Petri Net Toolbox for MATLAB

Mihaela-Hanako Matcovschi, Cristian Mahulea and Octavian Pastravanu

***Abstract*—The software *Petri Net Toolbox*, dealing with Petri nets under MATLAB, is presented. It can handle five types of Petri nets (untimed, transition-timed, place-timed, stochastic and generalized stochastic) with finite or infinite capacity. The toolbox is equipped with a user-friendly graphical interface and allows three simulation modes accompanied or not by animation. Its functions cover the key topics of analysis such as coverability tree, structural properties (including invariants), time-dependent performance indices, max-plus state-space representations. A design procedure is also available, based on parameterized models. The integration with the MATLAB philosophy enlarges the utilization of this popular software towards the area of discrete-event systems. The applicability of this toolbox in several domains of engineering, illustrated by three examples, demonstrates its value as an educational aid.**

***Index Terms*—Education and training, modeling and simulation, Petri nets, discrete event systems, manufacturing.**

I. Introduction

The ***Petri Net Toolbox*** (***PN Toolbox***) version 2.0 is a software tool for simulation, analysis and design of discrete event systems, based on Petri net (PN) models. This software is embedded in the MATLAB environment and its usage requires version 6.0 or higher. The orientation of the ***PN Toolbox*** was meant to permit further development in the sense of hybrid systems, because MATLAB incorporates comprehensive libraries for studying continuous and discontinuous dynamics. Such a development, of extreme importance for control engineering, would be hardly approachable relying on the already existing PNs packages [1] as requiring sophisticated interfaces for the cooperation with instruments devoted to the exploration of continuous dynamics. Moreover, the integration of the ***PN Toolbox*** with the MATLAB philosophy has the incontestable merit of broadening the MATLAB's utilization domain towards the area of discrete-event systems, which is now covered only by the State-Flow package. However, it is worth mentioning the existence of two products [2] and [3] using the algebraic resources of MATLAB for PN investigation, without incorporating graphical editors, which, at this time, are not included in the MATLAB Connections Program [4].

The ***PN Toolbox*** was designed, implemented and tested at the Department of Automatic Control and Industrial Informatics of the Technical University „Gh. Asachi" of Iasi. It was intensively exploited during the last two academic years for laboratory works accompanying a course on discrete event systems. Although initially intended just for internal usage, the current version of the ***PN Toolbox*** can serve various instructional tasks, due to the wide range of topics covered by its facilities. It is suitable for applications illustrating the theoretical concepts provided by courses on PNs with different levels of difficulty. Furthermore, the ***PN Toolbox*** allows 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.

This paper is organized as follows. Section II displays the main features of the ***PN Toolbox*** and comments on its role as an educational aid. The Graphical User Interface is briefly described in Section III. Section IV presents the main algorithms and their implementation. The examples in Section V illustrate the exploitation of the ***PN Toolbox***. Some conclusions are finally delivered in Section VI.

II. Petri Net Toolbox at a First Glance

In the current version of the ***PN Toolbox***, 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 Toolbox*** has an easy to exploit *Graphical User Interface* (GUI) [5] whose purpose is twofold. 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. All the net nodes (places, transitions) and arcs are handled as MATLAB objects whose properties depend on the model type. 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 Toolbox*** allows the assignment of priorities and/or probabilities to conflicting transitions.

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

Department of Automatic Control and Industrial Informatics
Technical University "Gh. Asachi" of Iasi
Blvd. Mangeron 53A, Iasi, 6600, Romania
Phone / Fax: +40–232–230751, E–mail: mhanako@ac.tuiasi.ro

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).

The overall programming philosophy relies on the effective exploitation of data structures, functions, flow-control statements, input/output and object-oriented capabilities offered by MATLAB as a high-level language. To ensure the flexibility, the set of all default values manipulated by the ***PN Toolbox*** are accessible to the user by means of a configuration file.

The main goal envisaged by the designers of the ***PN Toolbox*** was to provide a collection of instruments for education and training at a graduate level, exploitable under MATLAB. Therefore, the focus was placed on developing fundamental skills in mastering PN models as a generous framework for dealing with discrete-event systems. Although a large number of tools are advertised for various types of PN problems [1], the unified treatment permitted by the ***PN Toolbox*** for untimed, deterministic/stochastic *P*- and *T*-timed PNs, stochastic and generalized stochastic PNs, ensures the premises for an efficient instruction. Thus, the user needs a short time to learn how to handle the ***PN Toolbox*** and his major intellectual effort is invested in the construction and careful analysis of the PN models. The interest shown for the convenient usage of the ***PN Toolbox*** is reflected by the numerous improvements brought to its previous versions [6], [7]. For attaining the proposed teaching goal, we preferred to orient our work towards enhancing the quality and reliability of the procedures devoted to standard topics rather than developing new algorithms.

