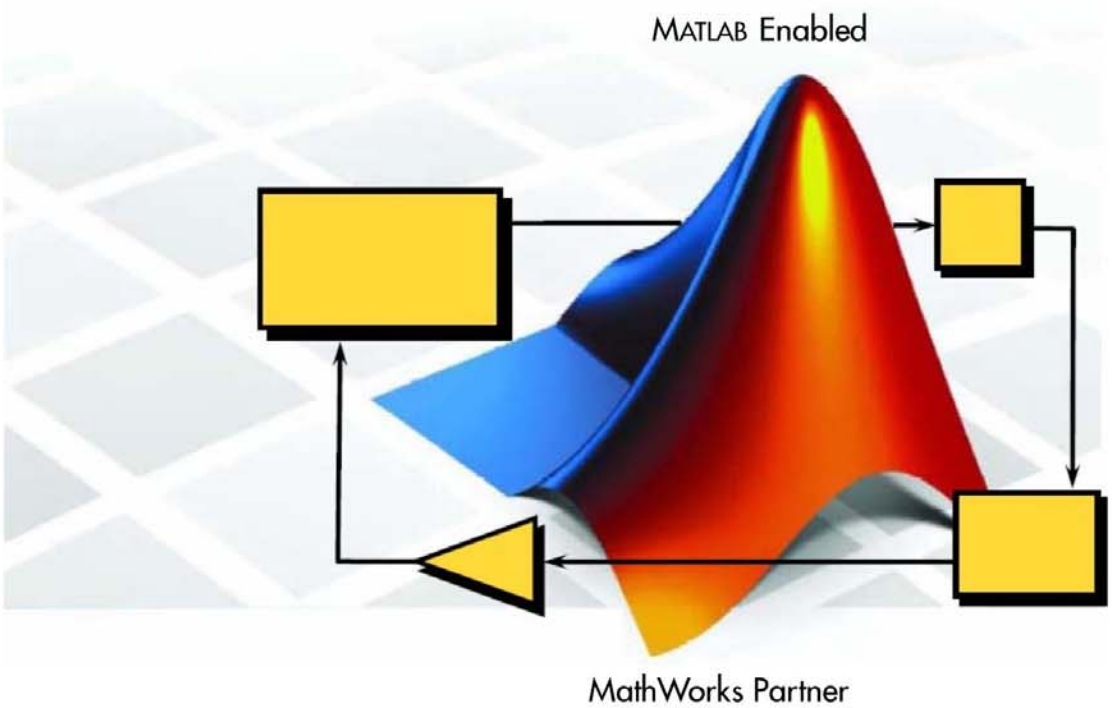
Petri Net Toolbox for MATLAB

Petri-net-based approaches to discrete event and hybrid systems in MATLAB-Simulink

Mihaela Matcovschi
Octavian Pastravanu



POLITEHNIUM
2008

MIHAELA MATCOVSCHI OCTAVIAN PĂSTRĂVANU

Petri Net Toolbox for MATLAB

Petri–net–based approaches to discrete event and hybrid systems in MATLAB–Simulink

Editura POLITEHNIUM

2008

Editura POLITEHNIUM
a Universităţii Tehnice „Gh. Asachi” din Iaşi
Bd. Dimitrie Mangeron nr. 67
RO-700050, Iaşi, România
Tel/Fax: +40 – 0232 – 231 343

Editura POLITEHNIUM (fostă „Gh. Asachi”) este recunoscută de
Consiliul Naţional al Cercetării Ştiinţifice
din Învăţământul Superior (CNCSIS)

Director editură:
Prof. univ. dr. ing. Mihail VOICU
Membru corespondent al Academiei Române

Referenţi ştiinţifici:
Prof. univ. dr. ing. Corneliu LAZĂR
Prof. univ. dr. ing. Doru PĂNESCU

Redactor:
Ing. Elena MATCU-ZBRANCA

Răspunderea pentru tot ceea ce conţine această carte aparţine
în întregime autorului (autorilor) ei.

Descrierea CIP a Bibliotecii Naţionale a României
MATCOVSCHI, MIHAELA HANAKO
Petri Net Toolbox for MATLAB: Petri-net-based approaches to discrete event and hybrid systems in MATLAB-Simulink / Mihaela-Hanako Matcovschi, Octavian Păstrăvanu – Iaşi: Politehnium, 2008.
Bibliogr.
Index
ISBN: 978-973-621-149-2

I. Păstrăvanu, Octavian

681.5:519.711+004.42 MATLAB

Universitatea Tehnică “Gh. Asachi” Iaşi
Facultatea de Automatică şi Calculatoare
Bd. D. Mangeron 53A, 700050 Iaşi
email: mhanako@ac.tuiasi.ro

Petri Net Toolbox at a First Glance

The modeling power of the Petri Nests (abbreviate as PNs) combined with the handy graphic representations stimulated the interest for their ample usage in various topics of discrete-event and hybrid systems. Consequently, during the last decade, many academic or research group s created software tools for studying PNs. Brief presentations of most of these tools are posted on the site Petri Nets World (http://www.informatik.uni-hamburg.de/TGI/PetriNets/) – maintained by the department of "Theoretical foundations of computer science" at the University of Hamburg, Germany.

For specialists in Control Engineering and adjacent areas dealing with discrete-event and hybrid systems, the most popular software is the MATLAB with its specialized toolboxes, produced by The MathWorks Inc. This software house produces two packages that allow the simulation of discrete-event and hybrid systems, namely the State-Flow and SimEvents. The former is based on finite state machine models, and the latter on queue and server models, which are compatible with the Simulink diagram philosophy. The MathWorks Inc. does not provide any tool for handling Petri Net models.

This book presents the package **Petri Net Toolbox** (abbreviated as **PN Toolbox**) **for MATLAB**, which bridges the PN formalism with the widely spread usage of the MATLAB. This package is equipped with simulation, analysis and design instruments; its simulation facilities are also integrated with the Simulink via the **Petri Net Simulink Block** (abbreviated as **PNSB**). The whole software was developed at the Department of Automatic Control and Applied Informatics of the Technical University "Gh. Asachi" of Iasi, Romania, and it is promoted by the Connections Program of The MathWorks Inc., as a third party product http://www.mathworks.com/products/connections/product_main.html?prod_id=556&prod_name=Petri%20Net%20Toolbox.

The book refers to the version 2.2 of our software, which is compatible with the MATLAB Release 2007b. The volume is organized in three parts, namely: Part I – PN

Toolbox User's Guide, Part II – PNSB User's Guide, Part III – Demonstrative Examples. The background of a reader must include theoretical knowledge about Petri nets in investigating the dynamics of discrete event and hybrid systems and practical skills in using the MATLAB-Simulink environment.

The **PN Toolbox** and **PNSB** accept three types of PN models, namely: untimed, transition-timed and place-timed. The timed nets can be deterministic or stochastic, and the stochastic case allows using the appropriate probability distribution function with positive support. The capacity of a place can be finite or infinite. The user can set priorities / probabilities for conflicting transitions. The PN model stored in a **PNSB** should contain synchronized (triggered) transitions. The firing of a synchronized transition depends on the occurrence of external (triggering) events. It is fired whenever (i) it is enabled by the net marking and (ii) one of its associated triggering events, defined at the PN level, occurs. Both finite and infinite server semantics can be used for transition firings. The ***PN Toolbox*** can also operate with stochastic and generalized stochastic PNs.

The **PN Toolbox** has an easy to exploit ***Graphical User Interface*** (GUI), whose purpose is twofold. First (***Draw Mode***), 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 (***Explore Mode***), it permits the simulation, analysis and design of the PNs, by exploiting all the computational resources of the environment, via the global variables stored in the MATLAB's Workspace. When starting to build a new PN model, the user must define its type by appropriate checking in a dialogue box. The properties of the net-objects (places, transitions, arcs) depend on the selected type of model. 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). The **PN Toolbox** uses the *XML* format for saving a model; a short description of the file format together with the specific commands is given in Appendix A.1. To ensure flexibility, the set of all default values manipulated by the **PN Toolbox** are accessible to the user by means of a configuration file presented in Appendix B.

The **PNSB** is also equipped with a GUI that allows the user: • to draw the PN model (***PNSB Editor***), • to define the triggering events (***PNSB Event Explorer***), • to debug the Simulink model (***PNSB Debugger***). The functions of the ***PNSB Editor*** are similar to the editing facilities available in the ***Draw Mode*** of the **PN Toolbox**; this ensures the immediate adaptation of the user's skills, once he/she has already been acquainted with the exploitation of the **PN Toolbox**. The **PNSB** uses the *XML* format for saving a model; a short description of the file format together with the specific commands is given in Appendix A.2. To ensure

flexibility, the set of all default values manipulated by the ***PNSB*** are accessible to the user by means of a configuration file presented in Appendix B.

The demonstrative examples in Part III illustrate the appropriate usage of the **PN Toolbox** and **PNSB** facilities for approaching simulation, analysis and synthesis problems. The considered problems refer to discrete event and hybrid systems that exhibit typical event-driven dynamics: two dinning philosophers, manufacturing system, flow-shop system, open queuing network, heating furnace.

The history of the **PN Toolbox** and **PNSB** goes back to the late nineties when we intended to design and implement a MATLAB-based Petri Net software for internal usage. In 2001 we already had a nice GUI for defining different types of PNs and a collection of good routines for simulation and analysis. We decided this was the birth of the **PN Toolbox,** version 1.0. In 2003 we produced the version 2.0 with a totally changed GUI and including many supplementary facilities for analysis and design. In 2004 - 2005 we created the **PNSB** for the integration with the Simulink, our researches being funded by the CNCSIS grant A541 / 2004. Since 2004 our software has been promoted by the MathWorks Inc, and we have distributed more than one hundred copies in the p-compiled form, on requests of researchers, educators, master and doctoral students all over the globe. Some of these beneficiaries have already published researches citing the **PN Toolbox**.

The gradual transformation of that initial project limited to internal usage into the software package presented by this book was supported by the enthusiastic contributions of our former colleagues Cristian Mahulea (University of Zaragoza, Spain), Claudiu Lefter (eScription, Inc., Needham, MA, USA) and Constantin Popescu (Siemens VDO Automotive SRL, Iasi, Romania). They developed a wonderful work in designing, implementing and testing various modules of the **PN Toolbox** and **PNSB**. The most important results of our fruitful collaboration have been disseminated by numerous publications listed in the References of this volume. We are greatly indebted to all of them for the invaluable aid.

March 2008 The Authors

Contents

PART I

Petri Net Toolbox
– User's Guide –

Chapter 1

DESCRIBING THE GRAPHICAL USER INTERFACE OF PN TOOLBOX

1.1. Overview

The ***PN Toolbox*** is equipped with an easy to exploit *Graphical User Interface* – GUI – (see fig. 1.1), which allows drawing PNs and permits simulation, analysis and synthesis. This GUI becomes operational once the command

```
>> pntool
```

is entered at the command-line prompt in MATLAB's Command Window.

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.

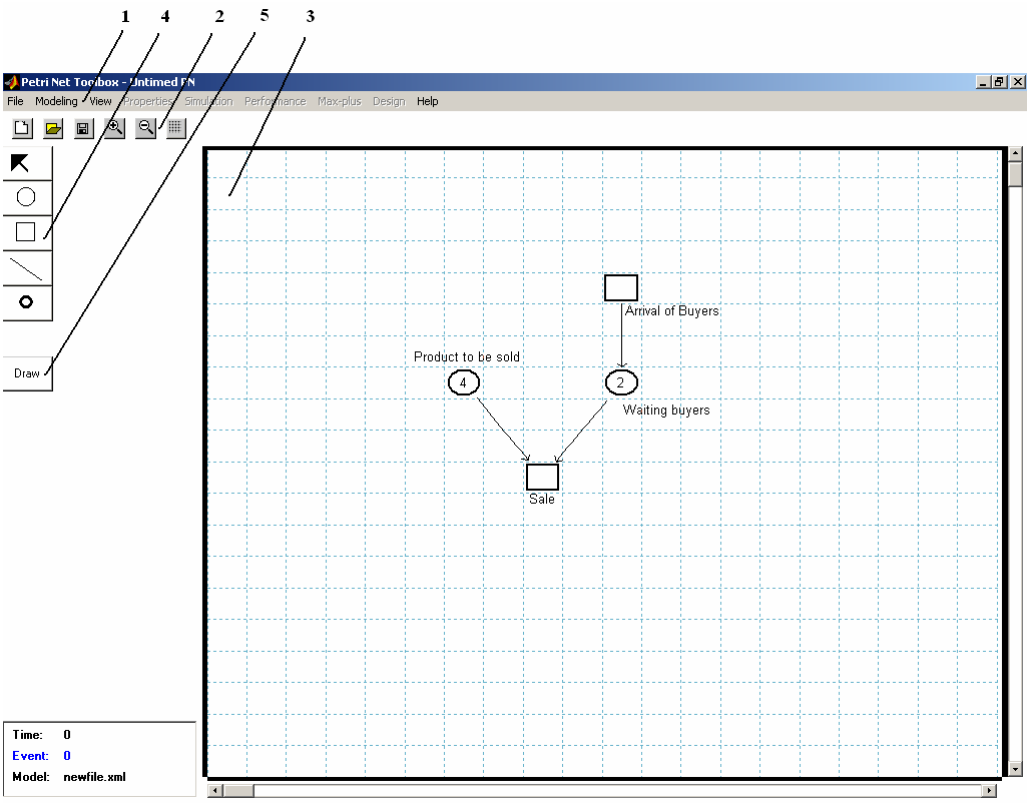
The GUI exhibits eight control panels (see fig. 1.1): ***Menu Bar*** (1), ***Quick Access Toolbar*** (2), ***Drawing Area*** (3), ***Drawing Panel*** (4), ***Draw/Explore Switch*** (5), ***Simulation Panel*** (6), ***Status Panel*** (7) and a ***Message Box*** (8). Further on, all these panels are described.

1.2. Menu Bar

The ***Menu Bar*** (placed horizontally, on top of the main window of the ***PN Toolbox***) 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***, namely the ***Draw Mode*** or the ***Explore Mode***.

The menus available under the ***Menu Bar*** are: ***File***, ***Modeling***, ***View***, ***Properties***, ***Simulation***, ***Performance***, ***Max-Plus***, ***Design*** and ***Help***.

(a)



(b)

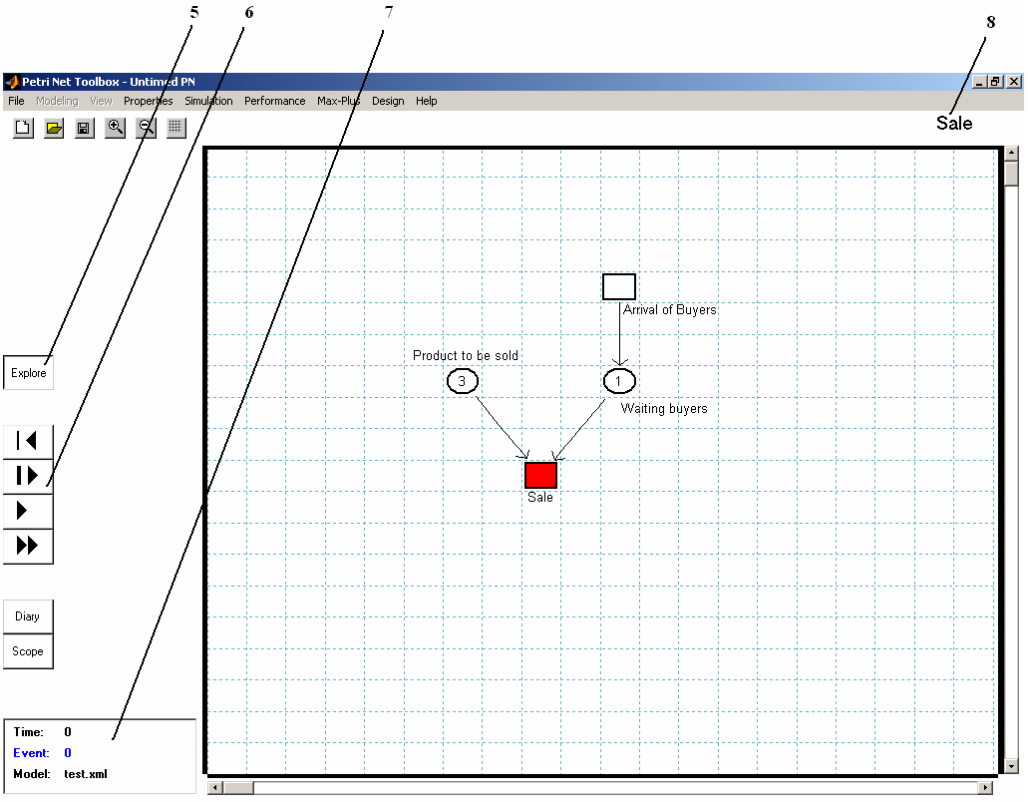


Fig. 1.1. *The main window of the* ***PN Toolbox****:*
(a) *Draw Mode,* (b) *Explore Mode.*

1.2.1. File Menu

The ***File*** menu (fig. 1.2) offers facilities for file-handling operations. This is the only menu available when the ***PN Toolbox*** GUI is started. This menu contains the commands:

- ***New Model***: opens a new board in the ***Drawing Area*** for the user to build a new PN model. This command first opens a dialogue box for selecting the type of the new model;
- ***Open Model***: loads a previously saved model;
- ***Close Model***: clears the ***Drawing Area*** asking for saving the current model if changes appeared in it;
- ***Save Model***: saves the PN model drawn in the ***Drawing Area***;
- ***Save Model As …***: saves the current model with a name given by the user;
- ***Save Model As Image***: saves the current model as a jpg file in a location and with a name given by the user;
- ***Print Model***: prints the current model on the printer selected by the user;
- ***Exit PNToolbox***: closes the current working session and clears all the global variables corresponding to the ***PN Toolbox***.

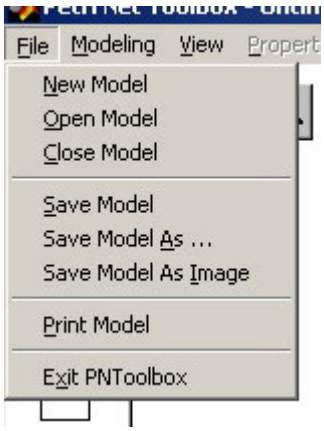


Fig. 1.2. *The **File** menu.*

Fig. 1.3. *The **Modeling** menu.*

1.2.2. Modeling Menu

The ***Modeling*** menu (fig. 1.3) provides tools for graphical editing (graph nodes, arcs, tokens, labels) a model in the ***Drawing Area***. The commands available in this menu are:

- ***Add Place***: adds a new place to the current PN model;
- ***Add Transition***: adds a new transition to the current PN model;
- ***Add Arc***: allows drawing an arc between two different nodes of the current PN model;
- ***Add Token***: adds a token in a specific place;
- ***Edit Objects***: opens the dialogue box associated with the net object that is selected in the ***Drawing Area*** by a click on the left button of the mouse. It allows the user to edit the properties of the selected object; these properties are in full accordance with the type of the PN model, so that part of them might be already disabled by the initial choice of the net type.

- ***Resolution for Conflicting Transitions***: allows assigning priorities or probabilities to conflicting transitions.

1.2.3. View Menu

The ***View*** menu (fig. 1.4) allows choosing specific conditions for visualization of the current model. The commands available in this menu are:

- ***Zoom In***: lets the user get a closer view to a smaller portion of the ***Drawing Area***;
- ***Zoom Out***: lets the user visualize a larger portion of the ***Drawing Area***;
- ***Show Grid:*** allows the user to add/remove grid lines to/from the ***Drawing Area***;
- ***Transitions Representation:*** allows the user to select the desired representation (as *Square*, *Horizontal Rectangle, Vertical Rectangle, Horizontal Bar* or *Vertical Bar*) for the transitions of a new model drawn in the ***Drawing Area***;
- ***Arc Weights:*** permits displaying or hiding the current values of arc weights.

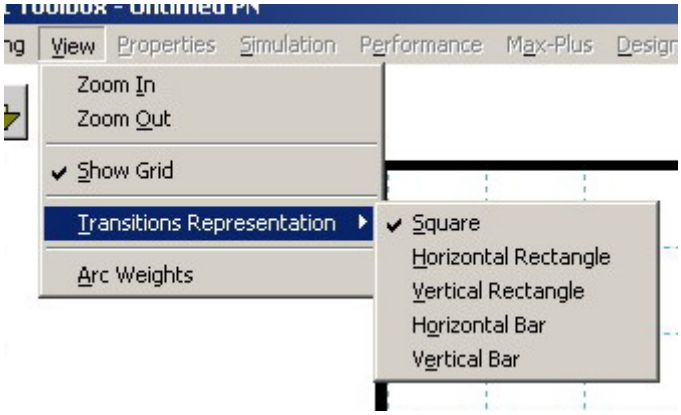


Fig. 1.4. *The **View** menu*.

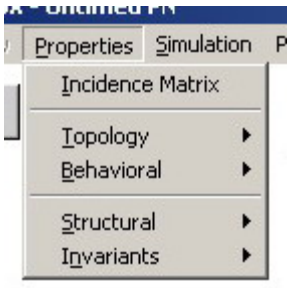


Fig. 1.5. *The **Properties** menu*.

1.2.4. Properties Menu

The ***Properties*** menu (fig. 1.5) provides computational tools for the analysis of the behavioral and structural properties of the current PN model. The following commands are available in this menu:

- ***Incidence Matrix:*** displays the incidence matrix of the current model in a separate window;
- ***Topology:*** performs topology analysis of ordinary Petri nets and displays the *Net Class*, *Traps* and *Siphons*;
- ***Behavioral*:** constructs the coverability tree of the current model, which can be visualized in two different formats: graphic and text. The *Liveness* of an ordinary Petri net can also be investigated.
- ***Structural***: performs the analysis of the structural properties of the current model (structural boundedness, conservativeness, repetitiveness and consistency);
- ***Invariants***: provides minimal-support P- and T-invariants.

1.2.5. Simulation Menu

Using the ***Simulation*** menu (fig. 1.6), the user can control the simulation progress and record the results (details in section 2.3). The following commands are available in this menu:

- ***Step***: simulates the PN model step by step, accompanied by animation;
- ***Run Slow***: simulates the PN model with a low speed, which is specified by the user (see the ***Preferences*** command), accompanied by animation. The simulation ends when the user clicks the right button of the mouse;
- ***Run Fast***: simulates the PN model with the maximum speed allowed by the environment. No animation is provided in this case. The simulation ends when the condition set by the user in ***Breakpoint*** is met;
- ***Breakpoint***: allows choosing the stop condition for simulation in the ***Run Fast*** mode;
- ***Reset***: brings the PN model to the initial state and resets all the results that were recorded in a previous simulation experiment;
- ***Log File***: stores the simulation history (recording all the states that the PN model passed through) in a file designated by the user. This command is operative only in the ***Step*** and ***Run Slow*** modes;
- ***View History***: opens the simulation log file;
- ***Preferences***: opens a dialogue box (fig. 1.7) allowing the user to set the conditions of the simulation (***Animation Delay*** is valid only for the ***Run Slow*** simulation mode; ***Token-In-Place Color*** is meaningful only for P-timed PNs; the value of the ***Seed*** is used to initialize the random number generator of MATLAB).

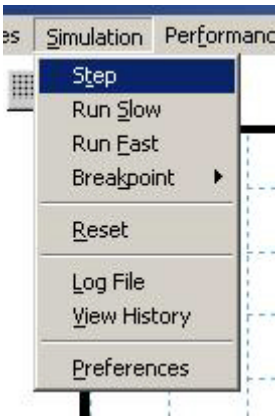


Fig. 1.6. *The **Simulation** menu.*

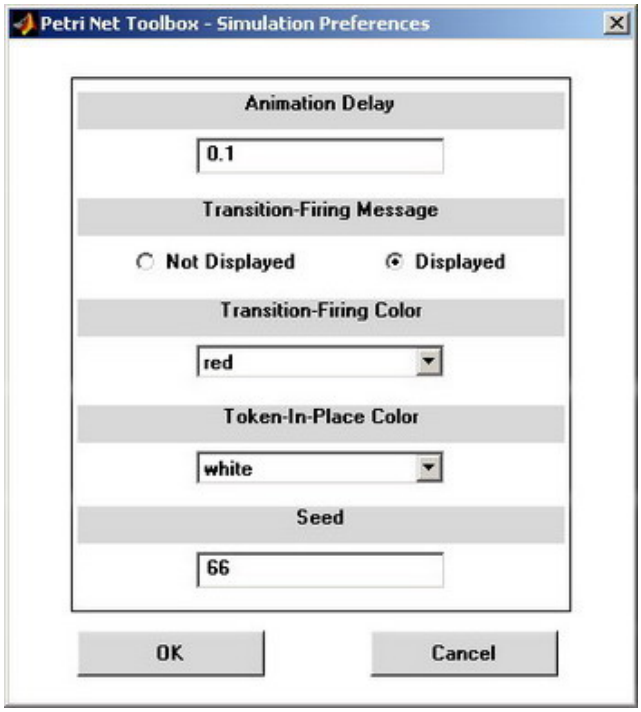


Fig. 1.7. *The **Preferences** dialogue box.*

1.2.6. Performance Menu

At the end of a simulation experiment, the ***Performance*** menu (fig. 1.8) allows the visualization of the ***global performance indices*** (see section 2.4 ***Analyzing Simulation Results***) that are written in an HTML file. These indices are separately recorded for transitions and for places. The commands available for this menu are:

- ***Place Indices***: displays the performance indices corresponding to the places;
- ***Transition Indices***: displays the performance indices corresponding to the transitions.
- ***Cycle Time***: provides the period of the steady state dynamics for a partially consistent deterministically timed PN, covered by *P-invariants*.

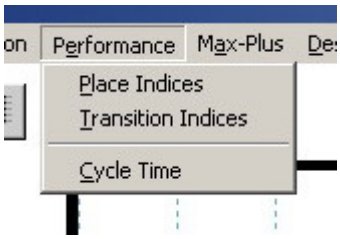


Fig. 1.8. *The **Performance** menu*.

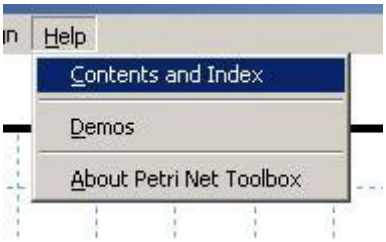


Fig. 1.9. *The **Help** menu*.

1.2.7. Max-Plus Menu

The ***Max-Plus*** menu (command) 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 (see section 2.5 **Max-Plus Models**).

1.2.8. Design Menu

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 (see section 2.6 ***Design***).

1.2.9. Help Menu

The ***Help*** menu (fig. 1.9) provides information about the exploitation of the ***PN Toolbox***. The commands available in this menu are as follows:

- ***Contents and Index***: opens the online help;
- ***Demos***: allows visualization of four **Flash** movies initiating the user in the exploitation of the ***PN Toolbox***;
- ***About Petri Net Toolbox***: contains information about the authors of the ***PN Toolbox***.

1.3. Quick Access Toolbar

The ***Quick Access Toolbar*** (fig. 1.10) is placed as a horizontal bar just below the ***Menu Bar*** and presents six image buttons. The actions of the first three buttons are identical to those controllable by the ***New*** (1), ***Open*** (2) and ***Save*** (3) commands from the ***File*** menu. The actions of the next three buttons are identical to those controllable by the ***Zoom In*** (4), ***Zoom Out*** (5) and ***Show Grid*** (6) commands from the ***View*** menu.

Fig. 1.10. *The **Quick Access Toolbar**.*

1.4. Drawing Area

The ***Drawing Area*** (located in the central and right side of the main window – see fig. 1.1) 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 with 50 rows and 50 columns. The bottom left-hand corner corresponds to the (0,0) cell, whereas the upper right-hand corner to the (49,49) cell. In one cell the user can draw a single node (place or transition), so that the total number of places and transitions in a model cannot exceed 2500.

Using the ***Zoom In*** / ***Zoom Out*** commands from the ***View*** menu, the number of the cells is decreased/increased with 10 cells on each axis. The ***Show Grid*** command adds/removes the grid lines to/from the Drawing Area.

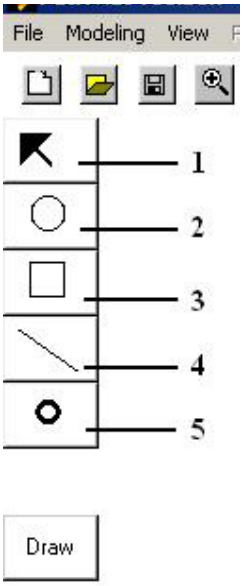


Fig. 1.11. *The **Drawing Panel**.*

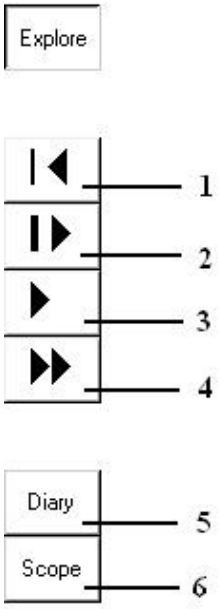


Fig. 1.12. *The **Simulation Panel**.*

1.5. Drawing Panel

The ***Drawing Panel*** (fig. 1.11) is placed vertically, in the left side of the main window, below the ***Quick Access Toolbar***, and presents five image buttons with actions identical to the commands ***Edit Objects*** (1), ***Add Place*** (2), ***Add Transition*** (3), ***Add Arc*** (4) and ***Add Token*** (5) available in the ***Modeling*** menu.

1.6. Draw/Explore Switch

The ***Draw / Explore Switch*** (identified as (5) in fig. 1.1) allows switching between the ***Draw Mode*** (in which the user can draw a new model or modify an existing one – button unpressed) and the ***Explore Mode*** (in which the user can access all functions available for simulation and analysis – button pressed).

When switching from the ***Explore Mode*** to the ***Draw Mode***, the marking of the PN model is automatically reset to the initial token configuration and the former values of all the performance indices are lost.

1.7. Simulation Panel

The ***Simulation Panel*** (fig. 1.12) is placed vertically, in the left side of the main window, just below the ***Drawing Panel***. It presents four image buttons with actions identical to the commands ***Reset*** (1), ***Step*** (2), ***Run Slow*** (3) and ***Run Fast*** (4) available in the ***Simulation*** menu. It also provides two visualization instruments for observing the progress of the simulation: ***Diary*** (5) and ***Scope*** (6) (see section 2.3 **Running a Simulation**).

1.8. Status Panel

The ***Status Panel*** (identified as (7) in fig. 1.1) is a message board (placed in the bottom left-hand corner of the main window), where the ***PN Toolbox*** displays the current simulation time and the total number of events. In addition, the ***Status Panel*** displays the file name corresponding to the current model.

1.9. Message Box

The ***Message Box*** (identified as (8) in fig. 1.1) is a MATLAB text object used by the ***PN Toolbox*** to display messages to the user. In the ***Draw Mode***, five types of messages may be displayed, corresponding to the actions enabled by the five buttons in the ***Drawing Panel***, respectively. In the ***Explore Mode***, messages are displayed only during the simulation of a model (***Step*** or ***Run Slow*** commands) if option ***Displayed*** is checked for ***Transition-Firing Message*** in the ***Preferences*** dialogue box (from the ***Simulation*** menu). When a transition fires, the ***Message Box*** displays a text associated with this transition; the text must be previously defined in the ***Edit Transition*** dialogue box (***Draw Mode*** - see section 2.1.3).

Chapter 2

EXPLOITING THE PN TOOLBOX

2.1. Building a Model

2.1.1. Overview

The ***PN Toolbox*** provides a set of commands for building a PN model, which are accessible from the ***Modeling*** menu or from the corresponding buttons in the ***Drawing Panel***. The model may be easily drawn, in a natural fashion, in the ***Drawing Area***. The user can also have a read / write access to the properties of the net nodes and arcs by opening the dialog box associated with the object selected in the ***Drawing Area***. In addition, the ***PN Toolbox*** allows the assignment of priorities and / or probabilities to conflicting transitions.

The ***Drawing Area*** displays a grid with dotted gray lines. The user may hide or show these lines by using the command ***View / Show Grid*** or by pressing the corresponding button from the ***Quick Access Toolbar***. The nodes of the PN may be placed only in the grid cells; each cell can contain a single net node. This way, each node is uniquely characterized by its coordinates in the ***Drawing Area***. A PN model created in the ***PN Toolbox*** is saved as an *XML* file. Besides the automatic generation of the XML file of a model, any text editor can be used to produce such a file if the user complies with the syntax given in Appendix A.1.

The first step in the creation of a new model is to define its type. By selecting the type of the PN, some of the work variables are automatically changed. This is because the simulation and analysis procedures differ from one type to another. The drawing board of a new model may be opened in the ***Drawing Area*** by selecting the ***File / New Model*** command or by pressing the ***New*** button from ***Quick Access Toolbar***. This command opens a dialog box (fig. 2.1) which permits the selection of the type of model to be built. The default type is ***Untimed PN***.

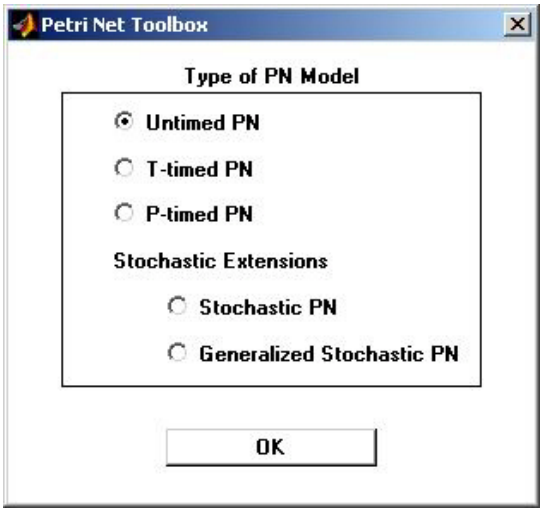


Fig. 2.1. *The dialogue box for selecting the type of a PN model.*

A previously created model may be loaded from the disk by means of the command ***Open*** available under the ***File*** menu or by pressing the ***Open*** button from the ***Quick Access Toolbar***. Changes in an existing model can be done directly in the GUI while in ***Draw Mode***, or by editing the corresponding **XML** file. If a model is already loaded in the window, the ***Drawing Area*** is cleared by selecting the command ***File / Close Model***.

2.1.2. Places

A place is graphically represented in the ***Drawing Area*** by a circle. To draw a place in the ***Drawing Area***, the user must press the ***Add Place*** button from the ***Drawing Panel*** or select the ***Add Place*** command from the ***Modeling*** menu. Then, the user must click only once into the desired grid cell of the ***Drawing Area***. A second click in the same grid cell has no effect. Once the circle corresponding to a place is drawn, a label is automatically attached to it. The default position of the label is below the circle.

After drawing a place, there are two ways to change its position in the ***Drawing Area*** while in ***Draw Mode*** and no button from the ***Drawing Panel*** is pressed. The first way is to left-click the desired place and then drag it in the new position. The second way is to right-click the place; as a result, a MATLAB uicontext menu (fig. 2.2) appears and the command ***Move Place*** becomes available. In both cases, the label is moved together with the place.

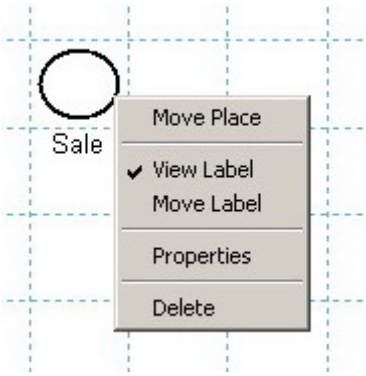


Fig. 2.2. *The uicontext menu of a place*

For a selected place, the uicontext menu also allows: (*i*) controlling the label's visibility on the screen (***View Label*** command), (*ii*) changing the position of the label (***Move Label*** command), (*iii*) deleting the place (***Delete***

command) and (*iv*) opening the ***Edit Place*** dialogue box (fig. 2.3) that lets the user modify the properties of the place as a MATLAB object (***Properties*** command).

Fig. 2.3. *The **Edit Place** dialogue box for modifying the properties of a place.*

The ***Edit Place*** dialogue box may be also opened by one of the procedures (EP1) or (EP2) described below, followed by a click on the desired place. (EP1) consists in selecting the ***Edit Objects*** command from the ***Modeling*** menu; (EP2) consists in pressing the ***Edit Objects*** button from the ***Drawing Panel***.

Each place is uniquely identified with an **id** that is automatically assigned by the ***PN Toolbox*** and cannot be changed by the user. This id appears in the title bar of the ***Edit Place*** dialogue box.

The option ***Label*** displays the string that is used as the label of the place. By default, this string coincides with the id of the place. The user can modify this string (without affecting the id) if necessary.

The option ***Color*** displays the color used for drawing the place. By default, this color is black, but the user can select another one from a list of eight predefined colors. Using different colors for drawing the places of a net might be helpful for complex topologies (e.g. for highlighting different phases requested by a multi-step design procedure).

The option ***Capacity*** displays the capacity of the place. By default, the capacity is **Inf** (the IEEE arithmetic representation for positive infinity). The user can set this field to a positive integer value.

The option ***Tokens*** displays the number of tokens in the place. By default, this number is 0. The user can set this field to a positive integer value. If the marking of a place is greater than 0, this marking is shown as a number inside the circle corresponding to that position. Void marking is not explicitly shown.

There are two more possibilities to add a token to a place (AT1) or (AT2) described below, followed by a click on the desired place. (AT1) consists in selecting the ***Add Token*** command from the ***Modeling*** menu; (AT2) consists in pressing the ***Add Token*** button from the ***Drawing Panel***.

In case of place-timed PN models (see section 2.3.4), the ***Time distribution*** option associated with a place allows the user to specify the probability distribution and the necessary parameter(s) which define the corresponding time-duration. This option is not available for untimed or transition-timed models.

The ***Delete object*** button placed at the bottom of the ***Edit Place*** dialogue box lets the user delete the place from the model.

2.1.3. Transitions

A transition is graphically represented in the ***Drawing Area*** by a square. To draw a transition in the ***Drawing Area***, the user must press the ***Add Transition*** button from the ***Drawing Panel*** or select the ***Add Transition*** command from the ***Modeling*** menu. Then, the user must click only once into the desired grid cell of the ***Drawing Area***. A second click in the same grid cell has no effect.

Once the square corresponding to a transition is drawn, a label is automatically attached to it. The default position of the label is below the square.

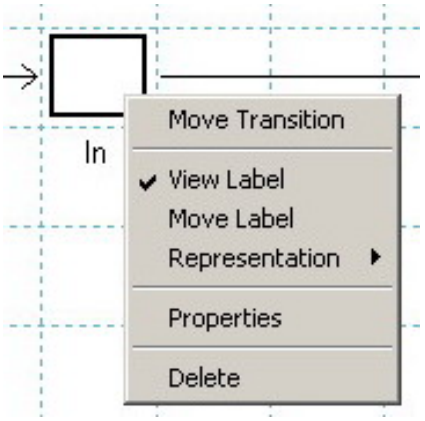


Fig. 2.4. *The uicontext menu of a transition.*

The same as for a place, after drawing a transition, there are two ways to change its position in the ***Drawing Area*** while in ***Draw Mode*** and no button from the ***Drawing Panel*** is pressed. The first way is to left-click the desired transition and then drag it in the new position. The second way is to right-click the transition; as a result, a MATLAB uicontext menu (fig. 2.4) appears and the command ***Move Transition*** becomes available. In both cases, the label is moved together with the transition.

For a selected transition, the uicontext menu also allows: (*i*) controlling the label's visibility on the screen (***View Label*** command), (*ii*) changing the position of the label (***Move Label*** command), (*iii*) deleting the transition (***Delete*** command) and (*iv*) opening the ***Edit Transition*** dialogue box (fig. 2.5) that lets the user modify the properties of the transition as a MATLAB object (***Properties*** command).

The ***Edit Transition*** dialogue box can be also opened by one of the procedures (ET1) or (ET2) described below, followed by a click on the desired transition. (ET1) consists in selecting the ***Edit Objects*** command from the ***Modeling*** menu; (ET2) consists in pressing the ***Edit Objects*** button from the ***Drawing Panel***.

Each transition is uniquely identified with an **id** that is automatically assigned by the ***PN Toolbox*** and cannot be changed by the user. This id appears in the title bar of the ***Edit Transition*** dialogue box.

The option ***Name*** displays the string that is used as the label of the transition. By default, this string coincides with the id of the transition. The user can modify this string (without affecting the id) if necessary.

The ***Message*** text-box contains the string that is displayed in the ***Message Box*** during the simulation of the model when firing that transition if option ***Displayed*** is checked for ***Transition-Firing Message*** in the ***Preferences*** dialogue box. By default, this string is "*Firing transition x*" where *x* is the id of that transition. The user can modify this string if necessary.

The option ***Color*** displays the color used for drawing the transition. By default, this color is black, but the user can select another one from a list of eight predefined colors. Using different colors for the transitions of a net might be helpful for complex topologies (e.g. for highlighting different phases requested by a multi-step design procedure).

In case of transition-timed PN models (see section 2.3.3), the ***Distribution*** option associated with a transition allows the user to specify the probability distribution and the necessary parameter(s) which define the corresponding time-duration. This option is not available for untimed or place-timed models.

For Stochastic PN models, only exponentially distributed firing rates can be assigned to the transitions in the net. In the case of Generalized Stochastic PN models, only constant and exponentially distributed firing rates can be assigned to the transitions in the net; obviously, an instantaneously fireable transition can be modeled by setting the constant distribution to 0. For these two types of models, the user has the possibility to enable or disable the dependence between the firing rate of a transition and the marking of its input places by checking the ***Marking dependent*** option.

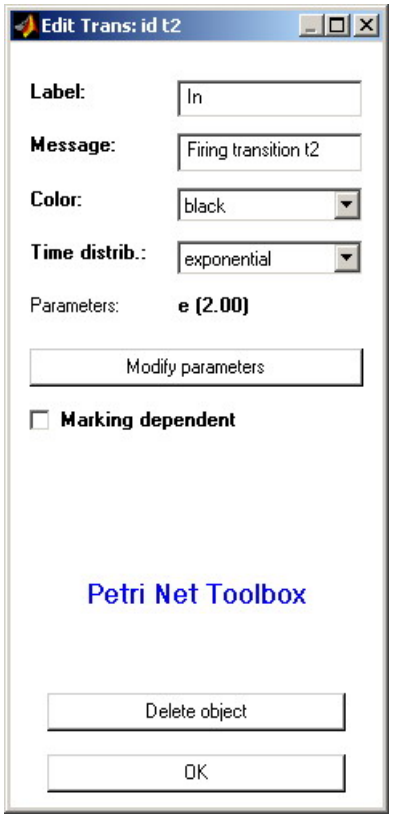


Fig. 2.5. *The **Edit Transition** dialogue box for modifying the properties of a transition*

The ***Delete object*** button placed at the bottom of the ***Edit Place*** dialogue box lets the user delete the transition from the model.

2.1.4. Arcs

To draw an arc in the ***Drawing Area***, the user must press the ***Add Arc*** button from the ***Drawing Panel*** or select the ***Add Arc*** command from the ***Modeling*** menu. Then, the user must click on the start node and then on the end node. The implementation in the ***PN Toolbox*** of PN models complies with the basic rule: *an arc of a PN can only connect a place to a transition* (pre-arc) *or a transition to a place* (post-arc)*, but never two nodes of the same kind*. The role of splitting arcs in two categories (pre- and post-arcs) will become apparent below, when talking about inhibitor arcs.

By default, an arc is represented as a straight arrow between the two selected nodes of the net. While in ***Draw Mode*** and no button from the ***Drawing Panel*** is pressed, a right-click on an arc of the net opens a MATLAB uicontext menu

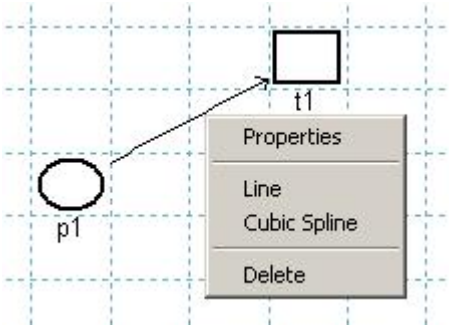


Fig. 2.6. *The uicontext menu of an arc.*

(fig. 2.6) that allows: (*i*) modifying the graphical representation in the ***Drawing Area*** (***Line*** command for a straight line or ***Cubic Spline*** command for a curve), (*ii*) deleting the arc (***Delete*** command) and (*iii*) opening the ***Edit Arc*** dialogue box (fig. 2.7) that lets the user modify the properties of the arc as a MATLAB object (***Properties*** command).

The ***Edit Arc*** dialogue box can be also opened by one of the procedures (EA1) or (EA2) described below, followed by a click on the desired arc. (EA1) consists in selecting the ***Edit Objects*** command from the ***Modeling*** menu; (EA2) consists in pressing the ***Edit Objects*** button from the ***Drawing Panel***.

Fig. 2.7. *The **Edit Arc** dialogue box for modifying the properties of an arc.*

Each arc is uniquely identified with an id that is automatically assigned by the ***PN Toolbox*** and cannot be changed by the user. This id appears in the title bar of the ***Edit Arc*** dialogue box.

The option ***Color*** displays the color used for drawing the arc. By default, this color is black, but the user can select another one from a list of eight predefined colors. Using different colors for the arcs of a net might be helpful for complex topologies (e.g. for highlighting different phases requested by a multi-step design procedure).

The option ***Type*** displays the type of an arc. By default, the type of an arc of a PN model is regular, but the user can change it into bidirectional or inhibitor, if necessary. A *regular* arc represents the standard connection between two nodes of different types. A *bidirectional* arc is equivalent to a pair of arcs with the same weight, one connecting a place to a transition and the other one connecting the same transition to the same place. The graphical representation of a bidirectional arc is a line with arrows at both ends.

An *inhibitor* arc can connect only a place to a transition. The transition is enabled only if the number of tokens in the input place is strictly smaller than the weight of the inhibitor arc. The graphical representation of an inhibitor arc is a line between the two nodes ending with a small circle (near the inhibited transition).

The option ***Weight*** displays the weight (multiplicity) of the arc. By default, the weight of an arc is equal to 1, but the user can set this field to a positive integer value.

By default, the weights of the arcs in a net are not shown in the ***Drawing Area***. At any stage of PN drawing, the user can visualize the current values of the weights for all the arcs in the net by selecting the ***Arc Weights*** command from the ***View*** menu. This command opens the dialogue box presented in fig. 2.8 and the user must click the ***Yes*** button.

Fig. 2.8. *The **View Arc Weights** dialogue box.*

2.1.5. Setting Priorities and / or Probabilities for Conflicting Transitions

By default, the ***PN Toolbox*** sets equal probabilities and / or priorities to conflicting transitions. These probabilities and / or priorities are used by the ***PN Toolbox***, when simulating a model, for selecting, from a set of conflicting transitions enabled at the same time, the next one to be fired. By selecting the ***Resolution for Conflicting Transitions*** option available under the ***Modeling*** menu, the corresponding dialogue window is opened (fig. 2.9) and the user can add, delete or edit the probabilities and / or priorities for each group of conflicting transitions.

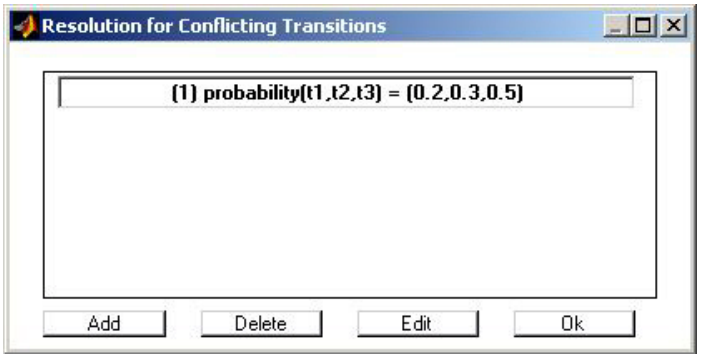


Fig. 2.9. *The **Resolution for Conflicting Transitions** dialogue window.*

2.2. Exploring Model Properties

2.2.1. Overview

The ***PN Toolbox*** offers several possibilities for analyzing discrete event systems modeled with PNs that are briefly presented in the sequel.

2.2.2. Incidence Matrix

The incidence matrix of a PN is computed directly from the graphical model and can be visualized in a MATLAB window opened by selecting the ***Incidence Matrix*** command from the ***Properties*** menu. This command is available for all types of PNs.

2.2.3. Behavioral Properties

For untimed PN models, the behavioral properties (e.g. boundedness, liveness, reversibility, etc.) may be studied based on the coverability tree of the net. Relying on the topology and the initial marking of the net, the ***PN Toolbox*** can automatically construct this tree and display it in either text or graphical mode. The coverability tree is built with or without the ω-convention (fig. 2.10). The ω-convention means the usage of a generic symbol (herein denoted by "ω") for referring to unbounded markings (Murata, 1989). Since the boundedness of a net cannot be a priori known, it is recommended to start the construction of the coverability tree by answering ***No*** to the question ***Do you want to use the "ω - convention"?***.

