complexity which cannot be reproduced by small scale laboratory set-ups. That is why the ***PN Toolbox*** **(**Mahulea *et al.* 2003) was meant to illustrate by short movies typical behaviors such as sequential/parallel sharing of resources, routing policies, services in queueing networks. Their 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 and Maya 4.5, respectively. Each movie shows the physical motion of a real-life system synchronized with the token dynamics in the associated PN model.

The four movies that illustrate the usage of the PN Toolbox in the simulation, analysis and design of discrete-event systems are accessible on the web site we have created for the ***PN Toolbox*** (Mahulea *et al.* 2003). *Movie 1* refers to a computer system with two processors sharing two disks (in parallel) which is a version of the "Two Dinning Philosophers" well-known problem (Dijkstra 1968). *Movie 2* refers to a manufacturing system with two machines and a sequentially shared robot (Zhou and DiCesare 1993) illustrated by the frame captured in figure 2. The role of this figure is to detail a

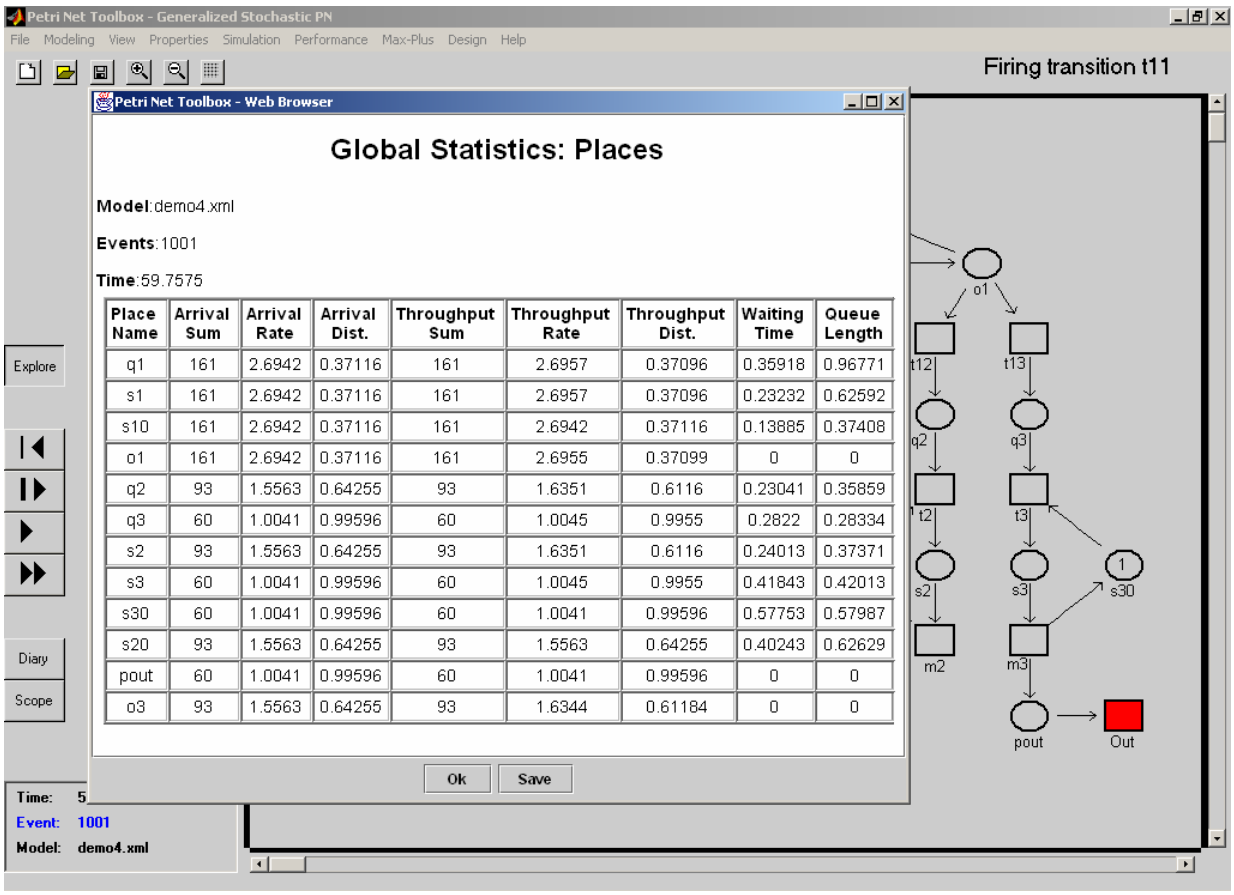

Fig. 1. *Screen capture illustrating the animation support for the simulation (image in the background – right side of the window) and the Java applet for the* ***PN Toolbox****.*