III. The Graphical User Interface (GUI)

There are two modes in which the ***PN Toolbox*** 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. 2): *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 Toolbox***. These menus are enabled in accordance with the exploitation mode of the ***PN Toolbox***.

The *File* menu offers facilities for file-handling operations. This is the only menu available when the ***PN Toolbox*** 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 MATLAB 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 exploitations of the ***PN Toolbox*** and allows visualization of four Flash demo-movies initiating the user in the exploitation of the ***PN Toolbox***.

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 an axes MATLAB object and it 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* (see fig. 2).

IV. Algorithms and Implementation

The simulation is driven by an asynchronous clock corresponding to the occurrence of events (e.g. [8]). 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 Toolbox***, 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 (see fig. 6). They refer to:

- transitions: *Service Sum* (the total number of firings during the simulation), *Service Rate* (the mean frequency of firings), *Service Distance* (the mean time between two successive firings), *Utilization* (the fraction of time when server is busy);
- places: *Arrival Sum*, *Throughput Sum* (the total number of arrived/departed tokens), *Arrival Distance*, *Throughput Distance* (the mean time between two successive instants when tokens arrive in/depart from the place), *Waiting Time* (the mean time a token spends

in a place), *Queue Length* (the average number of tokens weighted by time).

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 (fig. 7). The coverability tree is built with or without the ω-convention. The ω-convention means the usage of a generic symbol (herein denoted by "ω") for referring to unbounded markings [9].

The *structural properties* are approached as integer programming problems [7]; the minimal-support P- and T-invariants [10] are displayed, on request, in separate windows (see fig. 6).

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).

The ***PN Toolbox*** is able to derive, directly from the topology and initial marking of a place-timed event graph, the max-plus state-space representation [11]:

$$\begin{aligned} \boldsymbol{x}(k) &= \bigoplus_{i=0}^{M} \left[\boldsymbol{A}_i \otimes \boldsymbol{x}(k-i) \oplus \boldsymbol{B}_i \otimes \boldsymbol{u}(k-i) \right], \\ \boldsymbol{y}(k) &= \bigoplus_{i=0}^{M} \left[\boldsymbol{C}_i \otimes \boldsymbol{x}(k-i) \oplus \boldsymbol{D}_i \otimes \boldsymbol{u}(k-i) \right], \end{aligned} \qquad k = \overline{1, N} .$$

The following facilities are available for the max-plus analysis: • displaying the matrix-form of the equations; • max-plus simulation; • graphical plots of the simulation results (see fig. 5).

V. Exploiting the Toolbox – Illustrative Examples

A. PN with Stochastic Timing (*FMS Design*)

Simulation experiments addressed in the ***PN Toolbox*** for MATLAB allow the user to choose the most efficient solution for managing a discrete event system based on the computation of the mean production cycle time, as shown below. The FMS presented in fig. 1 was selected as an illustrative example and 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 (adaptation from [12]). 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 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.

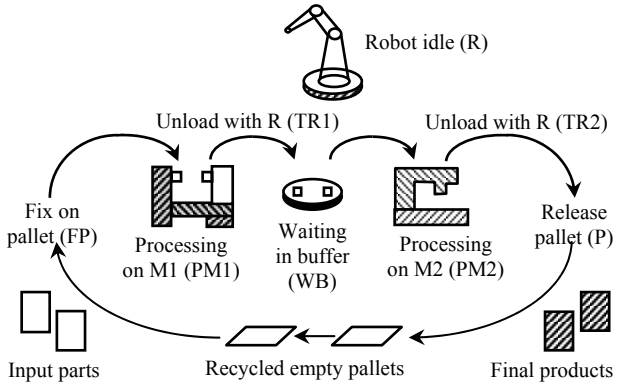

Fig. 1. Structure of the FMS used in Example A

Since the robot R constitutes a sequentially shared resource in the system, a Kanban controller [13] must be used so as to limit the number of parts in the critical subnet serviced by R and avoid the deadlock. The Petri net model is presented in fig. 2. The places are labeled according to the abbreviations used in fig. 1.