Fig 2.10. *The **Coverability Tree** dialogue window.*

2.2.4. Structural Properties

The *structural properties* of a PN are independent of the initial marking and may be characterized in terms of linear inequalities based on the incidence matrix of the net. The structural properties can be investigated from the ***Properties / Structural Properties*** menu, by means of the ***Structural Boundedness***, ***Conservativeness***, ***Repetitiveness*** and ***Consistency*** options.

2.2.5. Invariants

From the ***Properties / Invariants*** menu, the minimal-support *P*- and *T-invariants* (Martinez and Silva, 1982) can be visualized as vectors displayed in separate windows, automatically opened by the ***PN Toolbox***. New invariants may be constructed as linear combinations of these vectors.

2.3. Running a Simulation

2.3.1. Overview

After a model was drawn or retrieved from the disk, the user can simulate it. Three modes of simulation are implemented in the ***PN Toolbox***, namely: ***Step***, ***Run Slow*** and ***Run Fast***, that are called from the ***Simulation*** menu or from the ***Simulation Panel***.

The ***Step*** simulation mode ensures the progress of the simulation by a single step, when the status of a transition is changed from enabled to fired, or from fired to disabled. With the occurrence of each event two successive steps are associated, meaning the complete treatment of the firing transition.

The ***Step*** and ***Run Slow*** simulation modes are accompanied by animation. The animation displays the current number of tokens in the places of the net and colors transitions when they are fired. In the ***Run Slow*** mode, the speed of animation is adjustable from the ***Animation Delay*** option in the ***Simulation / Preferences*** dialogue box. The simulation can be stopped with a right-click. In these two modes, the ***Status Panel*** (placed in the bottom left-hand corner of the main window) presents the number of events and the current time.

Also, the user can record the progress of the simulation by using the ***Log File*** command from the ***Simulation*** menu. This journal is automatically saved in the project directory as an *HTML* file and can be read by means of any Internet browser using the ***Simulation / View history*** command available in the ***PN Toolbox***.

Note that in ***Step*** or ***Run Slow*** modes, the ***Scope*** and ***Diary*** facilities are available during the simulation. Their usage is relevant when studying timed PN.

The ***Run Fast*** simulation mode is not accompanied by animation and the simulation speed depends on the computer performances. This mode is recommended when the user is interested in estimating global performance indices of the PN model over a large time horizon (or, equivalently, for the occurrence of a large number of events).

From the ***Simulation / Breakpoint*** submenu the user can control the end point of the simulation in the ***Run Fast*** mode. The simulation is stopped by default when 1000 events have occurred or if no transition can fire any longer (i.e. when deadlock appears).

In all simulation modes, immediately after the end of the simulation, the values of the Performance Indices which characterize the simulated dynamics are available on request, by accessing the menu ***Performance***.

The simulation principle is different from one type of PN to another. Details about the simulation method implemented in the ***PN Toolbox*** for each type of PN model are given in the next pages.

2.3.2. Untimed Petri Nets

The sequencing of the events is reduced to simply ordering their occurrences. The simulation runs by firing the transitions one-by-one, in accordance with the *transition firing rule*. The ***PN Toolbox*** stores all the enabled transitions in a MATLAB array but only one transition fires at a time. The function that returns the next transition to be fired makes the decision according to the priorities or probabilities assigned to conflicting transitions from the ***Modeling / Conflicting Transitions*** command.

For the simulation modes ***Step*** and ***Run Slow***, the ***Events*** counter in the ***Status Panel*** is indexing and the currently firing transition gets the red color (by default) or an arbitrary color (selected by the user from the option ***Simulation / Preferences***). After it is fired, a transition returns to the background color.

The usage of the ***Scope*** and ***Diary*** facilities is not relevant for this type of PN.

2.3.3. T-timed Petri Nets

For *transition-timed* PN (*T*-timed PN), time durations can be assigned to the transitions; tokens are meant to spend that time as reserved in the input places of the corresponding transitions. In simulation, all the transitions that can fire due to the current marking are fired at the same time. For conflicting transitions, priorities or probabilities allow the choice of the transition(s) to fire. A transition can fire several times, in accordance with the marking of its input places and, from a theoretical point of view, an infinitesimal delay is considered to separate any two successive firings. After a transition fires, the enabling condition is tested for all the transitions of the net.

For the time durations assigned to the transitions, appropriate functions can be used to generate random sequences corresponding to probability distributions with positive support. When a transition is waiting to fire (i.e. during the period when its input places contain reserved tokens) the reserved tokens are graphically removed from the input places, but the

computation procedures of the performance indices consider them as remaining in those places (in full accordance with the theoretic approach).

The ***Diary*** facility opens a new window that displays (dynamically) the instant(s) when the next fireable transition(s) is (are) to fire (see section 2.3.7).

The ***Scope*** facility opens a new figure window that displays (dynamically) the evolution of the selected performance indices. The final value of the global index shown by the Scope is identical to that displayed on request, at the end of simulation, when the user explores the ***Performance*** menu (see section I.2.6).

2.3.4. P-timed Petri Nets

For *place-timed* PNs (*P*-timed PN), time durations can be assigned to the places; tokens are meant to spend that time as reserved in the corresponding places, immediately after their arrival. In simulation, all the transitions that can fire due to the current marking, fire at the same time. A transition can fire several times, in accordance with the marking of its input places and, from a theoretical point of view, an infinitesimal delay is considered to separate any two successive firings.

For the time durations assigned to the places, appropriate functions can be used to generate random sequences corresponding to probability distributions with positive support. During the simulation, the places that contain reserved tokens get the white color (default) or a color selected by using the option ***Simulation / Preferences***.

The ***Diary*** facility opens a new window that displays (dynamically) the instant(s) when the next fireable transition(s) is (are) to fire (see section 2.3.7).

The ***Scope*** facility opens a new figure window that displays (dynamically) the evolution of the selected performance indices. The final value of the global index shown by the Scope is identical to that displayed on request, at the end of simulation, when the user explores the ***Performance*** menu (see section I.2.6).

2.3.5. Stochastic Petri Nets

For *stochastic* PNs (SPNs), only exponential type distributions can be used to assign the time durations of the transitions. For conflicting transitions, it is the shortest time duration that allows the choice of the transition to fire, without using priorities or probabilities. Multiple firing of the same transition is not permitted, even if the token content of its input places allows this; i.e. the transition fires once and after the allocated time elapses, it will fire again if the current marking is appropriate. (Notice that the mechanism for selecting the firing transition and the condition for firing only once make the difference from the *T-timed PNs* with exponential type distributions for the time durations).

The sequencing of the firing transitions is exclusively controlled by the time durations of exponential type, which ensures the equivalence with the Markov chains. The firing rate of the transitions (i.e. the inverse of the mean time-duration) is, by default, marking dependent, but the user can select a marking-independent operation.

The ***Diary*** facility opens a new window that displays (dynamically) the instant(s) when the next fireable transition(s) is (are) to fire.

The ***Scope*** facility opens a new figure window that displays (dynamically) the evolution of the selected performance indices. The final value of the global index shown by the Scope is identical to that displayed on request, at the end of simulation, when the user explores the ***Performance*** menu (see section I.2.6).

2.3.6. Generalized Stochastic Petri Nets

Generalized Stochastic PNs (GSPNs) have two different classes of transitions: *immediate* transitions and *timed* transitions. Once enabled, immediate transitions fire in zero time. Timed transitions fire after a random, exponentially distributed enabling time as in the case of SPNs. For *timed* transitions, the firing rate (i.e. the inverse of the mean time-duration) is, by default, marking dependent, but the user can select a marking-independent operation (the same way as for SPNs).

The simulation procedure is similar to the SPN case, the only difference occurring in the case of the *immediate* transitions that fire first; priorities / probabilities can be associated to these transitions, in order to resolve the conflicts.

The ***Diary*** facility opens a new window that displays (dynamically) the instant(s) when the next fireable transition(s) is (are) to fire.

The ***Scope*** facility opens a new figure window that displays (dynamically) the evolution of the selected performance indices. The final value of the global index shown by the Scope is identical to that displayed on request, at the end of simulation, when the user explores the ***Performance*** menu (see section I.2.6).

2.3.7. Diary

When active, this facility opens a new MATLAB window (fig. 2.11) that displays (dynamically - during the simulation) the instant(s) when the next fireable transition(s) is (are) to fire. For several fireable transitions, the displayed list of transitions is ordered by the instants when the firings will occur.

The ***Diary*** facility is available only in the ***Step*** and ***Run Slow*** simulation modes. The usage of this facility is recommended for timed PN models and it gives no relevant information in the case of untimed models.

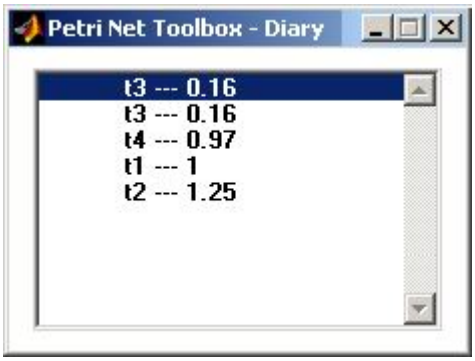


Fig 2.11. *The **Diary** window*.

2.3.8. Scope

The ***Scope*** facility opens a new MATLAB window (fig. 2.12) that displays (dynamically) the evolution of the selected performance index. Each index provides two types of information, namely a current value (that characterizes the PN at the current moment of the simulation) and a global value (that characterizes the whole evolution of the PN, as an average over the whole

time-horizon from the beginning of the simulation till the current moment). For the selected performance index, both current and global values are plotted versus time.

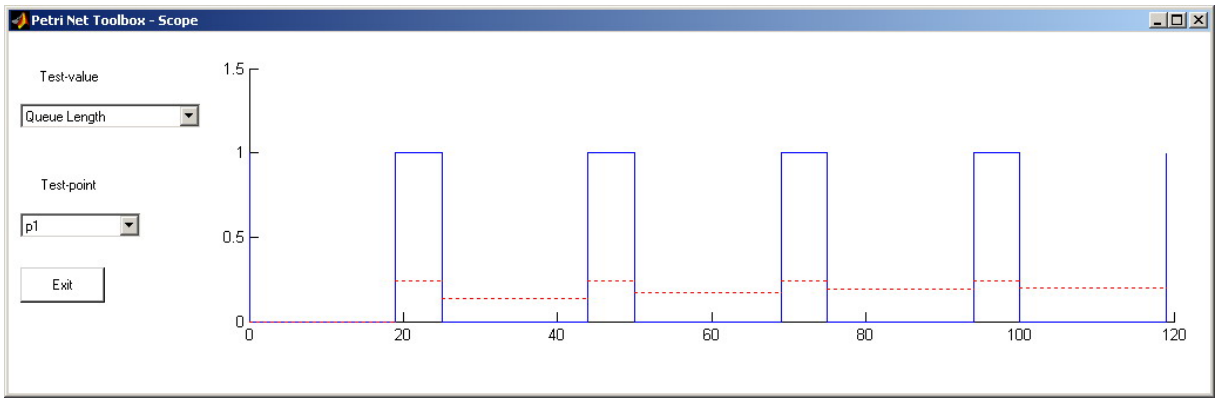


Fig 2.12. *The **Scope** window*.

The usage of the ***Scope*** facility is available only for timed PNs in the ***Step*** and ***Run Slow*** simulation modes.

The current / global performance indices displayed by ***Scope*** are:

- for a transition:
  - ***Service Distance***: means the duration between two successive firings of the selected transition;
  - ***Service Time***: means the duration between the enabling and the firing of the selected transition;
  - ***Utilization***: means the current status of the selected transition: 1 for the time-interval between the enabling and the firing of the selected transition; 0 otherwise;
- for a place:
  - ***Arrival Distance***: means the duration between two successive instants when tokens arrive in the selected place;
  - ***Throughput Distance***: means the duration between two successive instants when tokens leave the selected place;
  - ***Queue Length***: means the number of tokens in the selected place.

During a simulation experiment, only one of the six performance indices can be dynamically displayed by the ***Scope***. The red color is automatically set for plotting the evolution of the global performance index. The blue color is automatically set for plotting the evolution of the current performance index. The final value shown by the ***Scope*** for the global performance index is identical to the value displayed on request, at the end of simulation, by accessing the ***Performance*** menu.

2.4. Analyzing Simulation Results

After simulation ends, the global performance indices described in the ***Scope*** section are stored by the ***PN Toolbox*** and can be visualized by using the ***Performance*** menu. Besides these, there are also a number of global indices for which the current values are not defined.

The following two tables present the complete lists of global indices associated with the places (displayed by the ***Place Indices*** command) and the transitions (displayed by the ***Transition Indices*** command), respectively:

- for a transition:
  - ***Service Sum***: total number of firings;
  - ***Service Distance***: average value of the current index ***Service Distance***;
  - ***Service Rate***: average frequency of firings (inverse of ***Service Distance***);
  - ***Service Time***: average value of the current index ***Service Time***;
  - ***Utilization***: average value of the current index ***Utilization***;
- for a place:
  - ***Arrival Sum***: total number of arrived tokens;
  - ***Arrival Distance***: average value of the current index ***Arrival Distance***;
  - ***Arrival Rate***: average frequency of token-arrivals (inverse of ***Arrival Distance***);
  - ***Throughput Sum***: total number of departed tokens;
  - ***Throughput Distance***: average value of the current index ***Throughput Distance***;
  - ***Throughput Rate***: average frequency of token-departures (inverse of ***Throughput Distance***);
  - ***Waiting Time***: average waiting time per token;
  - ***Queue Length***: average value of the current index ***Queue Length***.

These indices may be saved in *HTML* format in a file placed in the working directory.

2.5. Max-Plus Models

For place-timed event graphs, the ***PN Toolbox*** is able to directly derive the max-plus state-space representation from the topology and initial marking of a marked graph, in an implicit form:

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

where M denotes the maximal number of tokens in the initial marking and N stands for the number of simulated iterations. The components of the input vector $\boldsymbol{u}(k) = \left[u_1(k)\ u_2(k)\ \ldots\ u_m(k) \right]^{\mathrm{T}}$ and those of the output vector $\boldsymbol{y}(k) = \left[y_1(k)\ y_2(k)\ \ldots\ y_p(k) \right]^{\mathrm{T}}$ represent the k-th firing moments of the m source transitions and of the p sink transitions, respectively. In a similar manner, the state vector $\boldsymbol{x}(k) = \left[x_1(k)\ x_2(k)\ \ldots\ x_n(k) \right]^{\mathrm{T}}$ corresponds to the n transitions in the net that have both input and output places (i.e. those transitions which are neither sources nor sinks).

The max-plus model is automatically constructed by the ***PN Toolbox*** by using the ***Max-Plus*** menu. This command opens the ***Max-Plus*** window (denoted by (1) in fig. 2.13) providing access to the following features available for max-plus analysis:

- ***Equations*** button: displays the max-plus equations in a separate window (denoted by (2) in fig. 2.13);
- ***Input*** button: allows the user to set the values of the input vectors (time instants). This can be done (*i*) by introducing the time instants as a matrix with *m* rows and *N* columns directly in the *dialogue box*, or (*ii*) by creating the matrix ***input_time*** with the structure described above in MATLAB's *Workspace*;
- ***Plot***: allows the user to select the components of the input, state or output vectors that will be plotted, by using vectors with components equal to 0 (enabled) or 1 (disabled);
- ***Iteration***: runs an iteration of the simulation at a time. The values of the state and output vectors are displayed in the ***Numerical Results*** box located on the left side of the ***Max-Plus*** window. The selected components are plotted;
- ***Reset***: resets the model to the initial state and clears the MATLAB axes;
- ***Exit***: closes the ***Max-Plus*** window and returns the control to the ***PN Toolbox*** GUI.

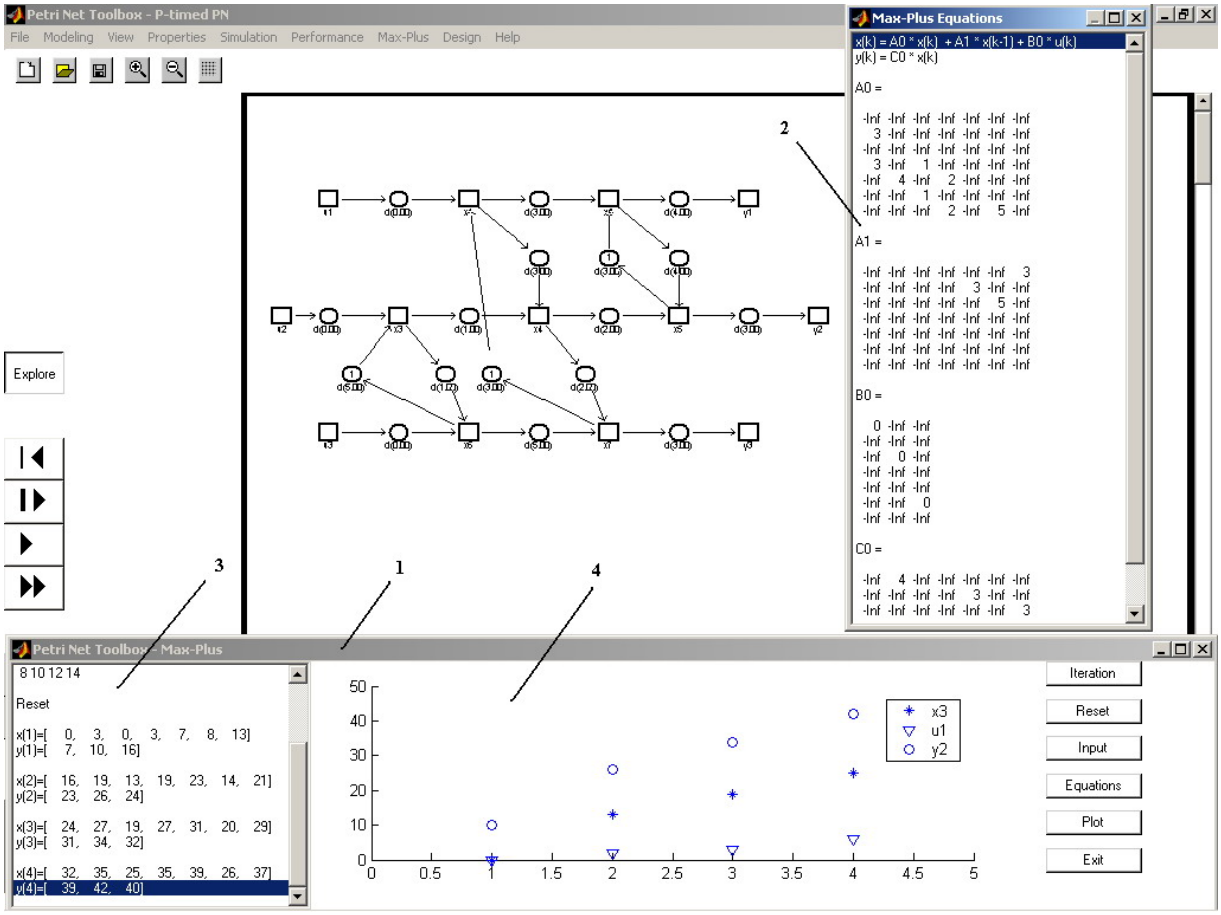


Fig. 2.13. *Facilities available in the* ***PN Toolbox*** *for max-plus analysis*.

The ***Numerical Results*** box (located in the left side of the ***Max-Plus*** window, denoted by (3) in fig. 2.13) displays the input vectors as well as information reflecting the stage of the analysis (the state and output vectors after each iteration).

In the MATLAB axes (denoted by (4) in fig. 2.13) placed in the middle of the ***Max-Plus*** window, all the selected components are plotted. The components of the input, state and output vectors are plotted with the symbols "∇", "$*$" and "$\bigcirc$", respectively. For two or more components of the same vector, different colors are used with the same symbol.

2.6. Design

A facility for the synthesis of timed or (generalized) stochastic PN models is ***Design***, which allows exploring the dependence of a ***Design Index* (*I*)** on one or two ***Design Parameters*** that vary within intervals defined by the user.

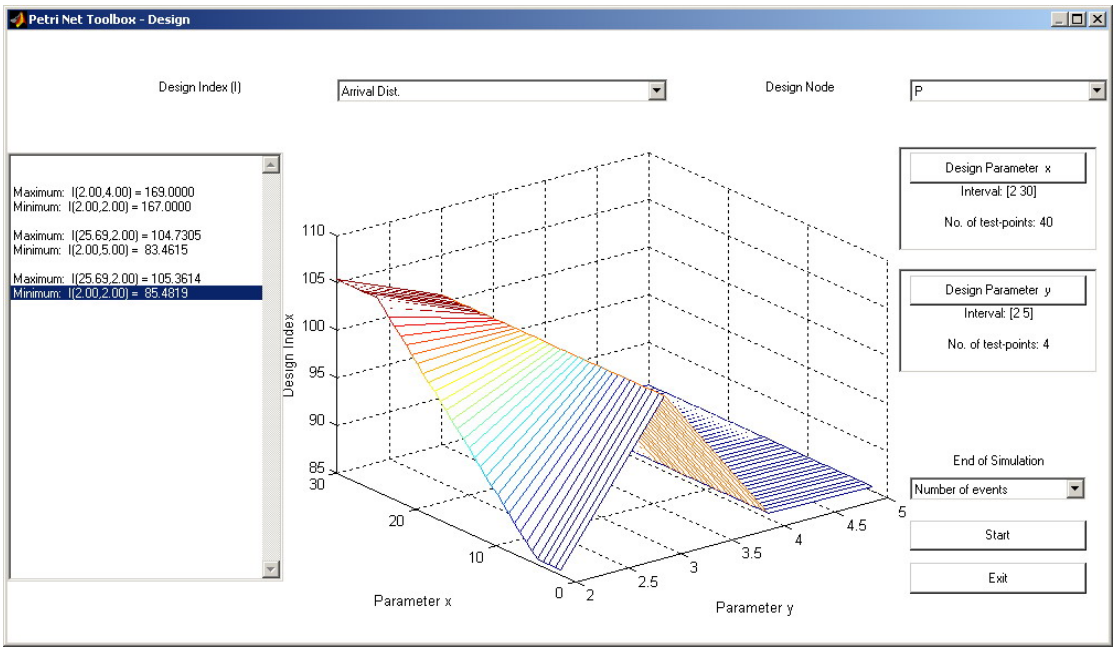


Fig. 2.14. *The window opened after selecting the* ***Design*** *option of the* ***PN Toolbox****.*

A ***Design Parameter*** may be selected as (*i*) the initial marking of a place, (*ii*) a parameter of the distribution function defining the duration associated with a place or a transition in timed PNs, (*iii*) the mean value of the exponential distribution function associated with a transition in (generalized) stochastic PNs. The ***Design Parameters*** are generically denoted by ***x*** and ***y***; any other notation is not accepted. To ensure the correspondence between the symbol ***x*** (or ***y***) and the selected ***Design Parameter***, this symbol must be used as a numerical value when filling out the appropriate dialogue box (exactly as detailed in sections 2.1.2 and 2.1.3). The place or transition subject to parameterization is automatically colored in red.

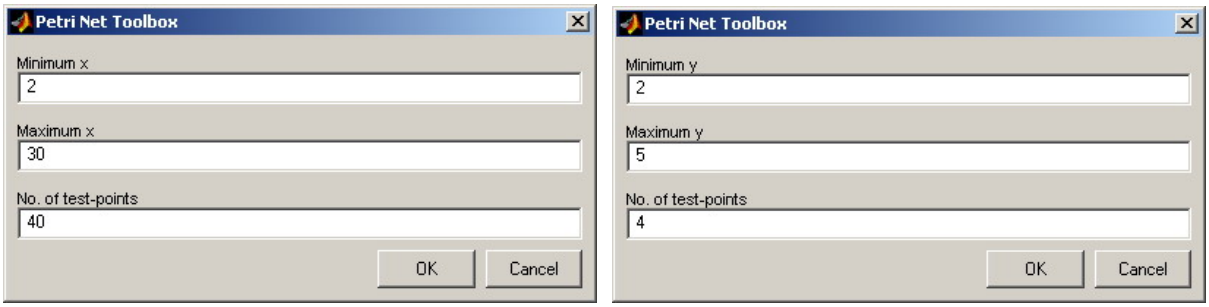
The ***Design Index*** may be selected as a global performance index associated with a ***Design Node***, namely ***Service Rate***, ***Service Distance***, ***Service Time*** or ***Utilization*** for a

transition, or ***Arrival Rate***, ***Arrival Distance***, ***Throughput Rate***, ***Throughput Distance***, ***Waiting Time*** or ***Queue Length*** for a place – see section 2.4. The ***Design Index*** and the ***Design Parameter*** do not necessarily refer to the same node.

When selecting the ***Design*** command from the ***Menu Bar*** of the ***PN Toolbox***, a new window is opened (see fig. 2.14). The choice of the ***Design Index*** and ***Design Node*** is made from the corresponding combo-boxes.

By pushing the ***Design Parameter x*** button, the dialogue box presented in fig. 2.15.(a) is opened, allowing the user to set the numerical information corresponding to ***x***, consisting in the extreme values to be considered and the number of equally-spaced test-points (where the ***Design Index*** will be calculated as commented bellow). The same way, numerical information can be set for ***y*** by pushing the ***Design Parameter y*** button, which opens the dialogue box presented in fig. 2.15.(b).

The set of simulation-experiments defined by the ***Design Parameters*** is launched by pushing the ***Start*** button and cannot be interrupted until scanning all the test-points. The ***Exit*** button serves only for closing the ***Design*** window (i.e. after finishing all the simulation-experiments).



(a) (b)

Fig. 2.15. *The dialogue boxes for setting* ***Design Parameter*** (a) ***x*** *and* (b) ***y***.

For each test-point (in the case of a single parameter) or for each pair of test-points (in the case of two parameters) automatically constructed by the ***PN Toolbox*** in accordance with the information given in the ***Design Parameter*** dialogue boxes, a simulation experiment is performed in the ***Run Fast*** mode. Each simulation experiment ends when the condition selected via the combo-box ***End of Simulation*** is fulfilled. 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 (I)*** on the ***Design Parameter(s)***. Besides the graphical plot, the toolbox displays the extreme values of the ***Design Index***, in the message box placed in the left side of the ***Design*** window.

Note that the simulation experiments performed for the given design parameter(s) provide the whole set of global indices associated with all the nodes of the net. Each of these indices can be visualized by appropriately choosing the ***Design Index*** and / or ***Design Node***. Such information remains available as long as the user does not press the buttons ***Design Parameter x***, ***Design Parameter y***, and does not alter the condition in the combo-box ***End of Simulation*** (because these operations reset the specific work variables).

PART II

Petri Net Simulink Block
– User's Guide –

Chapter 3

DESCRIBING THE GRAPHICAL USER INTERFACE OF THE PN SIMULINK BLOCK

3.1. Overview of the PNSB Editor

The ***PNSB*** can be found in the *Simulink Library Browser* (fig. 3.1).

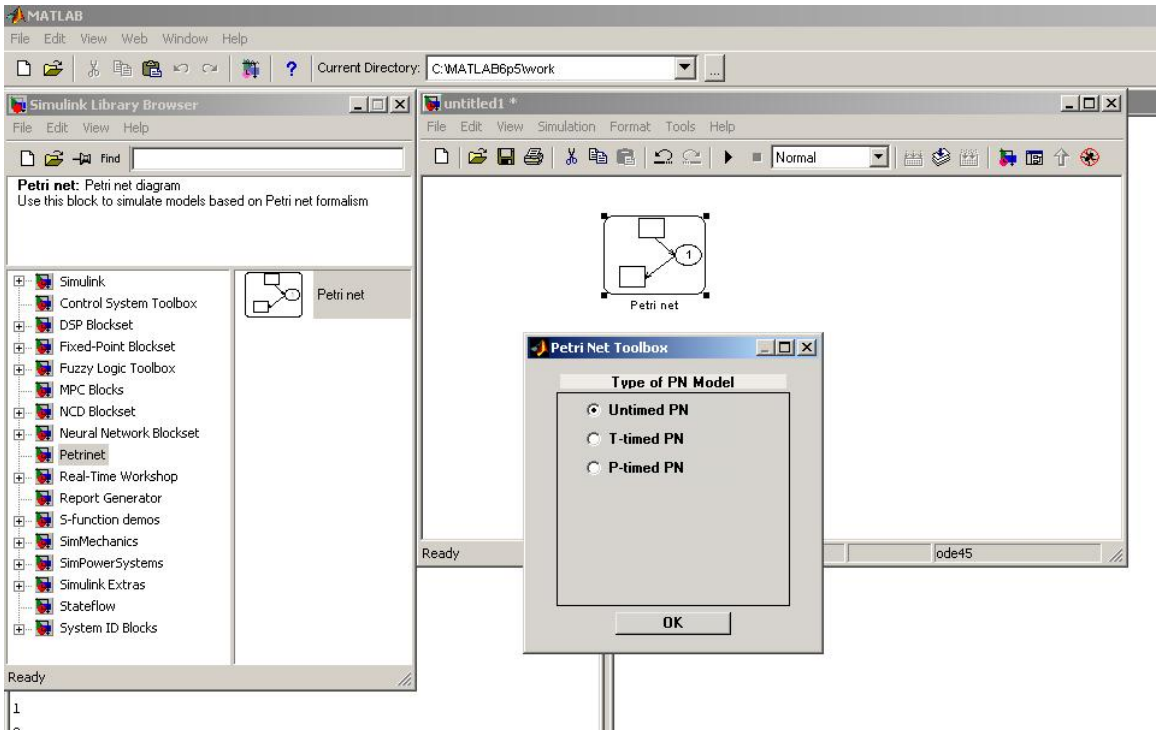


Fig. 3.1. *Inserting a **PNSB** in a Simulink model.*

Once a ***PNSB*** is inserted in a Simulink model, the user is requested first to choose the type of the Petri net model to be stored in that block. By double-clicking the PN block, it is opened a graphical interface that allows the user to draw the PN model (through the ***PNSB Editor*** – see figure 3.2), define the triggering events (***PNSB Event Explorer*** – see section 4.1.6) and debug the Simulink model (***PNSB Debugger*** – see section 4.2.5). The operation of the ***PNSB*** relies on callback functions that (i) initialize variables, (ii) generate the graphical interface, (iii) display the ***PNSB Editor*** when the user double clicks the Simulink block, (iv) hide the editor window when the user closes it and (v) save/load the PN model into/from an ***xml*** file.

It is worth noticing that the functions of the ***PNSB Editor*** are similar to the editing facilities available in the *Draw Mode* of the ***PN Toolbox***; this ensures the immediate adaptation of the user's skills, once he/she has already been acquainted with the exploitation of the ***PN Toolbox***. The ***PNSB Editor*** exhibits five control panels (see fig. 3.2): ***Menu Bar*** (PNE1), ***Quick Access Toolbar*** (PNE2), ***Drawing Panel*** (PNE3), ***Drawing Area*** (PNE4) and a ***Message Box*** (PNE5). Further on, all these panels are described.

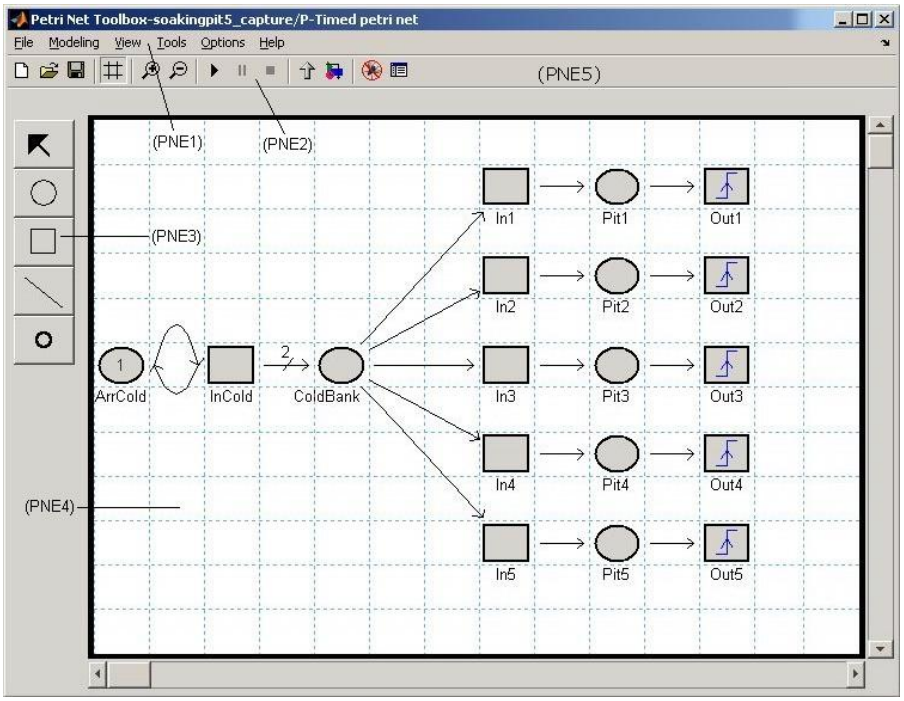


Fig. 3.2. *The main window of the* ***PNSB Editor***.

3.2. Menu Bar

The *Menu Bar* (PNE1 in fig 3.2) displays a set of six drop-down menus, from which the user can access all the facilities available in the application. The menus available under the ***Menu Bar*** are: ***File***, ***Modeling***, ***View***, ***Tools***, ***Options*** and ***Help***.

3.2.1. File Menu

The ***File*** menu (fig. 3.3) offers facilities for file-handling operations. This is the only menu available when the ***PN Toolbox*** GUI is started. This menu contains the commands:

- ***New Model***: opens a new board in the ***Drawing Area*** for the user to build a new PN model. This command opens a dialogue box for selecting the type of the new model;
- ***Open Model***: loads a previously saved model;
- ***Close Model***: closes the current PN model;
- ***Save Model***: saves the PN model drawn in the ***Drawing Area***;
- ***Save Model As …***: saves the current model with a name given by the user;
- ***Close System***: closes the current Simulink model.

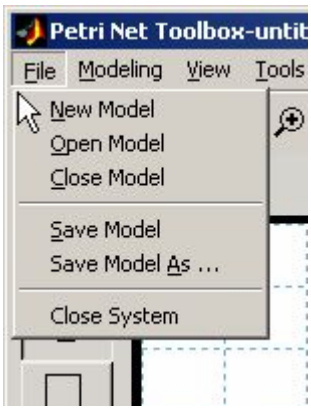


Fig. 3.3. *The **File** menu.*

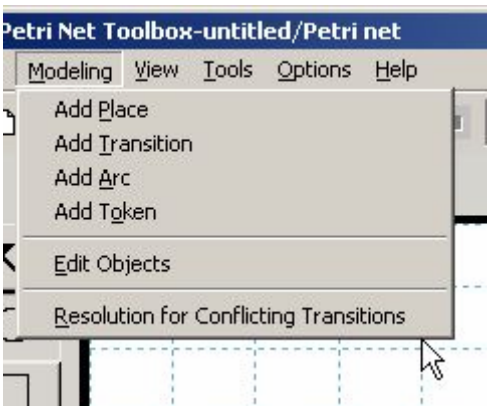


Fig. 3.4. *The **Modeling** menu.*

3.2.2. Modeling Menu

The ***Modeling*** menu (fig. 3.4) provides tools for graphical editing (graph nodes, arcs, tokens, labels) a model in the ***Drawing Area***. The commands available in this menu are:

- ***Add Place***: adds a new place to the current PN model;
- ***Add Transition***: adds a new transition to the current PN model;
- ***Add Arc***: allows drawing an arc between two different nodes of the current PN model;
- ***Add Token***: adds a token in a specific place;
- ***Edit Objects***: opens the dialogue box associated with the net object that is selected in the ***Drawing Area*** by a click on the left button of the mouse. It allows the user to edit the properties of the selected object; these properties are in full accordance with the type of the PN model, so that part of them might be already disabled by the initial choice of the net type.
- ***Resolution for Conflicting Transitions***: allows assigning priorities or probabilities to conflicting transitions.

3.2.3. View Menu

The ***View*** menu (fig. 3.5) allows choosing specific conditions for visualization of the current model. The commands available in this menu are:

- ***Zoom In***: lets the user get a closer view to a smaller portion of the ***Drawing Area***;
- ***Zoom Out***: lets the user visualize a larger portion of the ***Drawing Area***;
- ***Show Grid:*** allows the user to add/remove grid lines to/from the ***Drawing Area***;
- ***Arc Weights:*** permits displaying or hiding the current values of arc weights.

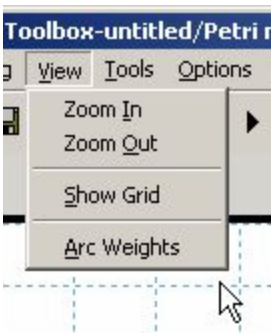


Fig. 3.5. *The **View** menu.*

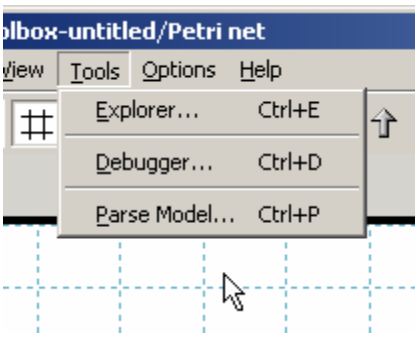


Fig. 3.6. *The **Tools** menu.*

3.2.4. Tools Menu

The ***Tools*** menu (fig. 3.6) allows opening the Event Explorer or the Debugger, and also permits the user to parse the Simulink model. The commands available from this menu are:

- ***Explorer…***: opens the Event Explorer;
- ***Debugger…***: opens the model Debugger;
- ***Parse Model…***: parses the Simulink model.

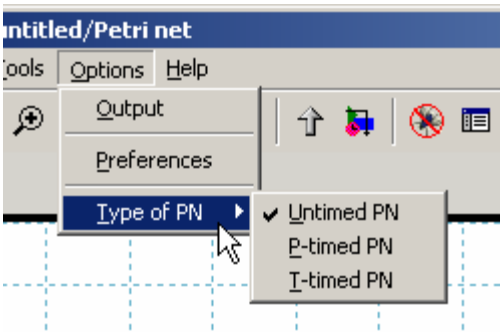


Fig. 3.7. *The **Options** menu.*

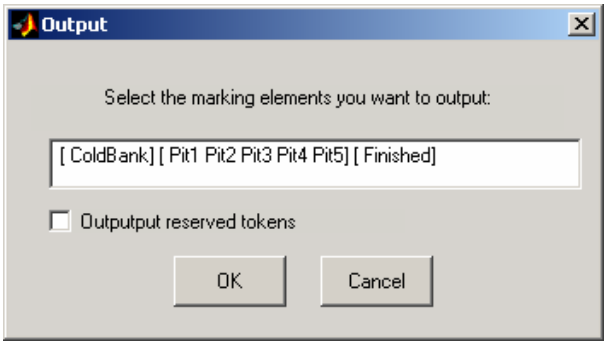


Fig. 3.8. *The **Output** dialog box.*

3.2.5. Options Menu

The ***Options*** menu (fig. 3.7) allows the user to set specific options for the current PN model. The commands available from this menu are:

- ***Output***: opens a dialogue box (fig. 3.8) allowing the user to choose the places whose markings form the output vector(s) of the ***PNSB***;
- ***Preferences***: opens a dialogue box (fig. 3.9) allowing the user to set the conditions of the simulation (***Token-In-Place Color*** is meaningful only for P-timed PNs; the value of the ***Seed*** is used to initialize the random number generator of MATLAB). These conditions are valid only when using the Debugger;

- ***Type of PN***: allows the user to change the type of PN model stored in the ***PNSB***.

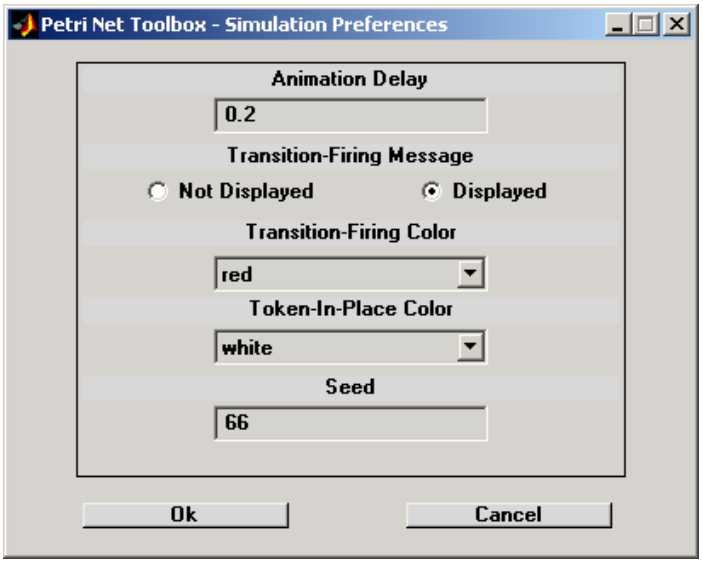


Fig. 3.9. *The **Preferences** dialog box.*

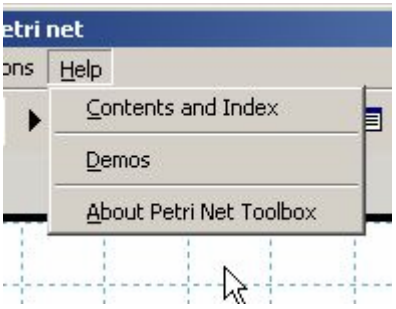


Fig. 3.10. *The **Help** menu.*

3.2.6. Help Menu

The ***Help*** menu (fig. 3.10) provides information about the exploitation of the ***PN Toolbox***. The commands available in this menu are as follows:

- ***Contents and Index***: opens the online help;
- ***Demos***: allows visualization of some **Flash** movies initiating the user in the exploitation of the ***PNSB***;
- ***About Petri Net Toolbox***: contains information about the authors of the ***PN Toolbox*** and of the ***PNSB***.

3.3. Quick Access Toolbar

The *Quick Access Toolbar* (PNE2 in fig. 3.1) maps the most frequently used facilities of the ***PNSB Editor***. It is placed as a horizontal bar just below the ***Menu Bar*** and presents six groups of image buttons, denoted by (1) to (6) in fig. 3.11.

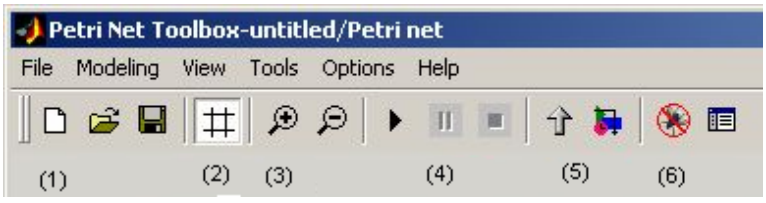


Fig. 3.11. *The **Quick Access Toolbar**.*

The actions of the first three buttons (1) are identical to those controllable by the ***New***, ***Open*** and ***Save*** commands from the ***File*** menu. The actions of the next three buttons are identical to those controllable by the ***Show Grid*** (2), ***Zoom In*** and ***Zoom Out*** (3) commands

from the ***View*** menu. The next three buttons (4) allow the user to ***Start***, ***Pause*** or ***Stop*** the simulation of the Simulink model. The buttons denoted by (5) permit bringing in the front window the Simulink model in which the current ***PNSB*** is included, or opening the Simulink Library Browser. The actions of the buttons in the last group (6) are identical to those controllable by the ***Debugger…*** and ***Explorer…*** commands from the ***Tools*** menu.

3.4. Drawing Panel

The ***Drawing Panel*** (PNE3 in fig. 3.2) is placed vertically, in the left side of the main window, below the ***Quick Access Toolbar***, and presents five image buttons (fig. 3.12) whose actions are identical to the commands ***Edit Objects*** (1), ***Add Place*** (2), ***Add Transition*** (3), ***Add Arc*** (4) and ***Add Token*** (5) available in the ***Modeling*** menu. When the user presses one of these buttons, the message corresponding to the employed command is displayed in the ***Message Box*** (PNE5 in fig. 3.2).

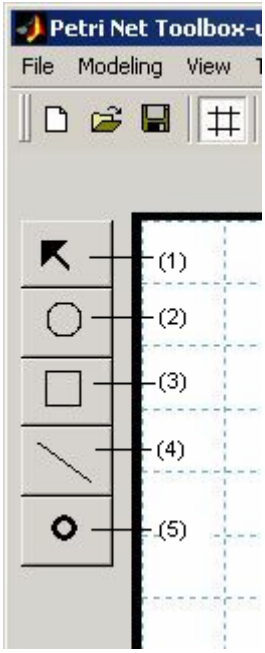


Fig. 3.12. *The* ***Drawing Panel***.

3.5. Drawing Area

The *Drawing Area* (PNE4 in fig. 3.2) is implemented as a matrix of cells, where the nodes of the PN graph are to be placed, with two scrollbars 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 with 50 rows and 50 columns. The bottom left-hand corner corresponds to the (0,0) cell, whereas the upper right-hand corner to the (49,49) cell. In one cell the user can draw a single node (place or transition), so that the total number of places and transitions in a model cannot exceed 2500.

Using the ***Zoom In*** / ***Zoom Out*** commands from the ***View*** menu, the number of the cells is decreased/increased with 10 cells on each axis. The ***Show Grid*** command adds/removes the grid lines to/from the Drawing Area.

3.6. Message Box

The ***Message Box*** (PNE5 in fig. 3.2) is a MATLAB text object used to display messages to the user. When the user presses one of the five buttons from the ***Drawing Panel***, the message corresponding to the employed command is displayed in the ***Message Box***. During simulation, a message is displayed only if option ***Displayed*** is checked for ***Transition-Firing Message*** in the ***Preferences*** dialogue box (from the ***Simulation*** menu). When a transition fires, the ***Message Box*** displays the text associated with this transition; the text must be previously defined in the ***Edit Transition*** dialogue box (***Draw Mode*** - see section 4.1.3).

Chapter 4

EXPLOITING THE PN SIMULINK BLOCK

4.1. Building a Model

4.1.1. Overview of the PNSB Editor

The functionality of the ***PNSB Editor*** is similar to the ***Draw Mode*** of the graphical user interface from the ***PN Toolbox***. It provides a set of commands for building a PN model, which are accessible from the ***Modeling*** menu or from the corresponding buttons in the ***Drawing Panel***. The model may be easily drawn, in a natural fashion, in the ***Drawing Area***. The user can also have a read / write access to the properties of the net nodes and arcs by opening the dialog box associated with the object selected in the ***Drawing Area***. In addition, the ***PN Toolbox*** allows the assignment of priorities and / or probabilities to conflicting transitions.

The ***Drawing Area*** displays a grid with dotted gray lines. The user may hide or show these lines by using the command ***View / Show Grid*** or by pressing the corresponding button from the ***Quick Access Toolbar***. The nodes of the PN may be placed only in the grid cells; each cell can contain a single net node. This way, each node is uniquely characterized by its coordinates in the ***Drawing Area***.

When a PNSB is inserted in a Simulink model, the user has to chose the type of the PN model from ***Untimed***, ***P-timed*** and ***T-timed*** (see figure 4.1). Only these three types are accepted since it makes no sense have stochastic or generalized stochastic PNs with synchronized transitions. By selecting the type of the PN, some of the work variables are automatically changed. This is because the simulation and analysis procedures differ from one type to another. The drawing board of a new model may be opened in the ***Drawing Area*** by selecting the ***File / New Model*** command or by pressing the ***New*** button from ***Quick Access***

Toolbar. This command opens a dialog box (fig. 4.1) which permits the selection of the type of model to be built. The default type of a new PN model created in the ***PNSB*** is ***Untimed***.

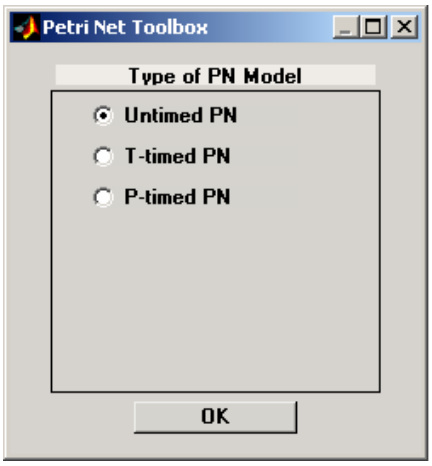


Fig. 4.1. *The dialogue box for selecting the type of a PN model.*

A PN model created in the ***PNSB*** is saved as an **xml** file. The syntax of such a file is given in Appendix A.2. To ensure compatibility between the models handled in the ***PN Toolbox*** and the ones in the ***PNSB***, all possible tags coincide. A model created in the ***PNSB*** may be loaded in the ***PN Toolbox*** and analyzed there, if necessary. It is also possible to load into the ***PNSB*** a model created in the ***PN Toolbox*** (obviously it must have one of the accepted types).

A previously created model may be loaded from the disk by means of the command ***Open*** available under the ***File*** menu or by pressing the ***Open*** button from the ***Quick Access Toolbar***. Changes in an existing model can be done directly in the GUI, or by editing the corresponding **xml** file.

Almost all the operations for building a PN model in the ***PNSB*** are similar to the ones performed under the GUI of the ***PN Toolbox***. The only differences regard setting the *Triggering Events* and the *Broadcasted Events* associated to transitions. Further on we present these operations in detail, but the user accustomed to building PN models under the ***PN Toolbox*** may skip this part.