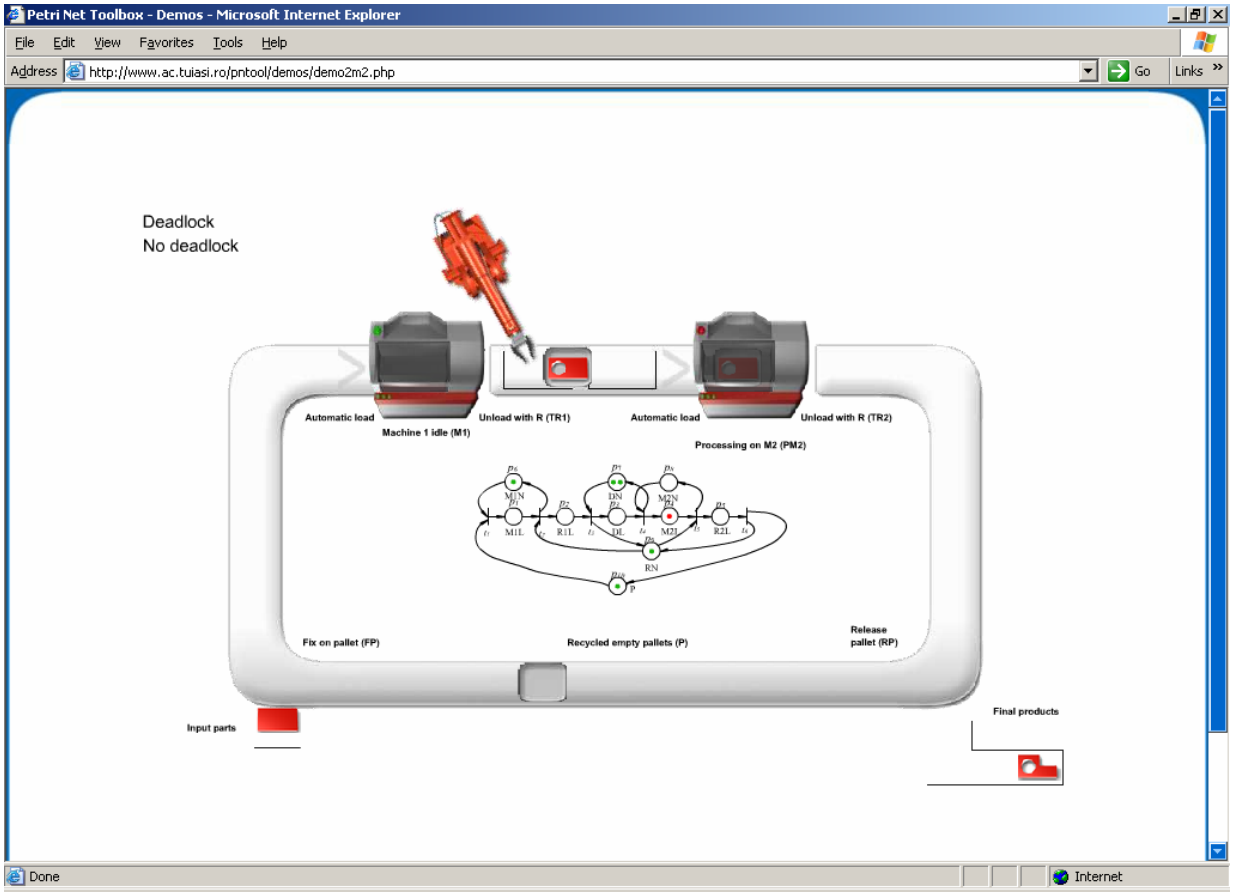

Fig. 2. *Screen capture of a frame in a movie illustrating the functioning of a manufacturing system concomitantly with the dynamics of the associated PN model.*

pose of the cell resources during the operation and the corresponding placement of the tokens in the associated Petri net model. *Movie 3* refers to a flow-shop system with three machines, adapted from (Bacelli *et al.* 1992). *Movie 4* refers to an open markovian queueing network (Cassandras 1993)