The design experiment performed in the ***PN Toolbox*** 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). Fig. 3 presents the dependence of the two parameters for the mean production cycle time (given by the *Throughput Distance* index for the place denoted by P in the net). The conclusion of this set of simulation experiments 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. As expected from analytic reasoning, 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.

B. Max-Plus Formalism (*Job-Shop Analysis*)

For the concrete analysis of event driven dynamics described by means of the max-plus formalism hand calculus is rather cumbersome even for problems of small complexity, fact that fully motivates our work in developing a max-plus simulator and integrating it into the ***PN Toolbox***. The example presented below (adapted from [11]) consists of a manufacturing system comprising three machines, denoted by M*i*, $i = 1, 2, 3$. It is supposed to produce three kinds of parts, namely P*i*, $i = 1, 2, 3$, according to a certain product mix. The routes to be followed by each part and each machine are depicted in fig. 4 and the corresponding processing times are given in Table 1. Parts are carried on a limited number of pallets; only one pallet is available for each part type. The sequencing of part types on the machines is known and it is (P2, P3) on M1, (P1, P2, P3) on M2 and (P1, P2) on M3. The final product mix can be obtained by means of a given input of parts.

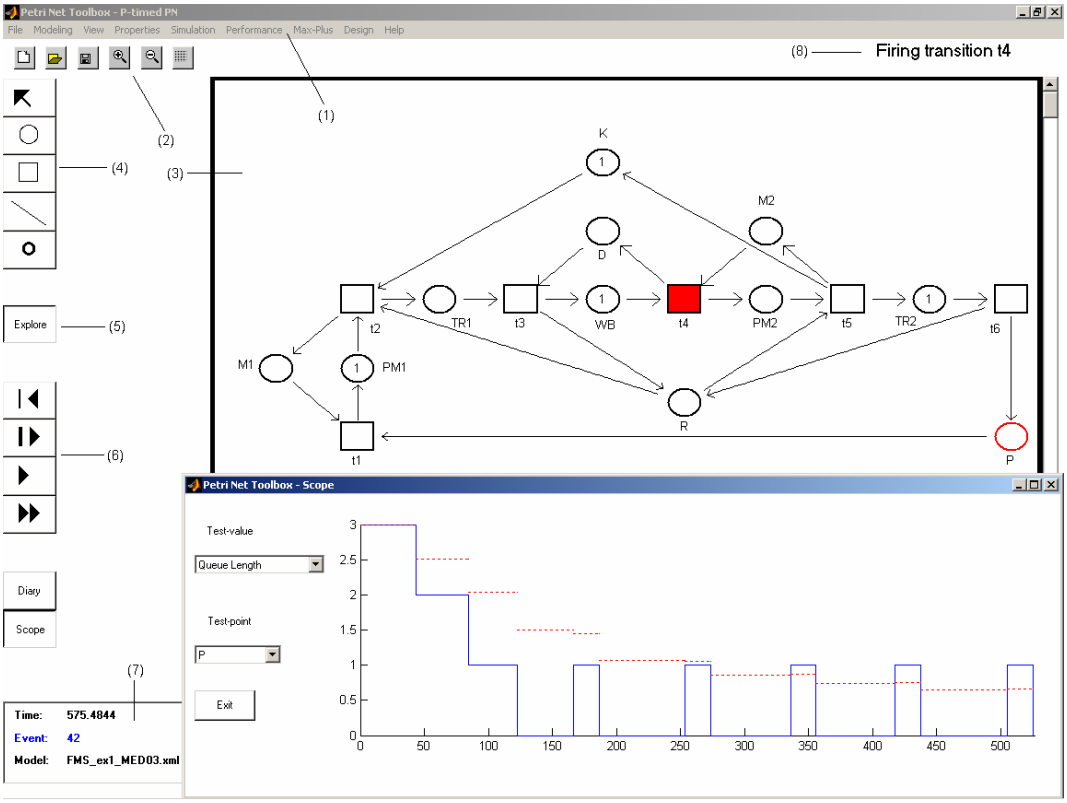

Fig. 2. Screen-capture of the main window of the ***PN Toolbox*** (containing the model of the FMS used in Example A)