4.1.2. Places

A place is graphically represented in the ***Drawing Area*** by a circle. To draw a place in the ***Drawing Area***, the user must press the ***Add Place*** button from the ***Drawing Panel*** or select the ***Add Place*** command from the ***Modeling*** menu. Then, the user must click only once into the desired grid cell of the ***Drawing Area***. A second click in the same grid cell has no effect.

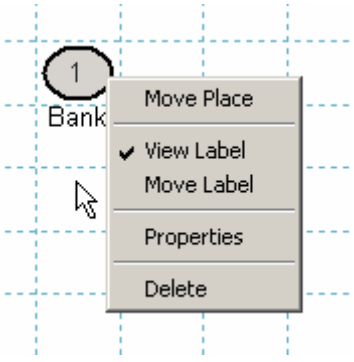


Fig. 4.2. *The uicontext menu of a place*

Once the circle corresponding to a place is drawn, a label is automatically attached to it. The default position of the label is below the circle.

After drawing a place, there are two ways to change its position in the ***Drawing Area*** while no button from the ***Drawing Panel*** is pressed. The first way is to left-click the desired place and then drag it in the new position. The second way is to right-click the place; as a result, a MATLAB uicontext menu (fig. 4.2) appears and the command ***Move Place*** becomes available. In both cases, the label is moved together with the place.

For a selected place, the uicontext menu also allows: (*i*) controlling the label's visibility on the screen (***View Label*** command), (*ii*) changing the position of the label (***Move Label*** command), (*iii*) deleting the place (***Delete*** command) and (*iv*) opening the ***Edit Place*** dialogue box (fig. 4.3) that lets the user modify the properties of the place as a MATLAB object (***Properties*** command).

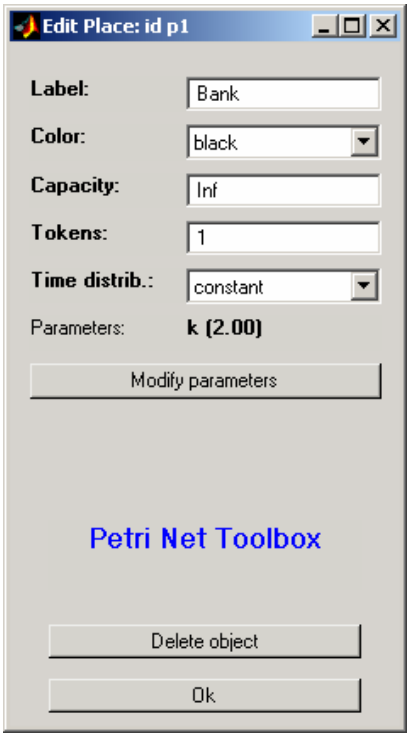


Fig. 4.3. *The **Edit Place** dialogue box for modifying the properties of a place.*

The ***Edit Place*** dialogue box may be also opened by one of the procedures (EP1) or (EP2) described below, followed by a click on the desired place. (EP1) consists in selecting the ***Edit Objects*** command from the ***Modeling*** menu; (EP2) consists in pressing the ***Edit Objects*** button from the ***Drawing Panel***.

Each place is uniquely identified with an **id** that is automatically assigned by the ***PN Toolbox*** and cannot be changed by the user. This id appears in the title bar of the ***Edit Place*** dialogue box.

The option ***Label*** displays the string that is used as the label of the place. By default, this string coincides with the id of the place. The user can modify this string (without affecting the id) if necessary.

The option ***Color*** displays the color used for drawing the place. By default, this color is black, but the user can select another one from a list of eight predefined colors. Using different colors for drawing the places of a net might be helpful for complex topologies (e.g. for highlighting different phases requested by a multi-step design procedure).

The option ***Capacity*** displays the capacity of the place. By default, the capacity is **Inf** (the IEEE arithmetic representation for positive infinity). The user can set this field to a positive integer value.

The option ***Tokens*** displays the number of tokens in the place. By default, this number is 0. The user can set this field to a positive integer value. If the marking of a place is greater than 0, this marking is shown as a number inside the circle corresponding to that position. Void marking is not explicitly shown.

There are two more possibilities to add a token to a place (AT1) or (AT2) described below, followed by a click on the desired place. (AT1) consists in selecting the ***Add Token***

command from the ***Modeling*** menu; (AT2) consists in pressing the ***Add Token*** button from the ***Drawing Panel***.

In case of place-timed PN models (see section 4.2.4), the ***Time distribution*** option associated with a place allows the user to specify the probability distribution and the necessary parameter(s) which define the corresponding time-duration. This option is not available for untimed or transition-timed models.

The ***Delete object*** button placed at the bottom of the ***Edit Place*** dialogue box lets the user delete the place from the model.

4.1.3. Transitions

A transition is graphically represented in the ***Drawing Area*** by a square. To draw a transition in the ***Drawing Area***, the user must press the ***Add Transition*** button from the ***Drawing Panel*** or select the ***Add Transition*** command from the ***Modeling*** menu. Then, the user must click only once into the desired grid cell of the ***Drawing Area***. A second click in the same grid cell has no effect.

Once the square corresponding to a transition is drawn, a label is automatically attached to it. The default position of the label is below the square.

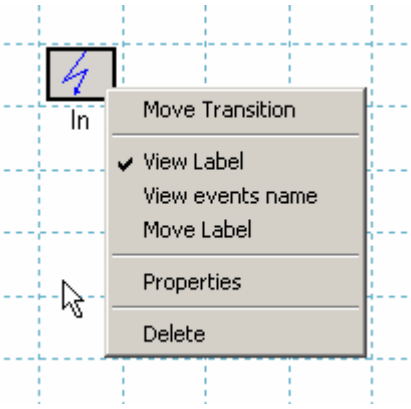


Fig. 4.4. *The uicontext menu of a transition.*

The same as for a place, after drawing a transition, there are two ways to change its position in the ***Drawing Area*** while no button from the ***Drawing Panel*** is pressed. The first way is to left-click the desired transition and then drag it in the new position. The second way is to right-click the transition; as a result, a MATLAB uicontext menu (fig. 4.4) appears and the command ***Move Transition*** becomes available. In both cases, the label is moved together with the transition.

For a selected transition, the uicontext menu also allows: (*i*) controlling the label's visibility on the screen (***View Label*** command), (*ii*) displaying the associated triggering events on the screen (***View Events Name*** command), (*iii*) changing the position of the label (***Move Label*** command), (*iv*) deleting the transition (***Delete*** command) and (*v*) opening the ***Edit Transition*** dialogue box (fig. 4.5) that lets the user modify the properties of the transition as a MATLAB object (***Properties*** command).

The ***Edit Transition*** dialogue box can be also opened by one of the procedures (ET1) or (ET2) described below, followed by a click on the desired transition. (ET1) consists in selecting the ***Edit Objects*** command from the ***Modeling*** menu; (ET2) consists in pressing the ***Edit Objects*** button from the ***Drawing Panel***.

Each transition is uniquely identified with an **id** that is automatically assigned by the ***PN Toolbox*** and cannot be changed by the user. This id appears in the title bar of the ***Edit Transition*** dialogue box.

The option ***Label*** displays the string that is used as the label of the transition. By default, this string coincides with the id of the transition. The user can modify this string (without affecting the id) if necessary.

If necessary, the user can associate one or more ***Triggering Events*** to a certain transition. The triggering mode must be later defined in the ***Event Explorer*** (see Section 4.1.6)**.**

The option ***Broadcast Events*** allows the user to associate one or more events to the transition firing. These events represent outputs for the ***PNSB*** and can be used to trigger other blocks included in the Simulink model.

The option ***Color*** displays the color used for drawing the transition. By default, this color is black, but the user can select another one from a list of eight predefined colors. Using different colors for the transitions of a net might be helpful for complex topologies (e.g. for highlighting different phases requested by a multi-step design procedure).

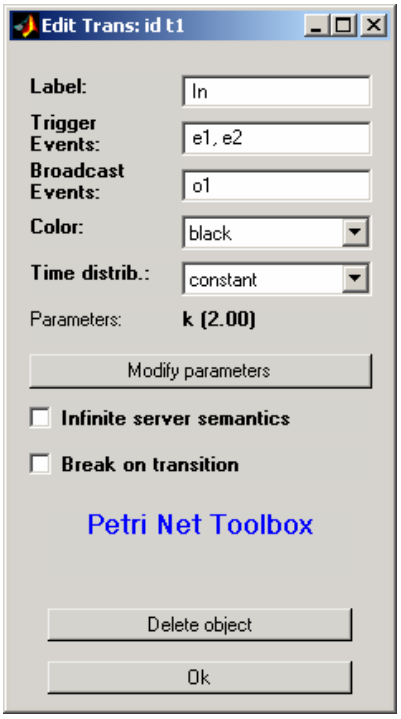


Fig. 4.5. *The **Edit Transition** dialogue box for modifying the properties of a transition*

In case of transition-timed PN models (see section 4.2.3), the ***Distribution*** option associated with a transition allows the user to specify the probability distribution and the necessary parameter(s) which define the corresponding time-duration. The user has the possibility to enable or disable the dependence between the firing rate of a transition and the marking of its input places by checking the ***Marking dependent*** option.

The ***Delete object*** button placed at the bottom of the ***Edit Place*** dialogue box lets the user delete the transition from the model.

4.1.4. Arcs

To draw an arc in the ***Drawing Area***, the user must press the ***Add Arc*** button from the ***Drawing Panel*** or select the ***Add Arc*** command from the ***Modeling*** menu. Then, the user must click on the start node and then on the end node. The implementation in the ***PN Toolbox*** of PN models complies with the basic rule: *an arc of a PN can only connect a place to a transition* (pre-arc) *or a transition to a place* (post-arc)*, but never two nodes of the same kind*. The role of splitting arcs

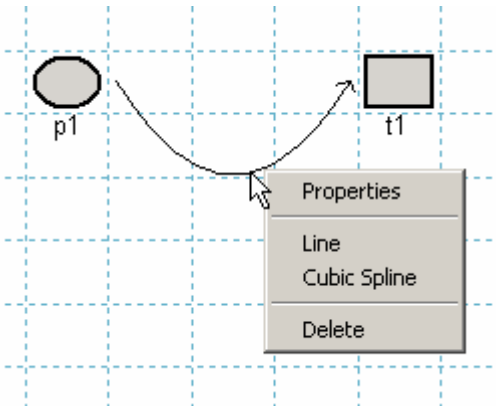


Fig. 4.6. *The uicontext menu of an arc.*

in two categories (pre- and post-arcs) will become apparent below, when talking about inhibitor arcs.

By default, an arc is represented as a straight arrow between the two selected nodes of the net. While in ***Draw Mode*** and no button from the ***Drawing Panel*** is pressed, a right-click on an arc of the net opens a MATLAB uicontext menu (fig. 4.6) that allows: (*i*) modifying the graphical representation in the ***Drawing Area*** (***Line*** command for a straight line or ***Cubic Spline*** command for a curve), (*ii*) deleting the arc (***Delete*** command) and (*iii*) opening the ***Edit Arc*** dialogue box (fig. 4.7) that lets the user modify the properties of the arc as a MATLAB object (***Properties*** command).

The ***Edit Arc*** dialogue box can be also opened by one of the procedures (EA1) or (EA2) described below, followed by a click on the desired arc. (EA1) consists in selecting the ***Edit Objects*** command from the ***Modeling*** menu; (EA2) consists in pressing the ***Edit Objects*** button from the ***Drawing Panel***.

Fig. 4.7. *The **Edit Arc** dialogue box for modifying the properties of an arc.*

Each arc is uniquely identified with an id that is automatically assigned by the ***PN Toolbox*** and cannot be changed by the user. This id appears in the title bar of the ***Edit Arc*** dialogue box.

The option ***Color*** displays the color used for drawing the arc. By default, this color is black, but the user can select another one from a list of eight predefined colors. Using different colors for the arcs of a net might be helpful for complex topologies (e.g. for highlighting different phases requested by a multi-step design procedure).

The option ***Type*** displays the type of an arc. By default, the type of an arc of a PN model is regular, but the user can change it into bidirectional or inhibitor, if necessary. A *regular* arc represents the standard connection between two nodes of different types. A *bidirectional* arc is equivalent to a pair of arcs with the same weight, one connecting a place to a transition and the other one connecting the same transition to the same place. The graphical representation of a bidirectional arc is a line with arrows at both ends.

An *inhibitor* arc can connect only a place to a transition. The transition is enabled only if the number of tokens in the input place is strictly smaller than the weight of the inhibitor arc. The graphical representation of an inhibitor arc is a line between the two nodes ending with a small circle (near the inhibited transition).

The option ***Weight*** displays the weight (multiplicity) of the arc. By default, the weight of an arc is equal to 1, but the user can set this field to a positive integer value.

Fig. 4.8. *The **View Arc Weights** dialogue box.*

By default, the weights of the arcs in a net are not shown in the ***Drawing Area***. At any stage of PN drawing, the user can visualize the current values of the weights for all the arcs in the net by selecting the ***Arc Weights*** command from the ***View*** menu. This command opens the dialogue box presented in fig. 4.8 and the user must click the ***Yes*** button.

4.1.5. Setting Priorities and / or Probabilities for Conflicting Transitions

By default, the ***PN Toolbox*** sets equal probabilities and / or priorities to conflicting transitions. These probabilities and / or priorities are used by the ***PNSB***, when simulating a model, for selecting, from a set of conflicting transitions enabled at the same time, the next one to be fired. By selecting the ***Resolution for Conflicting Transitions*** option available under the ***Modeling*** menu, the corresponding dialogue window is opened (fig. 4.9) and the user can add, delete or edit the probabilities and / or priorities for each group of conflicting transitions.

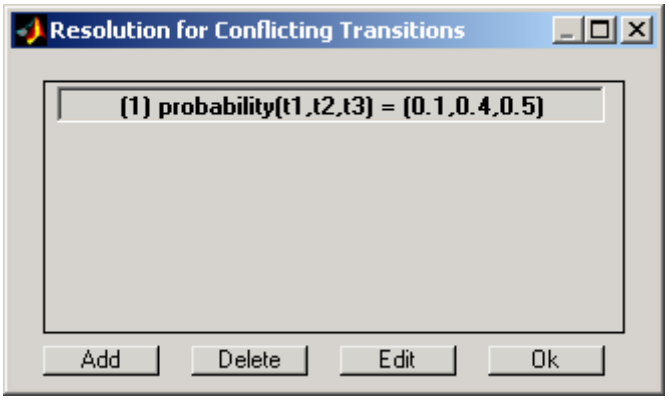


Fig. 4.9. *The* ***Resolution for Conflicting Transitions*** *dialogue window.*

4.1.6. The PNSB Event Explorer

The ***PNSB Event Explorer*** (fig. 4.10) allows the user to manage the triggering events of the current PN model. The user can edit the name (EE1) and the triggering mode (EE2) for each external event. The number and order of PNSB input signals has to match the number and order of triggering events defined in the Event Explorer for the PN model. The selection of a certain event displayed in the Event panel is accomplished by pressing the corresponding *Select event* button (EE3). The user can *Add* (EE4) or *Delete* (EE5) events, and change their order by moving up (EE6) or down (EE7).

There are three types of events that can be defined by the user, the differences between them resulting from the triggering conditions that must be met by the corresponding Simulink signal. Thus, a *rising edge* event triggers the PNSB when the input signal rises from a zero or negative value to a positive value (or zero if the initial value is negative), while a *falling edge* event is activated whenever the input signal falls from a positive value to a zero or negative value (or zero if the initial value is positive). An *either edge* event triggers the PNSB when the input signal is either rising or falling. The default triggering mode for a newly created event is *either edge*.

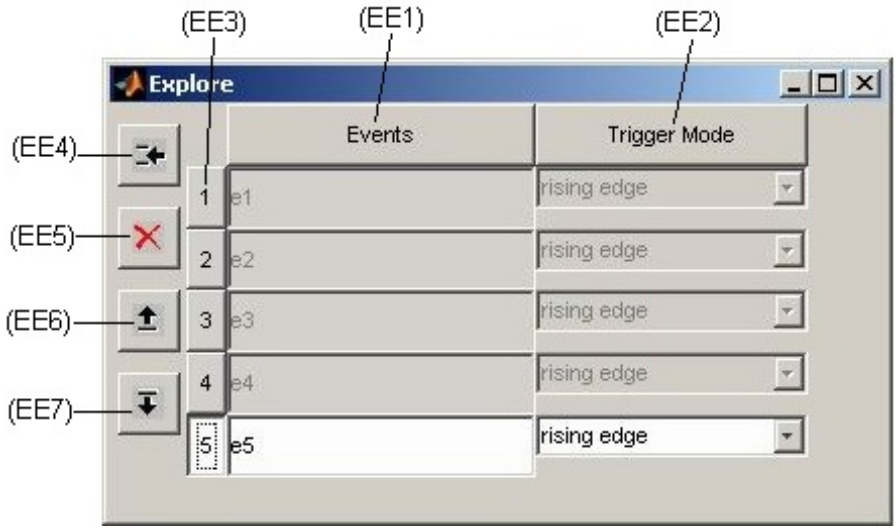


Fig. 4.10. *The **PNSB Event Explorer**.*

For the triggered transitions, special icons were introduced as shown in fig. 4.11, associated with the following types of events: (a) - defined by rising edges; (b) - defined by falling edges; (c) – defined by either rising or falling edges. A transition triggered by an event that is still undefined (in the PNSB Event Explorer) is represented by icon (d).

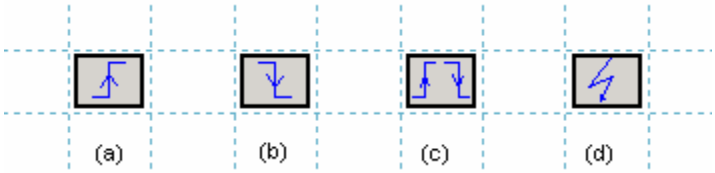


Fig. 4.11. *The icons used for synchronized (triggered) transitions.*

4.2. Running a Simulation

4.2.1. Overview

The ***PNSB*** accepts, as input, a set of signals in a multiplexed form; the evolution of each signal can generate Simulink events. The number of input signals has to be equal to the number of events defined for the PN. Once an input signal generates an event, the simulation time of Simulink "freezes", the ***PNSB*** identifies the generated event and fires all the enabled transitions. After that, the simulation is resumed until a new event is generated. At each simulation step, the ***PNSB*** outputs the net marking (as defined by means of the ***Options / Output*** command) and the broadcasted internal events (generated by transitions firings and defined in the corresponding ***Edit Transition*** dialogue boxes).

The simulation principle is different from one type of PN to another. Details about the simulation method implemented in the ***PN Toolbox*** for each type of PN model are given in the next pages.

4.2.2. Untimed Petri Nets

The sequencing of the events is reduced to simply ordering their occurrences. The simulation runs by firing the transitions one-by-one, in accordance with the *transition firing rule*. The ***PN Toolbox*** stores all the enabled transitions in a MATLAB array but only one transition fires at a time. The function that returns the next transition to be fired makes the decision according to the priorities or probabilities assigned to conflicting transitions from the ***Modeling / Conflicting Transitions*** command.

The currently firing transition gets the red color (by default) or an arbitrary color (selected by the user from the option ***Simulation / Preferences***). After it is fired, a transition returns to the background color.

4.2.3. T-timed Petri Nets

For *transition-timed* (*T*-timed) PNs, time durations can be assigned to the transitions; tokens are meant to spend that time as reserved in the input places of the corresponding transitions. In simulation, all the transitions that can fire due to the current marking are fired at the same time. For conflicting transitions, priorities or probabilities allow the choice of the transition(s) to fire. A transition can fire several times, in accordance with the marking of its input places and, from a theoretical point of view, an infinitesimal delay is considered to separate any two successive firings. After a transition fires, the enabling condition is tested for all the transitions of the net.

For the time durations assigned to the transitions, appropriate functions can be used to generate random sequences corresponding to probability distributions with positive support. When a transition is waiting to fire (i.e. during the period when its input places contain reserved tokens) the reserved tokens are graphically removed from the input places, but the computation procedures of the performance indices consider them as remaining in those places (in full accordance with the theoretic approach).

4.2.4. P-timed Petri Nets

For *place-timed* PNs (*P*-timed PN), time durations can be assigned to the places; tokens are meant to spend that time as reserved in the corresponding places, immediately after their arrival. In simulation, all the transitions that can fire due to the current marking, fire at the same time. A transition can fire several times, in accordance with the marking of its input places and, from a theoretical point of view, an infinitesimal delay is considered to separate any two successive firings.

For the time durations assigned to the places, appropriate functions can be used to generate random sequences corresponding to probability distributions with positive support. During the simulation, the places that contain reserved tokens get the white color (default) or a color selected by using the option ***Simulation / Preferences***.

4.2.5. The PNSB Debugger

The ***PNSB Debugger*** is a useful tool when simulating a Simulink model containing both time-driven and event-driven systems. The debugger pauses the simulation at each simulation step, allowing the user to inspect the current state of the Petri net or to visualize the evolution of some particular signals from the Simulink model. The MATLAB figure opened when the user selects the ***Debugger*** option from the ***Tools*** menu is presented in fig. 4.12. The upper buttons (D1) start and stop the simulation, the time progress being displayed in a bar (D2) together with the simulation time, the start and the stop simulation time. If the *Enable Debugger* checkbox (D3) is checked, the simulation is interrupted on each occurrence of a triggering event associated with the PNSB. The *Step* button (D4) fires one transition of the Petri net at a time, while the *Continue* button (D5) runs the simulation introducing, between successive firings, a delay set by the *Delay* listbox (D6). The maximum number of firings is set by the value entered in the *Breakpoint* edit item (D7).

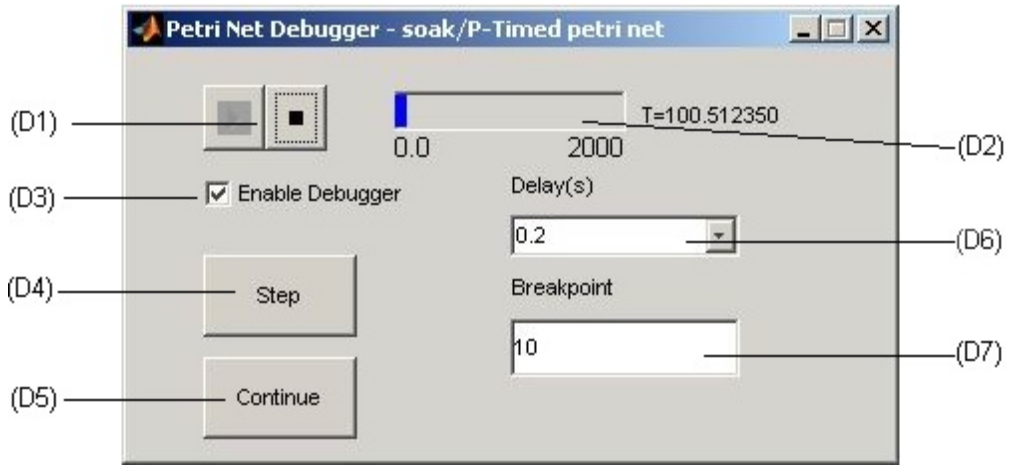


Fig. 4.12. *The* ***PNSB Debugger***.

PART III

Demonstrative Examples

Demo 1

TWO DINNING PHILOSOPHERS

The dinning philosophers' problem is a classic example of multi-process synchronization. It was first presented in 1971 by Edsger Dijkstra (a well-known Dutch computer scientist) as a problem where five computers competed for access to five shared tape drive peripherals. Afterwards, it was reformulated as the dining philosophers' problem

We consider this problem in the case when there are two philosophers sitting in front of each other at a table. On the table there is a plate with food and two chopsticks on each side of the plate. In order to eat, each philosopher needs both chopsticks. He first takes the stick placed on his left, then the stick placed on his right. After eating, the philosopher places both chopsticks simultaneously back on the table.

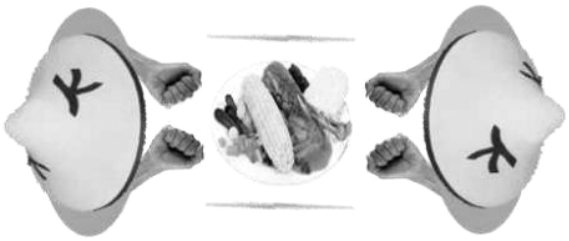
The current demonstration illustrates the usage of the ***PN Toolbox*** for the analysis of a version of this problem that involves a computer system with two processors sharing two disks (in parallel).

This demo illustrates:

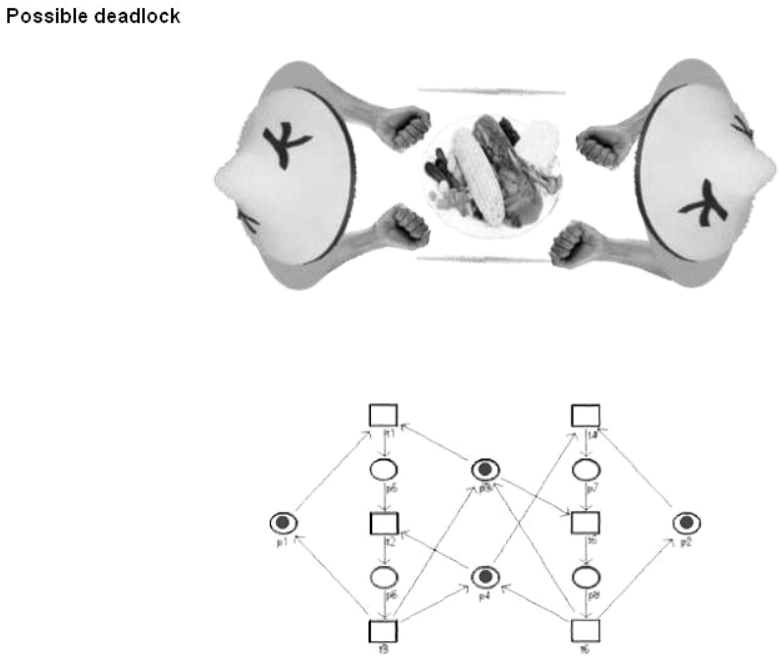
- the construction of an untimed Petri net model
- the analysis of deadlock (via the coverability tree)
- the prevention of deadlock through lookahead feedback
- the access to the following information about a Petri net model:
    - incidence matrix
    - minimal-support P- and T-invariants
    - structural properties

D1.1. Illustration of the Physical System

1. The two philosophers

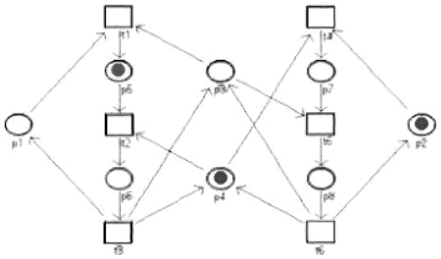


2. Model of the system that allows the analysis of deadlock



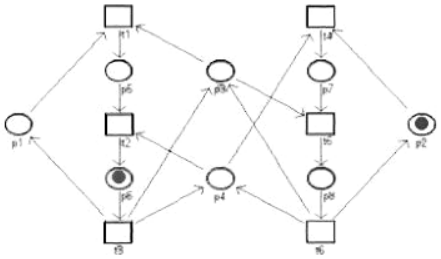
3. The first philosopher starts to eat (takes a chopstick)

Possible deadlock

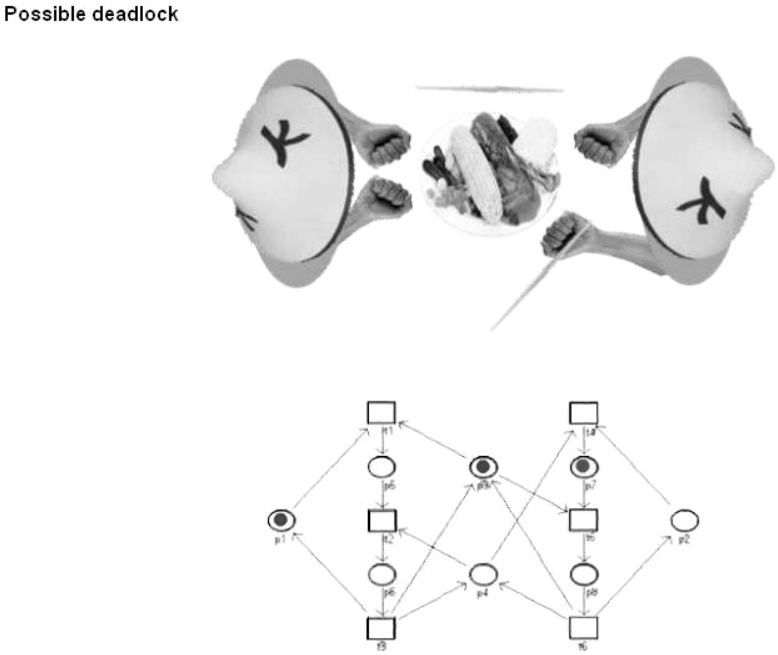


4. The first philosopher eats (he has both chopsticks)

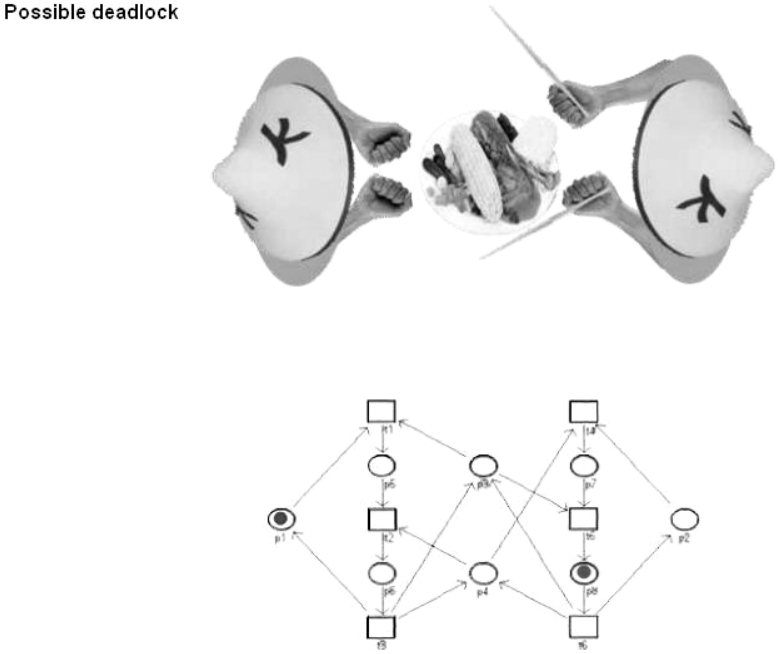
Possible deadlock



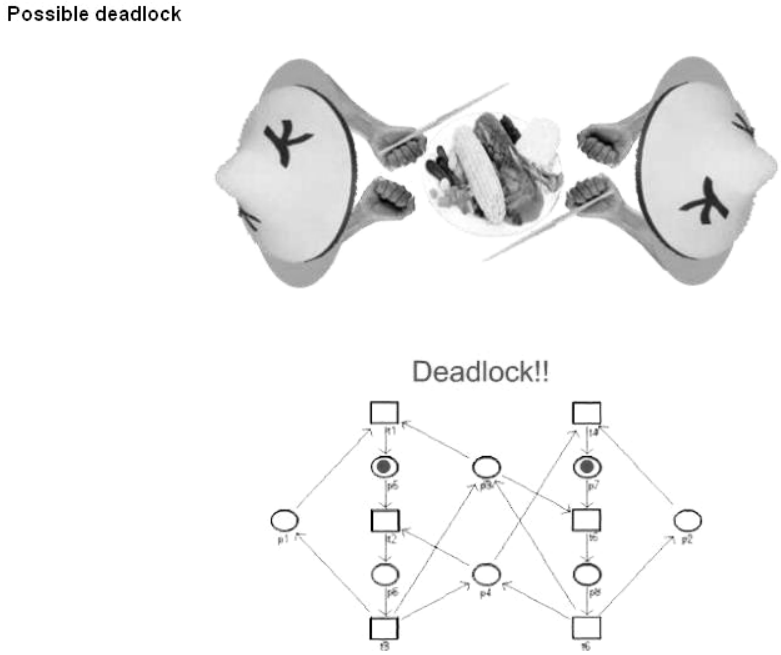
5. The second philosopher starts to eat (takes a chopstick)



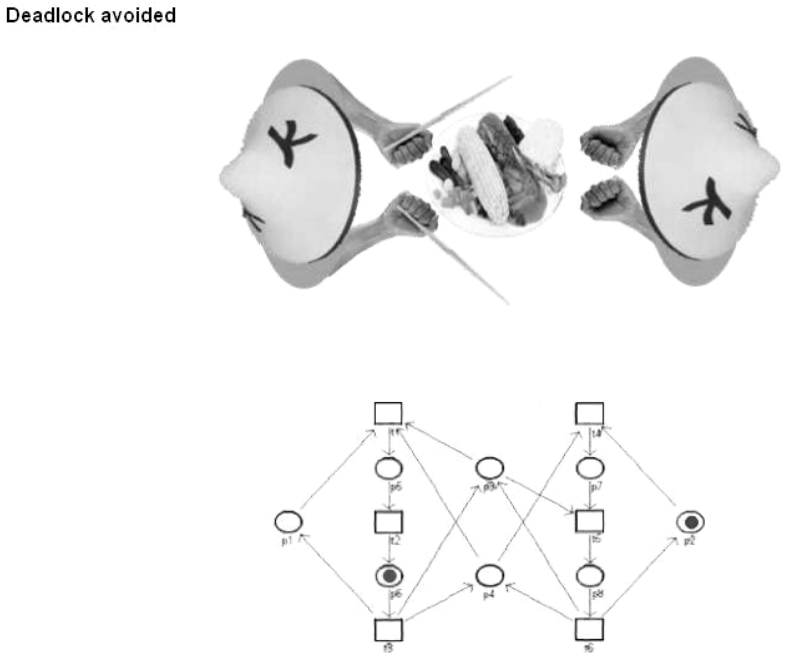
6. The second philosopher eats (he has both chopsticks)



7. Deadlock situation – each philosopher has a chopstick, neither one can finish eating

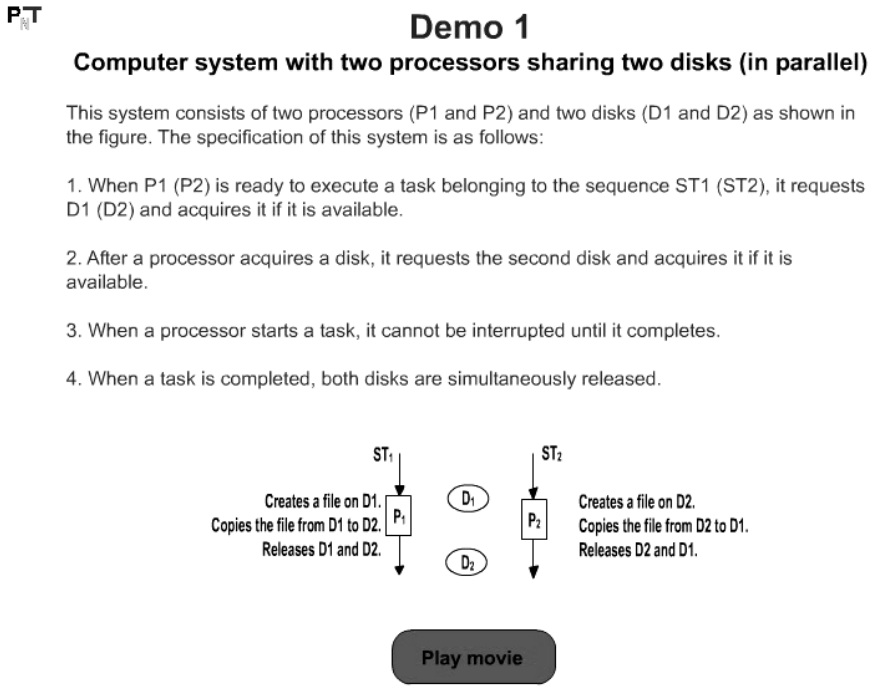


8. Model of the system with deadlock avoidance policy

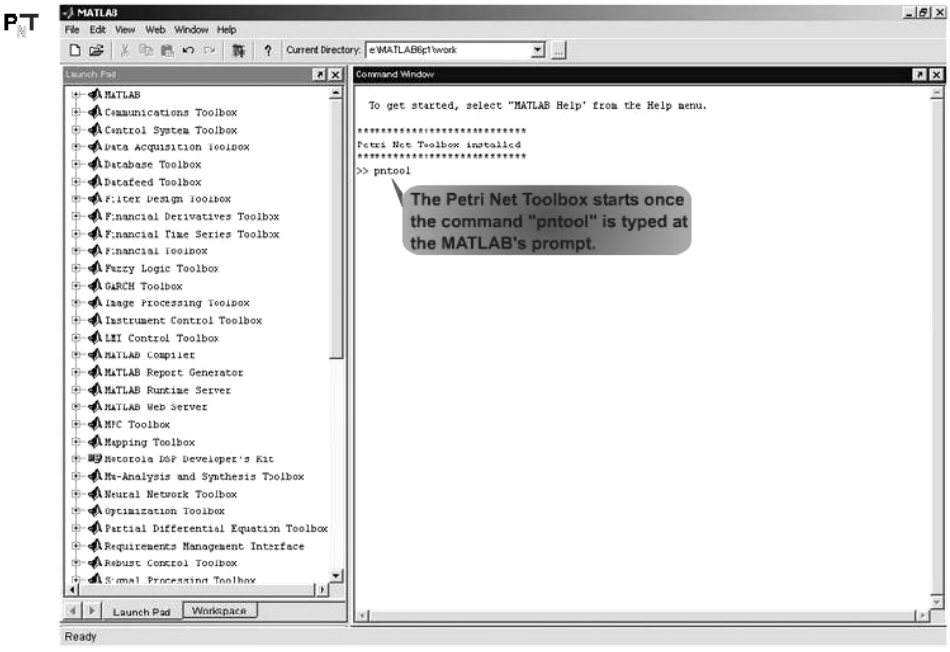


D1.2. Usage of the PN Toolbox for System Modeling and Analysis

1. Problem description



2. Start the PNTOOL graphical interface

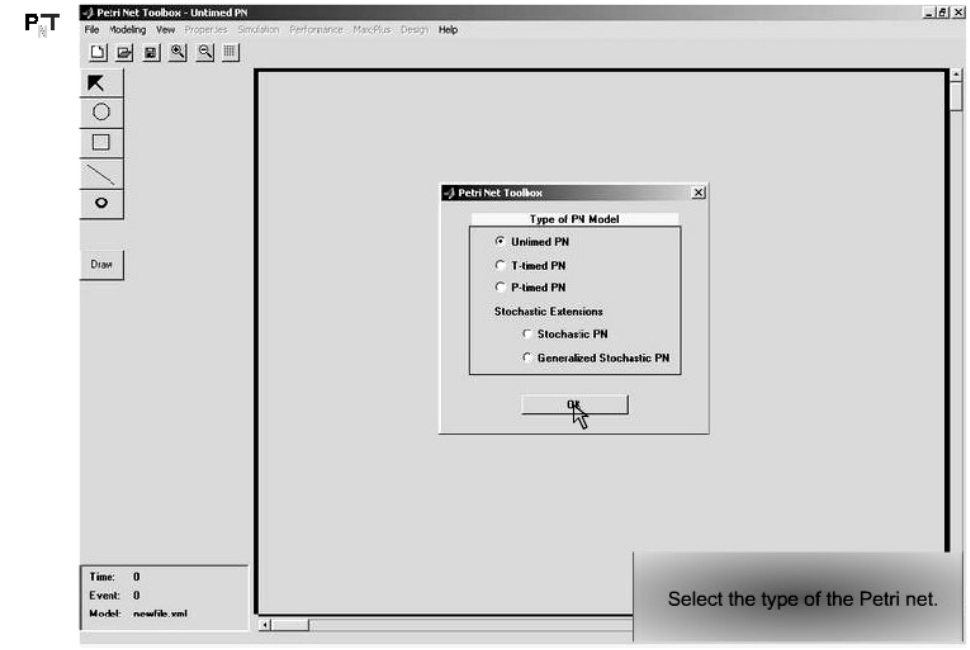


Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

3. Create a new PN model

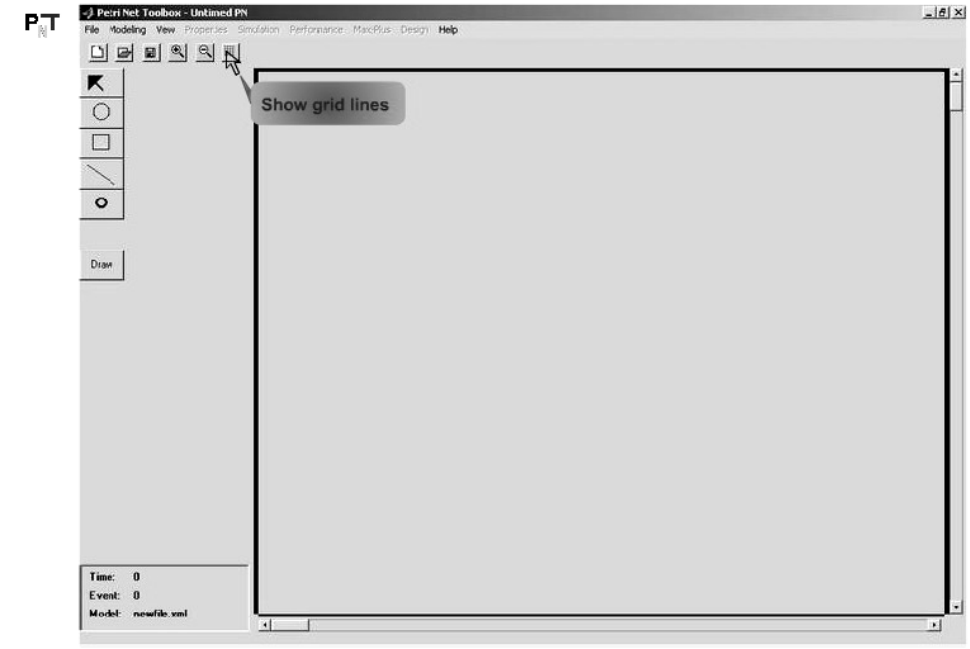
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

4. Select the type of the PN model



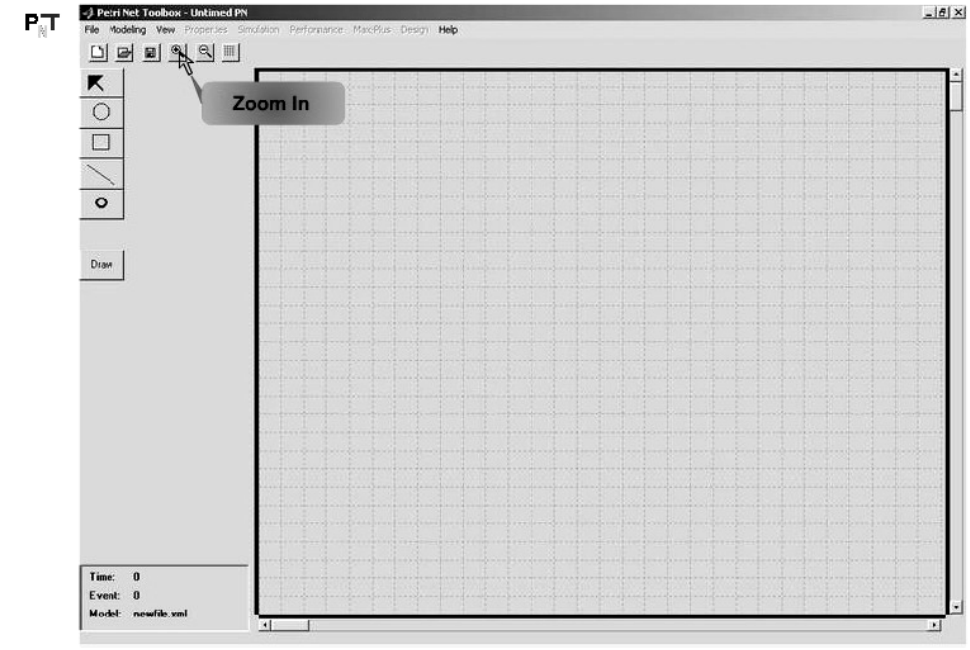
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

5. Show the grid lines



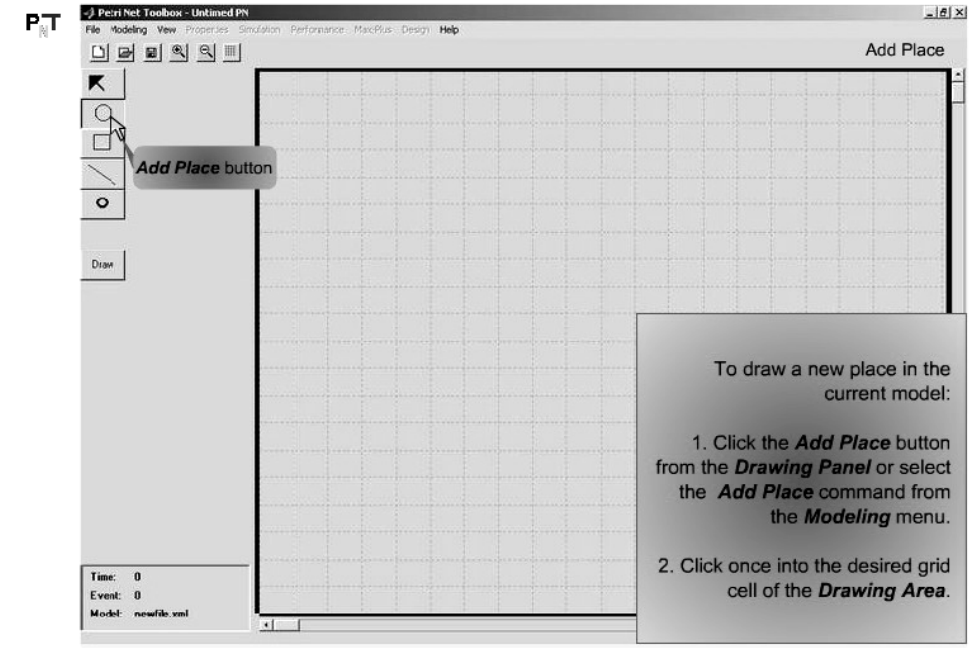
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

6. Use the ***Zoom In*** button



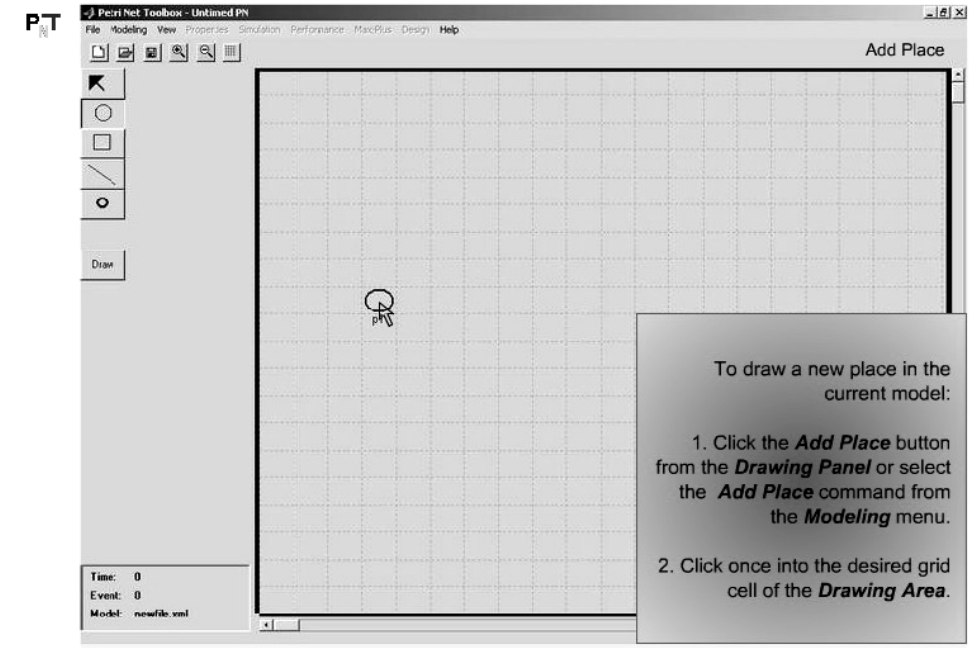
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