5.3. Animated Guide for Learning by Examples

The Macromedia technology is also used in the on-line help of the ***PN Toolbox*** **(**Mahulea *et al.*, 2003) for giving the beginners a full assistance by animated demos. Watching these demos, the user learns how to handle the key problems of discrete event systems within a PN framework: usage of adequate PN type (untimed, P/T-timed, stochastic or generalized stochastic) in model construction, study of behavioral/structural properties, analysis of max-plus representation, simulation and interpretation of the results, parameterized design, etc.

To illustrate such aspects, references to the problems addressed within the contexts of the four movies mentioned in subsection 5.2 are extremely profitable. *Movie 1* exemplifies the following topics: construction of an untimed Petri net model; analysis of deadlock (via the coverability tree); prevention of deadlock through lookahead feedback (Lewis *et al.* 1995); access to the following information about the Petri net model: incidence matrix, minimal-support P- and T-invariants, structural properties. The problems illustrated by *Movie 2* are: construction of a P-timed Petri net model; analysis of deadlock (via simulation); prevention of deadlock by limiting the number of pallets; analysis of time-dependent performance indices (figure 3); study of a performance index depending on two design parameters (figure 4). *Movie 3* demonstrates the following issues: simulation and animation in the Run Slow mode; record of the simulation results in a log file; computation of the cycle time; max-plus analysis of a place-timed event graph: max-plus state-space representation, setting of the values for the input vectors, max-plus based simulation and plots of the components for the input, state or output vectors. In *Movie 4* the subsequent themes are approached: construction of a generalized stochastic Petri net model; usage of the Scope and Diary facilities; analysis of time-dependent performance indices.

6. CONCLUSIONS

This work was devoted to the role that animation can have in complementing the conventional Control Engineering education by meaningful visual information resulting from the application of animation in various contexts of training. Three main directions were identified as beneficial for such applications and an extensive case-study reveals the coordinates of implementation and exploitation in the ***PN Toolbox*** for MATLAB. In the development of this software, primarily designed for education purposes, the authors proved a great interest in taking advantage of the deeper insight created by animation.

7. REFERENCES

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

Cassandras C.G. (1993). *Discrete Event Systems: Modeling and Performance Analysis*, Irwin.

Dinasim AB (2003), *Home Page*, http://www.dynasim.se/.

Djikstra E. W. (1968). Co-operating sequential processes, in Genyus, F. (ed.) *Programming Languages*, New York Academic, pp. 43-112.

Lewis F., Huang H.H., Pastravanu O. and Gurel A. (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.

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

Mahulea C. (2002). *Petri Net Toolbox – A MATLAB Library for Discrete Event Systems*, MS thesis, The Sheffield University, Department of Automatic Control & System Engineering, UK.

Mahulea C., Matcovschi M.H. and **Pastravanu O.** (2003). *Home Page of the Petri Net Toolbox*, http://www.ac.tuiasi.ro/pntool.

Matcovschi M.H., Mahulea C. and **Pastravanu O.** (2003a). Petri Net Toolbox for MATLAB, *11th IEEE Mediterranean Conference on Control and Automation MED'03*, Rhodes, Greece.

Matcovschi M.H., Mahulea C. and **Pastravanu O.** (2003b). Modeling, Simulation and Analysis of Petri Nets in MATLAB, *14th International Conference on Control Systems and Computer Science - CSCS14*, Bucharest, Romania.

Matcovschi M.H. and **Pastravanu O.** (2002). Developing Practicals for a Course on Queueing Systems Delivered to Control Engineering Students, *5th International Conference on Technical Informatics CONTI'2002*, Timişoara, Romania.

National Instruments Corporation (2003), *Home Page*, http://www.ni.com/labview.