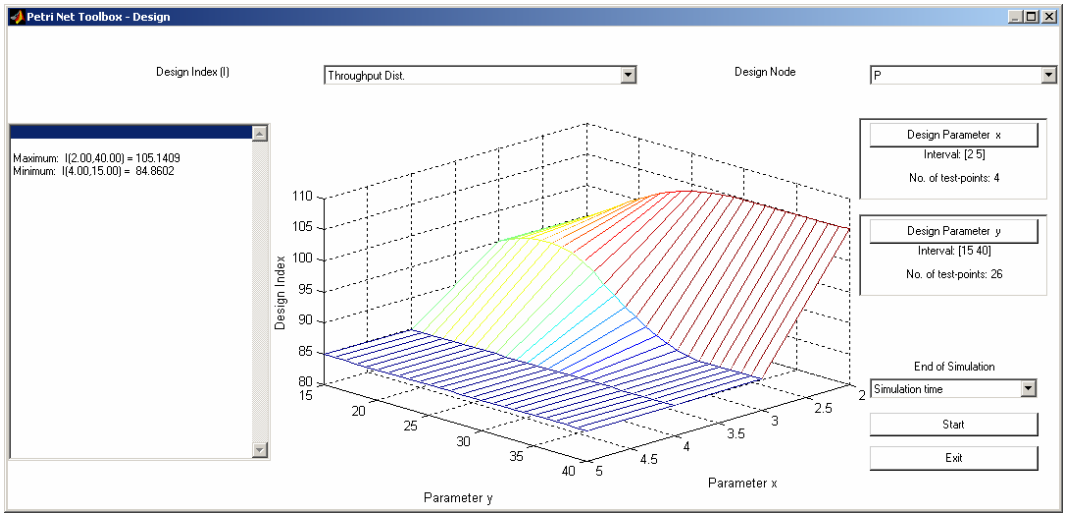

Fig. 3. Screen capture of the window corresponding to the *Design* facility

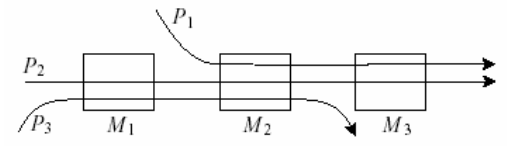

Fig. 4. Routing of parts along the machines in Example B

TABLE 1. Processing times of parts on machines in Example 2

	P1	P2	P3
M1	-	1	5
M2	3	2	3
M3	4	3	-

The place-timed Petri net model is presented in the main window of the ***PN Toolbox*** captured in fig. 5. The implicit form of the max-plus state-space representation is given by

$$\boldsymbol{x}(k) = \boldsymbol{A}_0 \otimes \boldsymbol{x}(k) \oplus \boldsymbol{A}_1 \otimes \boldsymbol{x}(k-1) \oplus \boldsymbol{B}_0 \otimes \boldsymbol{u}(k),$$
$$\boldsymbol{y}(k) = \boldsymbol{C}_0 \otimes \boldsymbol{x}(k), \qquad k = 1, 2, \ldots$$

The numerical values of the matrices involved in the previous equations, automatically derived by the ***PN Toolbox***, are presented by the upper left window in fig. 5. The max-plus simulator uses the explicit form:

$$\boldsymbol{x}(k) = \boldsymbol{A} \otimes \boldsymbol{x}(k-1) \oplus \boldsymbol{B} \otimes \boldsymbol{u}(k-1),$$

where $\boldsymbol{A} = \boldsymbol{A}_0^* \otimes \boldsymbol{A}_1$ and $\boldsymbol{B} = \boldsymbol{A}_0^* \otimes \boldsymbol{B}_0$. Matrix $\boldsymbol{A}_0^*$ is associated to $\boldsymbol{A}_0$ by means of the Kleene-star operator [11]. The lower window in fig. 5 presents the graphical plot of firing time vs. firing count for transitions u1, x1, x2 and y1, associated to the occurrences of events driving the progress in processing a part of type P1, obtained through simulation.