7. Draw a new place in the model – click the ***Add Place*** button



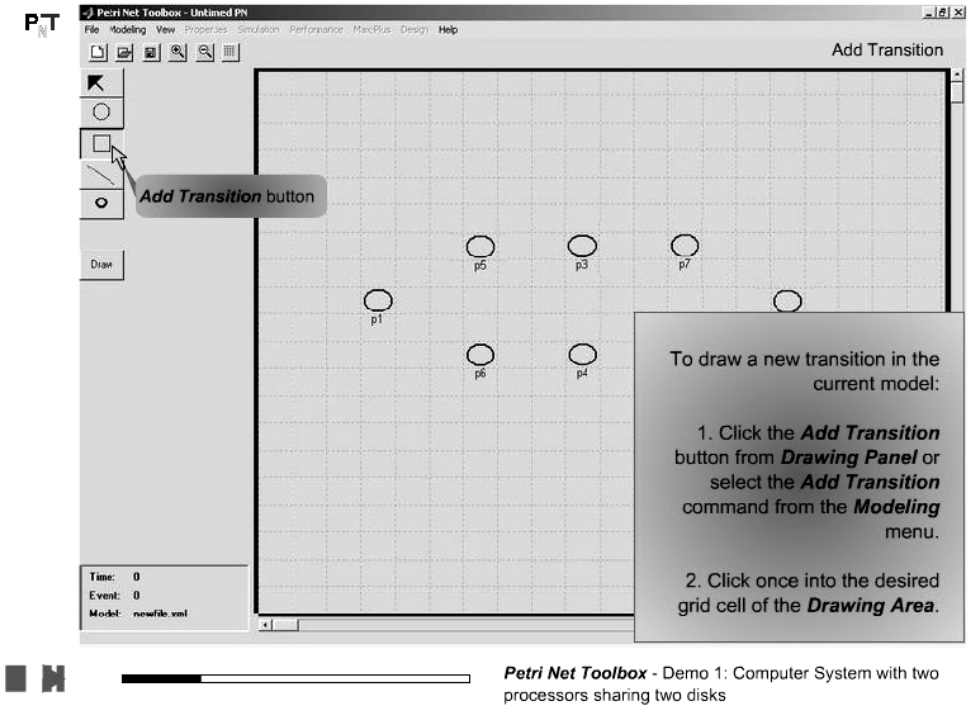
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

8. Click in the ***Drawing Area*** to add a new place to the model



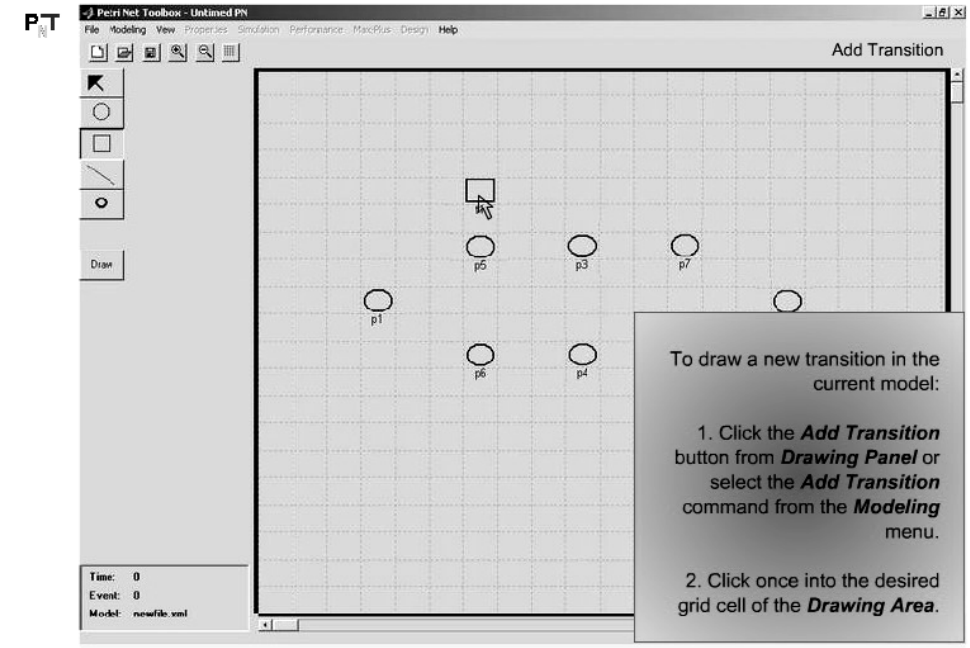
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

9. Draw a new transition in the model – click the ***Add Transition*** button



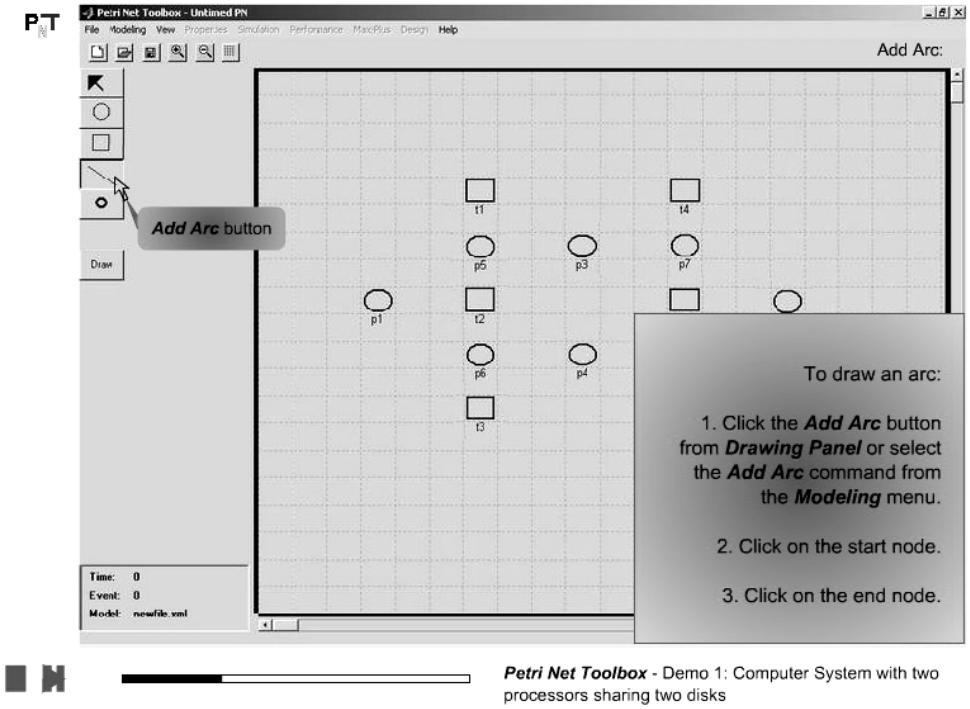
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

10. Click in the ***Drawing Area*** to add a new transition to the model



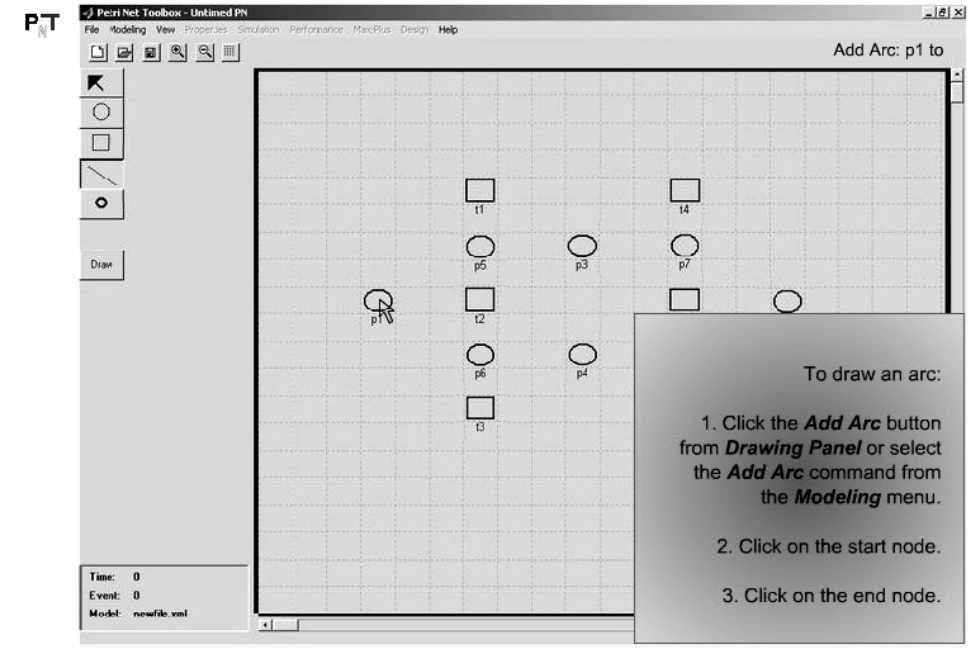
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

11. Draw a new arc in the model – click the ***Add Arc*** button



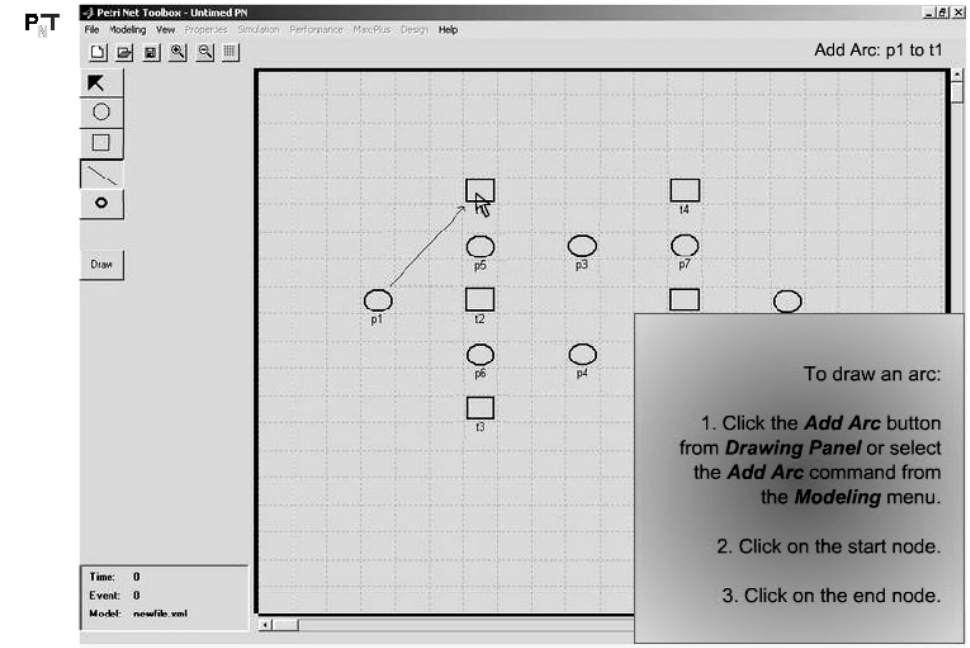
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

12. Click on the start node



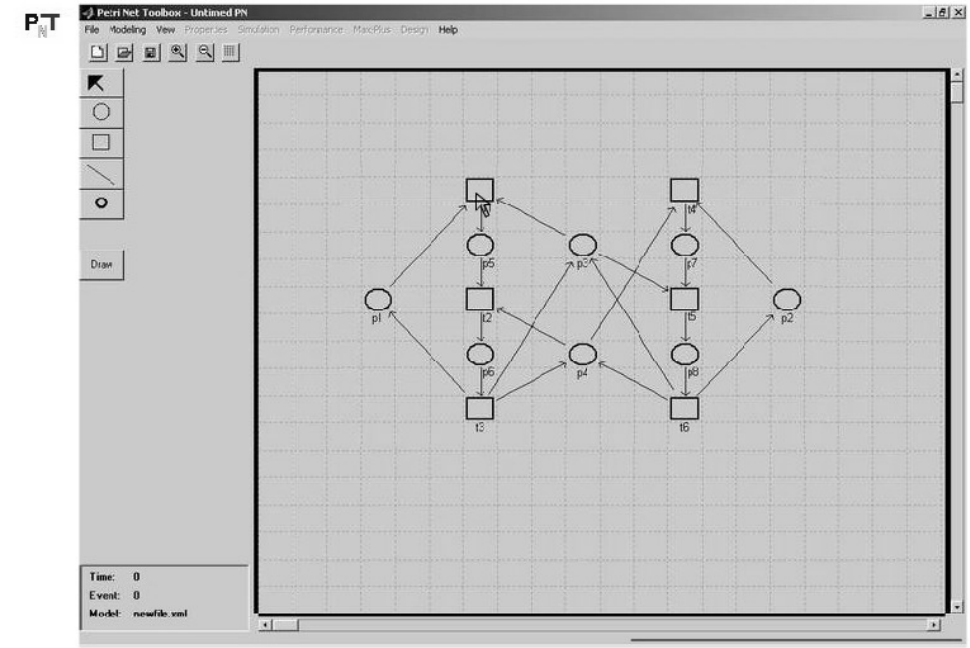
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

13. Click on the end node



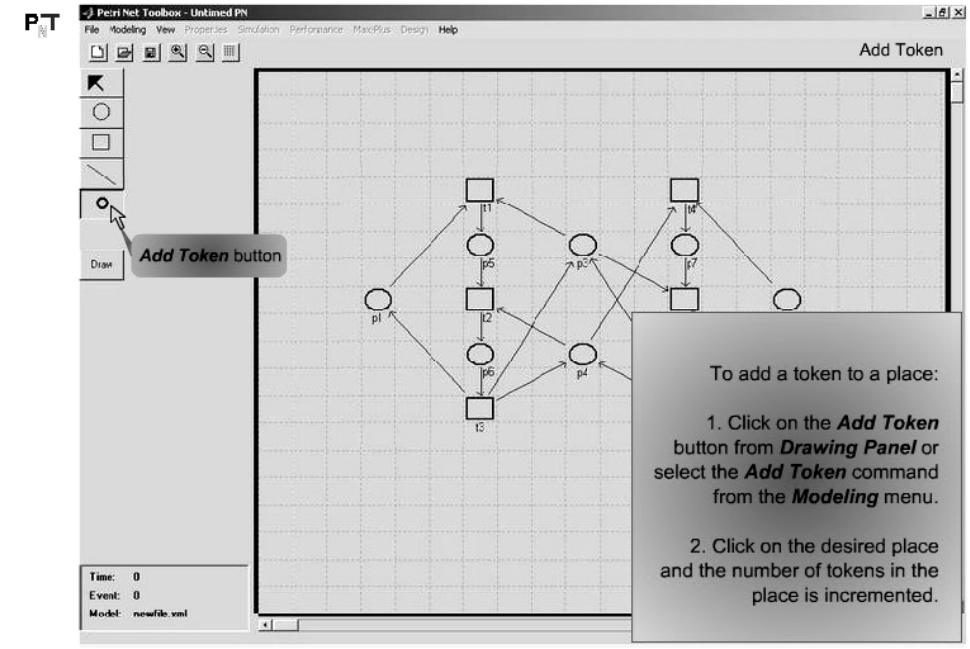
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

14. Inspect the topology of the constructed model



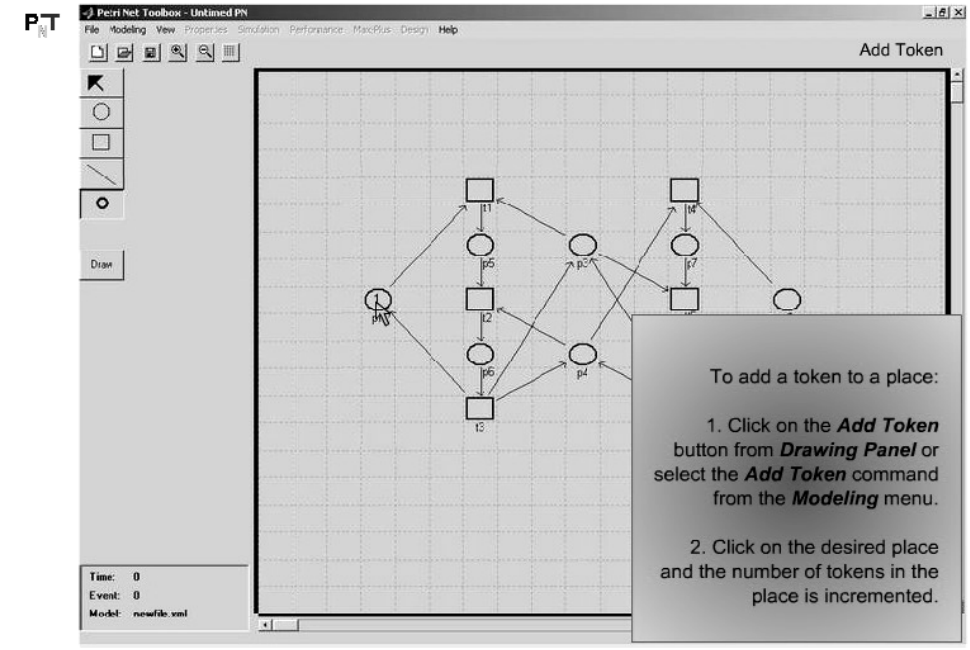
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

15. Add tokens to places – click the ***Add Token*** button



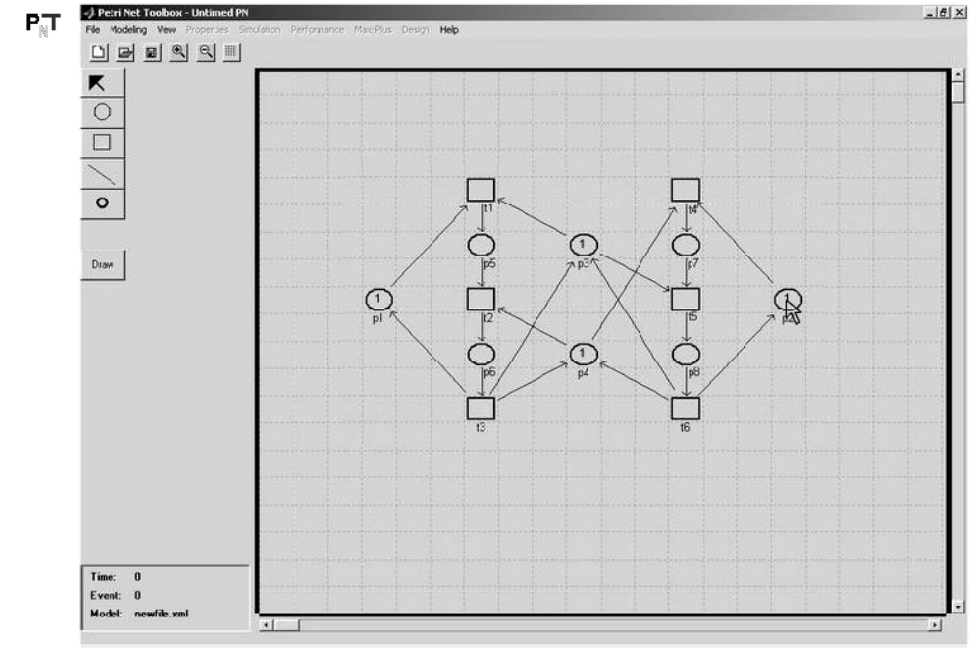
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

16. Click on the desired place to increase the number of tokens



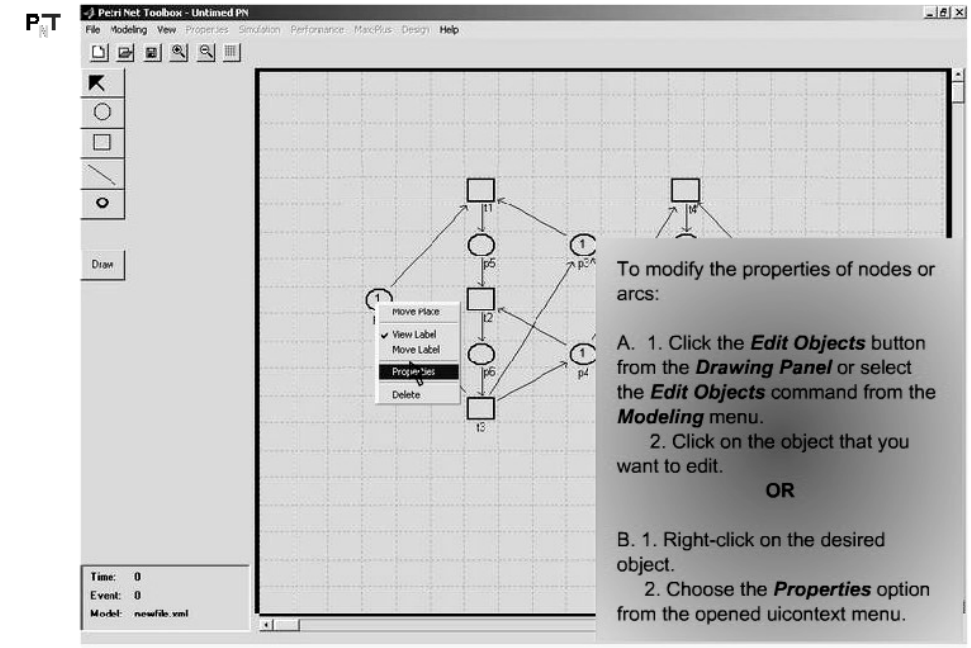
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

17. Inspect the final model (topology and initial marking)



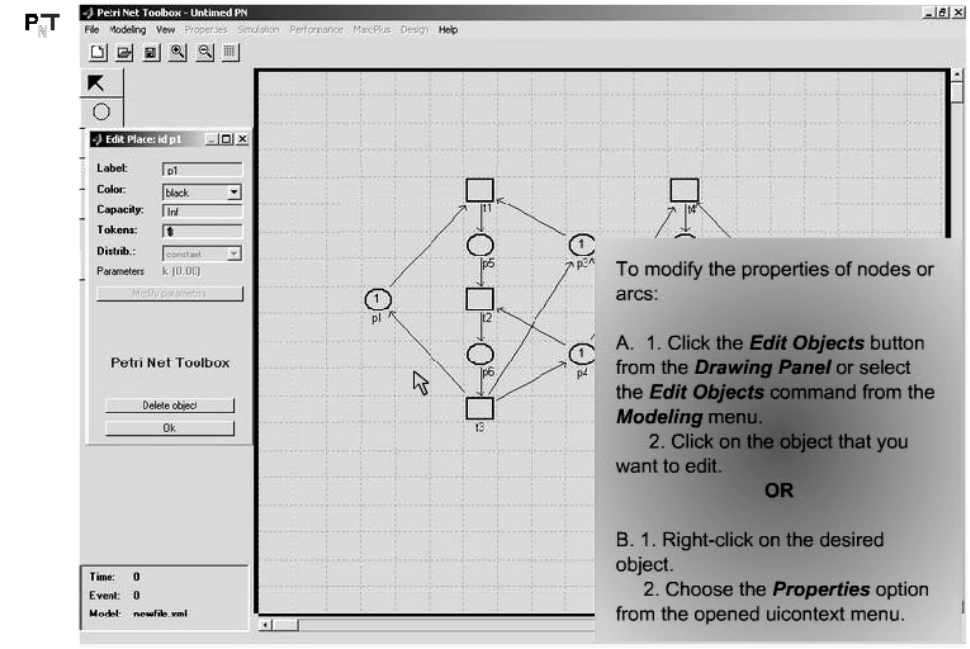
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

18. Access the properties of nodes or arcs



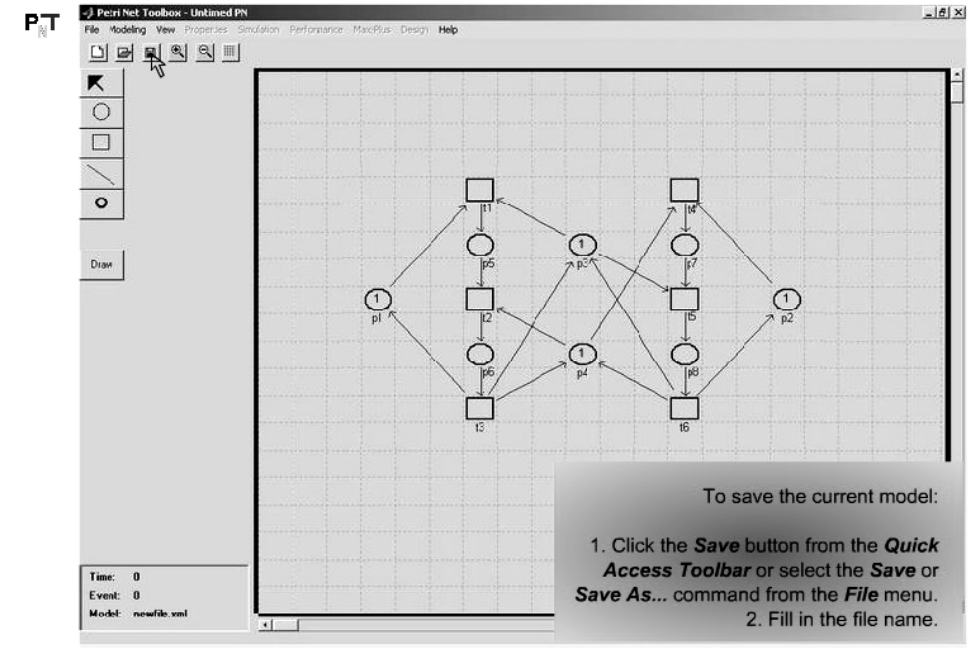
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

19. Change (if desired) properties in the corresponding ***Edit*** dialogue window



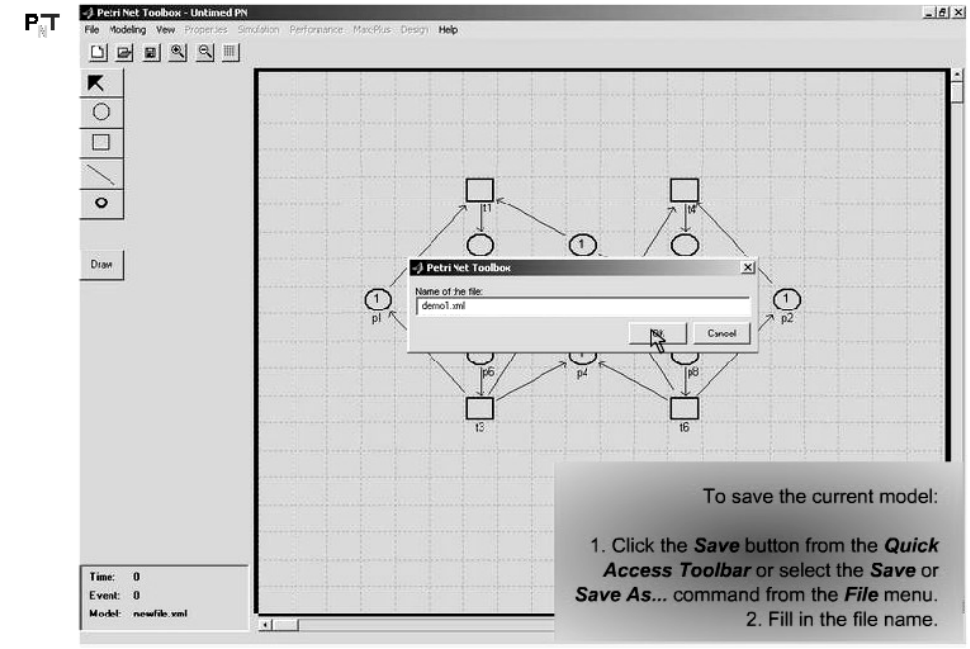
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

20. Save the current model



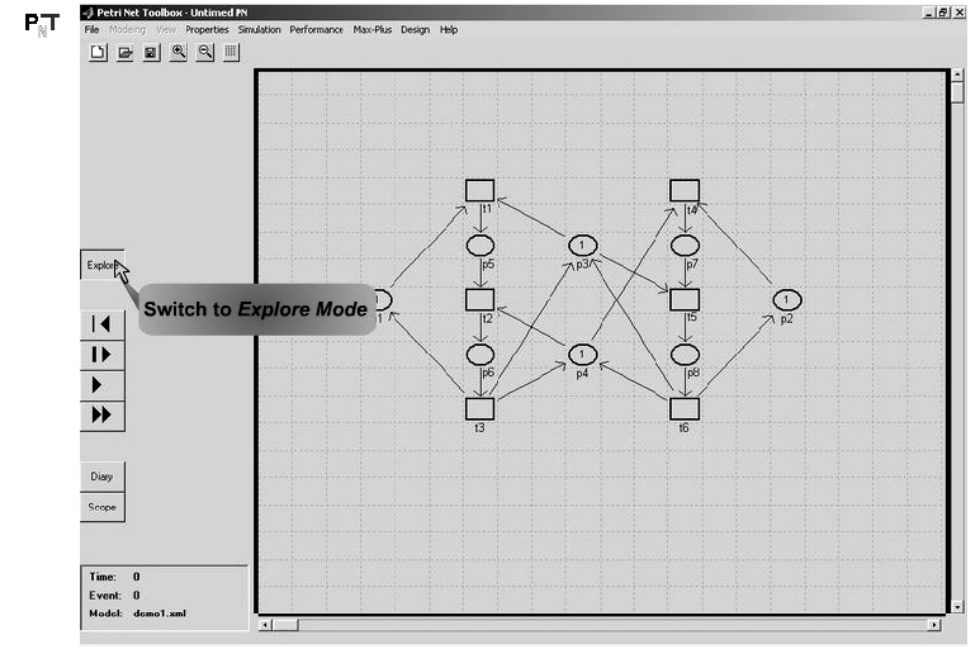
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

21. Fill in the file name



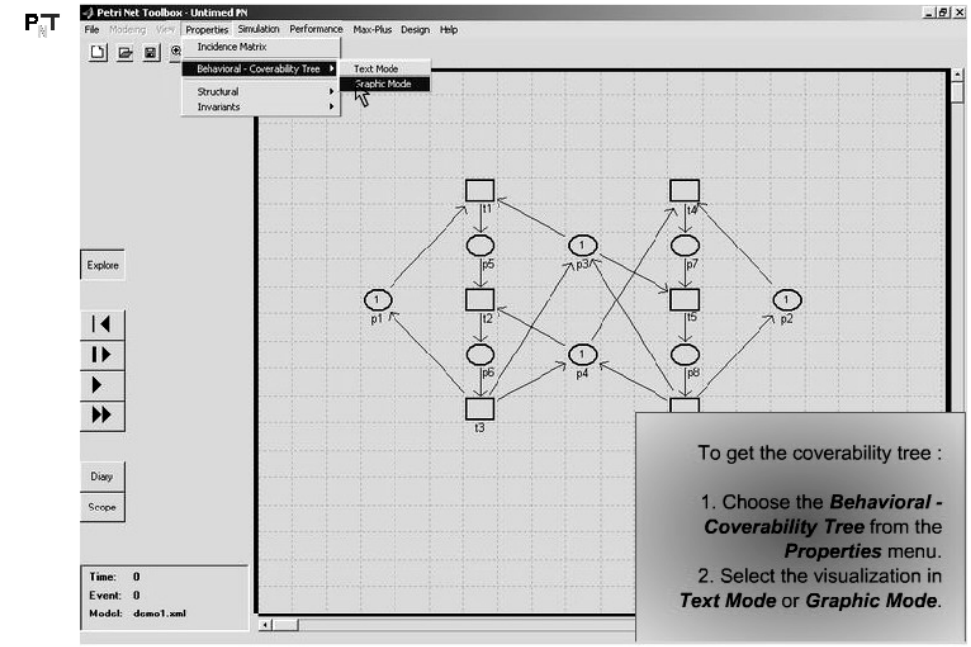
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

22. Switch from ***Drawing Mode*** to ***Explore Mode***



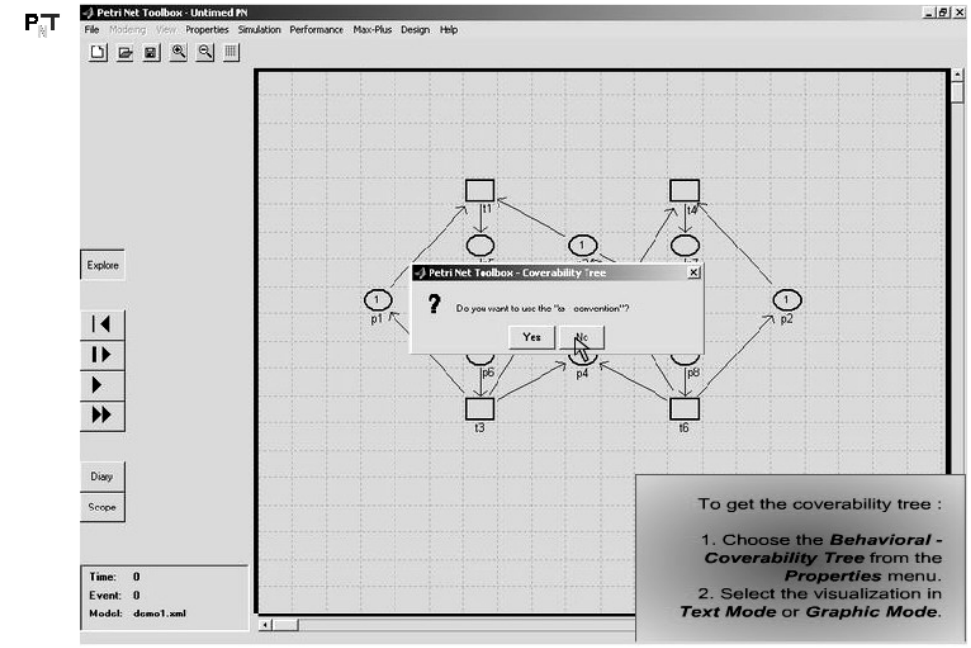
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

23. Construct the ***Coverability Tree*** of the PN model in ***Graphic Mode***



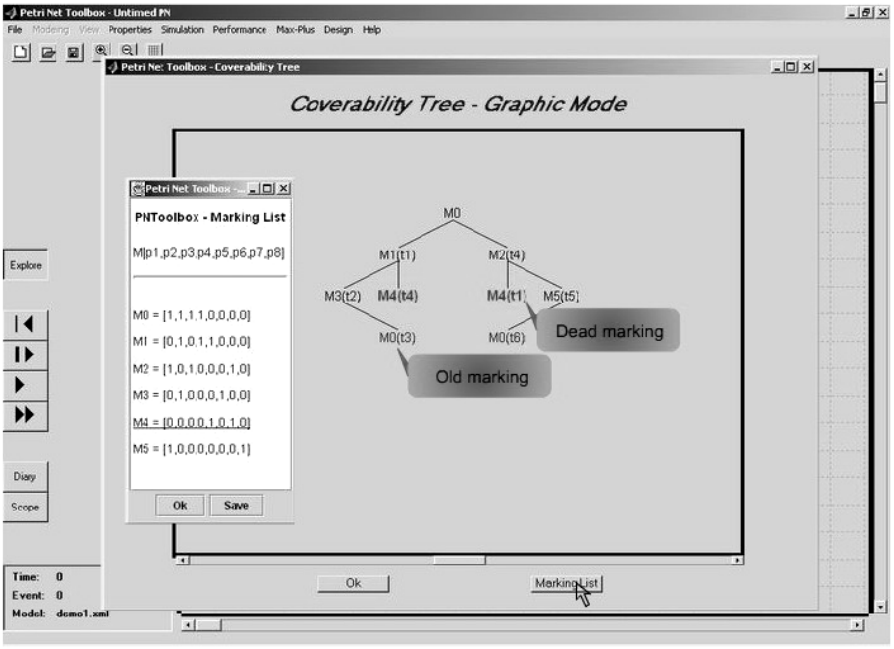
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

24. No need to use the "ω convention" in the analysis of this model



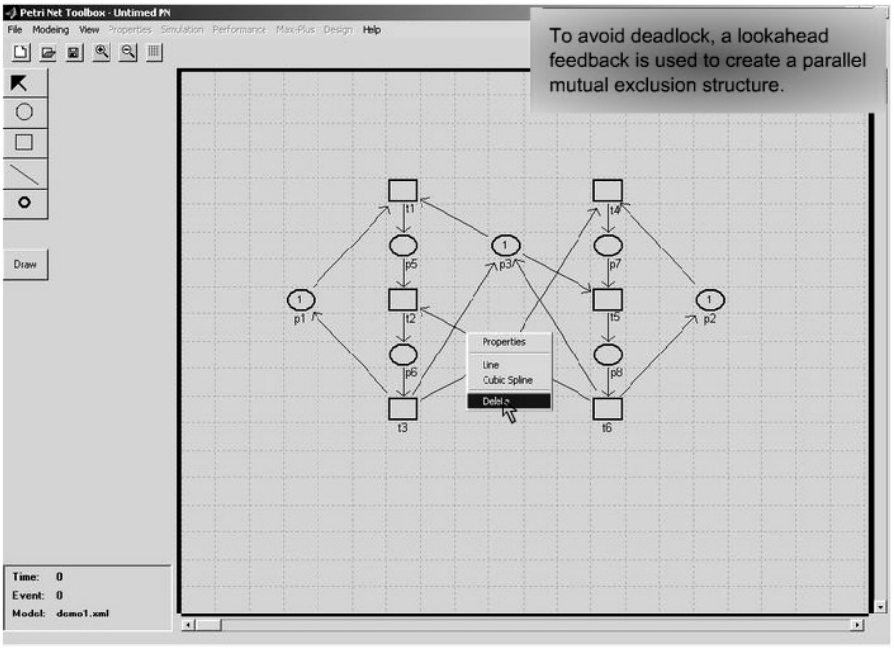
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

25. View the corresponding marking list. Notice the dead marking!



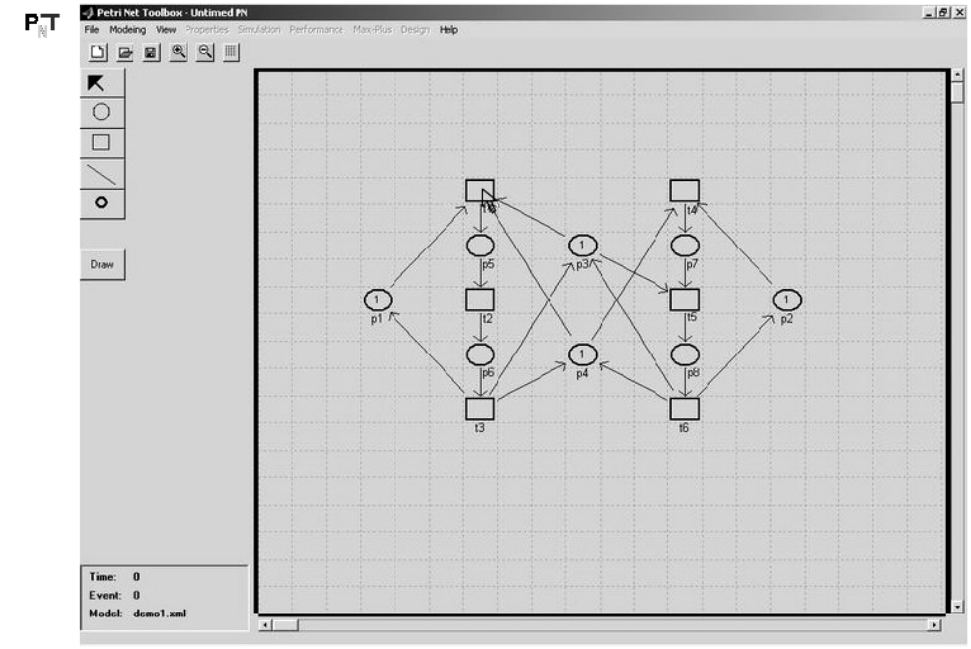
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

26. Modify the model to include lookahead feedback. Delete an arc



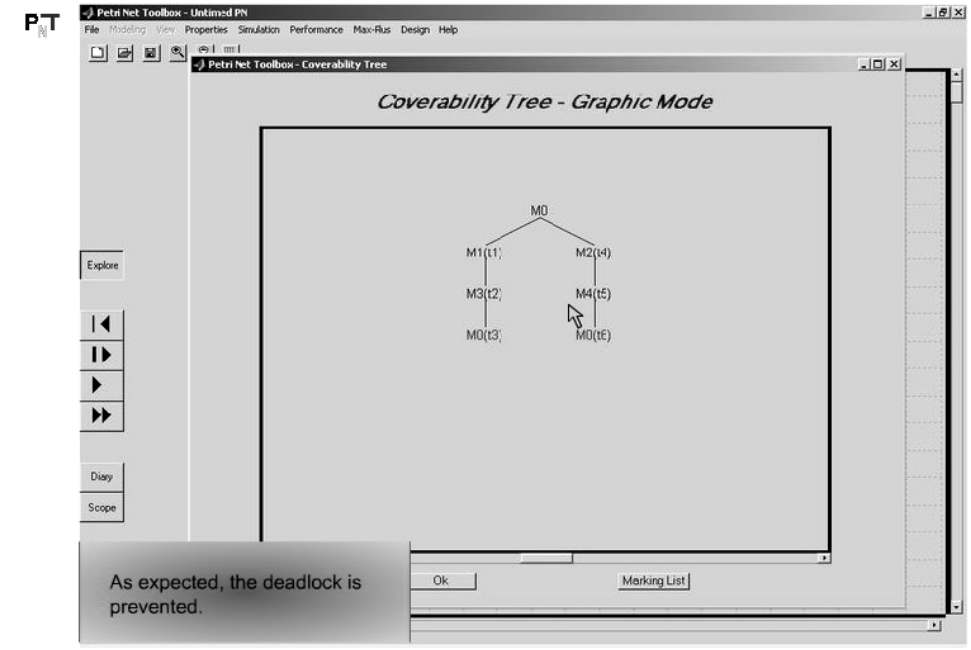
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

27. Draw a new arc



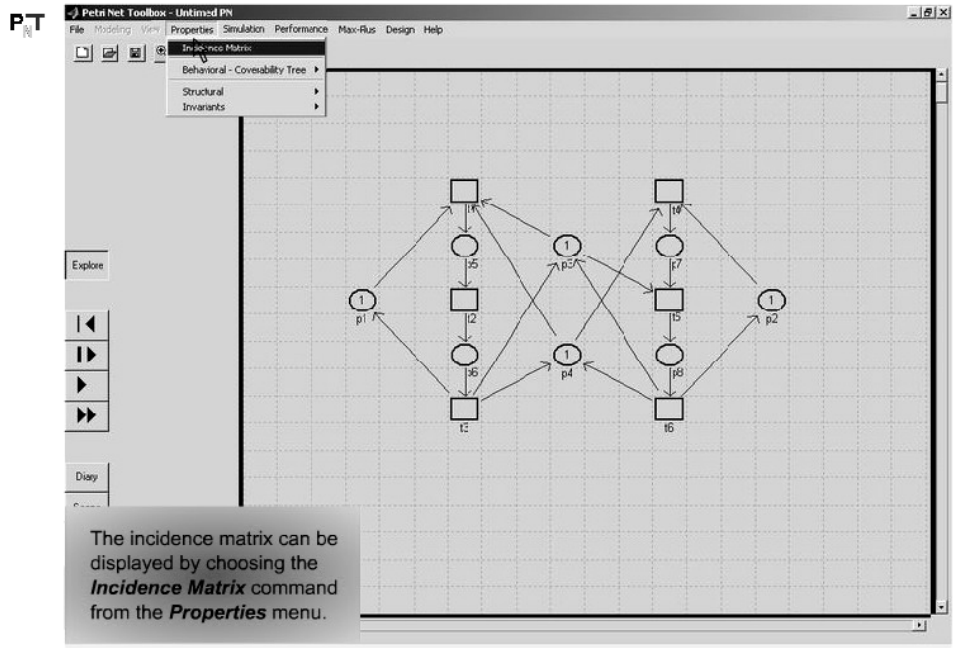
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

28. Inspect the coverability tree of the new model. Notice that deadlock is avoided!



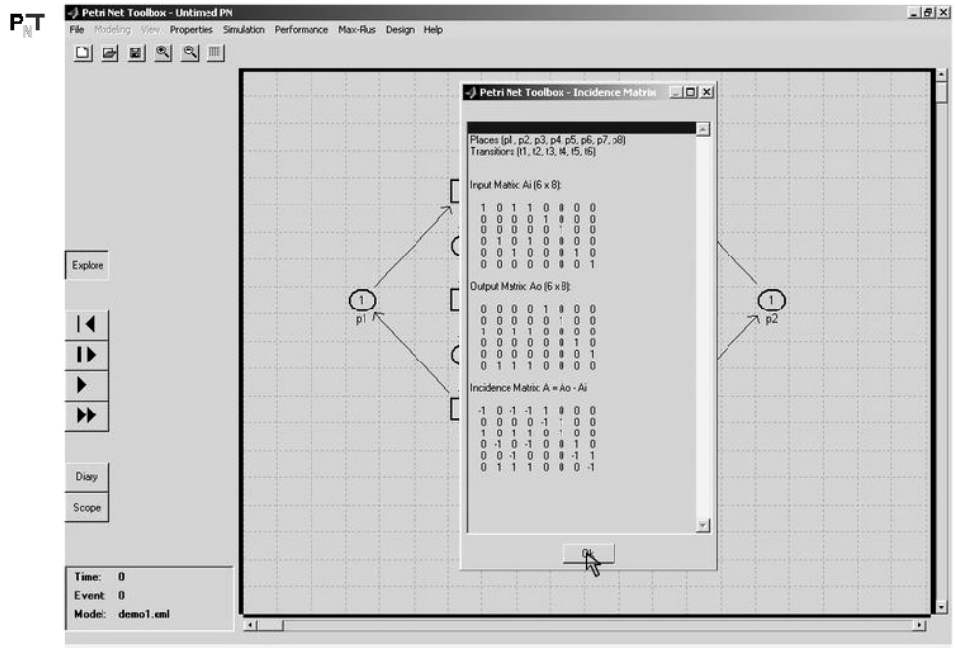
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

29. Display the ***Incidence Matrix*** of the PN model



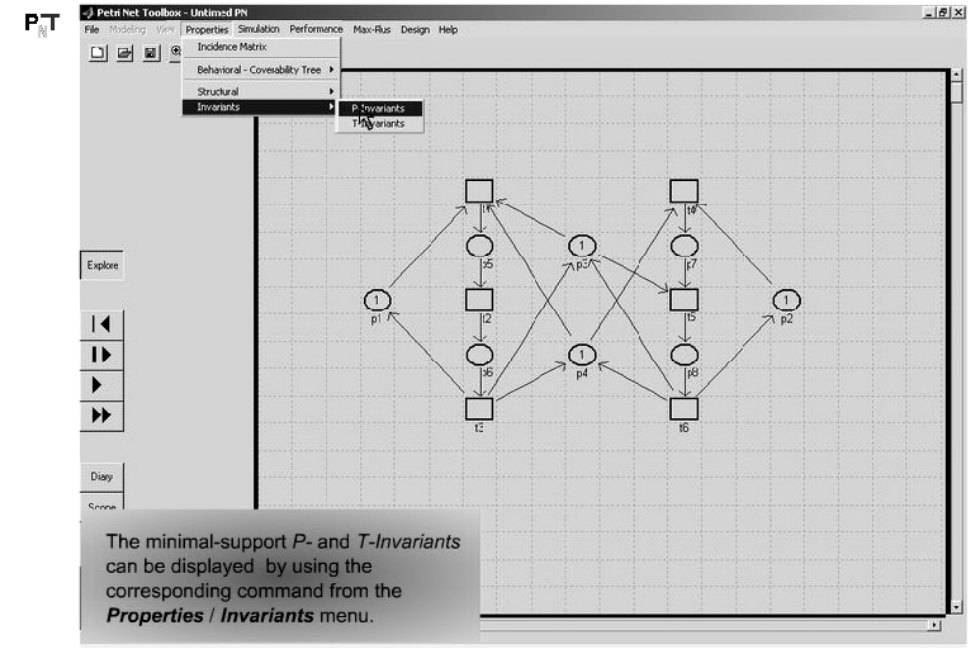
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

30. Visualize the input, output and incidence matrix



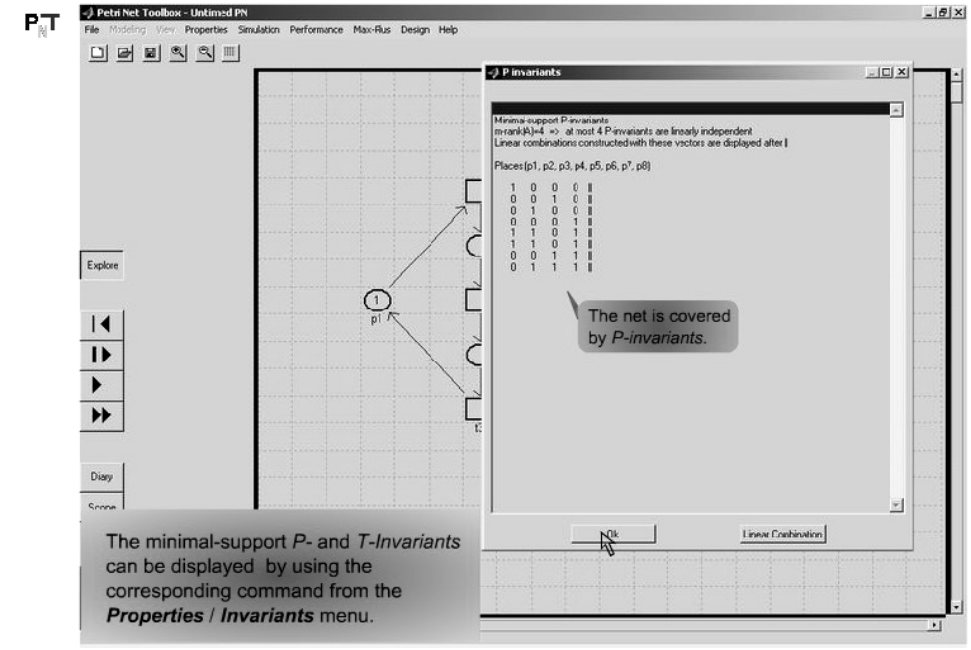
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