Pastravanu O. (1997). *Discrete Event Systems. Qualitative techniques in a Petri Net Framework* (in Romanian), Ed. Matrix Rom.

Pastravanu O., Matcovschi M.H. and Mahulea C. (2002). *Applications of Petri Nets in Studying Discrete Event Systems* (in Romanian), Ed. Gh. Asachi.

Schmid C. (2003). *Virtual Action Group on Teachware for CACSD*, http://www.esr.ruhr-uni-bochum.de/cacsd/teachware.

The MathWorks Inc. (2001a). *Building GUIs with MATLAB*. Natick, Massachusetts.

The MathWorks Inc. (2001b). *Using MATLAB Graphics*. Natick, Massachusetts.

The MathWorks Inc. (2003). Developers of MATLAB and Simulink for Technical Computing. http://www.mathworks.com.

Zhou M.C. and DiCesare F. (1993). Petri Net Synthesis for Discrete Event Control of Manufacturing Systems, Kluwer, Boston.

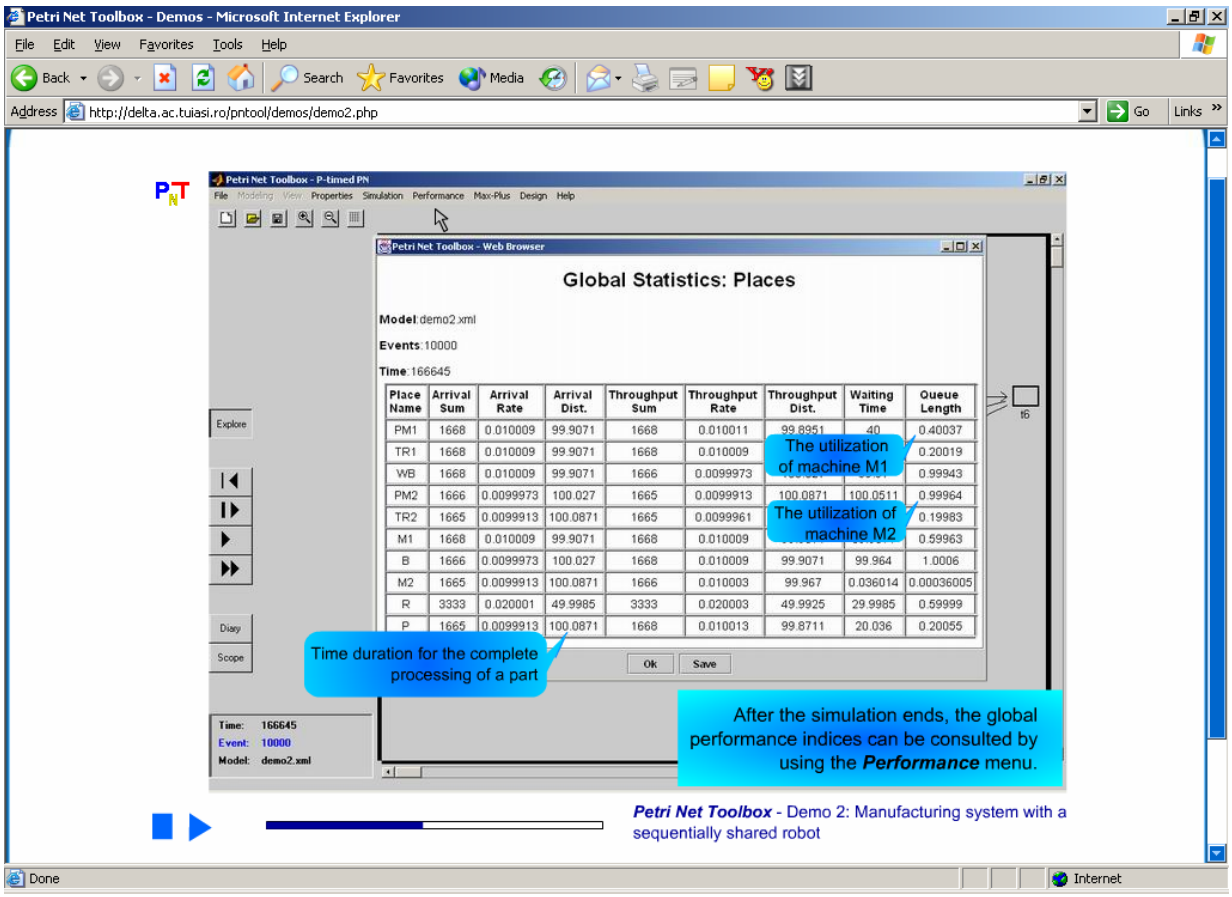

Fig. 3. *Screen capture of a frame in an animated demo presenting the table of the performance indices and advising on their interpretation.*