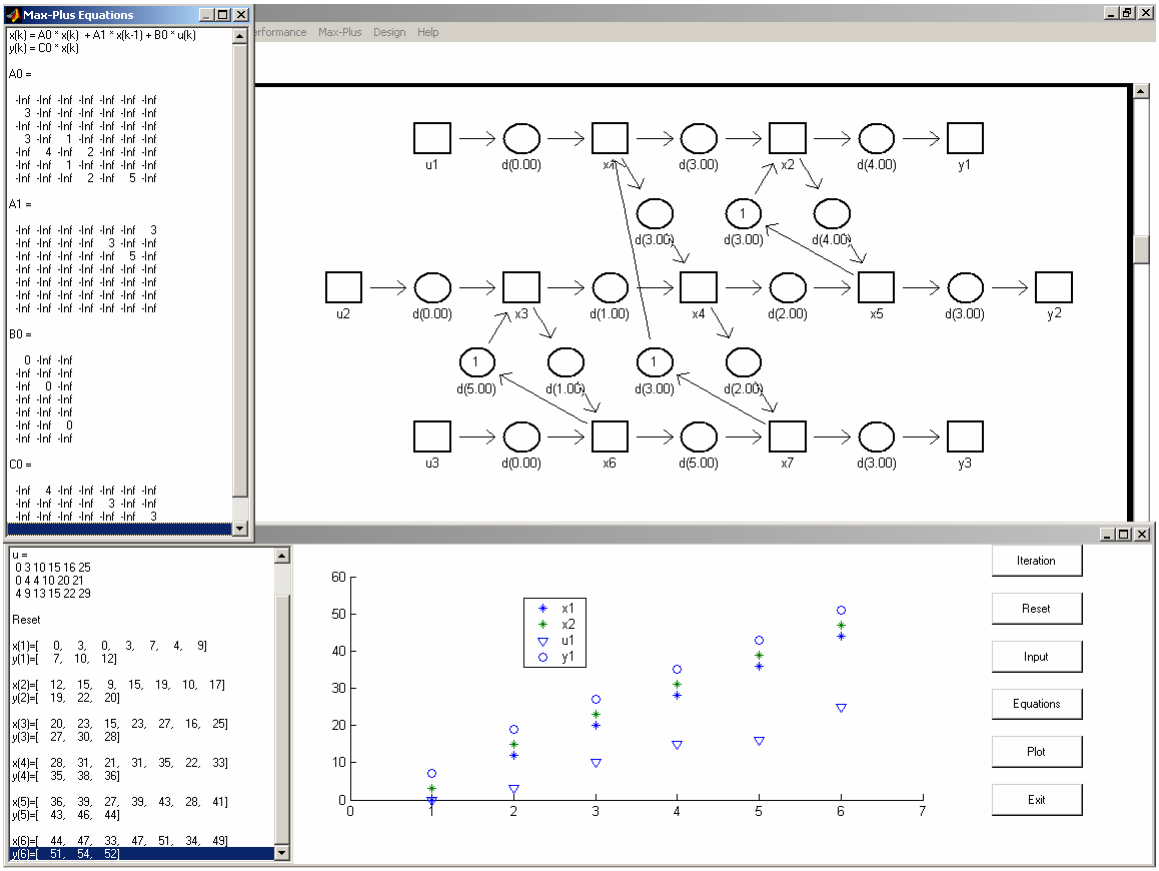

Fig. 5. Screen capture of the windows opened by the ***PN Toolbox*** for the max-plus state-space representation in Example B