31. Inspect the minimal support ***P-invariants***



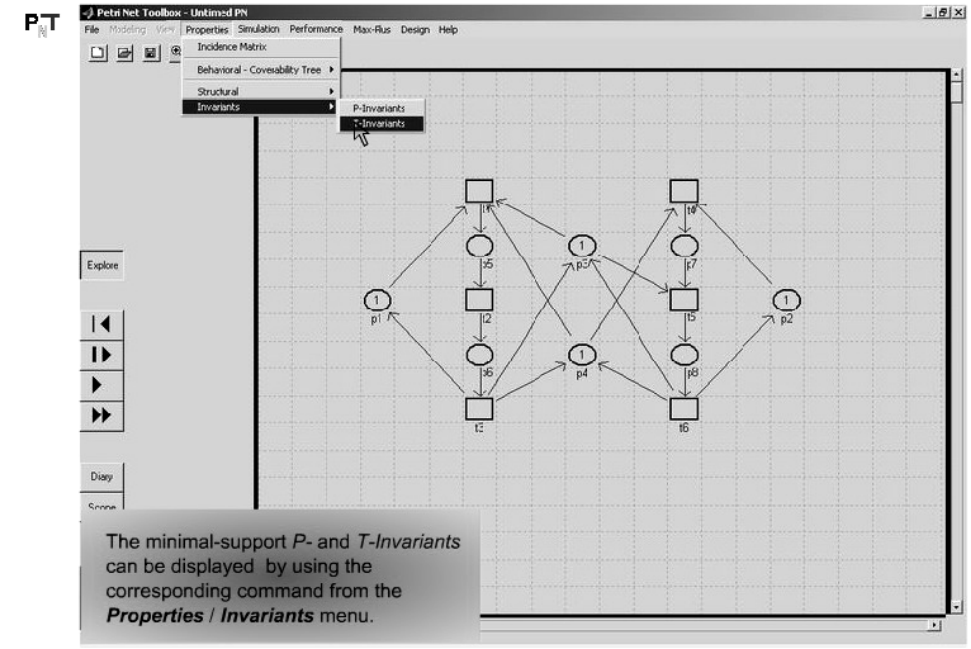
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

32. Notice that the PN model is covered by P-invariants



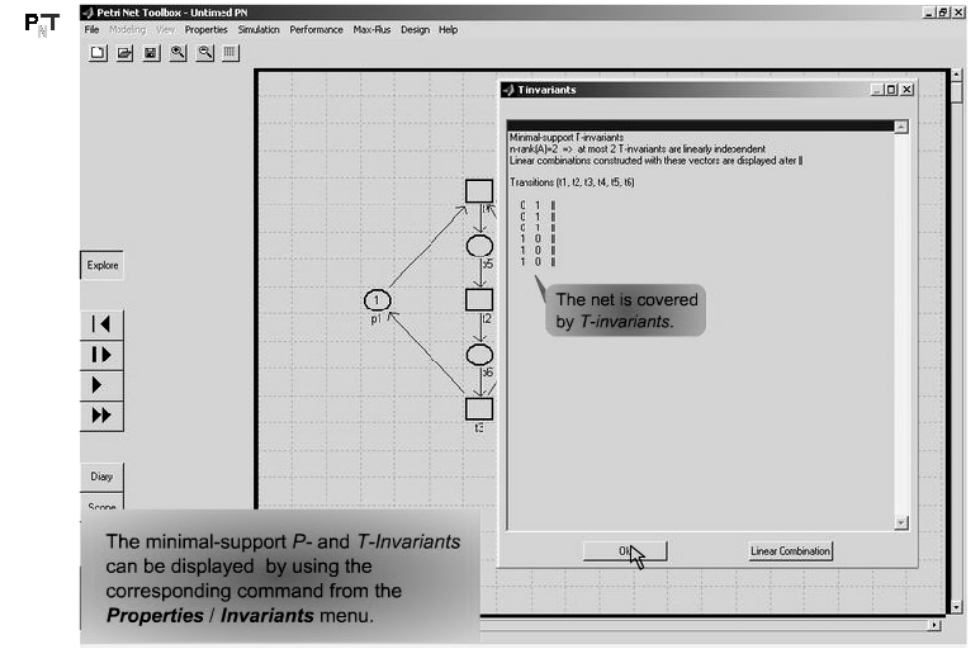
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

33. Inspect the minimal support ***T-invariants***



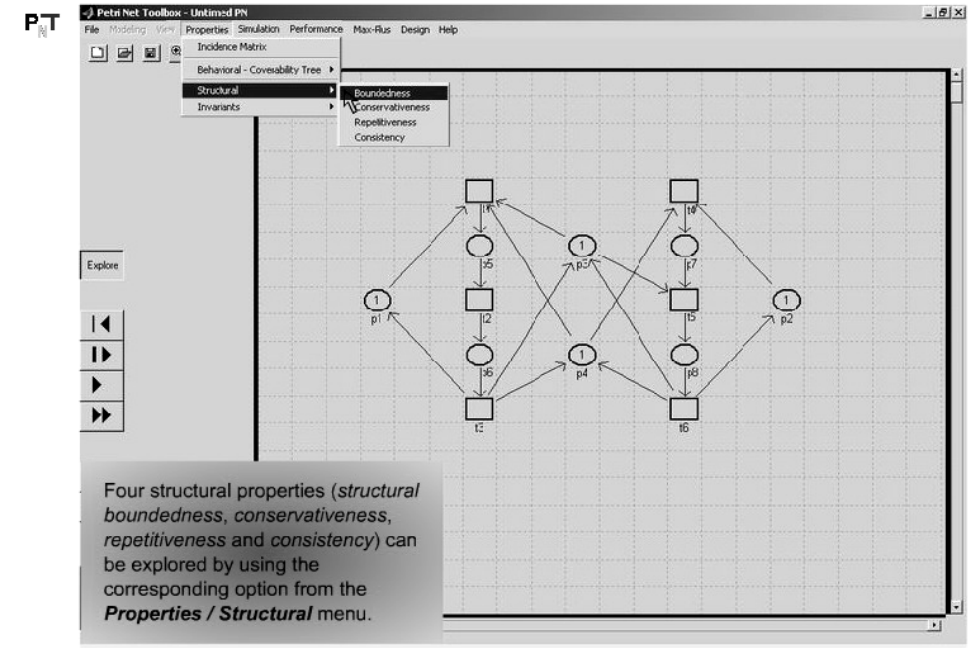
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

34. Notice that the PN model is covered by T-invariants



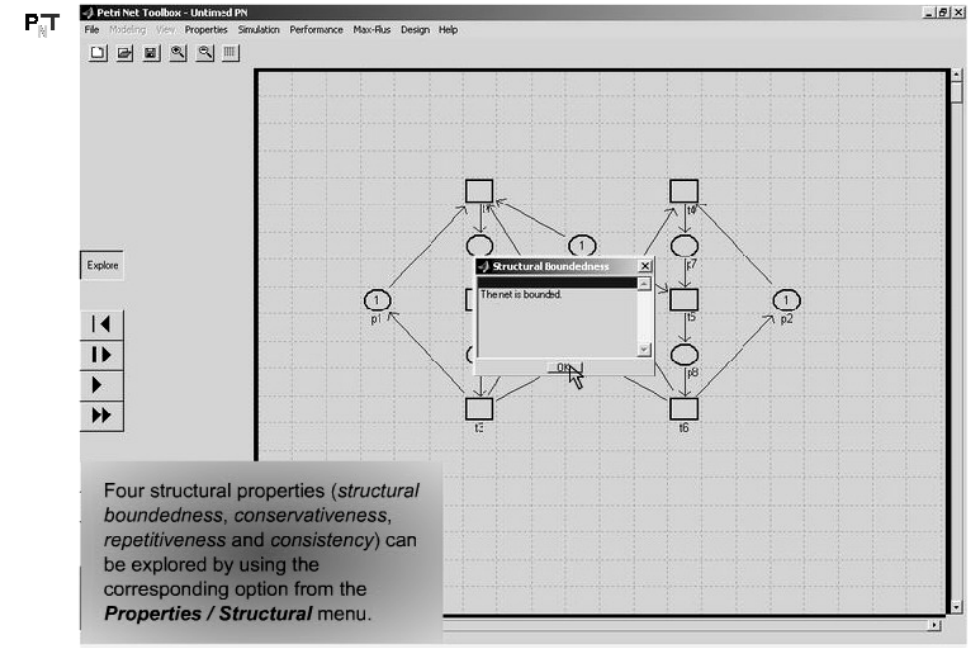
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

35. Study the ***Structural Properties*** of the PN model



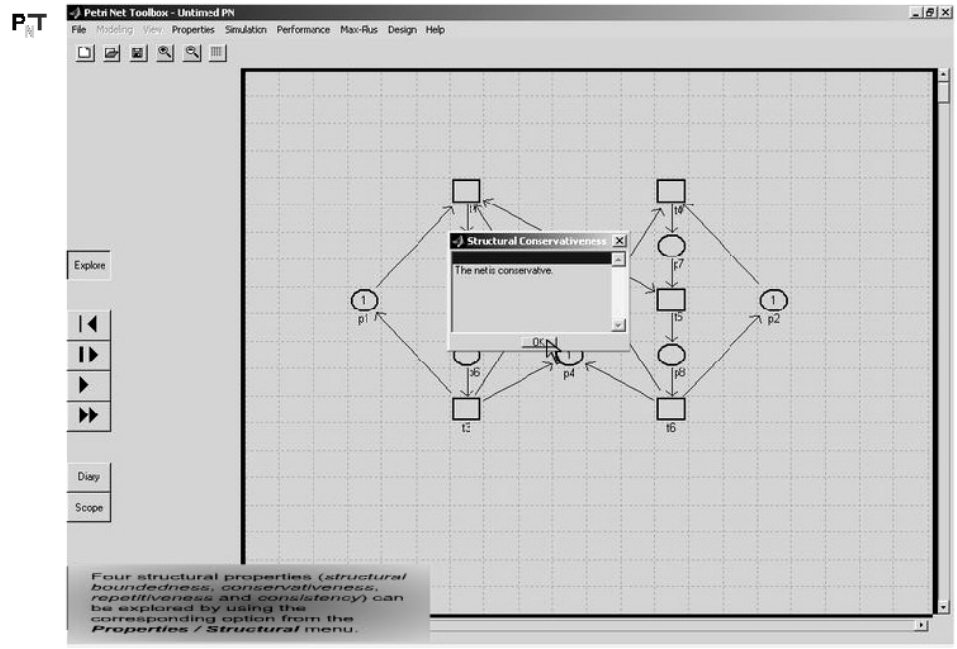
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

36. Notice that the net is structurally bounded



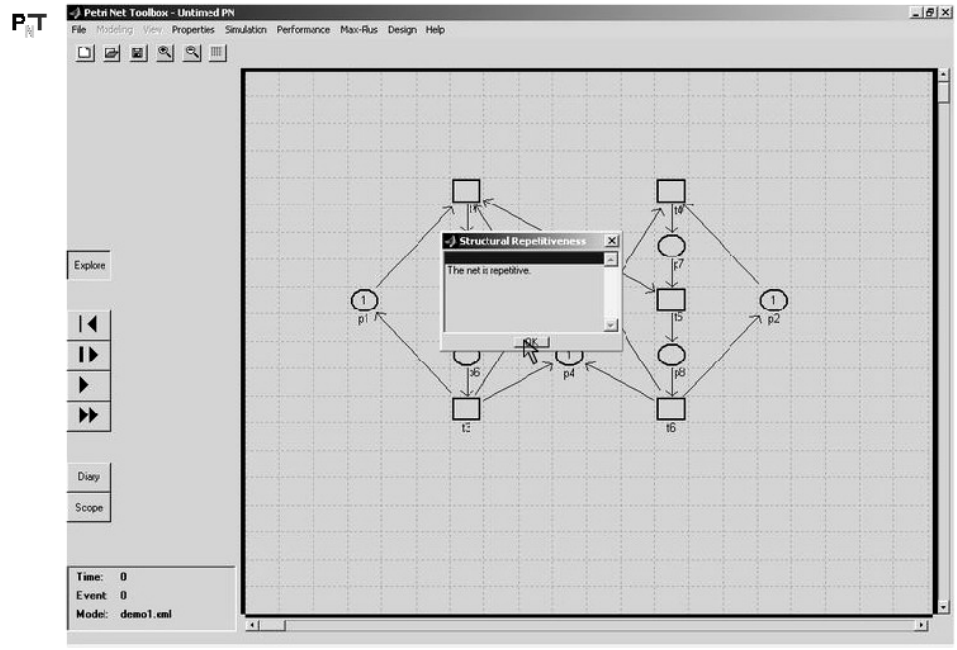
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

37. Notice that the net is conservative



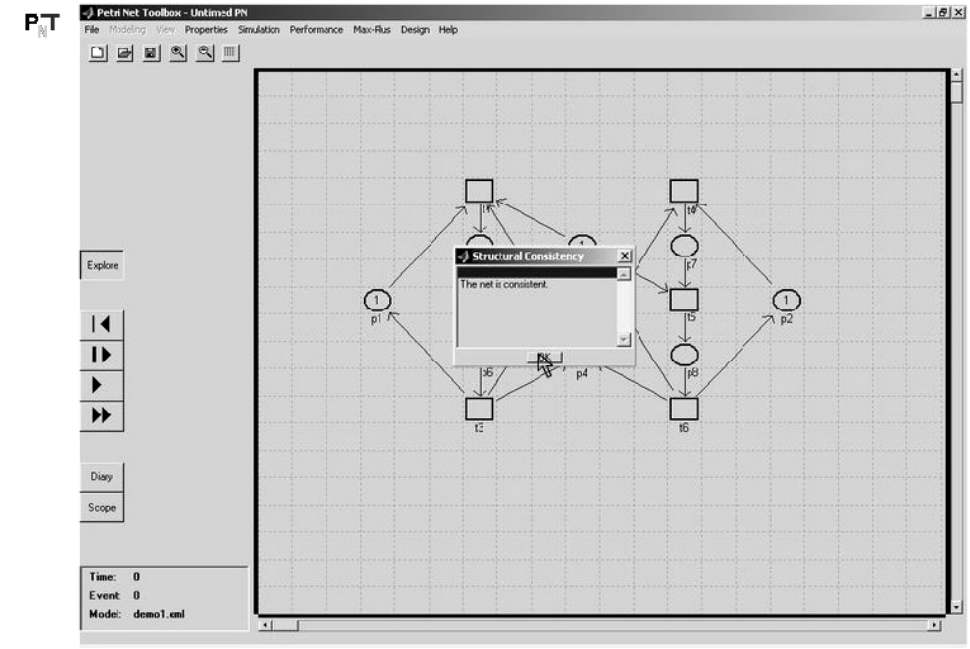
Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

38. Notice that the net is repetitive



Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

39. Notice that the net is repetitive



Petri Net Toolbox - Demo 1: Computer System with two processors sharing two disks

Demo 2

FLEXIBLE MANUFACTURING SYSTEM

We consider a *flexible manufacturing system* (FMS) that consists of two different machines (a lathe (M1) and a drilling machine (M2)), a robot and a buffer with two slots between the two machines. 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, the durations for transport operations and for releasing the pallets are known. At the initial moment, all resources are idle.

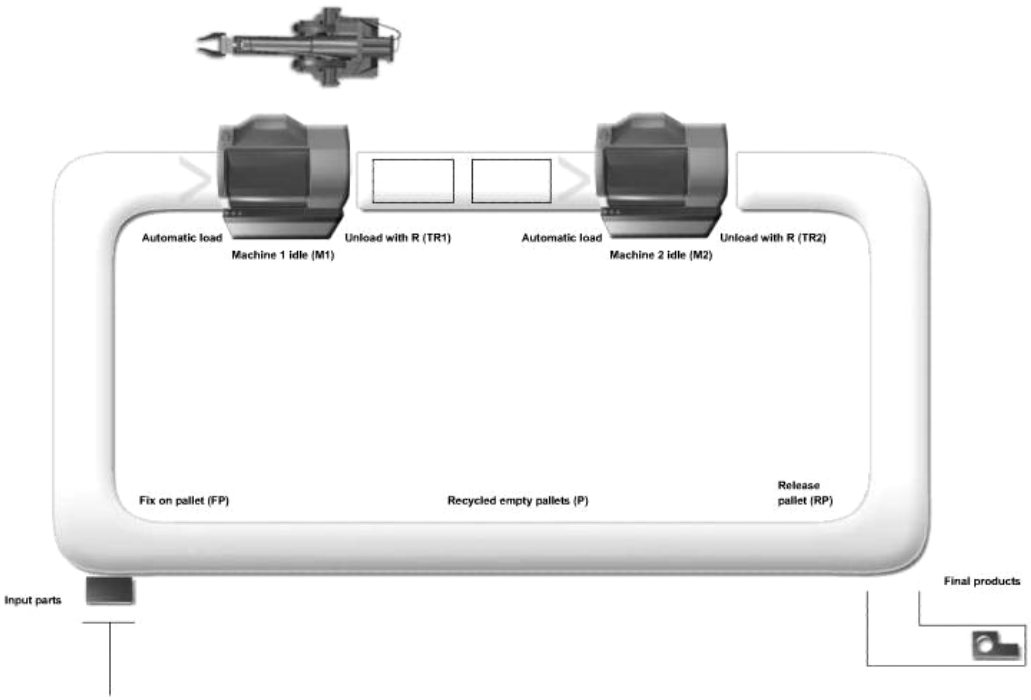
The current demonstration illustrates the usage of the ***PN Toolbox*** in the design of this FMS. The purpose is to find the optimal number of pallets and the optimal duration (considered constant but unknown) for releasing and recycling a pallet so as to ensure the best value for the mean production cycle time.

This demo illustrates:

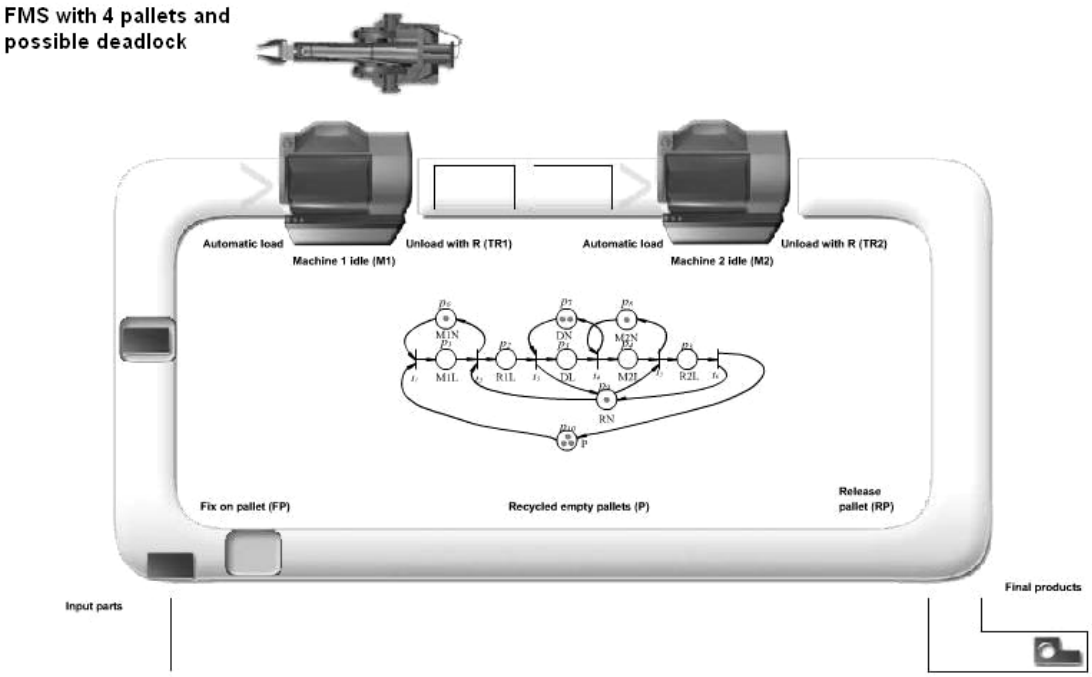
- the construction of a P-timed Petri net model
- the analysis of deadlock (via simulation)
- the prevention of deadlock by limiting the number of pallets
- the analysis of time-dependent performance indices
- the study of a performance index depending on two design parameters

D2.1. Illustration of the Physical System

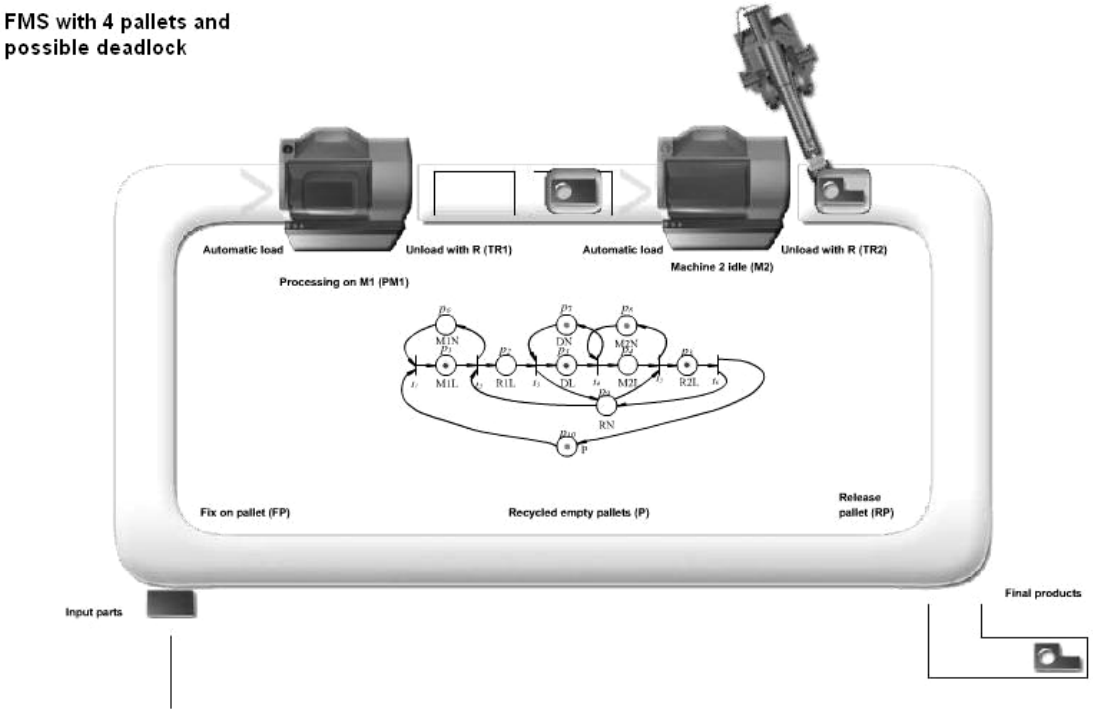
1. The flexible manufacturing system with a shared resource (namely the robot)



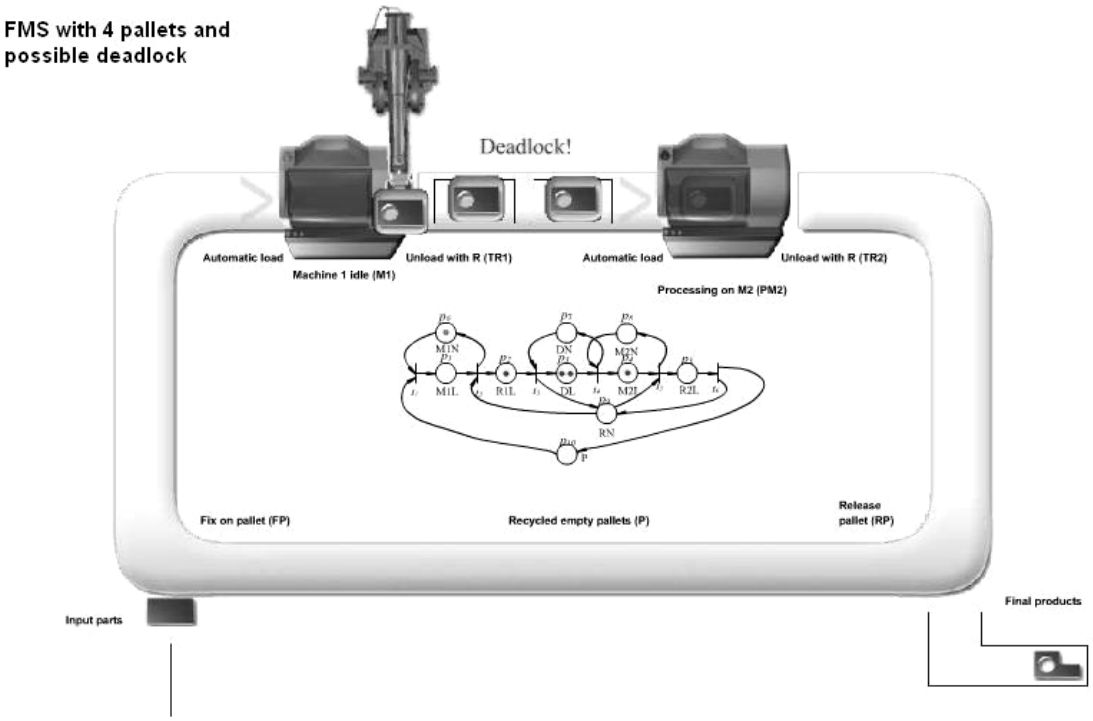
2. Model of the system that allows the analysis of deadlock



3. The robot transports a part to the buffer and both machines are idle

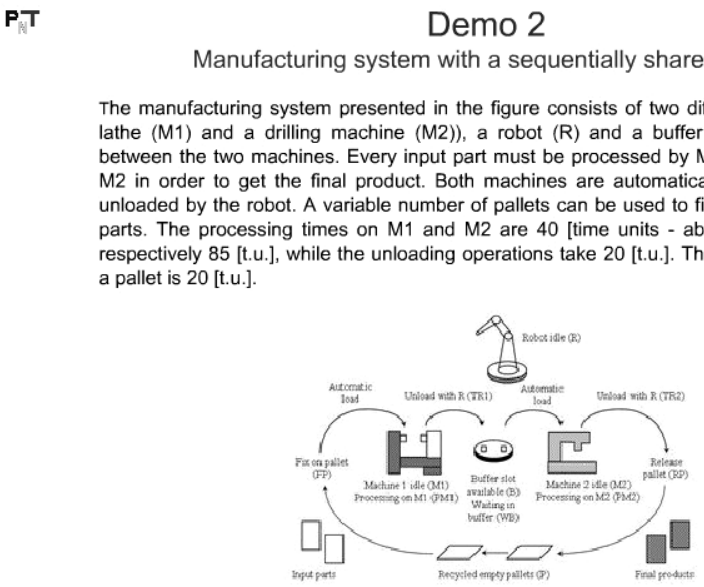


4. Deadlock situation – the buffer is full, machine M2 is occupied and the robot cannot be released

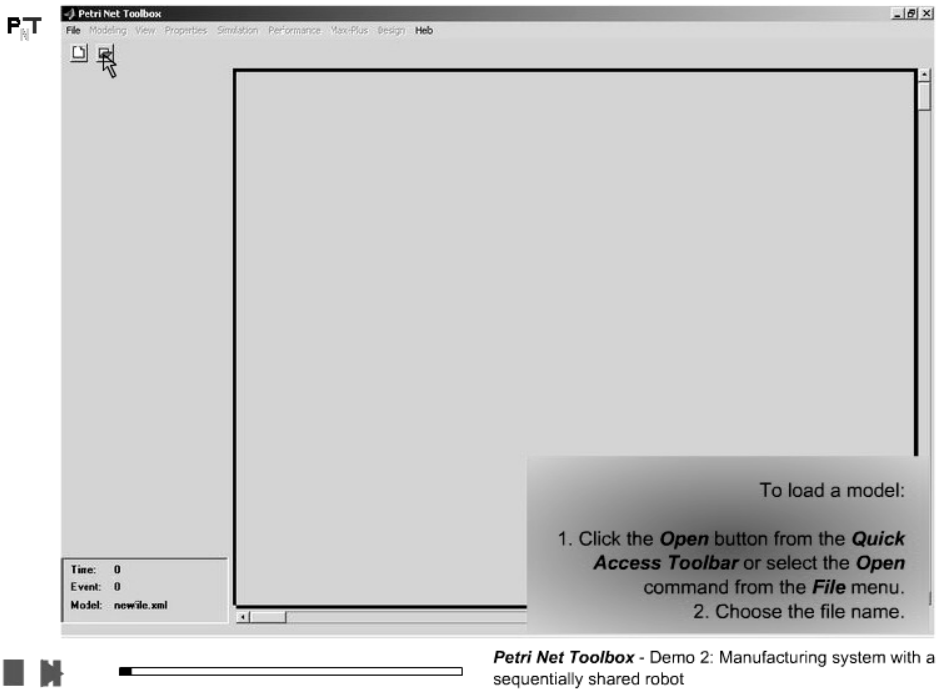


D2.2. Usage of the PN Toolbox for System Modeling and Analysis

1. Problem description

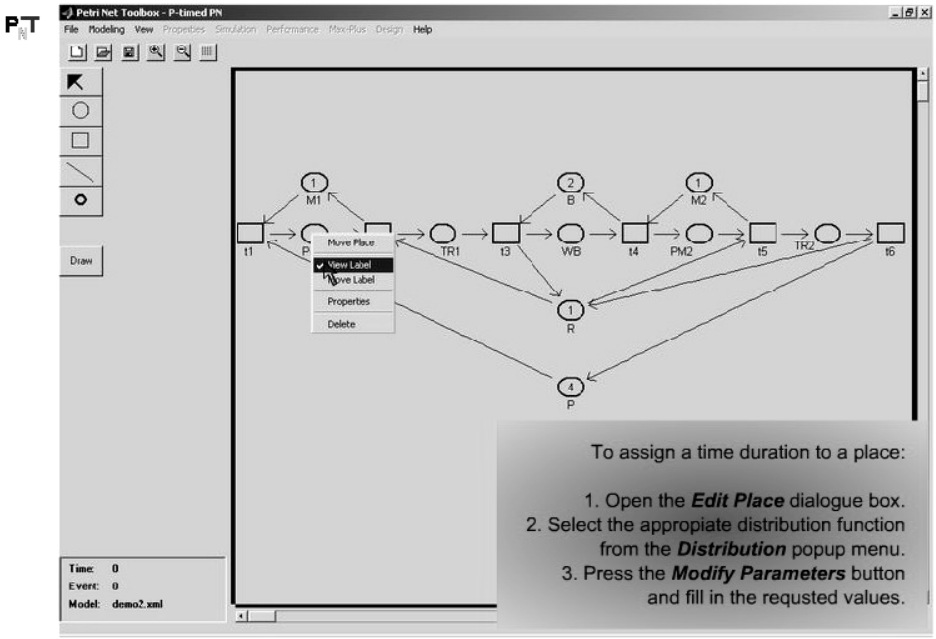


2. Open the model previously saved on the hard-disk



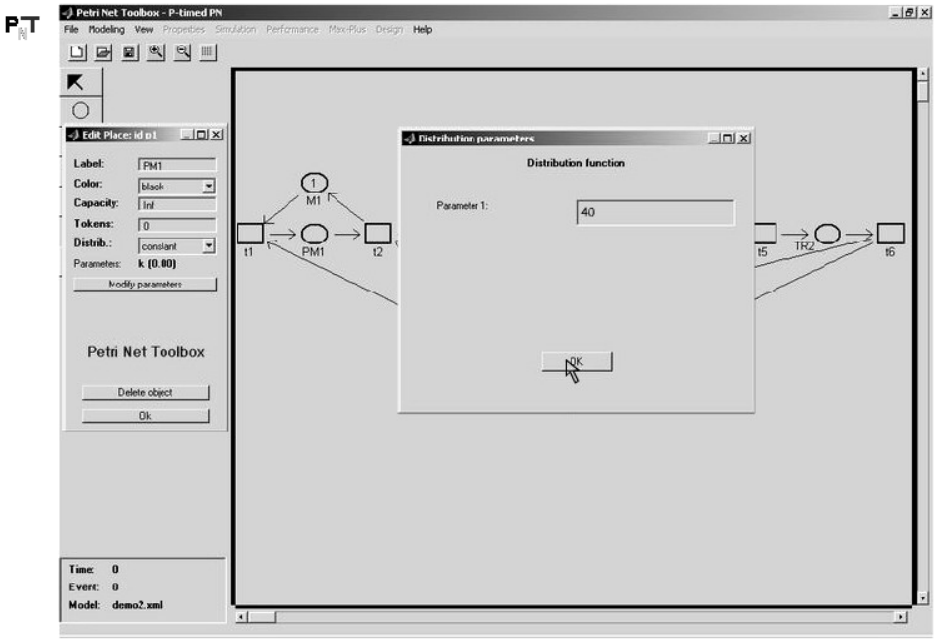
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

3. Assign a time duration to a place



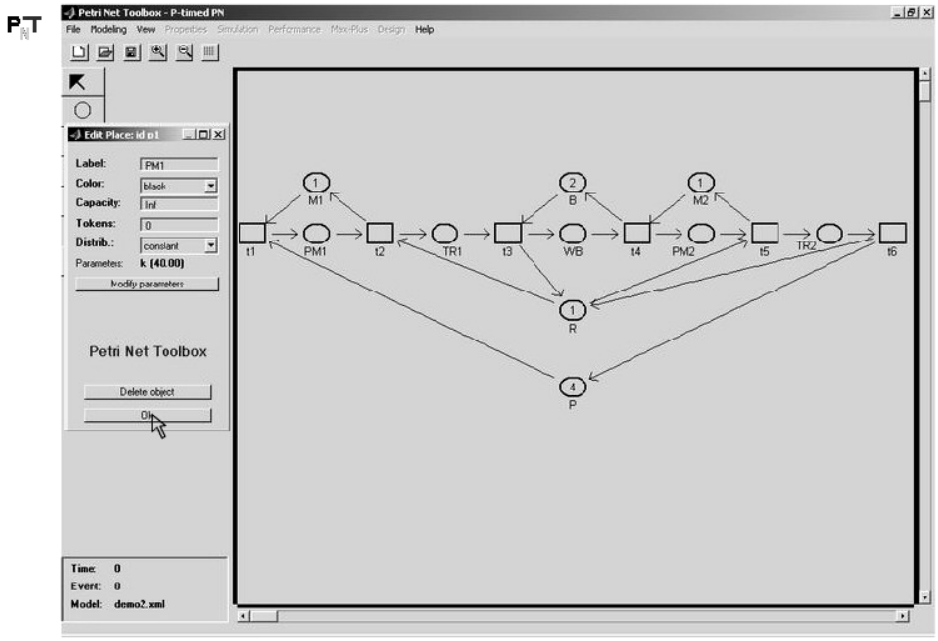
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

4. Select the desired time-distribution and set its parameters



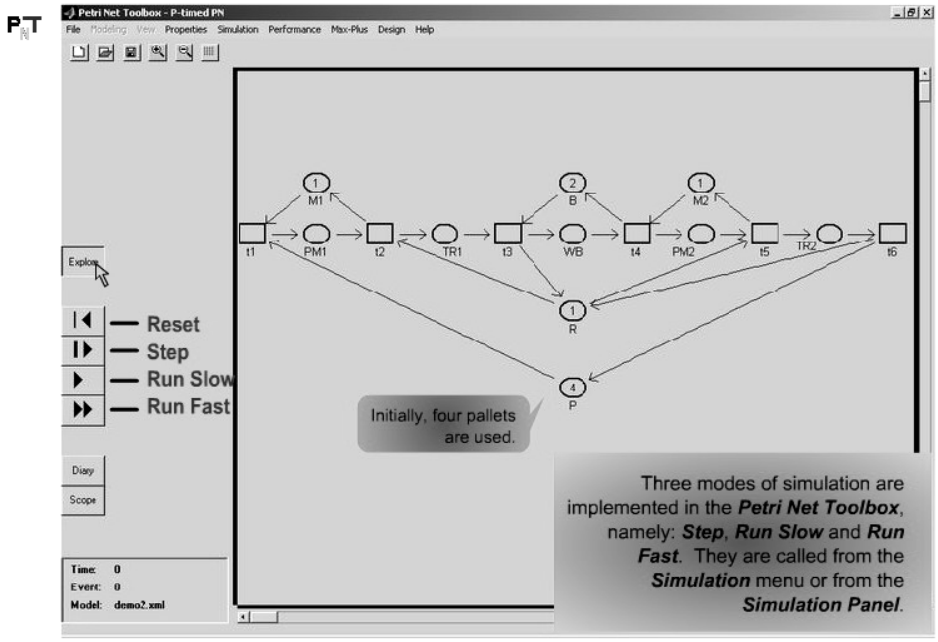
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

5. Click the ***OK*** button to close the ***Edit Place*** dialogue window



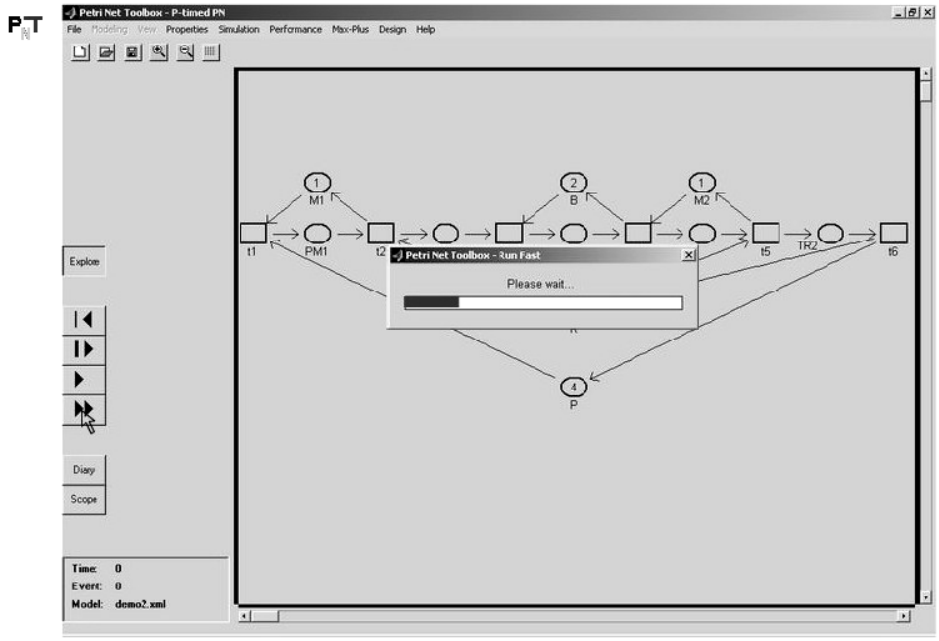
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

6. View the significance of the simulation buttons



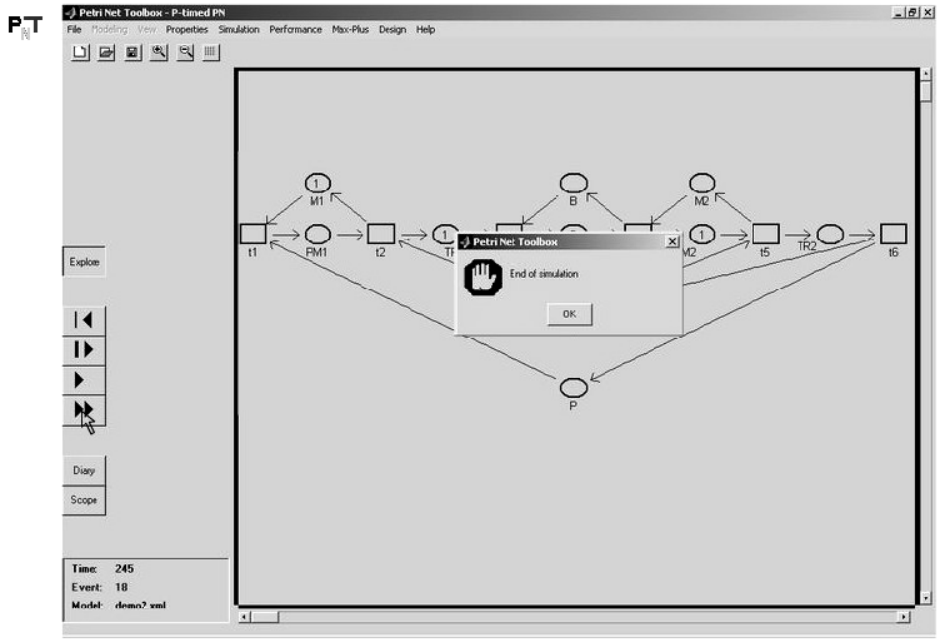
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

7. Run a simulation experiment in ***Run Fast*** mode



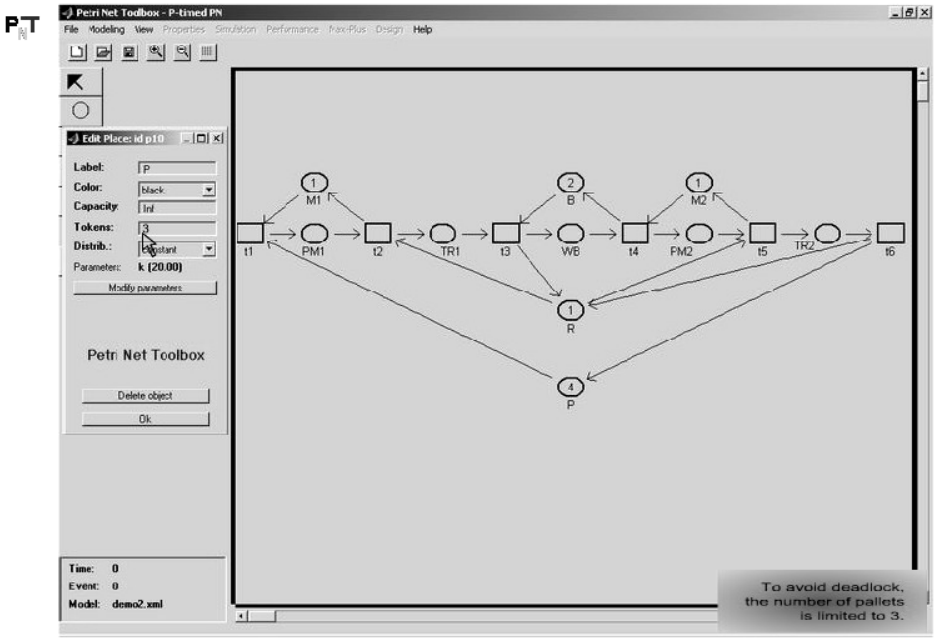
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

8. Deadlock marking in the PN model



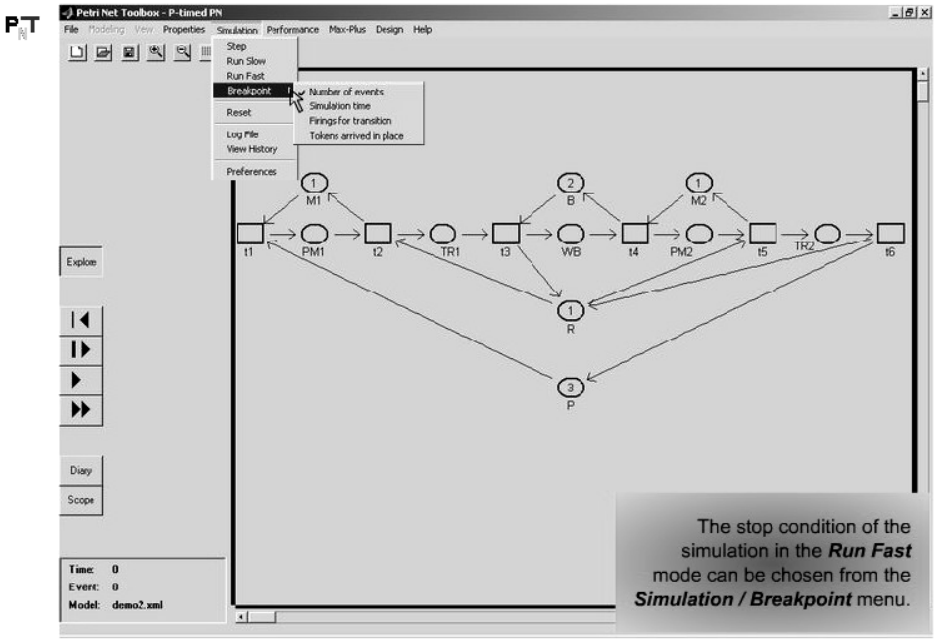
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

9. To avoid deadlock reduce the number of pallets in the FMS



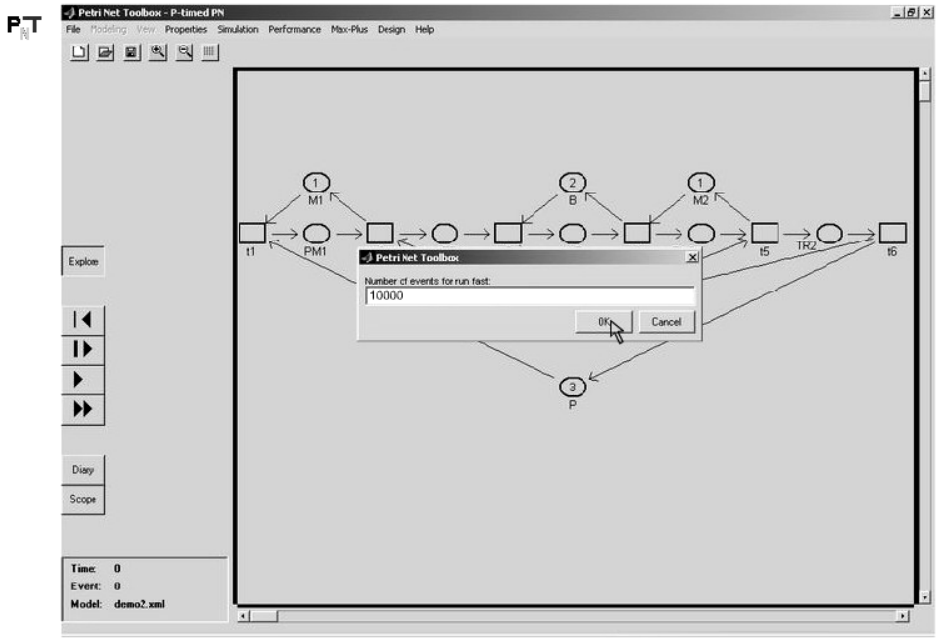
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

10. Set a ***Breakpoint*** for simulation on a given ***Number of Events***



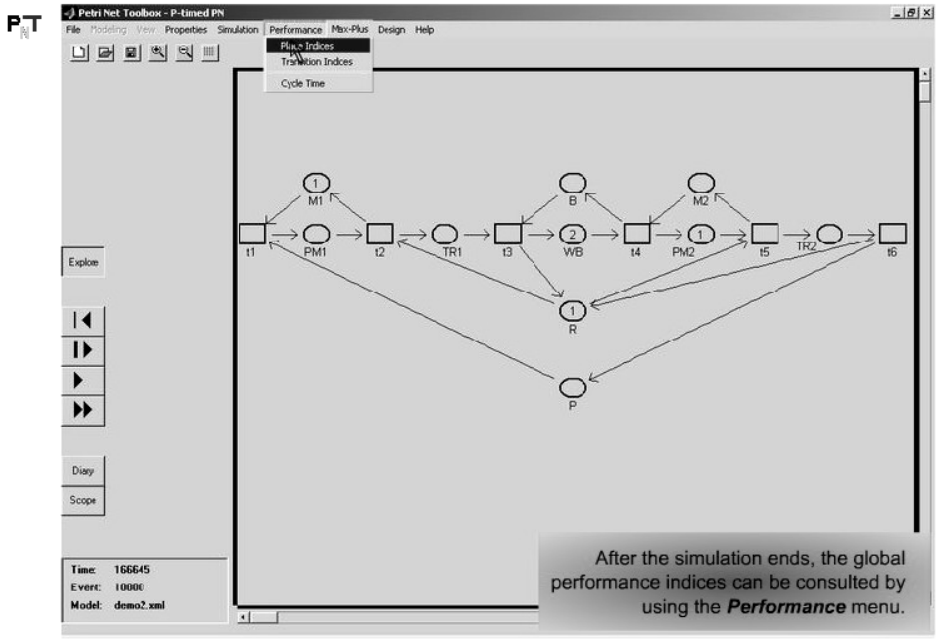
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