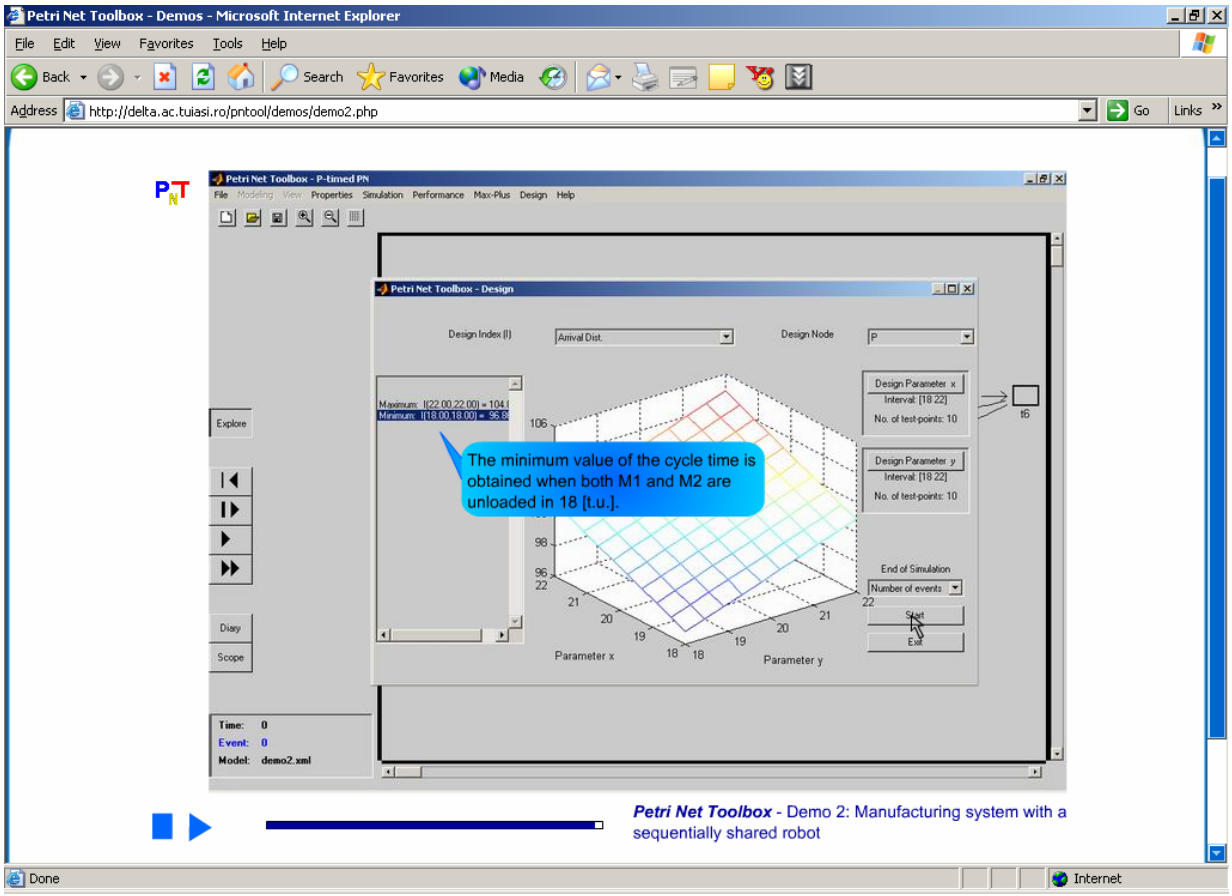

Fig. 4. *Screen capture of a frame in an animated demo introducing the usage of parameterized design.*