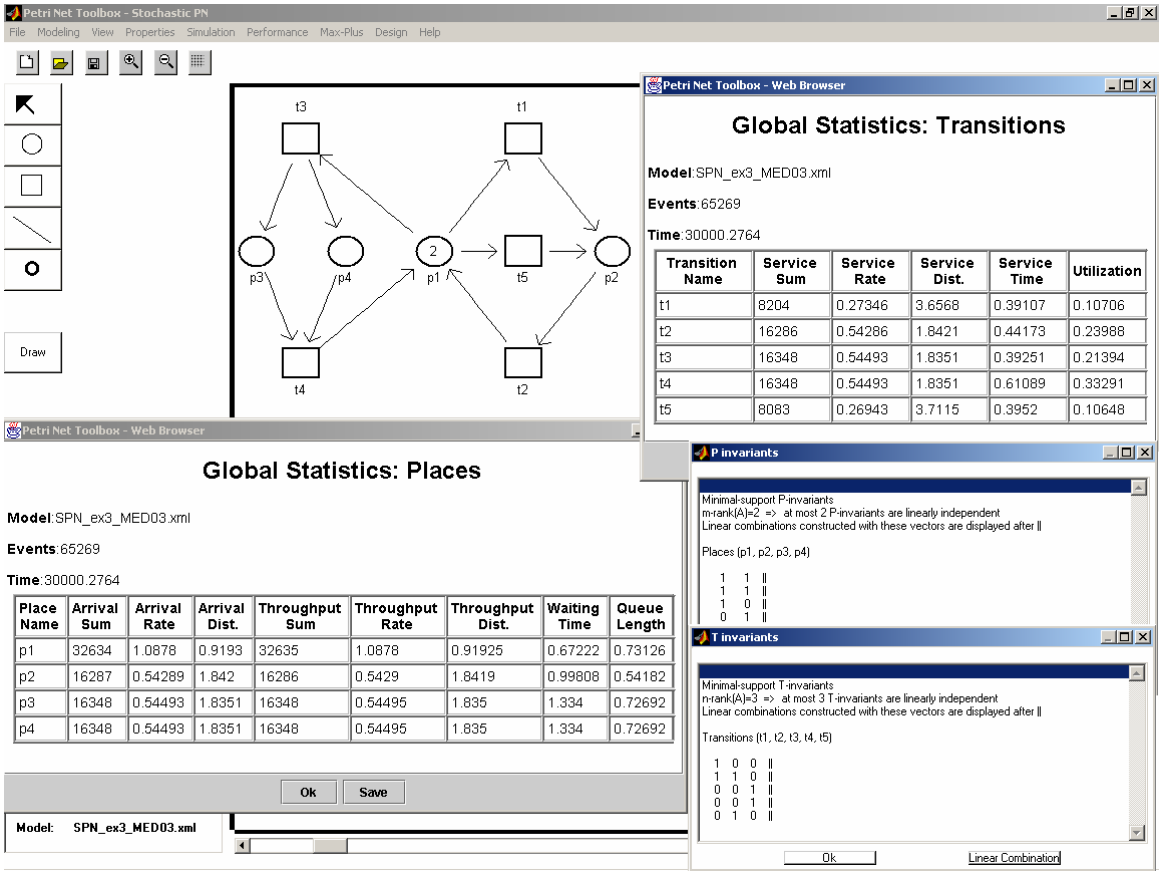

Fig. 6. Screen capture of the windows opened by the ***PN Toolbox*** for the performance analysis of the net used in Example C

C. *Stochastic PN*

The third example, that illustrates the usage of the ***PN Toolbox*** for the analysis of stochastic PNs (SPNs), refers to the model in fig. 6 (studied in [9]). Transition t2 fires at a marking-dependent rate given by $m_2\lambda_2$, where m_2 is the number of tokens in p2. Transitions t1, t3, t4 and t5 have (marking-independent) firing rates $\lambda_1, \lambda_3, \lambda_4$ and λ_5, respectively, with $\lambda_1 = \lambda_5 = 0.5$ [s^{-1}] and $\lambda_2 = \lambda_3 = \lambda_4 = 1$ [s^{-1}]. The coverability tree of this SPN, automatically computed by the ***PN Toolbox***, is presented in fig. 7 and serves for deriving its associated Markov chain. The analytical study of this model is based on the transition rate matrix given by

$$\boldsymbol{Q} = \begin{bmatrix} -2 & 1 & 1 & 0 & 0 & 0 \\ 1 & -3 & 0 & 1 & 1 & 0 \\ 1 & 0 & -3 & 0 & 1 & 1 \\ 0 & 2 & 0 & -2 & 0 & 0 \\ 0 & 1 & 1 & 0 & -2 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}.$$

Let $\Pi = [\pi_0 \ \pi_1 \ \pi_2 \ \pi_3 \ \pi_4 \ \pi_5]$ stand for the steady-state probability distribution vector, whose element π_i denotes the probability of being in state Mi, $i = 0,\ldots,5$. The following numerical values are obtained: $\pi_3 = 1/11$ and $\pi_0 = \pi_1 = \pi_2 = \pi_4 = \pi_5 = 2/11$.Various performance indices of the SPN model can be computed based on the steady state-distribution Π [9], [14]. For example, the mean number of tokens (*Queue Length*) in p1 is given by

$$M[m_1] = 2\pi_0 + \pi_1 + \pi_2 = 8/11 = 0.7273,$$

whereas the mean firing rate (*Service Rate*) and utilization (*Utilization*) for t2 are

$$M[f_2] = \lambda_2\pi_1 + 2\lambda_2\pi_3 + \lambda_2\pi_4 = 6/11 = 0.5455 \text{ and}$$

$$M[u_2] = \frac{\lambda_2}{-q_{11}}\pi_1 + \frac{2\lambda_2}{-q_{33}}\pi_3 + \frac{\lambda_2}{-q_{44}}\pi_4 = 6/11 = 0.5455,$$

respectively, where $q_{11} = -(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_5)$, $q_{33} = -2\lambda_2$ and $q_{44} = -(\lambda_2 + \lambda_4)$ are diagonal elements of matrix $\boldsymbol{Q}$.