11. Fill in the dialogue box the desired number of events



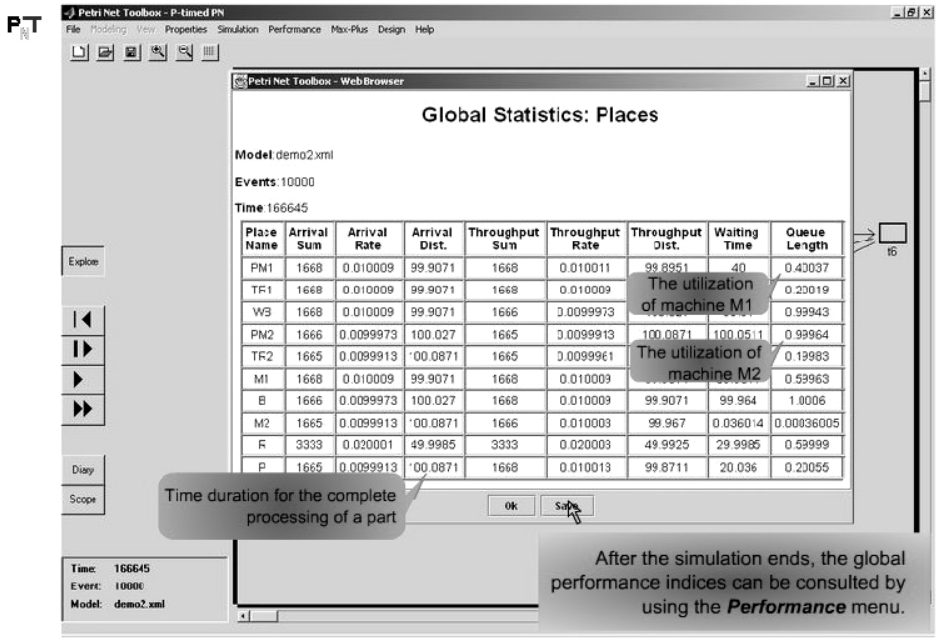
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

12. After the end of simulation inspect the global ***Place Indices***



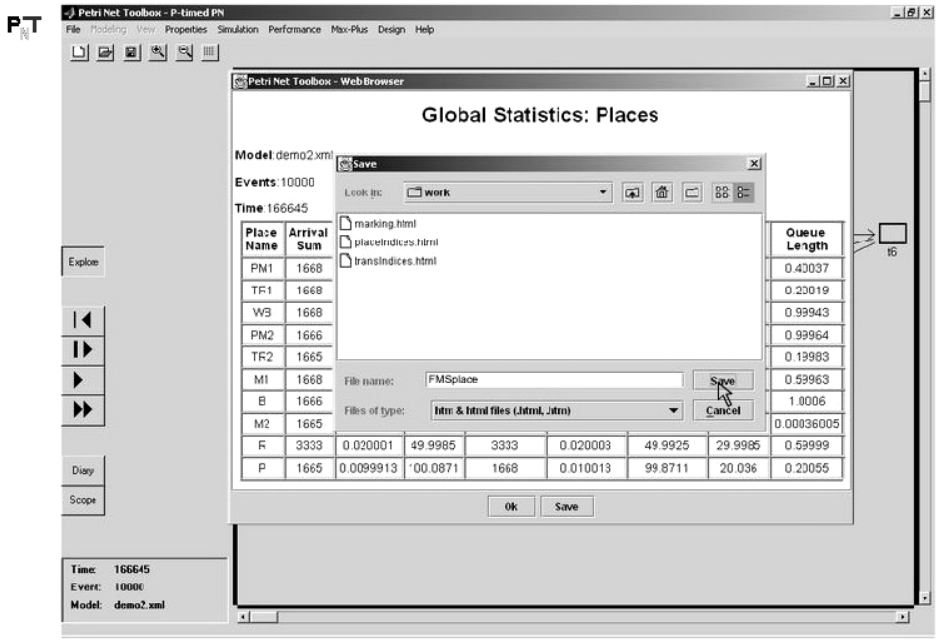
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

13. Inspect the values with physical significance



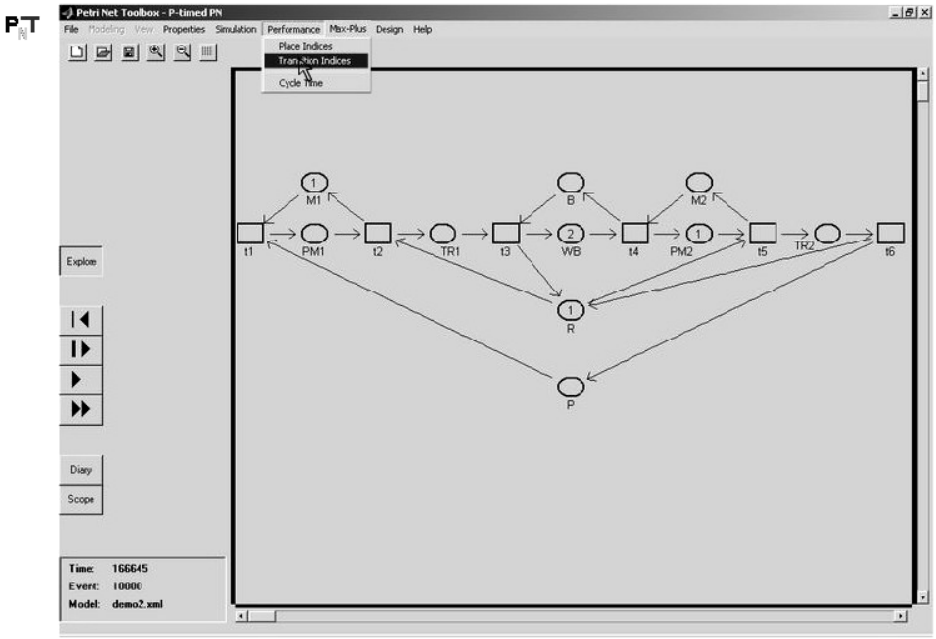
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

14. Save the performance indices as an *html* file



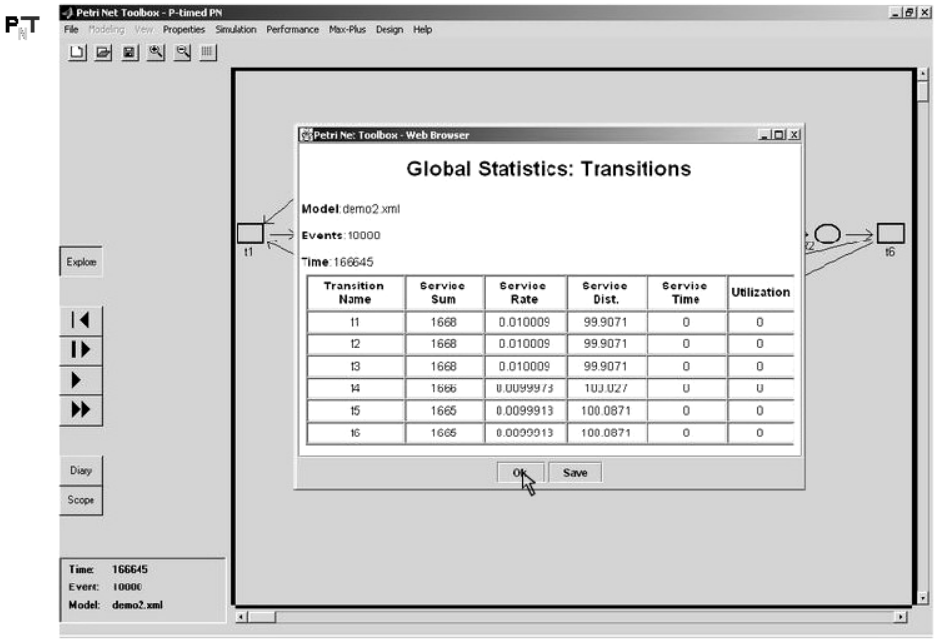
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

15. After the end of simulation inspect the global ***Transition Indices***



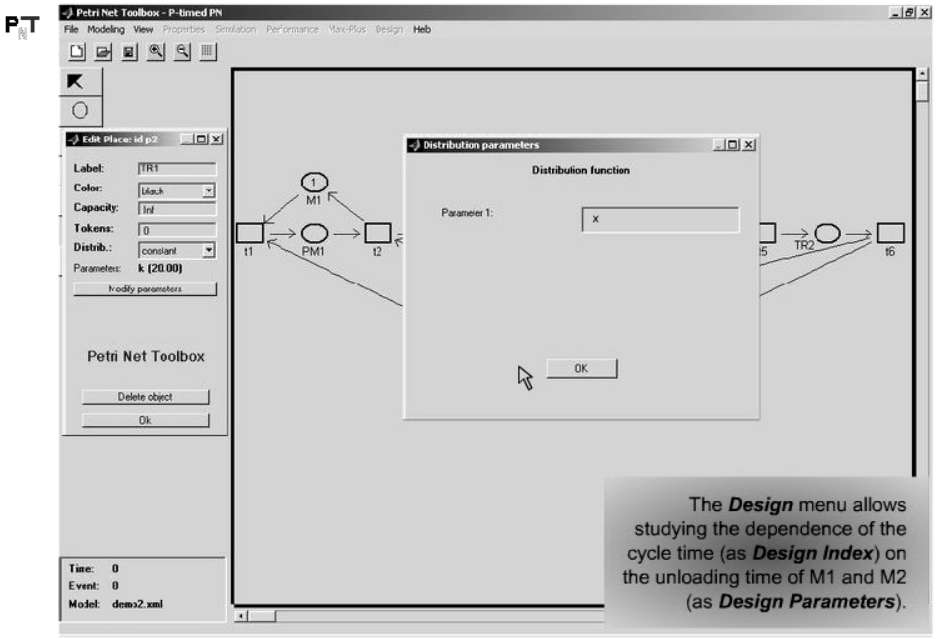
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

16. Notice the correspondence with the global place indices



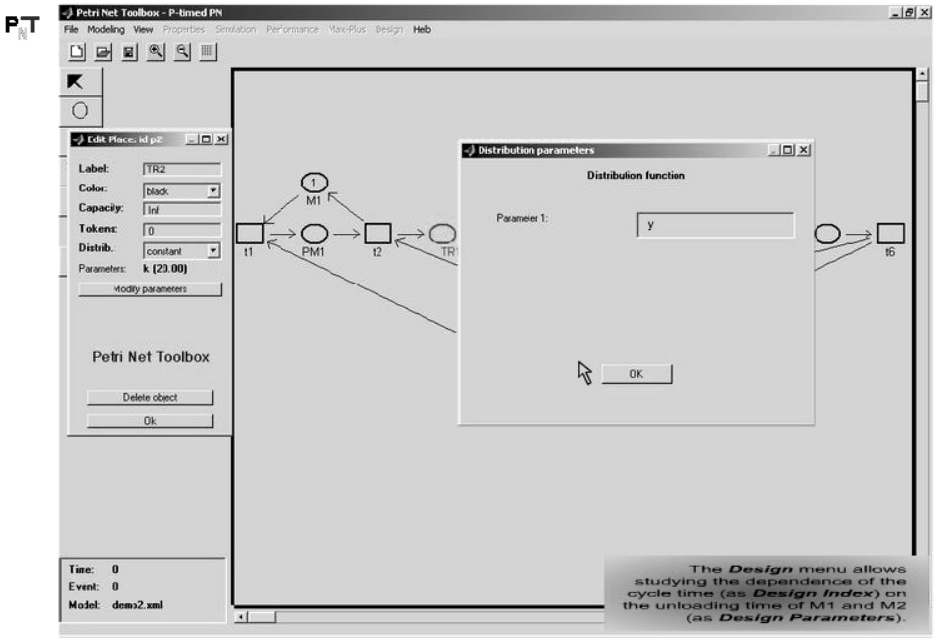
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

17. Set the design parameters. Denote by **x** the duration of the first unloading operation



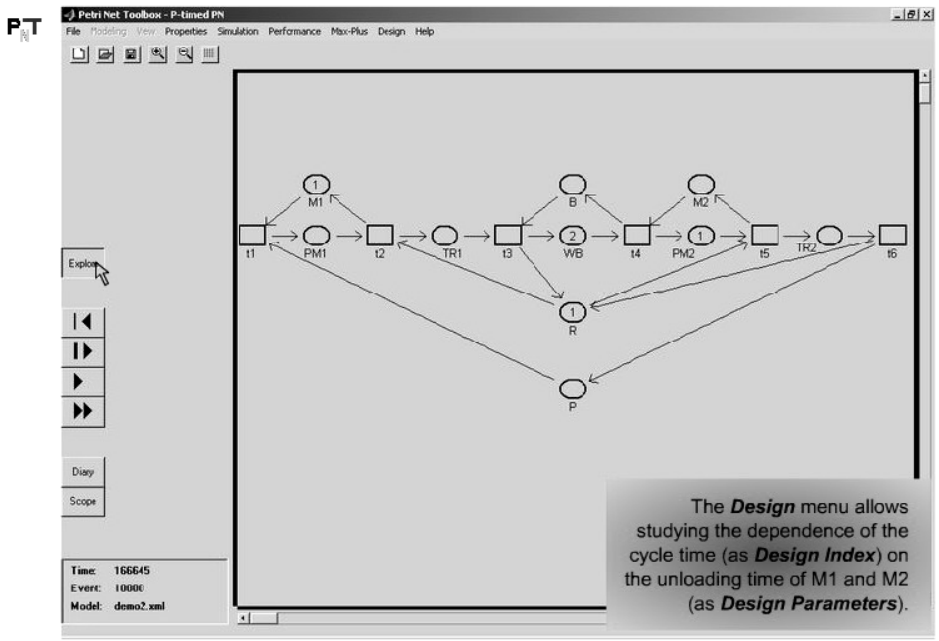
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

18. Denote by **y** the duration of the second unloading operation



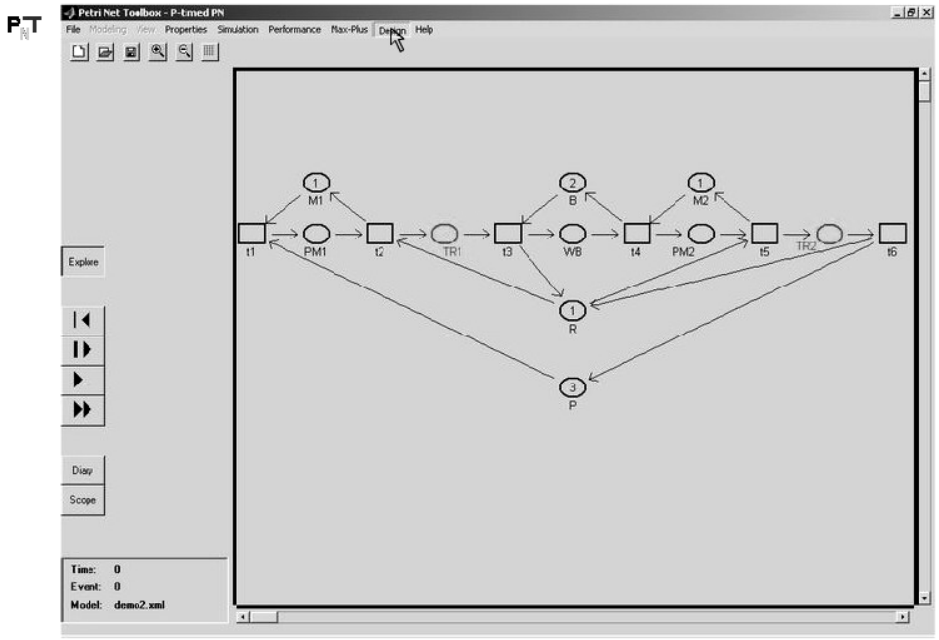
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

19. Switch to ***Explore Mode***



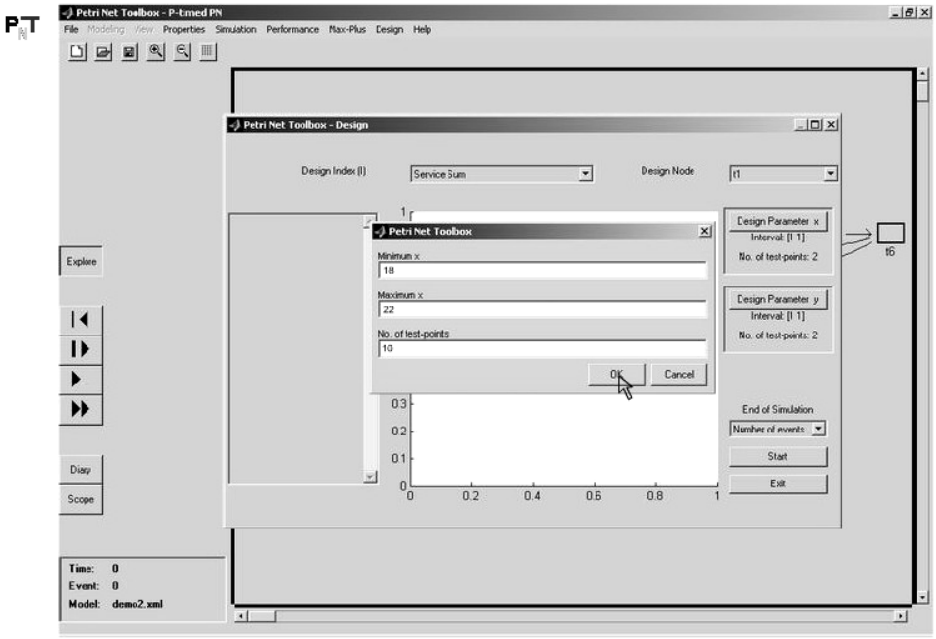
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

20. Select the ***Design*** command from the ***Menu Bar***



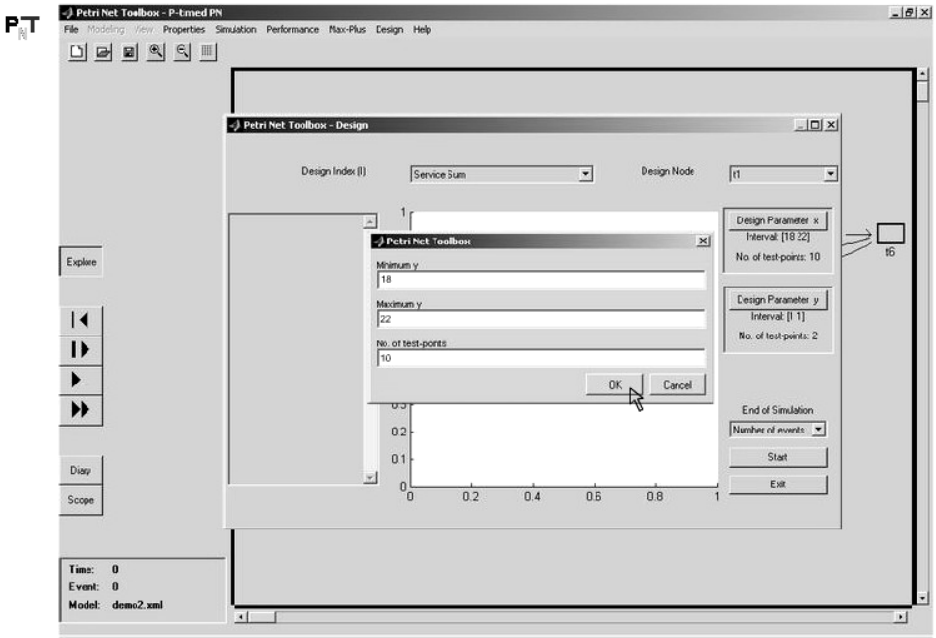
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

21. Set the minimum and maximum values for **x** and the number of simulation points



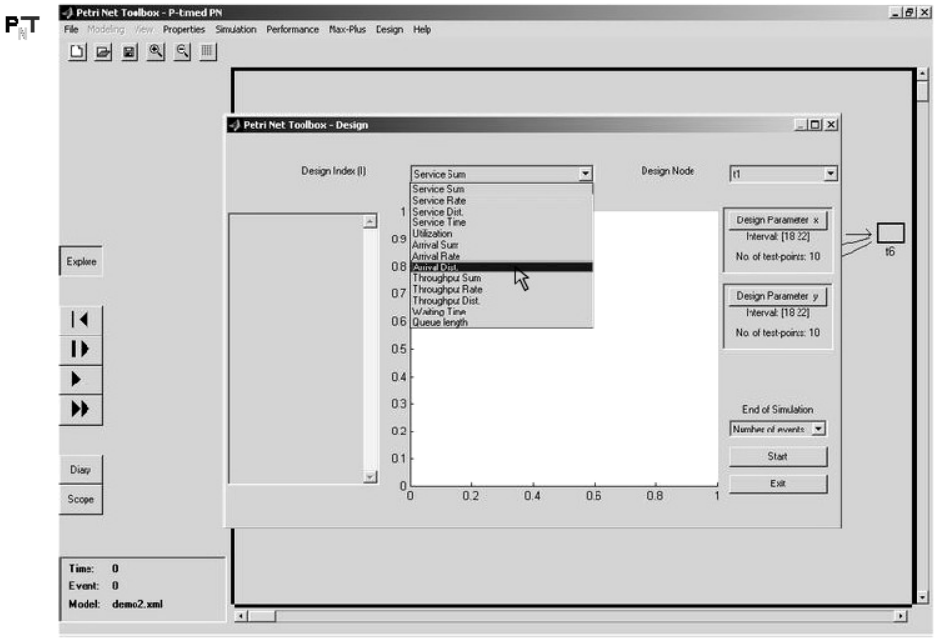
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

22. Set the minimum and maximum values for **y** and the number of simulation points



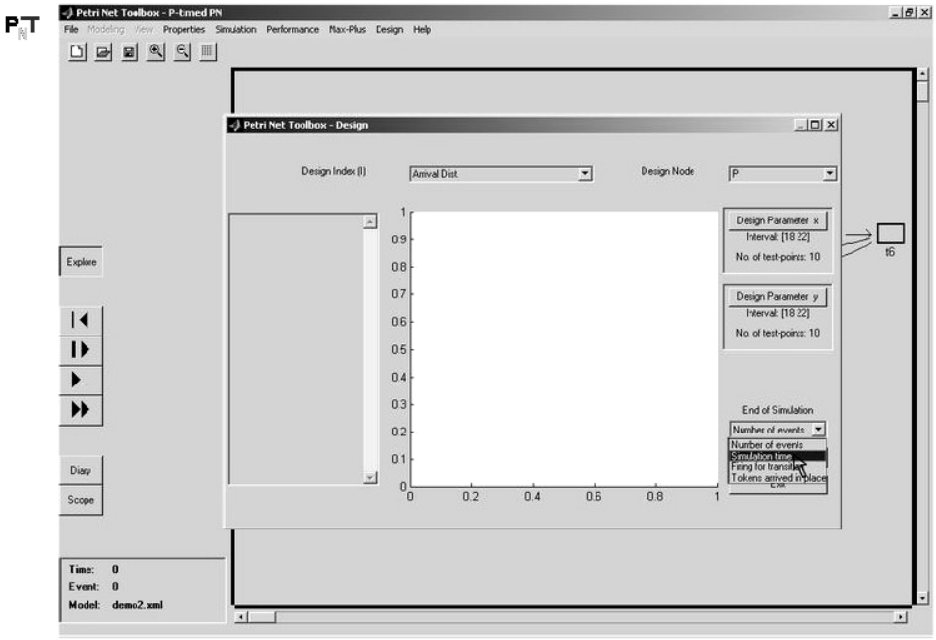
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

23. Select the performance index to be studied



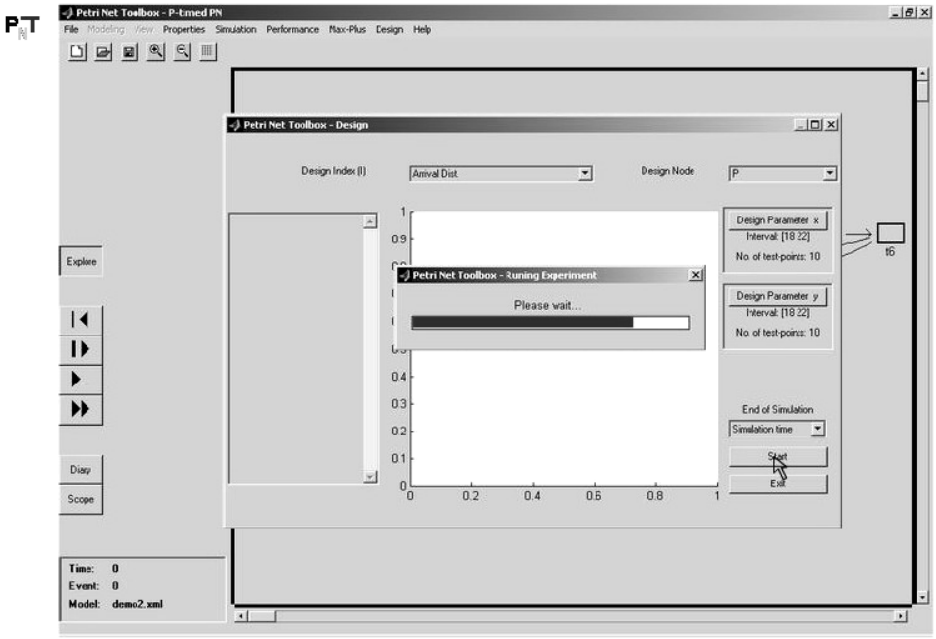
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

24. Set the simulation breakpoint



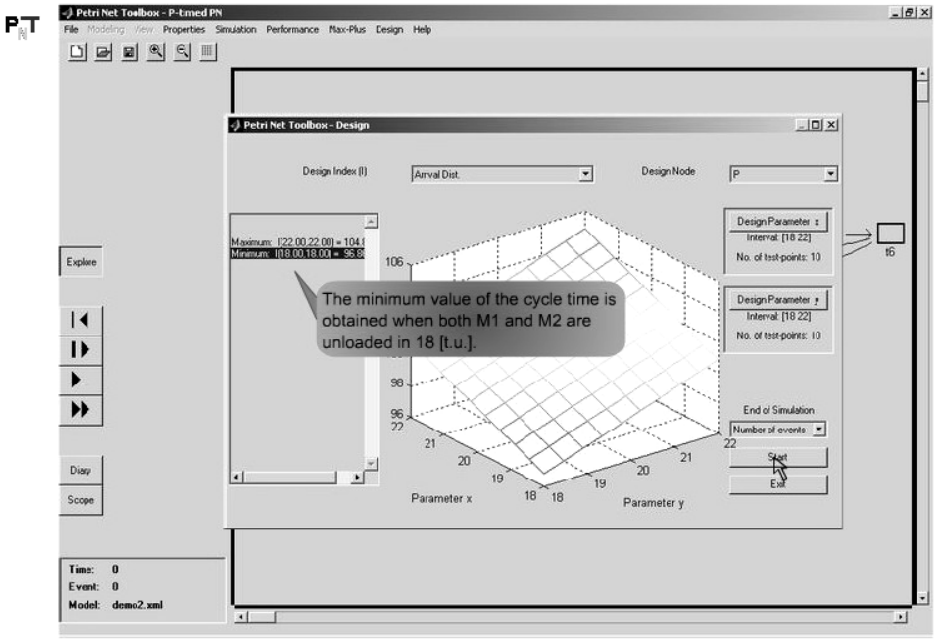
Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

25. Run the simulation experiments



Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

26. Study the results and reach a conclusion on the studied subject



Petri Net Toolbox - Demo 2: Manufacturing system with a sequentially shared robot

Demo 3

FLOW-SHOP SYSTEM

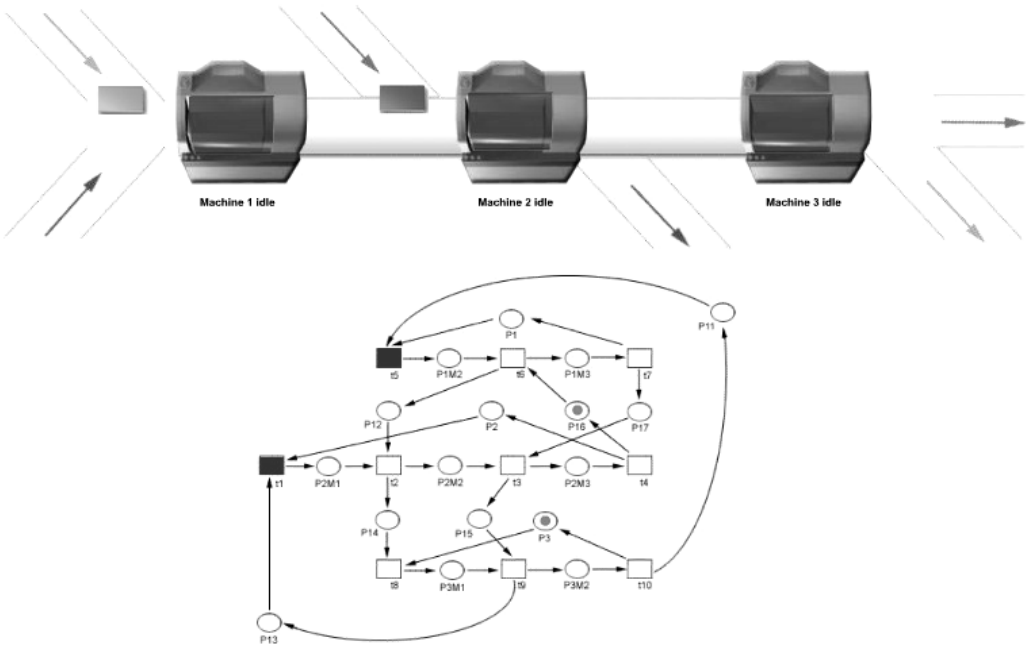
We consider a flow-shop type manufacturing system consisting of three machines. It is supposed to produce three kinds of parts according to a certain product mix. All parts of a certain type follow the same sequence on the machines (although they may skip some) and every machine is visited at most once by each part. 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. Each processing operation takes a given amount of time. There are no set-up times on machines when they switch from one part type to another and also no traveling times for parts between the machines.

The current demonstration illustrates the following aspects of ***PN Toolbox*** usage:

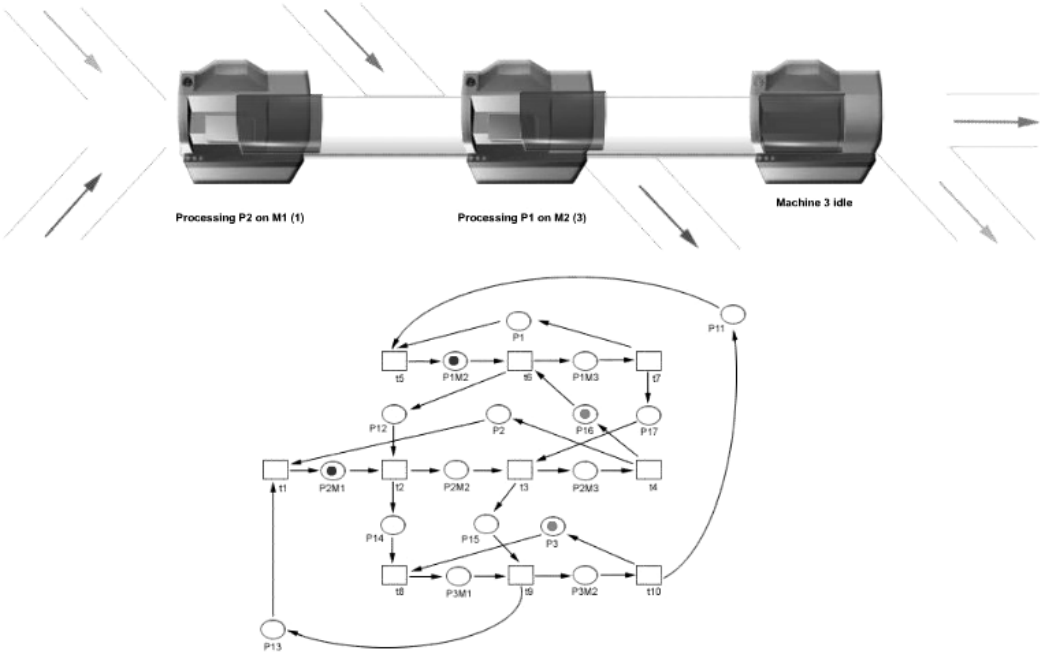
- simulation and animation in the Run Slow mode
- recording the simulation results in a log file
- computation of the cycle time
- max-plus analysis of a place-timed event graph
- max-plus state-space representation
- setting the values for the input vectors (moments for firing source transitions)
- max-plus based simulation
- plots of the components for the input, state or output vectors

D3.1. Illustration of the Physical System

1. The flow-shop system: arrival of two parts



2. State when two machines are working



D3.2. Usage of the PN Toolbox for System Modeling and Analysis

1. Problem description

Demo 3

Flow-shop system with three machines

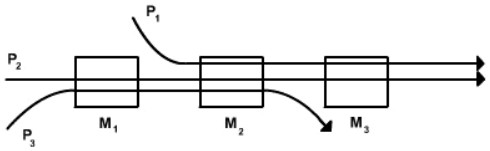
A manufacturing system consists of three machines (M1, M2 and M3). It is supposed to produce three kinds of parts (P1, P2 and P3) according to a certain product mix. The routes to be followed by each part on the machines are depicted in the bellow figure and the corresponding processing times are given in the table.

There are no set-up times on machines when they switch from one part type to another and also no traveling times for parts between the machines. 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 system behavior is studied in the following two situations:

A. For each part type, no more than one part can be present in the system at a time. The processing of a part can start as soon as a part of the same type has been completed and the required machine is available.

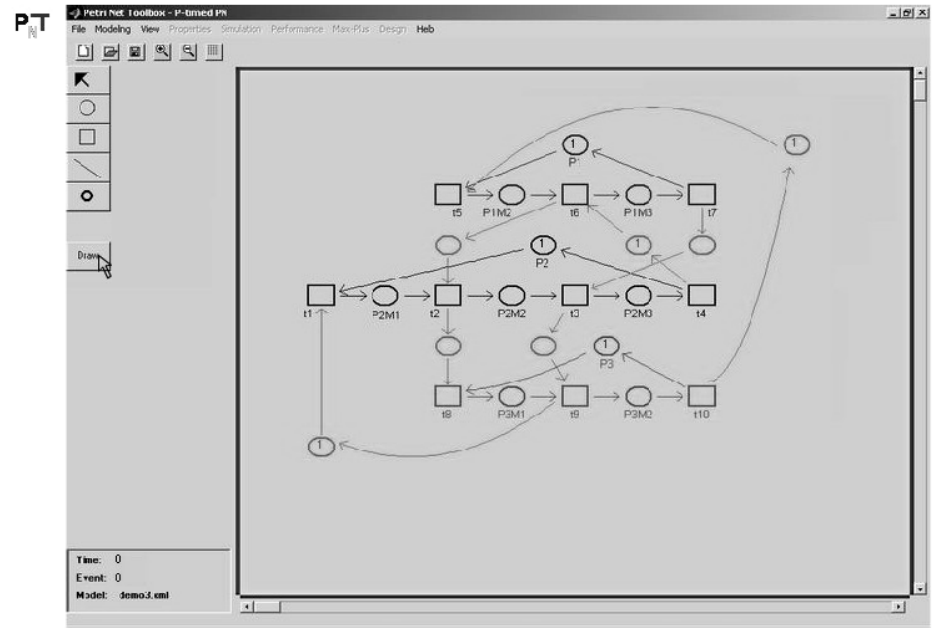
B. The final product mix is obtained by means of a given input of parts.



	P_1	P_2	P_3
M_1	-	1	5
M_2	3	2	3
M_3	4	3	-

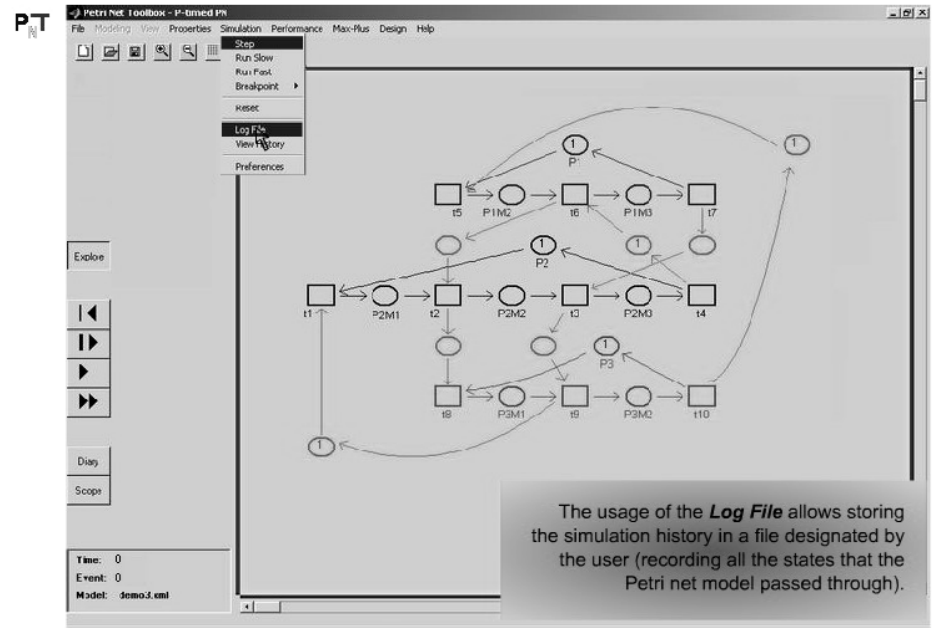
Play movie

2. Draw the P-timed PN model then switch to ***Explore Mode***



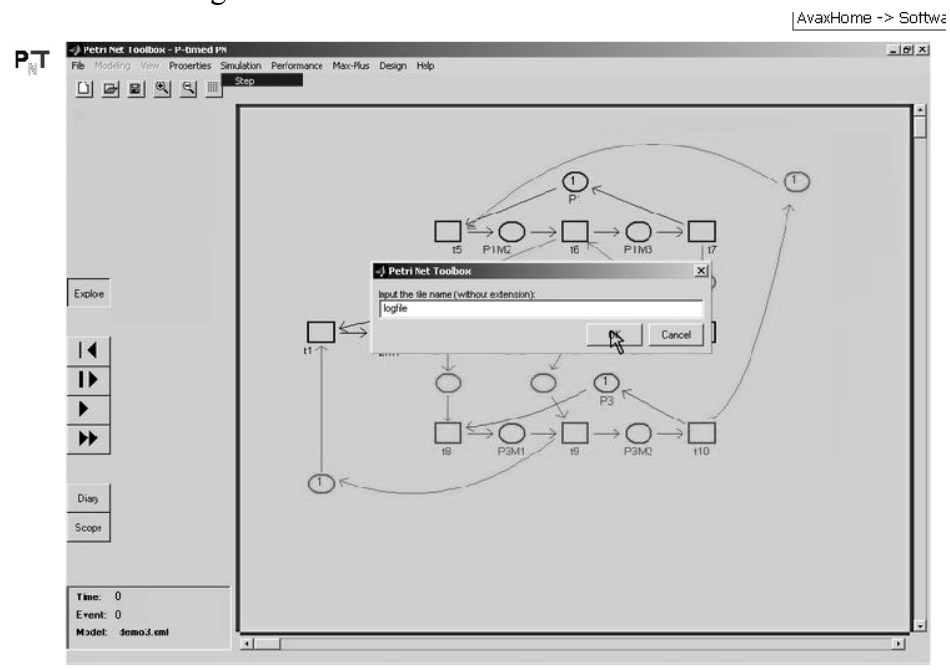
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

3. Before simulation, select the ***Log File*** command



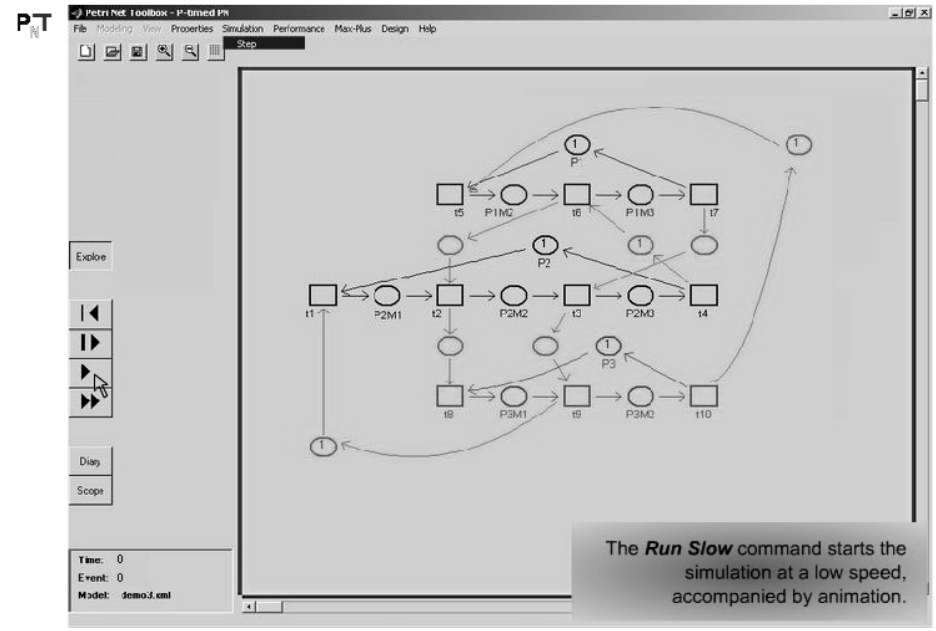
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

4. Set the name for the log file to be saved



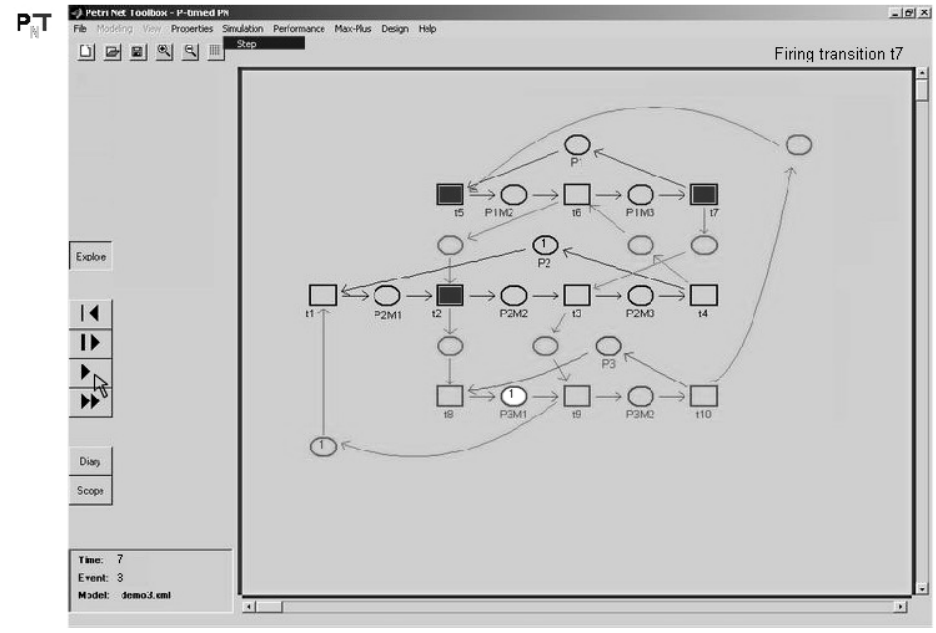
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

5. Simulate the system in ***Run Slow*** mode



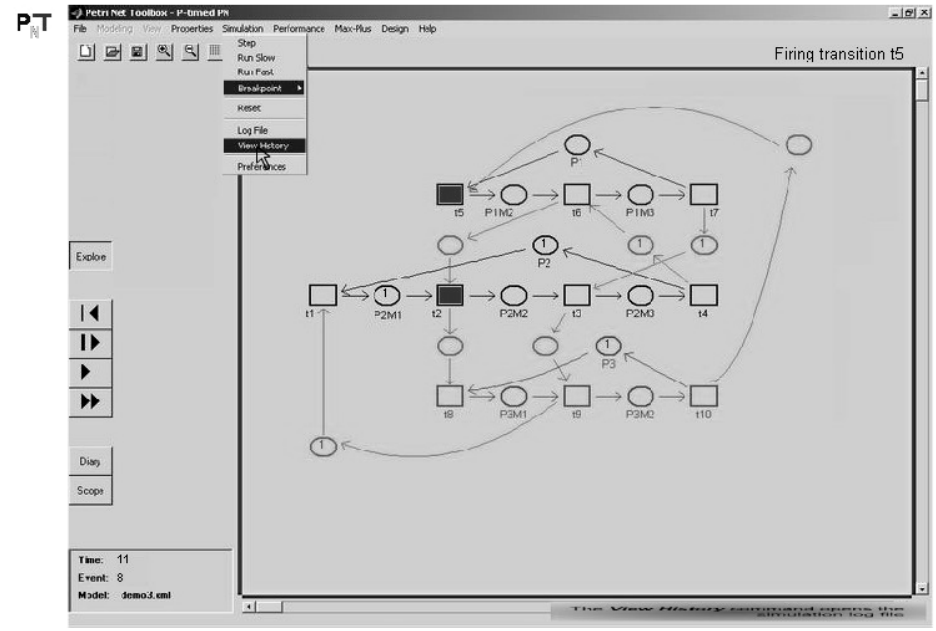
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

6. Watch the sequencing of events in the system



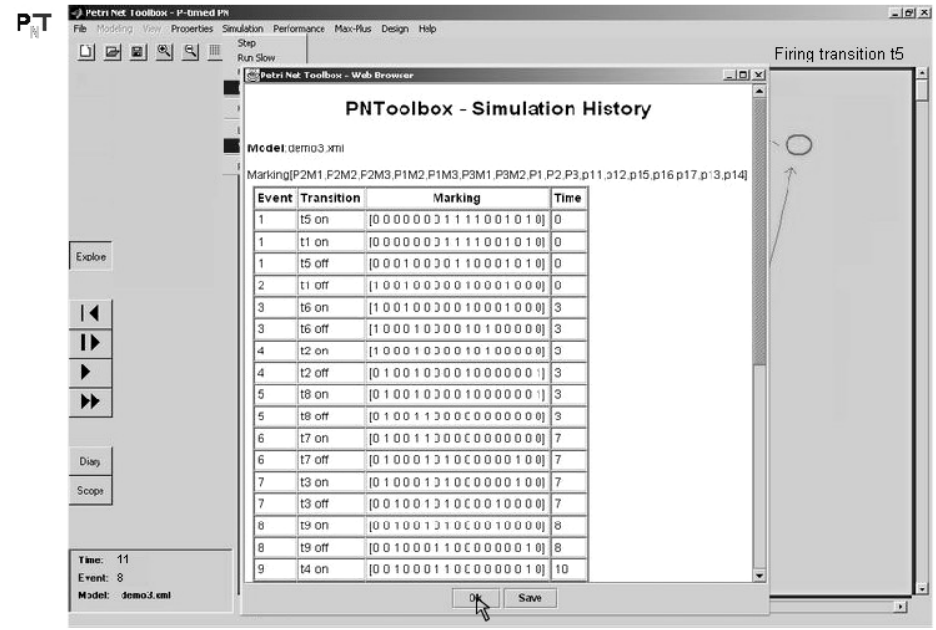
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

7. Open the saved log file by selecting the ***View History*** command



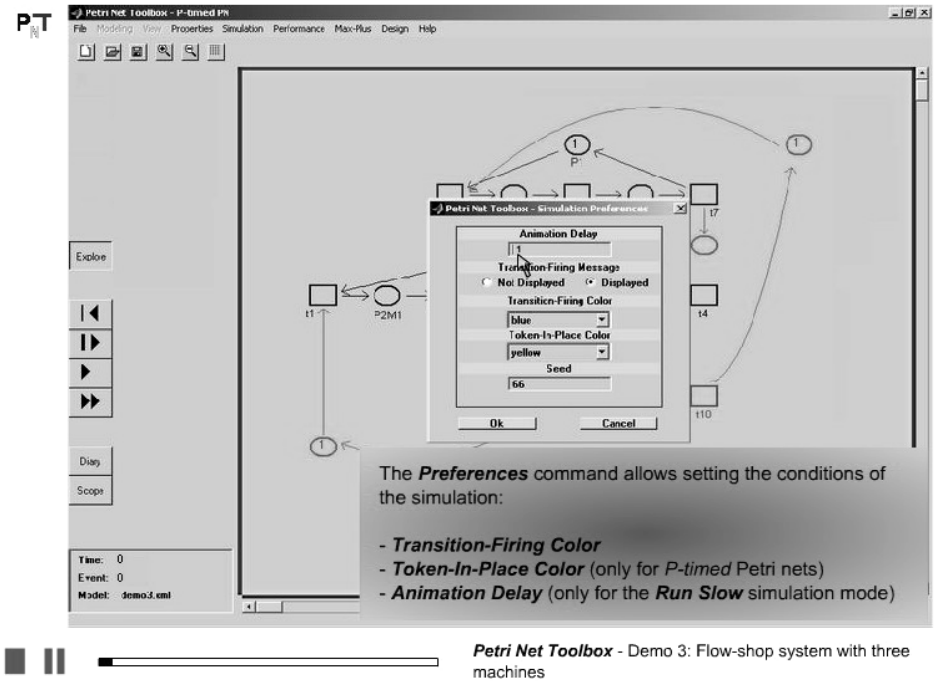
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