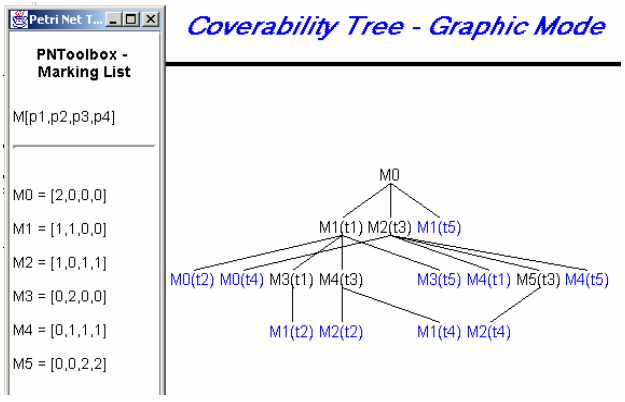

Fig. 7. Screen capture of the coverability tree automatically constructed by the ***PN Toolbox*** for the SPN model used in Example C

The analytical results may be compared to the corresponding ones obtained through the simulation of the SPN model in the ***PN Toolbox*** (fig. 6), validating the usability of our software for the analysis of (generalized) stochastic PNs.

VI. Conclusions

The development of the ***PN Toolbox*** has been motivated by the fact that MATLAB is a very popular software in various areas of engineering and especially in automatic control. Although the Petri net theory has been frequently used in technical scenarios during the last decade, the MathWorks has not released yet a product covering this topic. The ***PN Toolbox*** is able to fill this gap and offers instruments that are easy to exploit by people accustomed to the MATLAB philosophy. Compared to the two MATLAB-based products mentioned in the introductory section, our toolbox provides a user-friendly graphical interface that ensures its independence of other software resources and makes it more attractive by concealing the direct manipulation of the mathematical functions. By the facilities created for simulation, analysis and design, the ***PN Toolbox*** is utilizable in many types of applications encompassing a wide range of event-driven dynamics, as illustrated by the three examples briefly presented in the text.

VII. References

[1] K.H. Mortensen, *Petri Nets Tools and Software*, http://www.daimi.au.dk/PetriNets/tools, 2003.

[2] M.V. Iordache and P.J. Antsaklis, "Software Tools for the Supervisory Control of Petri Nets Based on Place Invariants", Technical Report ISIS-2002-003, University of Notre Dame, IN, USA, http://www.nd.edu/~isis/techreports/isis-2002-003.pdf, 2002.

[3] M. Svádová and Z. Hanzálek, "Matlab Toolbox for Petri Nets", *22nd International Conference ICATPN 2001*, Newcastle, UK, pp. 32-36, 2001.

[4] The MathWorks Inc., *MATLAB Connections - Third Party Products and Services*, http://www.mathworks.com/products/connections, 2003.

[5] The MathWorks Inc., *Building GUIs with MATLAB*, Natick, Massachusetts, 2000.

[6] C. Mahulea, L. Bârsan and O. Pastravanu, "Matlab Tools for Petri-Net-Based Approaches to Flexible Manufacturing Systems", In: F.G. Filip, I. Dumitrache and S. Iliescu (Eds.), *9th IFAC Symposium on Large Scale Systems LSS 2001*, Bucharest, Romania, pp. 184-189, 2001.

[7] M.H. Matcovschi, C. Mahulea, and O. Pastravanu, "Exploring Structural Properties of Petri Nets in MATLAB", *7th International Symposium on Automatic Control and Computer Science SACCS 2001*, Iasi, Romania, CD Rom, 2001.

[8] C.G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*, Irwin, 1993.

[9] T. Murata, "Petri Nets: Properties, Analysis and Applications", *Proc. of the IEEE*, vol. 77, pp. 541-580, 1989.

[10] J. Martinez and M. Silva, "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, 1982.

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

[12] M.C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*, Kluwer, Boston, 1993.

[13] F. Lewis, H.H. Huang, O. Pastravanu, and A. Gurel, "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, 1995.

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