8. Inspect the simulation progress and save the history as an *html* file



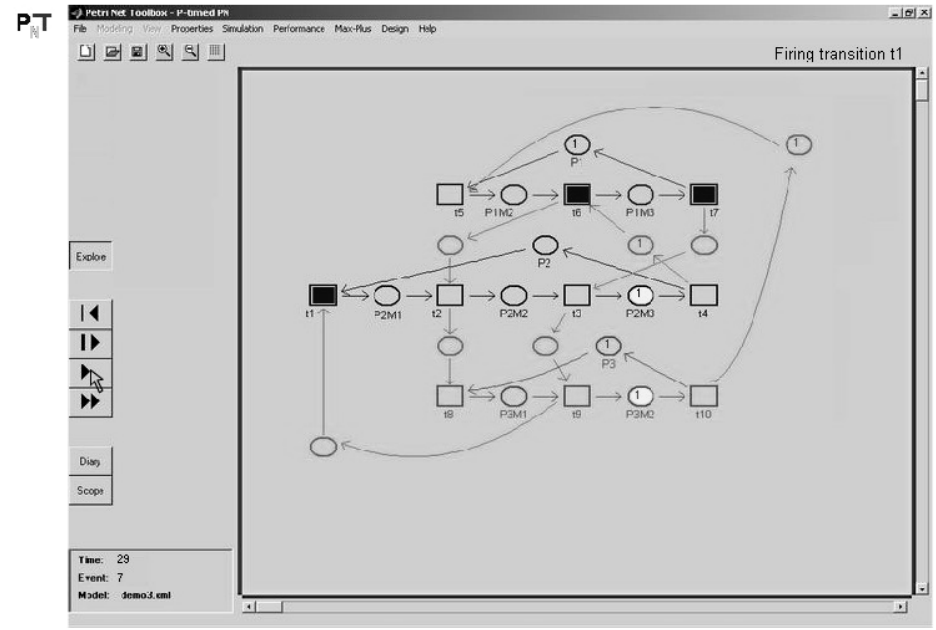
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

9. Using the ***Preferences*** command, set the conditions for visualization of simulation



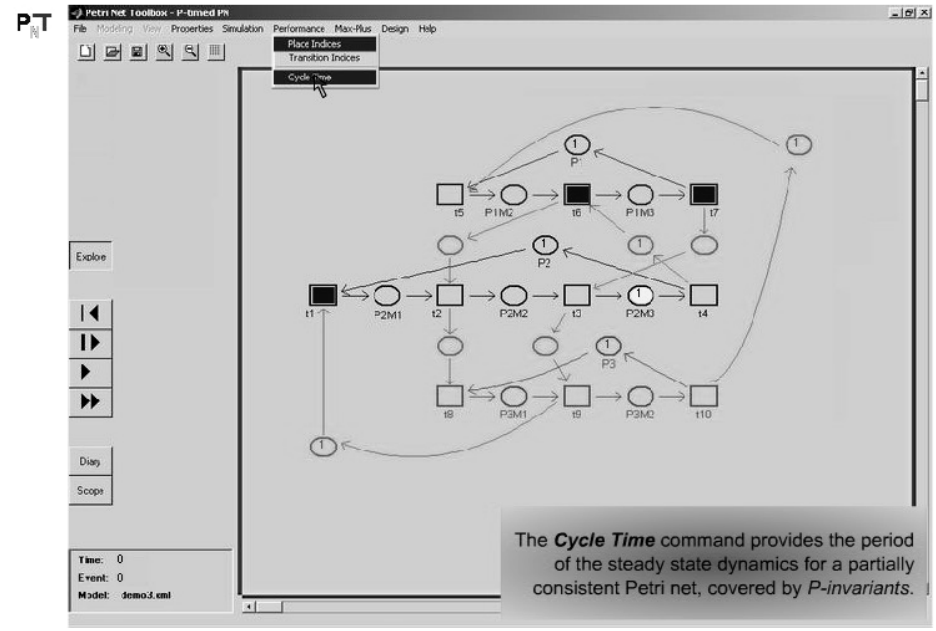
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

10. Simulate the model in ***Run Slow*** mode



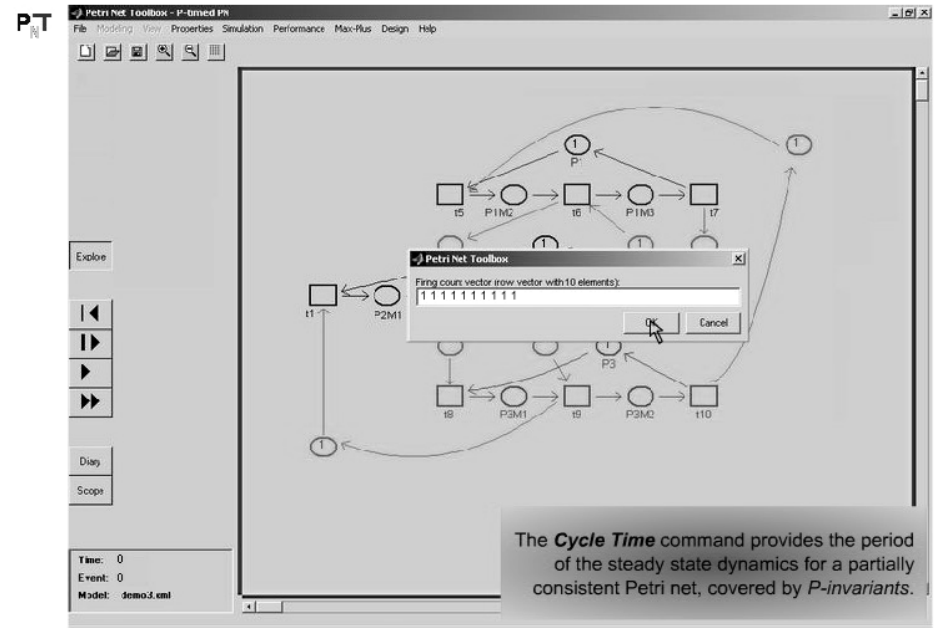
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

11. Compute the ***Cycle Time*** for the autonomous P-timed event-graph



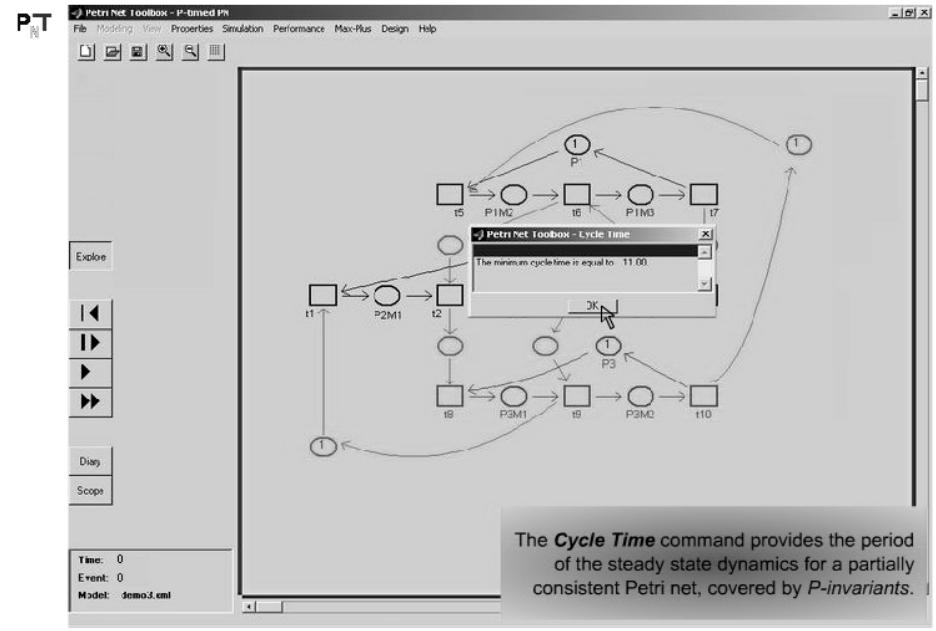
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

12. Introduce the firing count vector corresponding to a production cycle



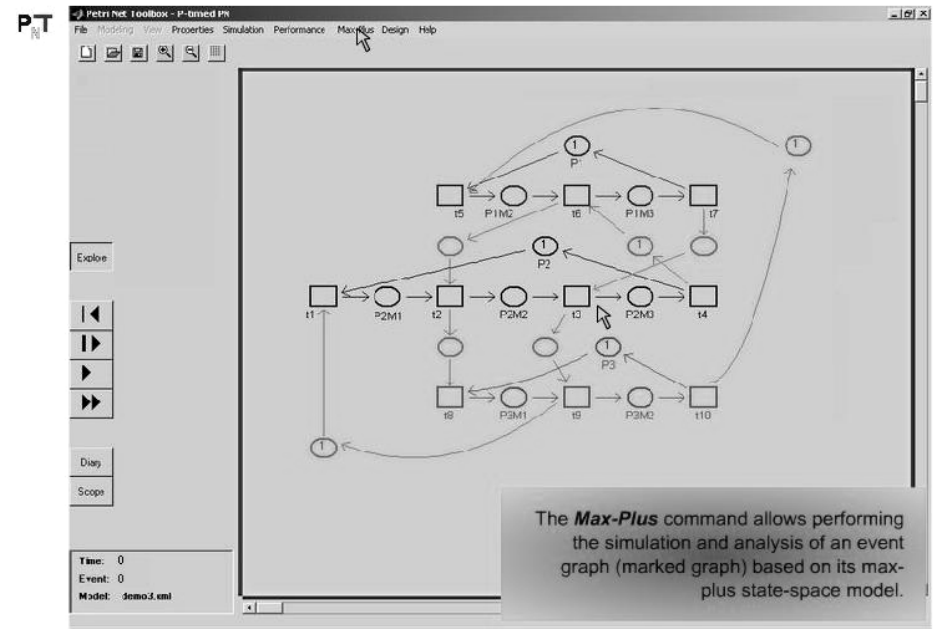
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

13. The computed cycle time is displayed in a dialogue window



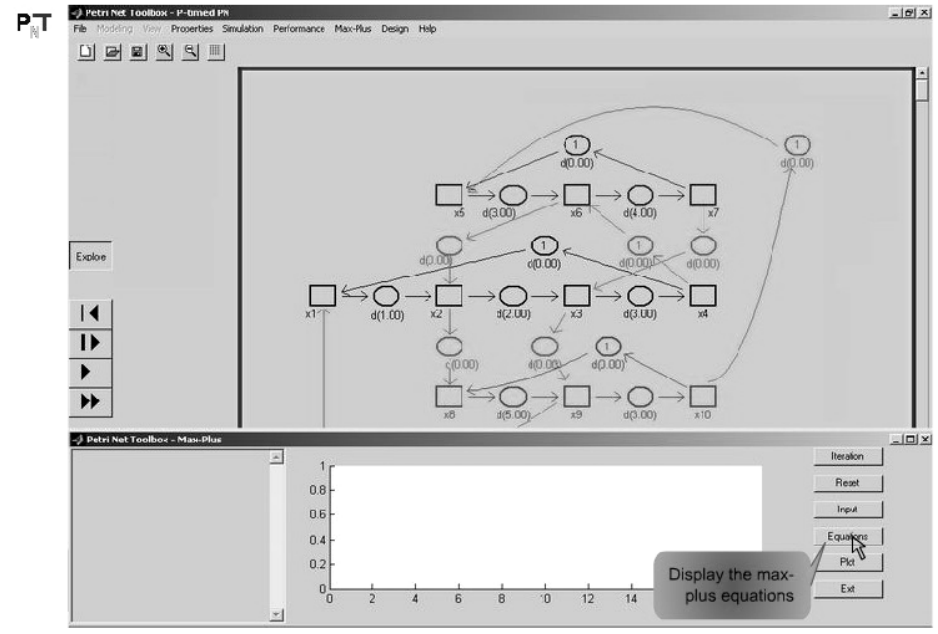
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

14. Open the ***Max-Plus*** window, to analyze and simulate the system based on its max-plus state-space representation



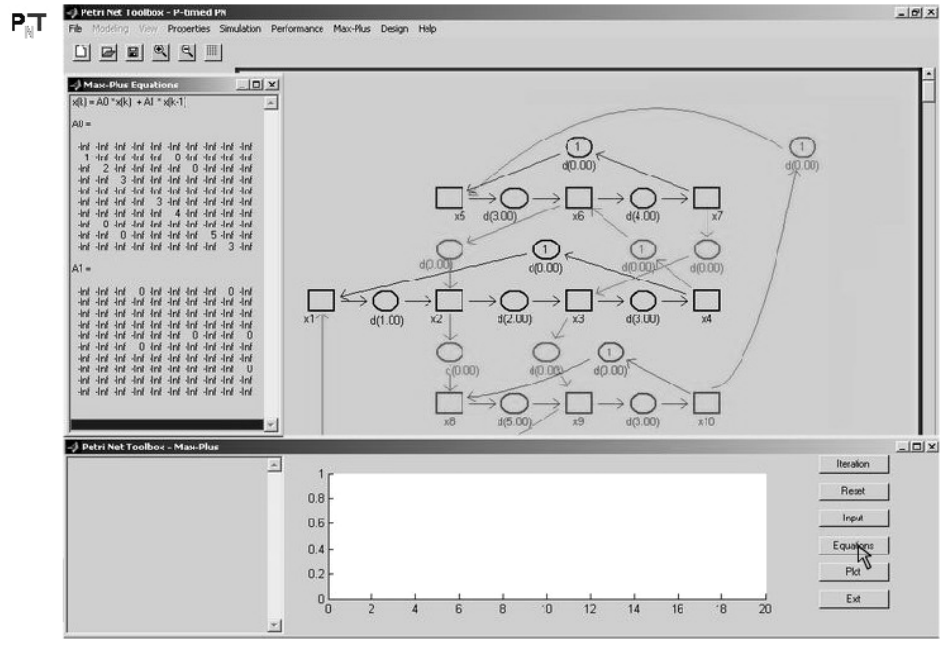
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

15. Display the max-plus equations



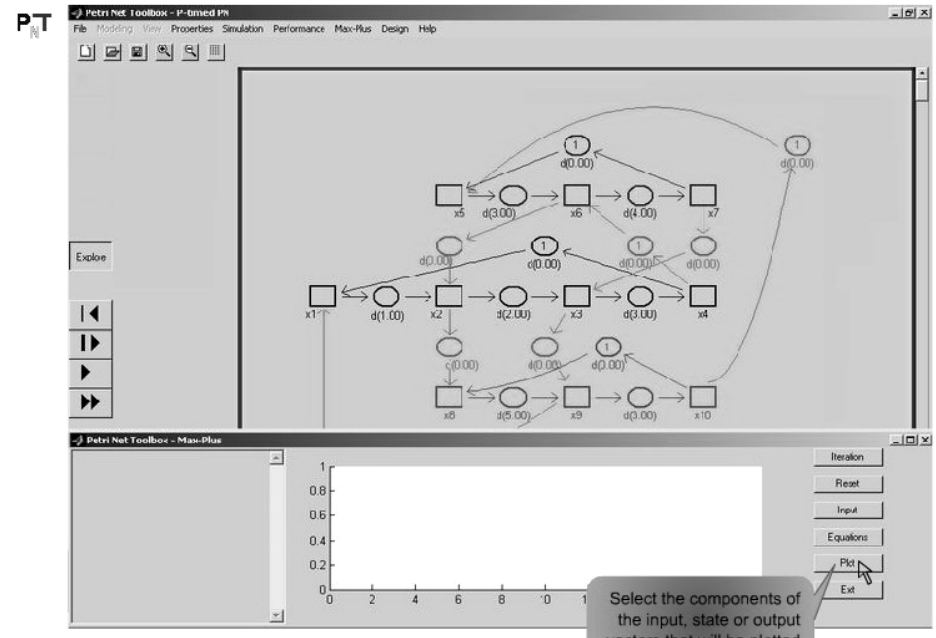
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

16. Study the max-plus equations and the significance of the state-variables



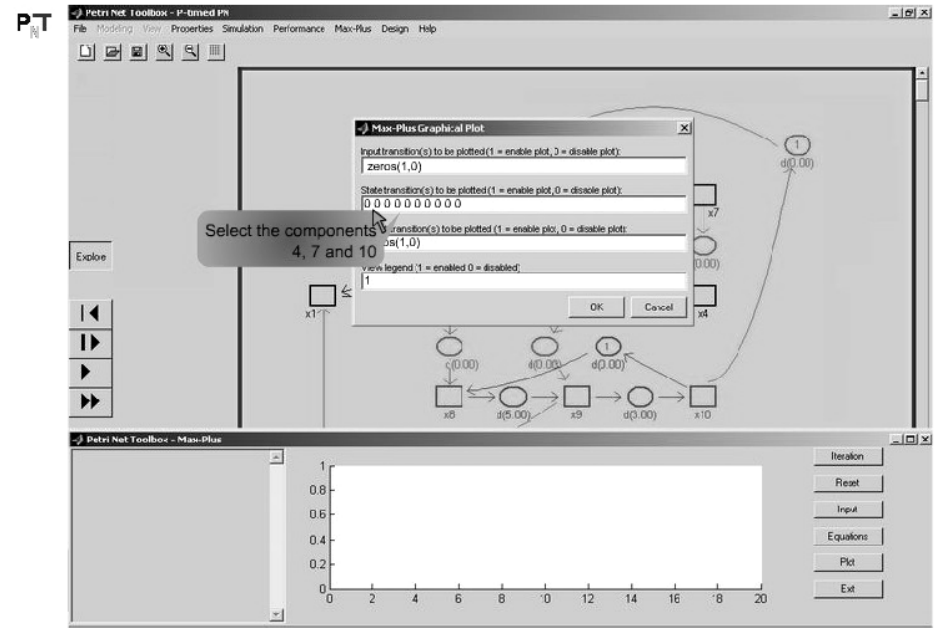
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

17. Select the state-variables to be plotted during simulation



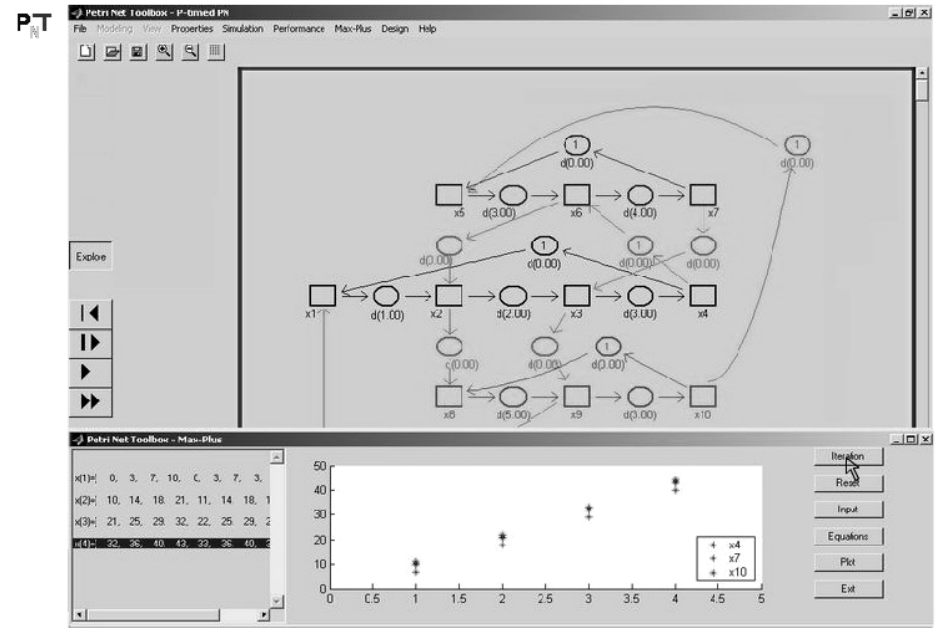
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

18. Since the model is a strongly connected graph, no input is required



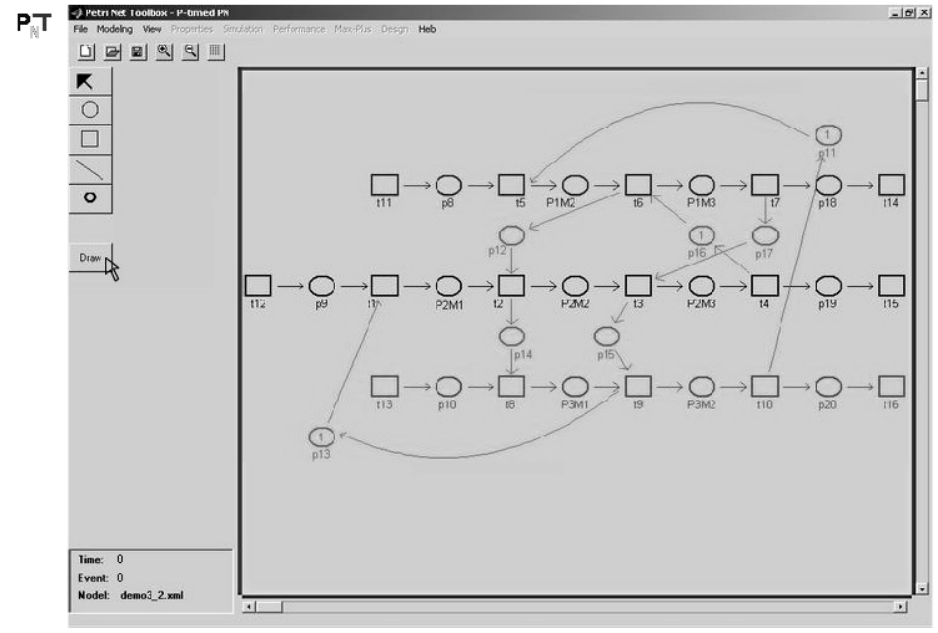
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

19. Simulate the model step-by-step and interpret the obtained graphical representation



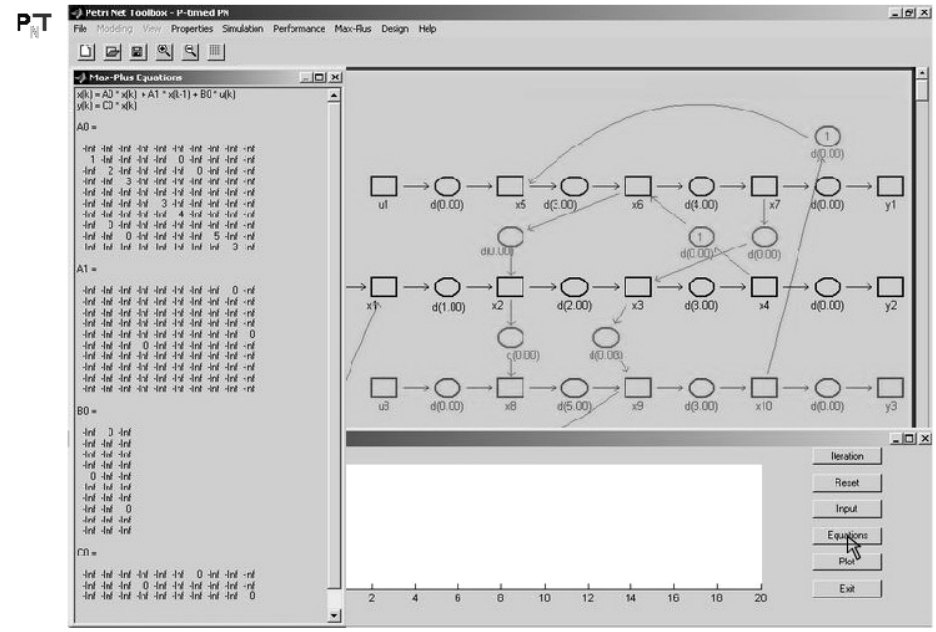
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

20. Open the model of a second flow-shop system, which is non-autonomous



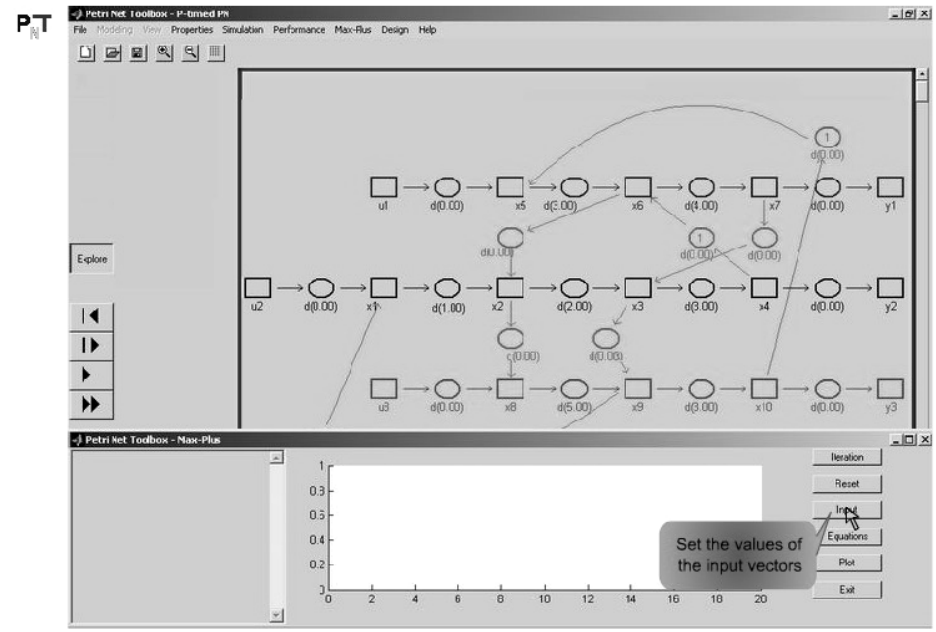
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

21. Study the max-plus equations and the significance of the state-variables, inputs and outputs



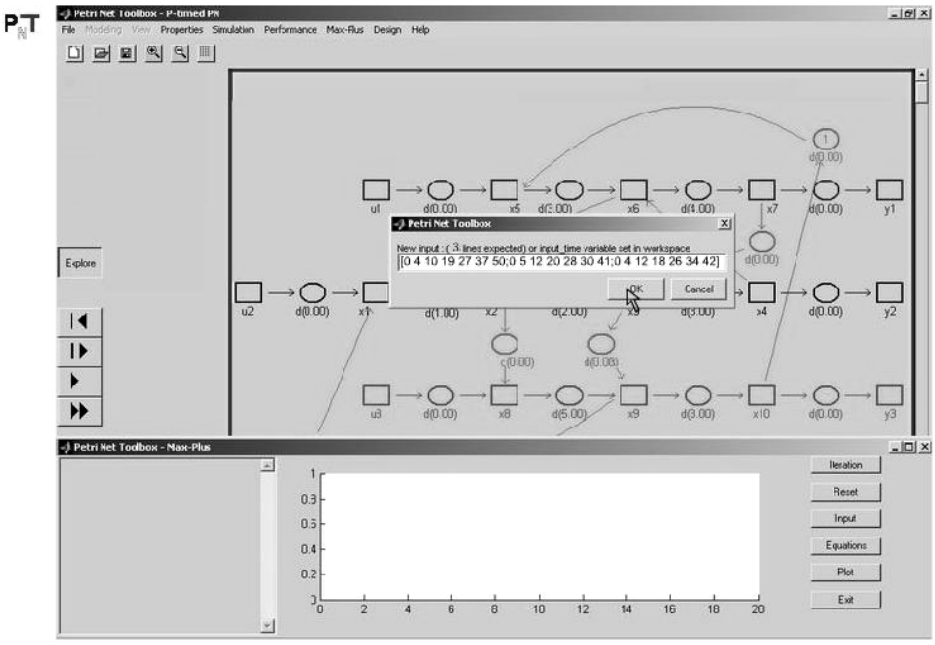
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

22. For a non-autonomous model, it is necessary to set the values of the inputs



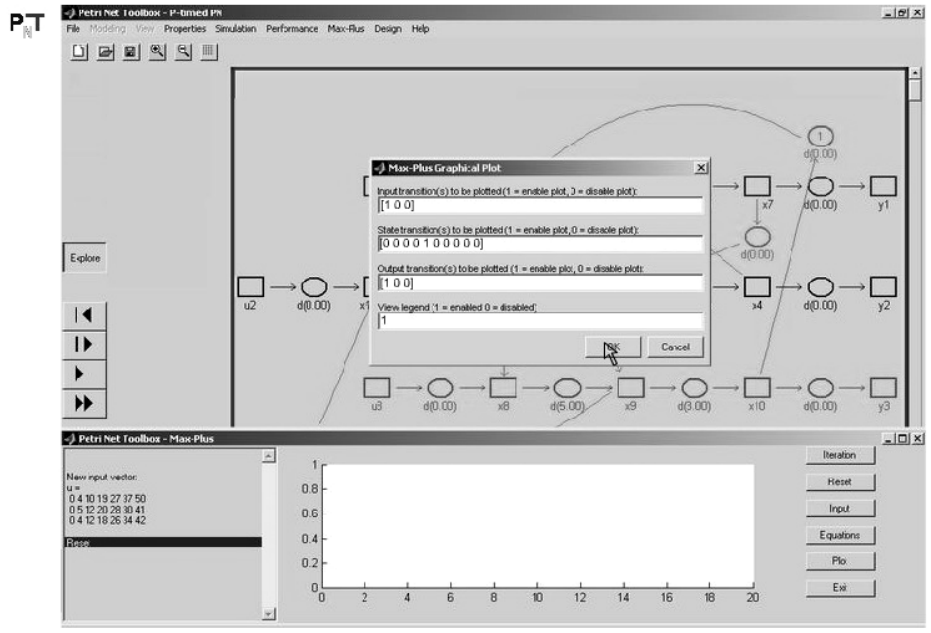
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

23. Give the matrix containing the moments when the source transitions are fired



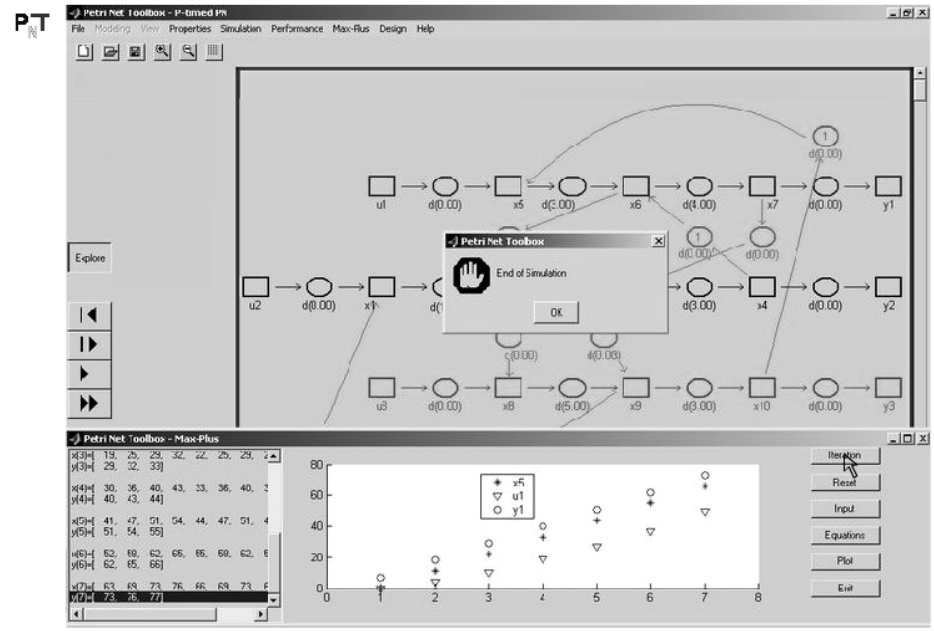
Petri Net Toolbox - Demo 3: Flow-shop system with three machines

24. Select the variables to be plotted during simulation



Petri Net Toolbox - Demo 3: Flow-shop system with three machines

25. Simulate the model step-by-step and interpret the obtained graphical representation



Petri Net Toolbox - Demo 3: Flow-shop system with three machines

Demo 4

OPEN QUEUEING NETWORK

We consider an open queueing network consisting of three nodes with infinite capacity queues. The service times of the servers are exponentially distributed, their rates being $\mu_1 = \mu_2 = 4\ \mathrm{s}^{-1}$ and $\mu_3 = 2\ \mathrm{s}^{-1}$ respectively. Each server may process only job at a time.

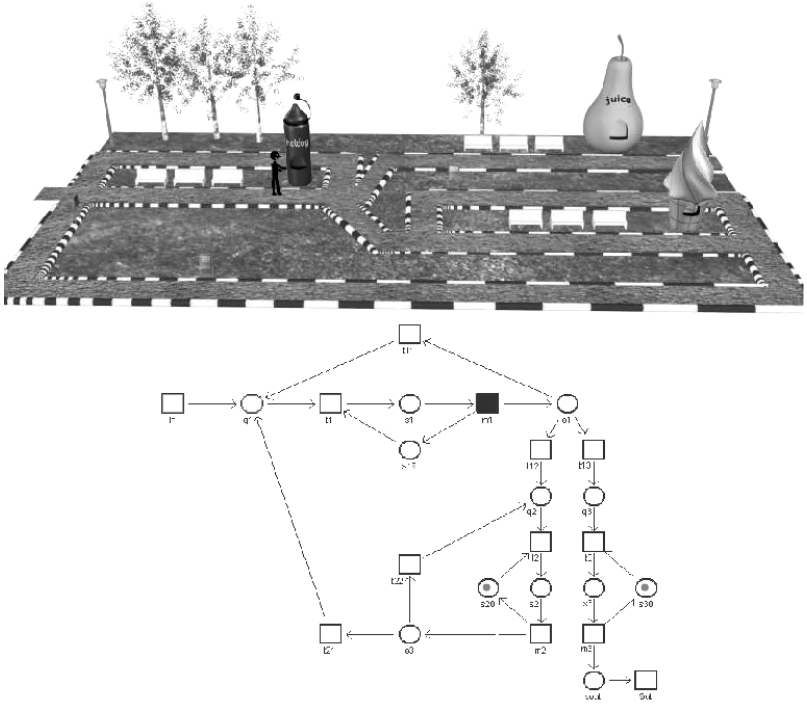
There is a single external Poisson arrival process at node 1 with rate $r_1 = 1\ \mathrm{s}^{-1}$. This queueing network includes feedback paths from node 1 to itself and from node 2 to both nodes 1 and 2. Numerical values of the routing probabilities p_{ij}, $i, j = 1,2,3$, from node i to node j are: $p_{11} = 30\%$, $p_{12} = 20\%$, $p_{13} = 50\%$, $p_{21} = 20\%$ and $p_{22} = 80\%$.

This demo illustrates:

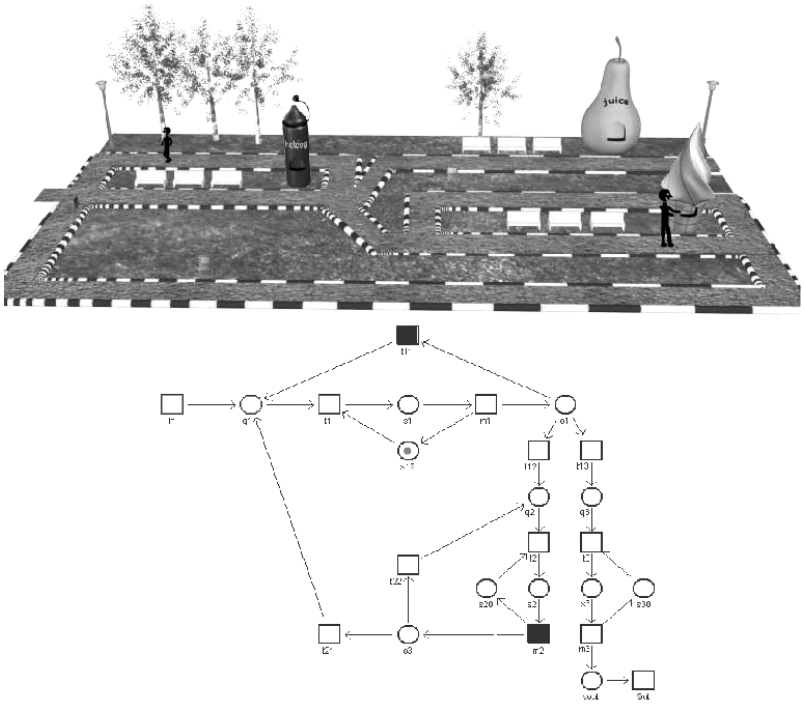
- the construction of a generalized stochastic Petri net model
- the usage of the Scope and Diary facilities
- the analysis of time-dependent performance indices

D4.1. Illustration of the Physical System

1. The two philosophers

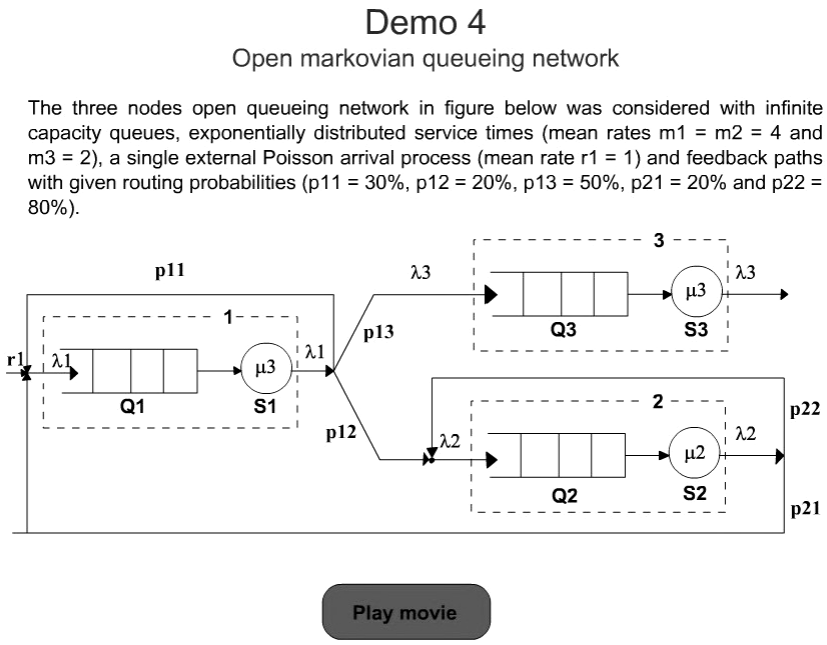


2. Model of the system that allows the analysis of deadlock

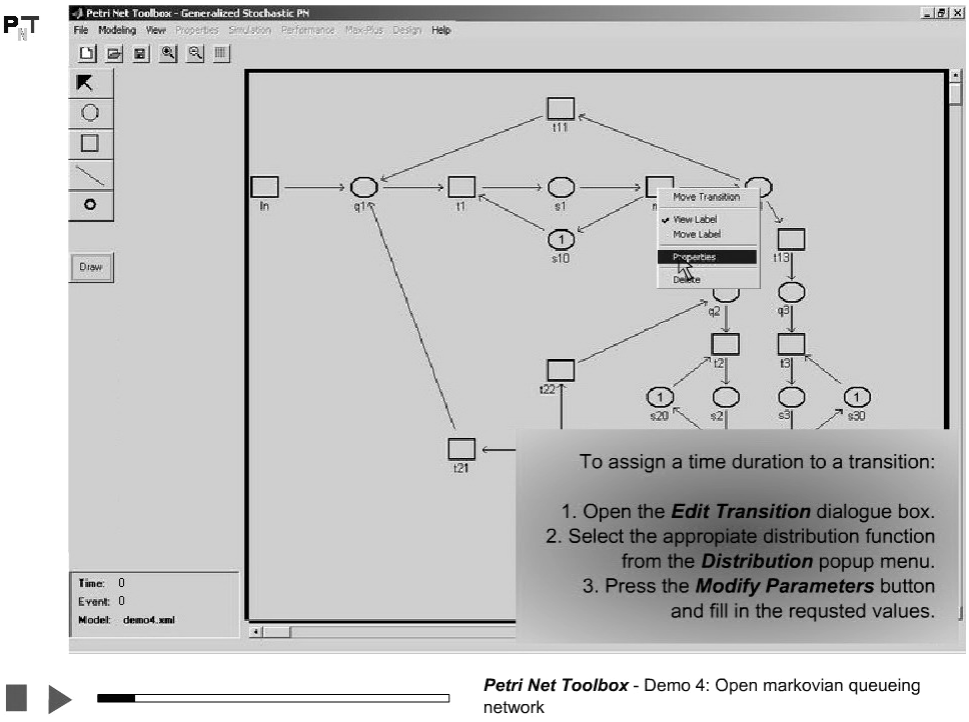


D4.2. Usage of the PN Toolbox for System Modeling and Analysis

1. Problem description

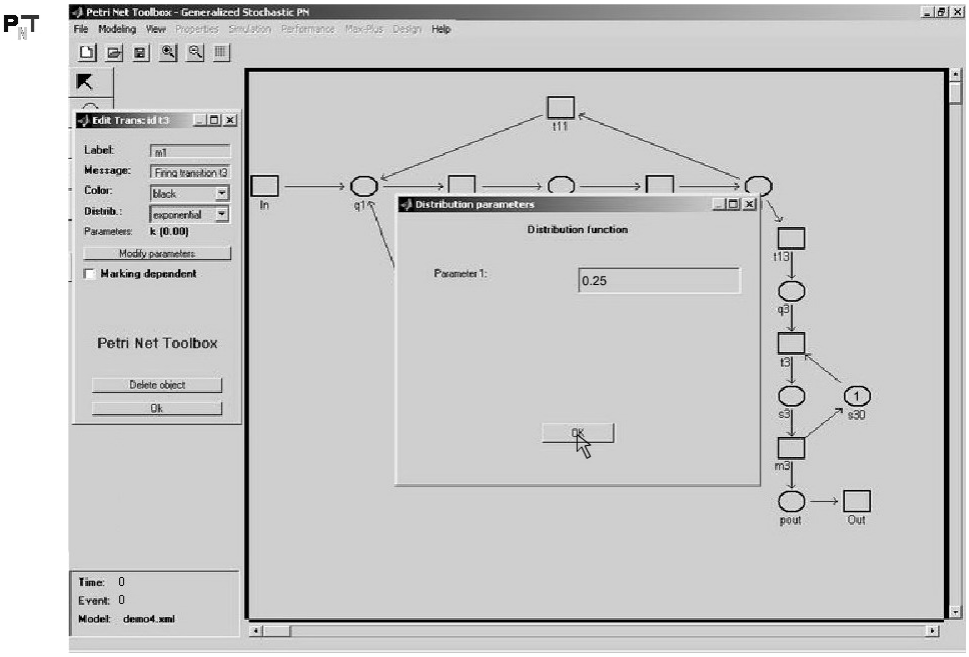


2. After drawing the topology of the net, assign time durations to the transitions in the model



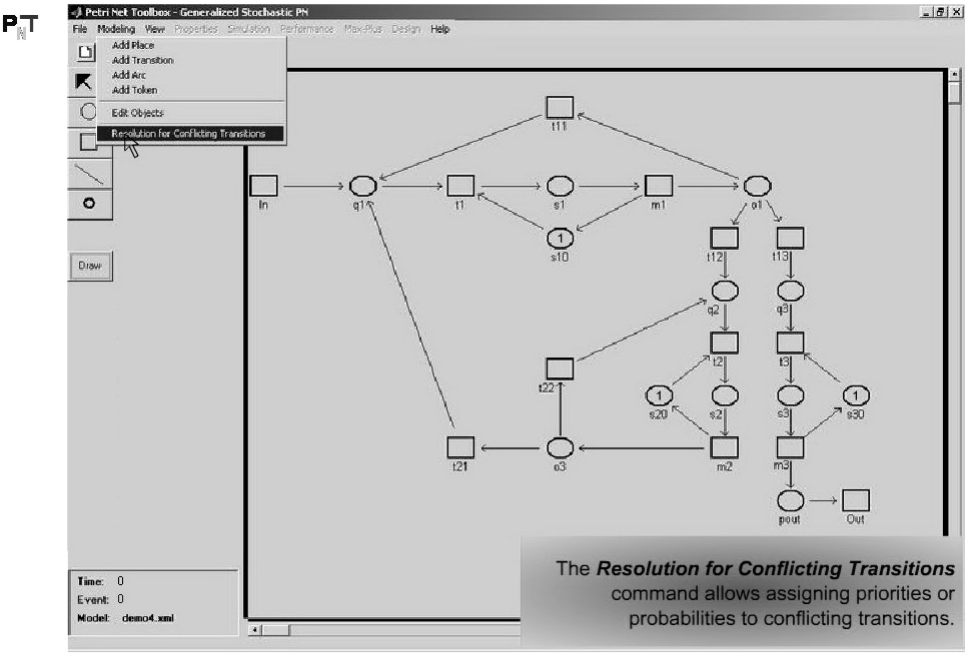
Petri Net Toolbox - Demo 4: Open markovian queueing network

3. Select the desired probability distribution and set its parameters



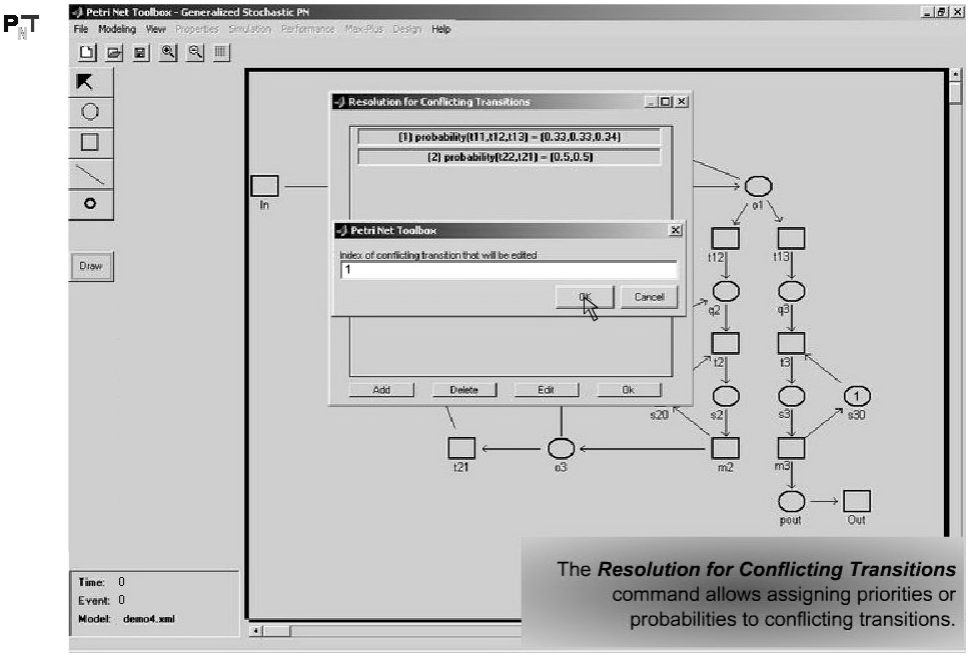
Petri Net Toolbox - Demo 4: Open markovian queueing network

4. Set the probabilities corresponding to conflicting transitions. Select the ***Resolution for Conflicting Transitions*** command from the ***Modeling*** **menu**

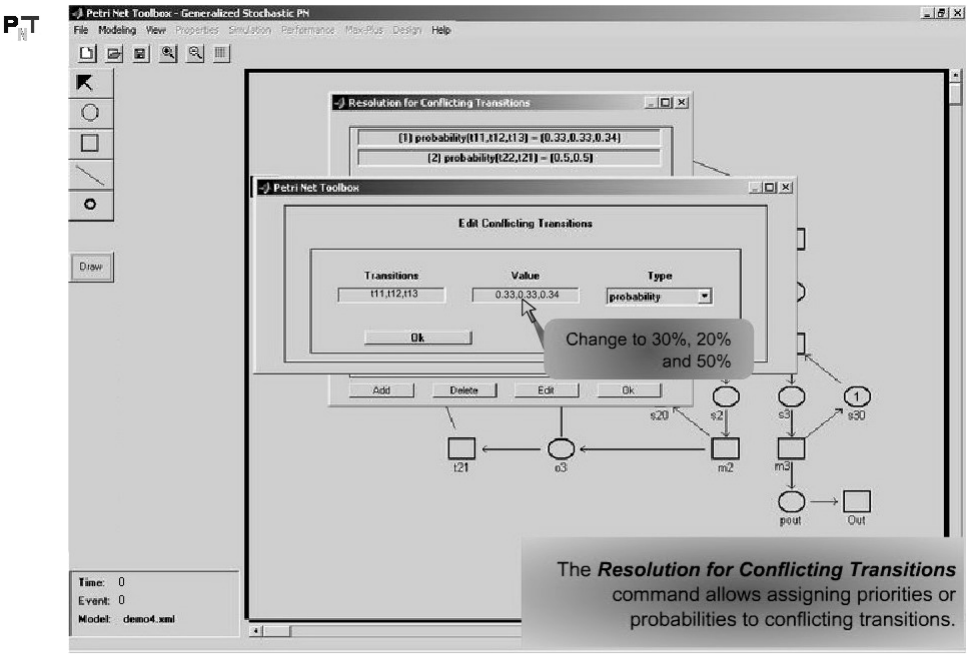


Petri Net Toolbox - Demo 4: Open markovian queueing network

5. For each group of conflicting transitions edit the probabilities set by default

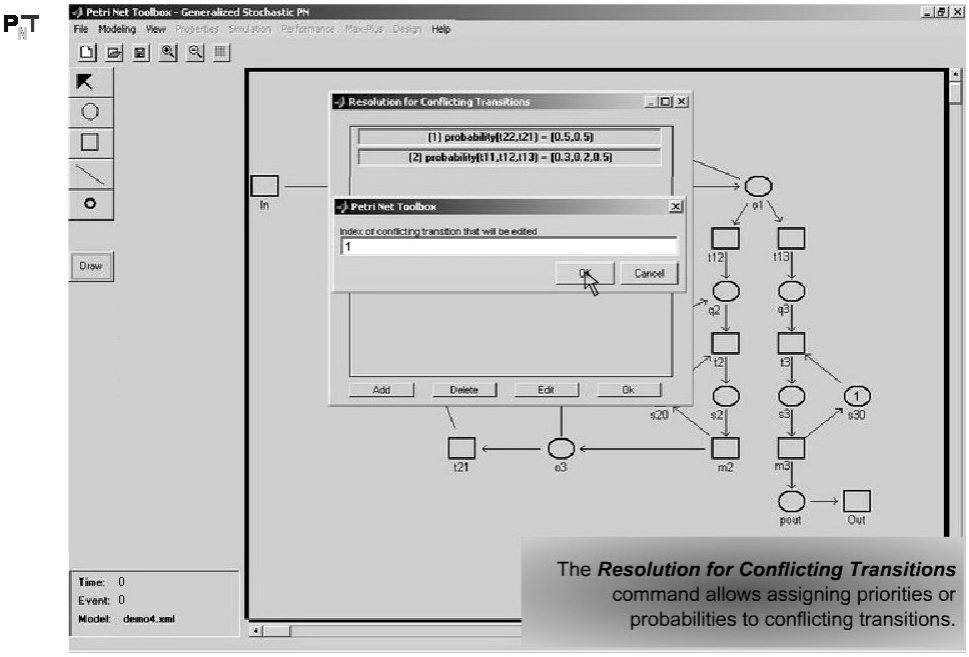


Petri Net Toolbox - Demo 4: Open markovian queueing network

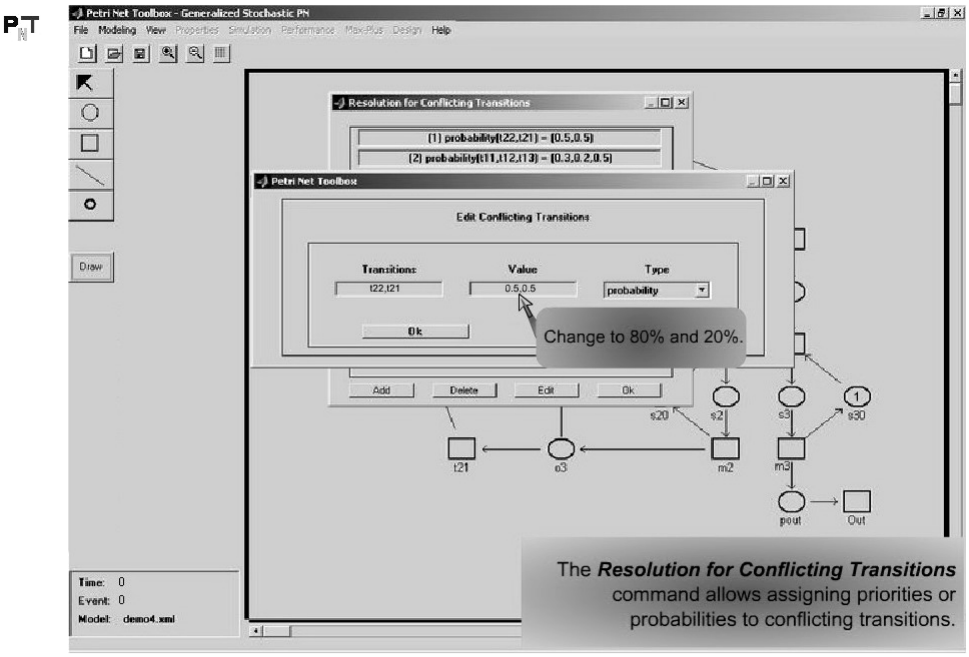


Petri Net Toolbox - Demo 4: Open markovian queueing network

6. Repeat the operation for the remaining group of conflicting transitions

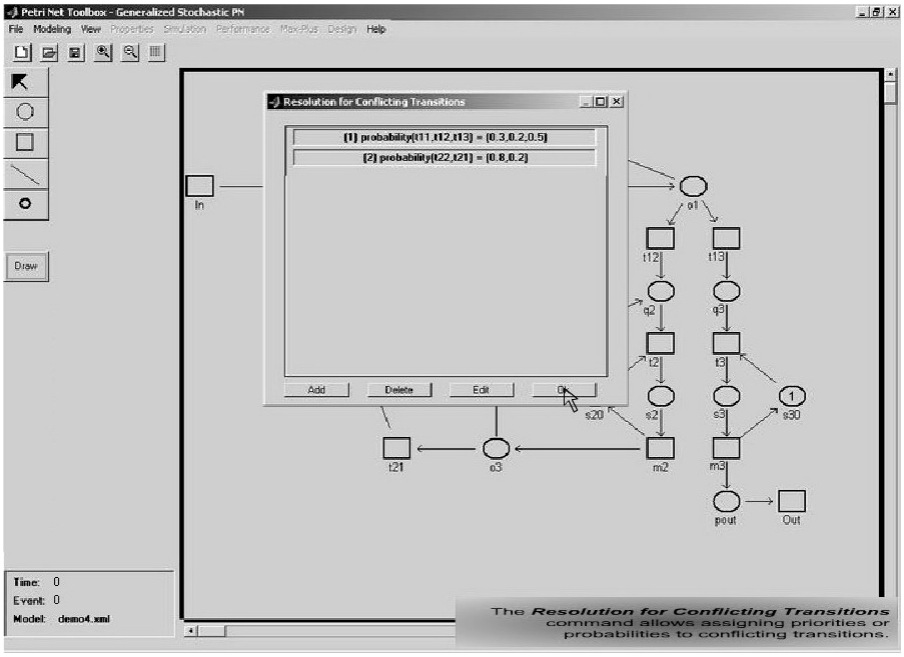


Petri Net Toolbox - Demo 4: Open markovian queueing network



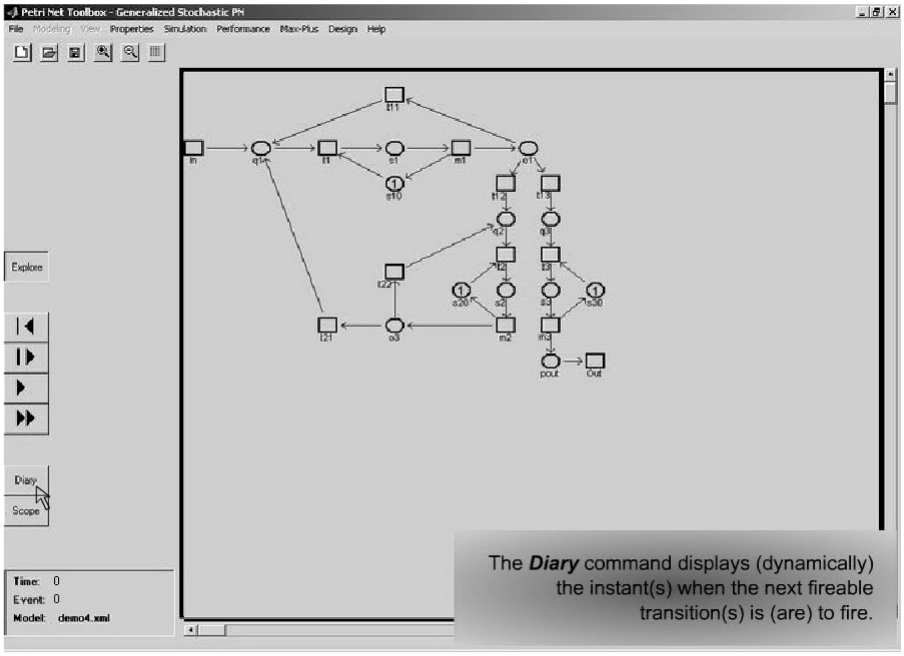
Petri Net Toolbox - Demo 4: Open markovian queueing network

7. Check the correctness of the settings



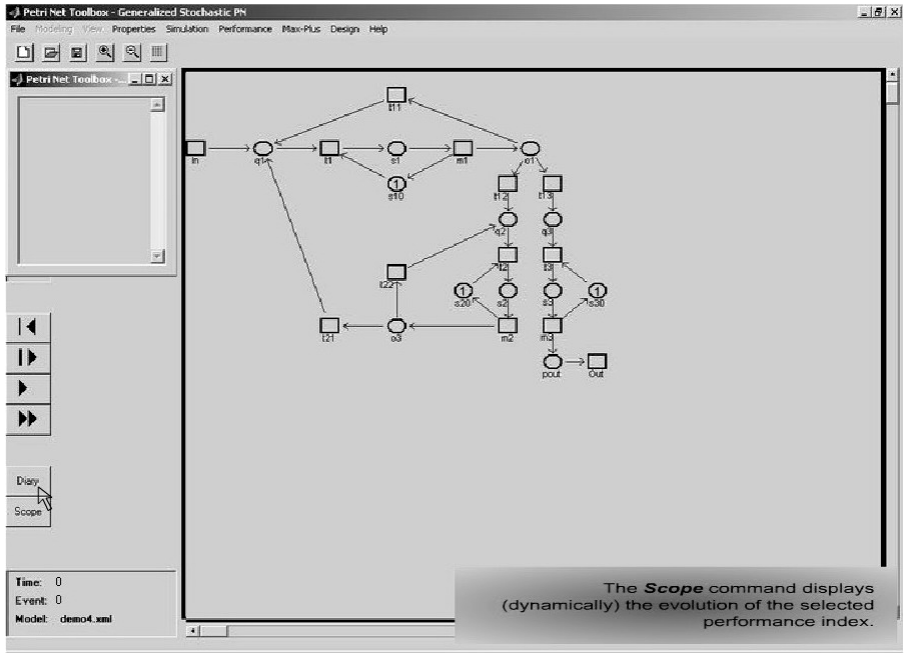
Petri Net Toolbox - Demo 4: Open markovian queueing network

8. Switch to ***Explore Mode*** and open the ***Diary*** window to watch the moments of firings



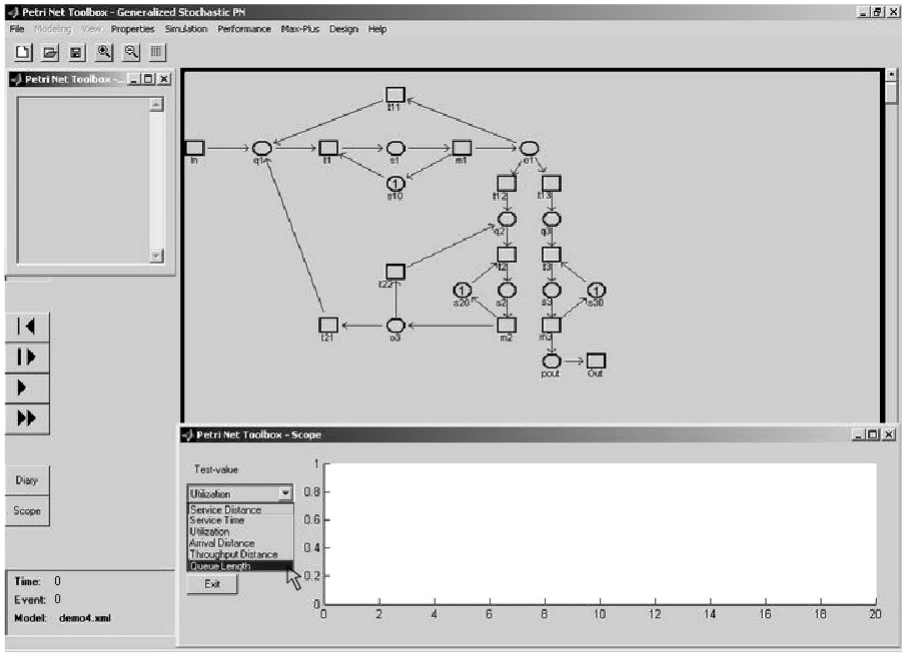
Petri Net Toolbox - Demo 4: Open markovian queueing network

9. Open the ***Scope*** window



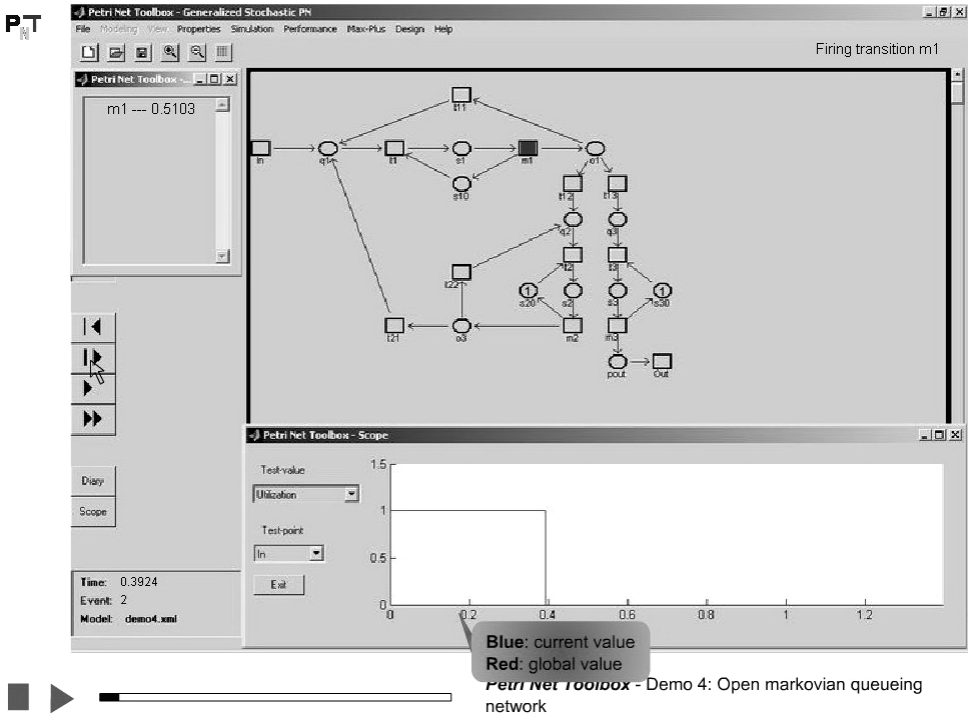
Petri Net Toolbox - Demo 4: Open markovian queueing network

10. Select the performance index to be plotted during simulation

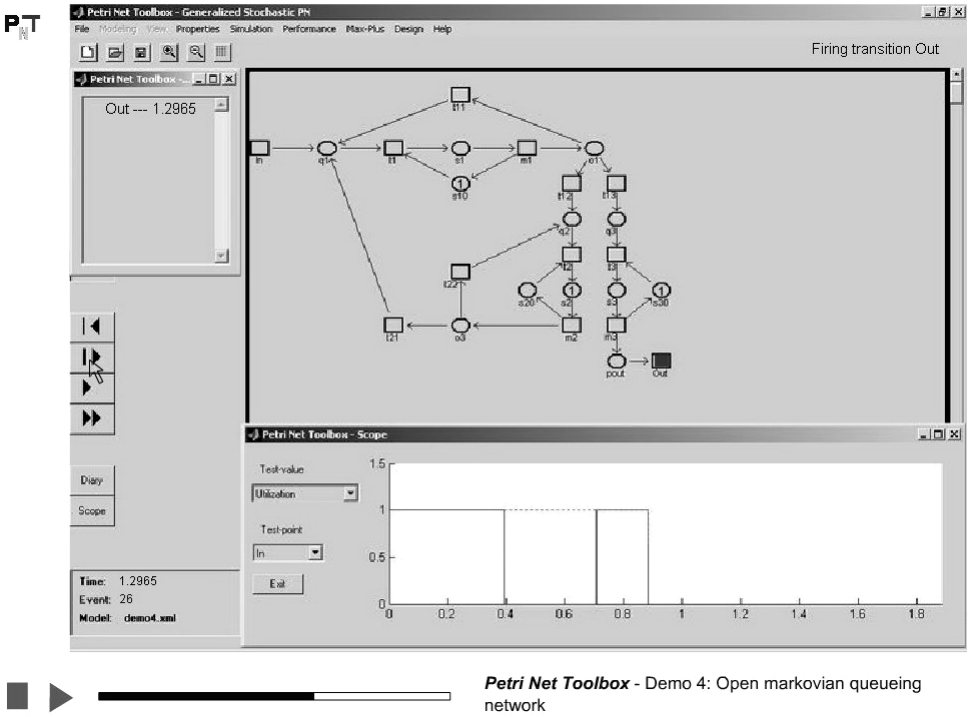


Petri Net Toolbox - Demo 4: Open markovian queueing network

11. Simulate the system in ***Run Slow*** mode

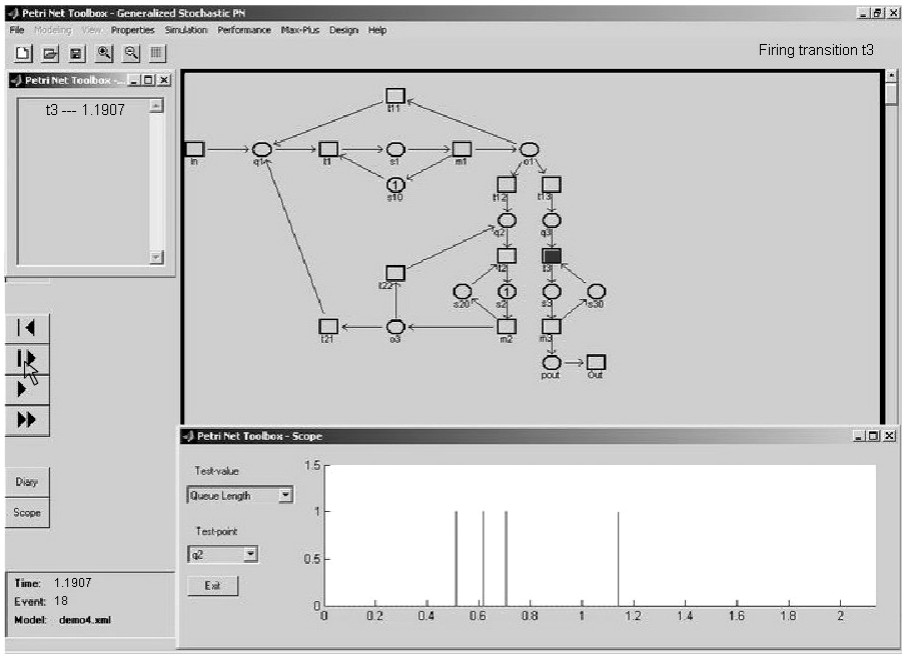


Petri Net Toolbox - Demo 4: Open markovian queueing network



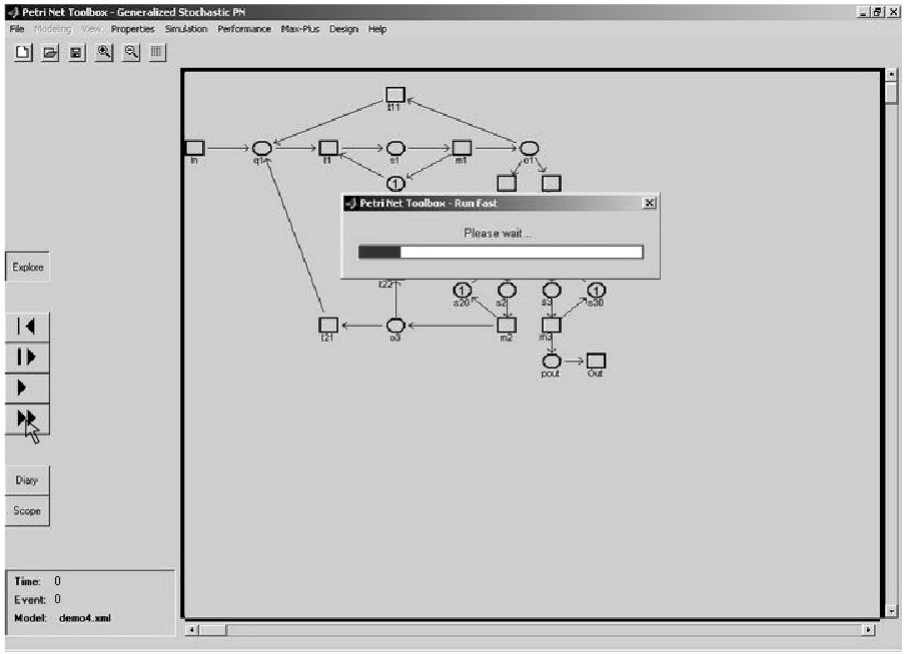
Petri Net Toolbox - Demo 4: Open markovian queueing network

12. Select another performance index and watch its evolution during simulation



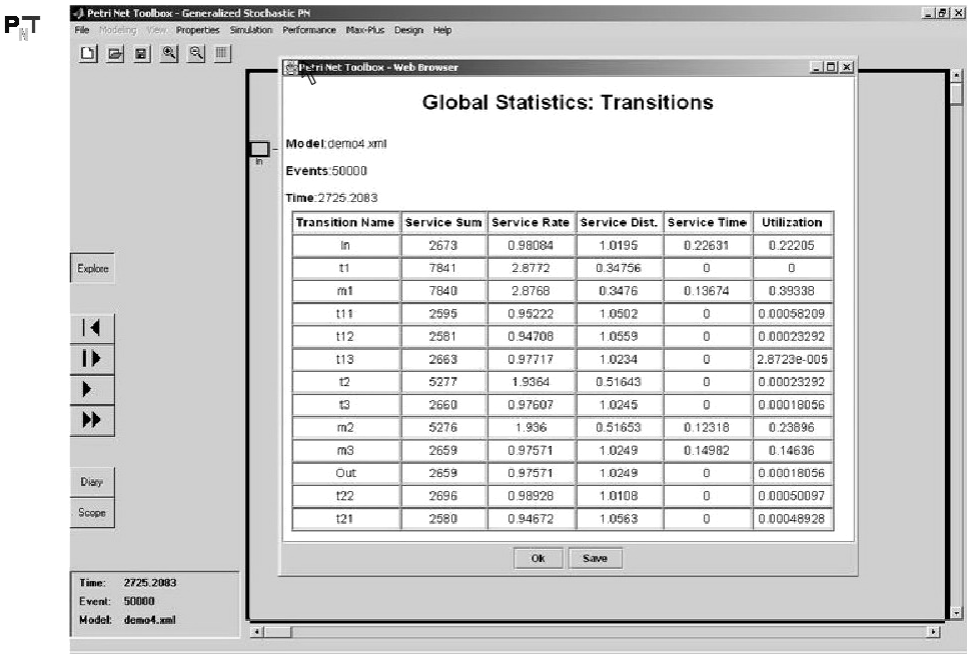
Petri Net Toolbox - Demo 4: Open markovian queueing network

13. Simulate the system in ***Run Fast*** mode



Petri Net Toolbox - Demo 4: Open markovian queueing network

14. Analyze the global performance indices for all the transitions or places in the model



Petri Net Toolbox - Demo 4: Open markovian queueing network

Demo 5

SOAKING PIT FURNACE

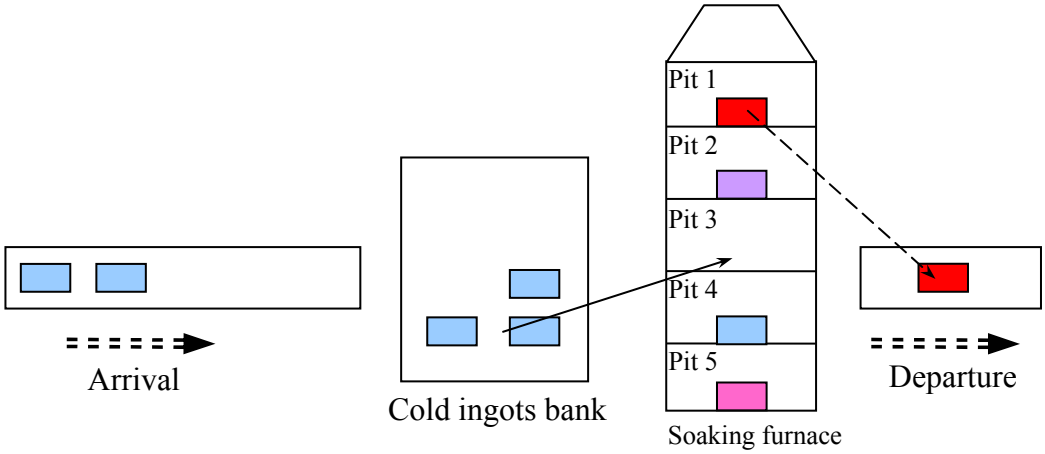
In a steel plant, steel ingots arrive in pairs at a soaking pit furnace where they are heated so they can be rolled in the next stage of the process. There is space for five ingots in the furnace. When an ingot arrives, it is placed into the furnace if space is available; otherwise, it is placed in the cold ingots bank to wait for free space. The interarrival times are exponentially distributed, the mean value being μ min. The initial furnace temperature is 2,200 degrees Fahrenheit. The furnace is heated according to the differential equation: $dF / dt = 2(2600 - F)$, where F is the furnace temperature. The initial temperature of an arriving ingot is uniformly distributed between 300 and 500 degrees Fahrenheit. When an ingot is inserted into the furnace, it reduces the furnace temperature by the difference between the furnace temperature and the ingot temperature, divided by the number of ingots in the furnace. The temperature change of ingots as they are heated in the furnace is described by the differential equation: $dP_j / dt = 0.15(F - P_j)$, where P_j is the temperature of the ingot in the j-th position in the pit. Each ingot is heated in the furnace until it reaches 2,200 degrees and then it is removed.

This demo illustrates:

- the construction of a synchronized PN model
- the definition of the events that trigger the firing of transitions in the PN model
- the definition of the output for a PN Simulink Block
- how to debug a Simulink model of a hybrid system

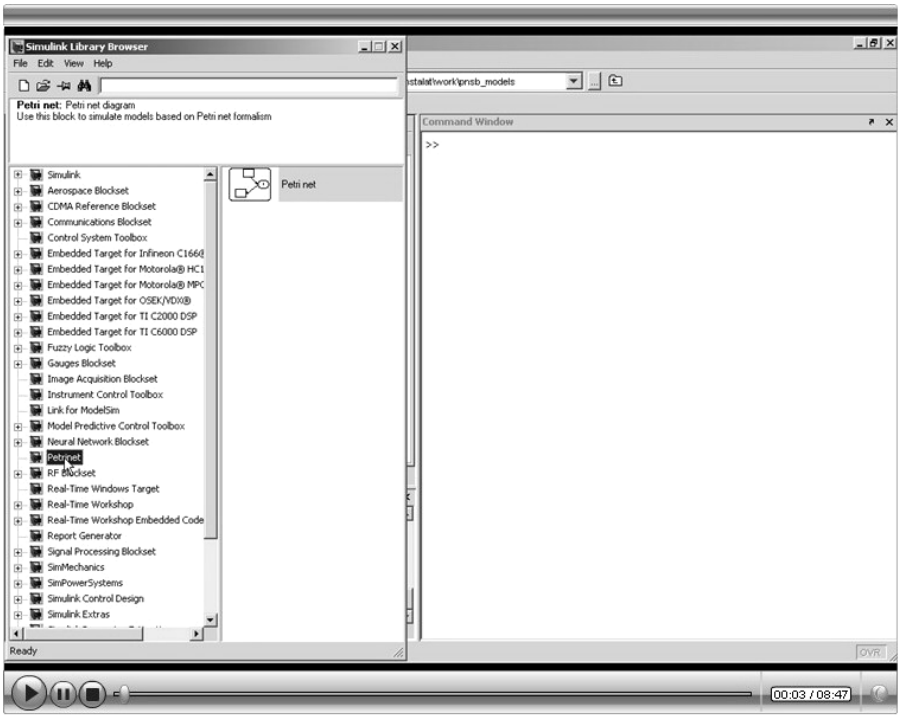
D5.1. Illustration of the Physical System

1. The soaking pit furnace

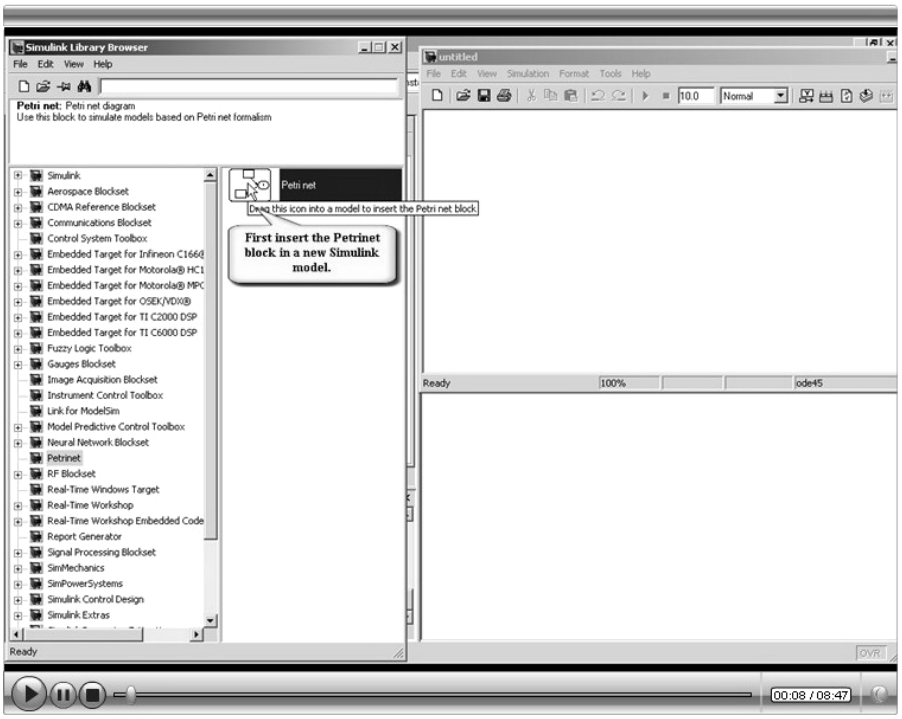


D5.2. Usage of the PN Simulink Block for System Modeling and Analysis

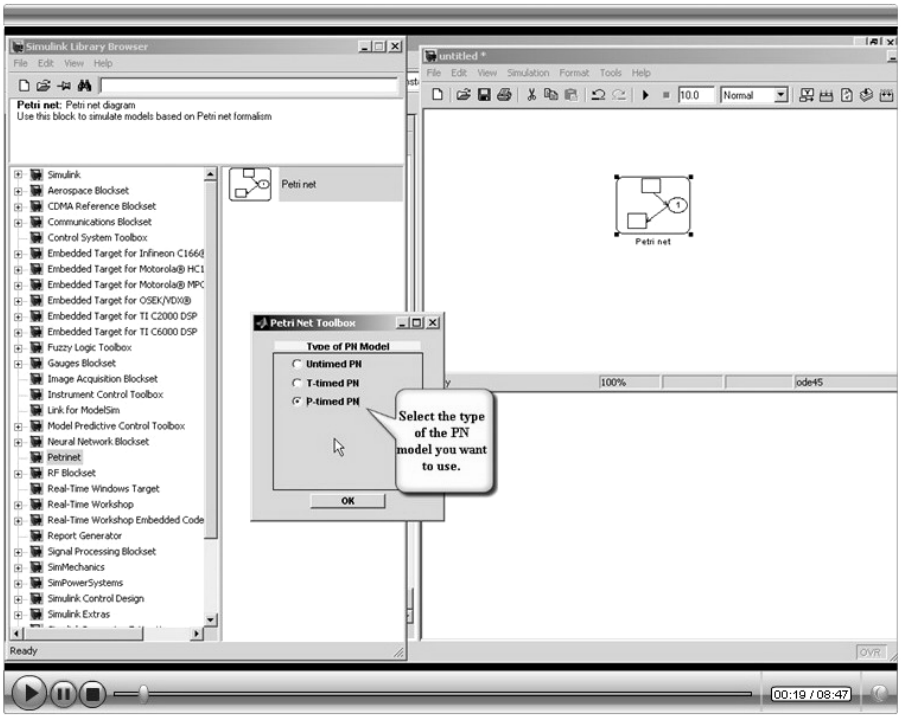
1. Locate the PN Simulink Block in the Simulink Library browser



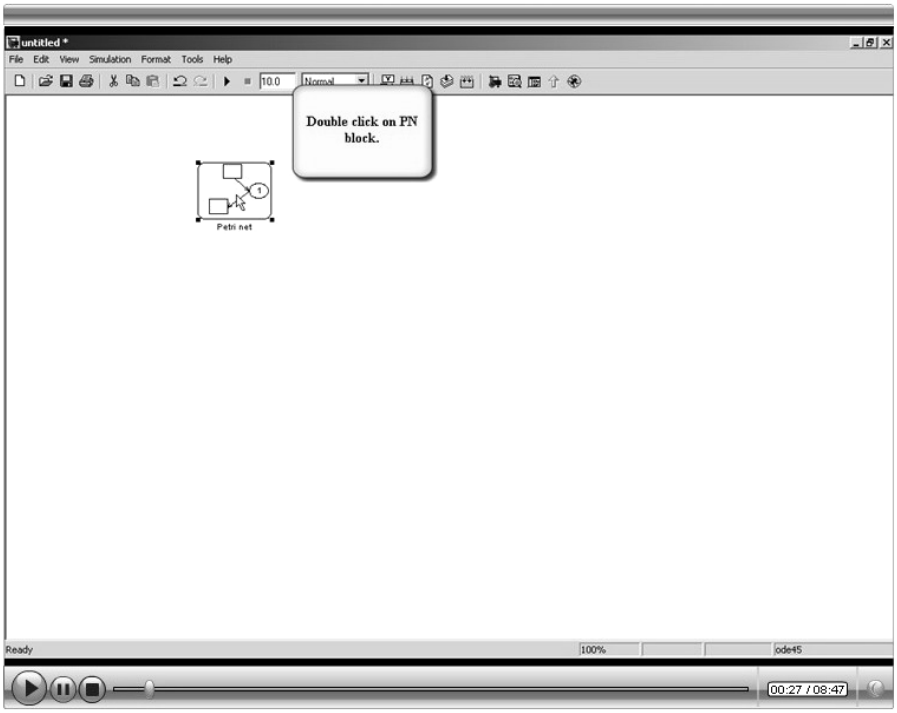
2. Open a new Simulink model and insert a PN block



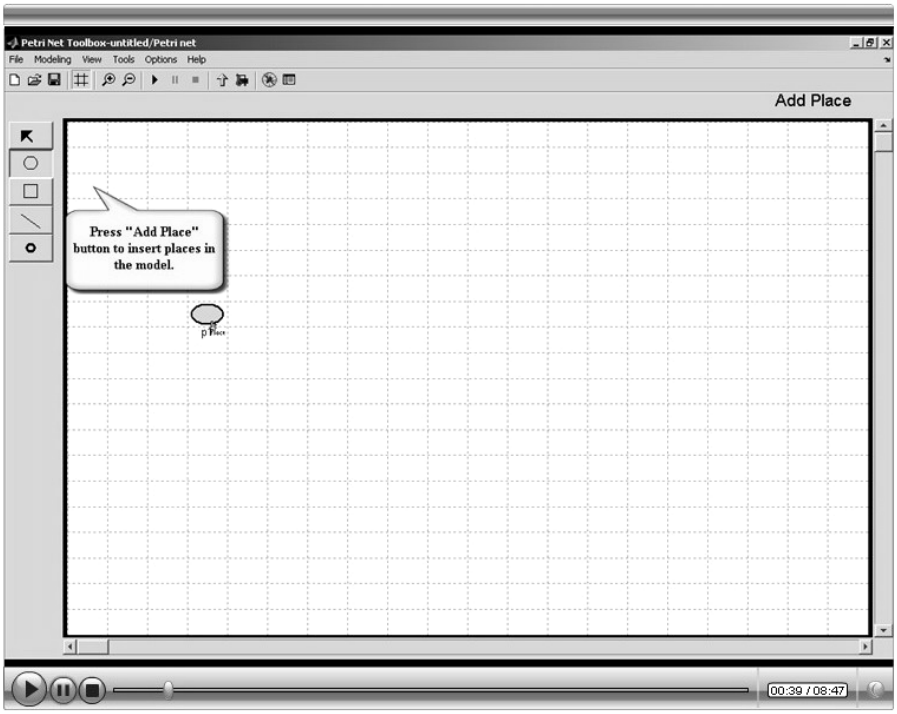
3. Select the type of the PN model to be created



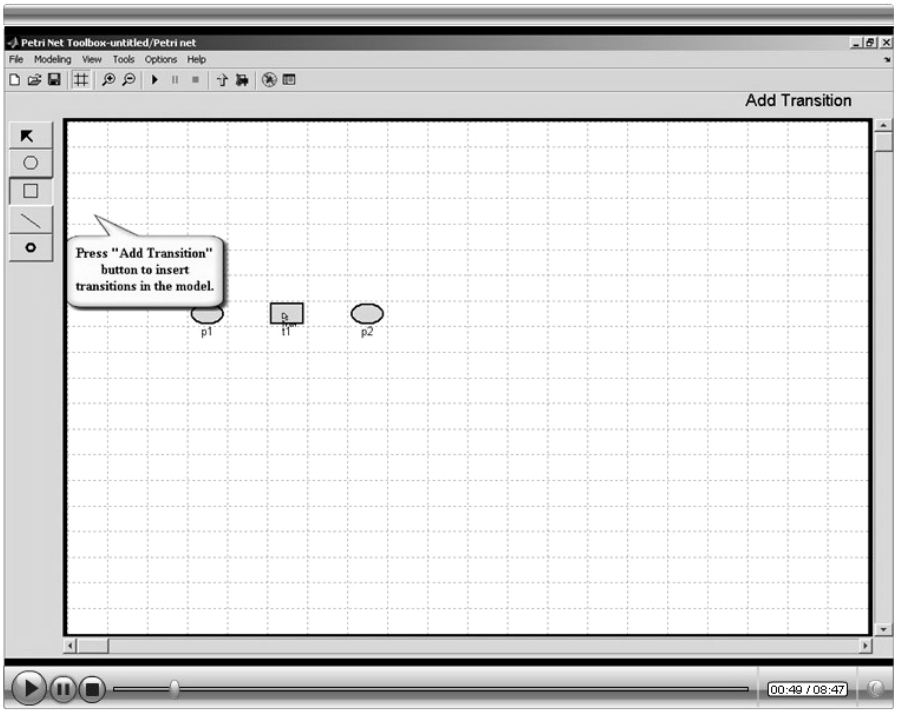
4. Double click on the PN block to open the graphical interface



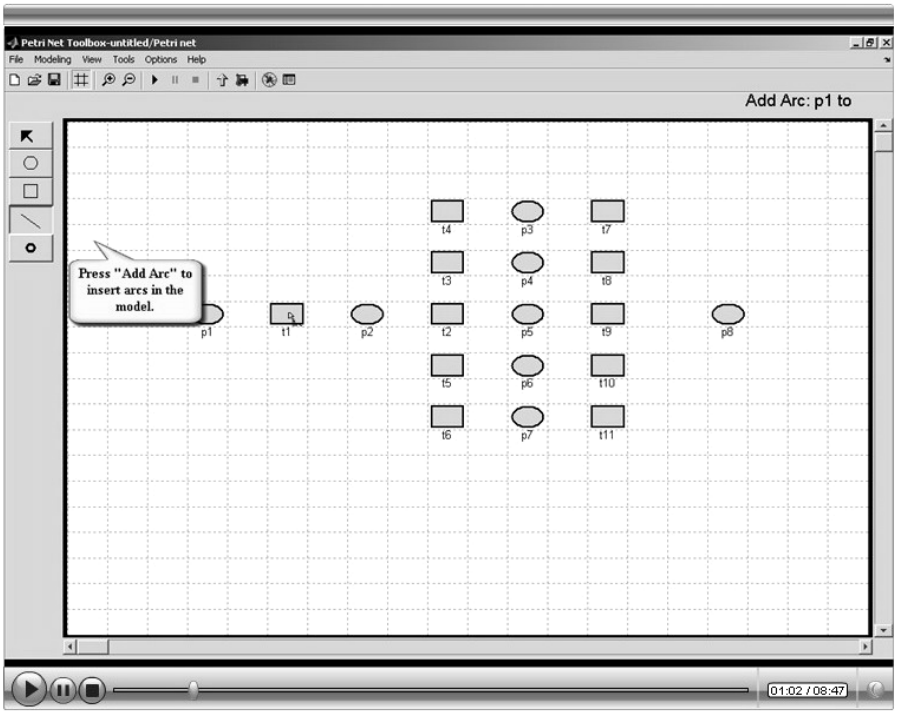
5. Click the ***Add Place*** button to insert places in the model



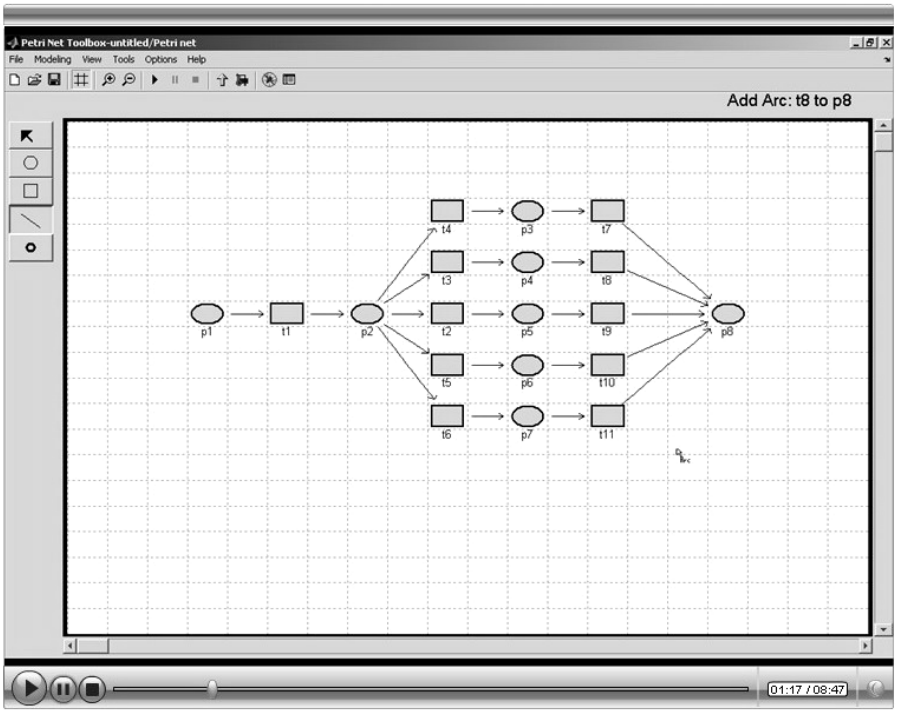
6. Click the ***Add Transition*** button to insert transitions in the model



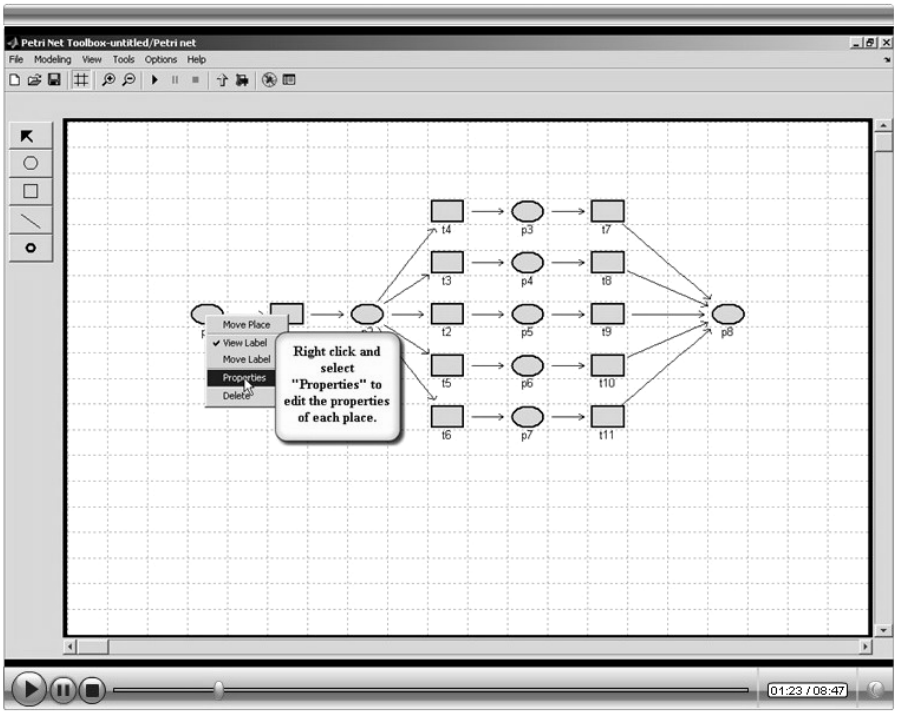
7. Click the ***Add Arc*** button to insert arcs in the model



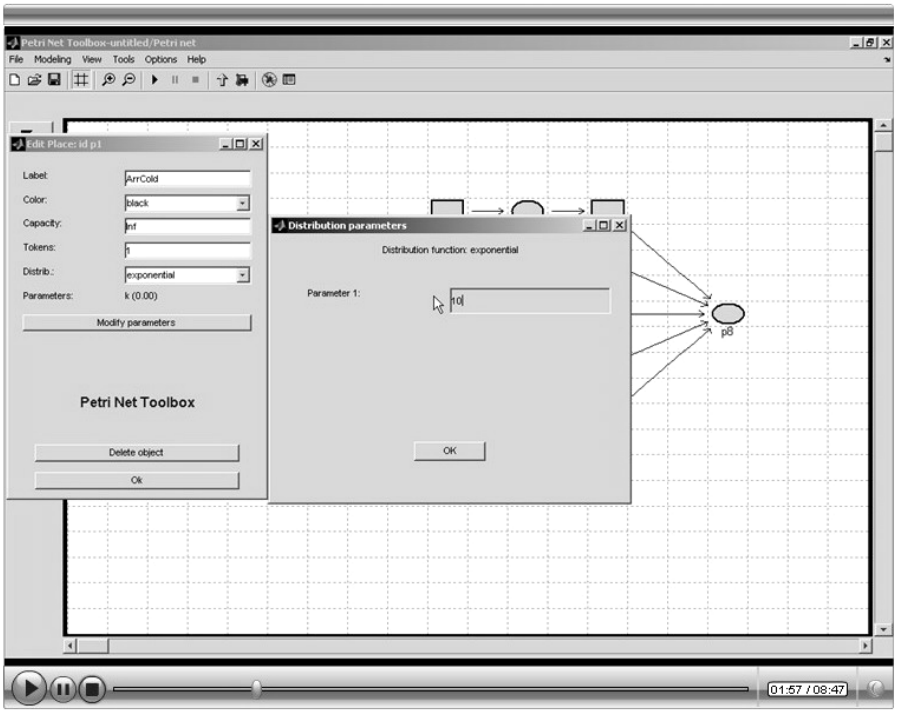
8. Examine the topology of the model



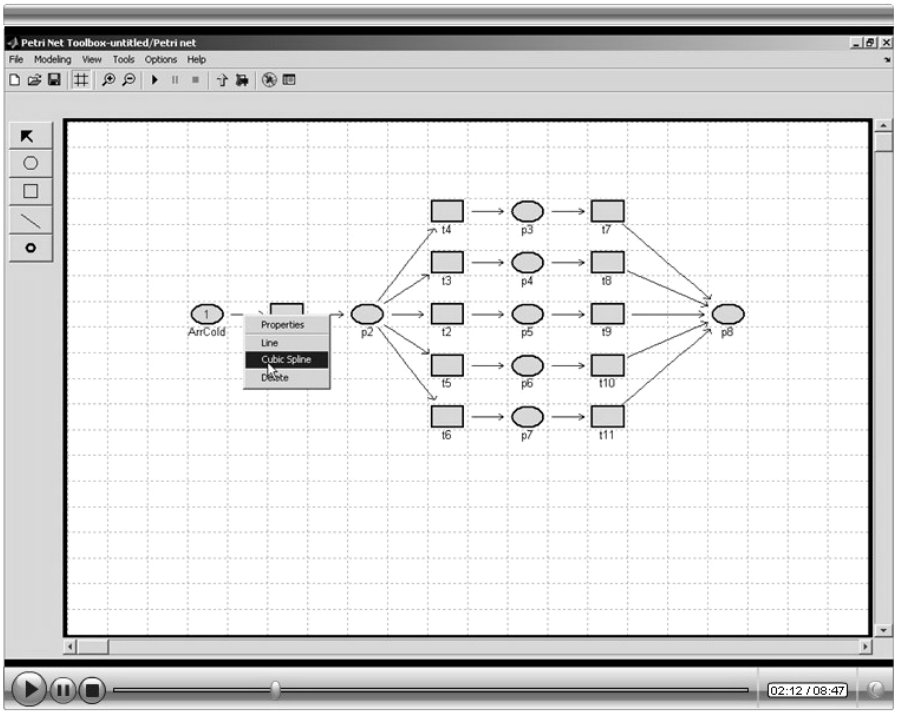
9. Edit the properties of each place in the net



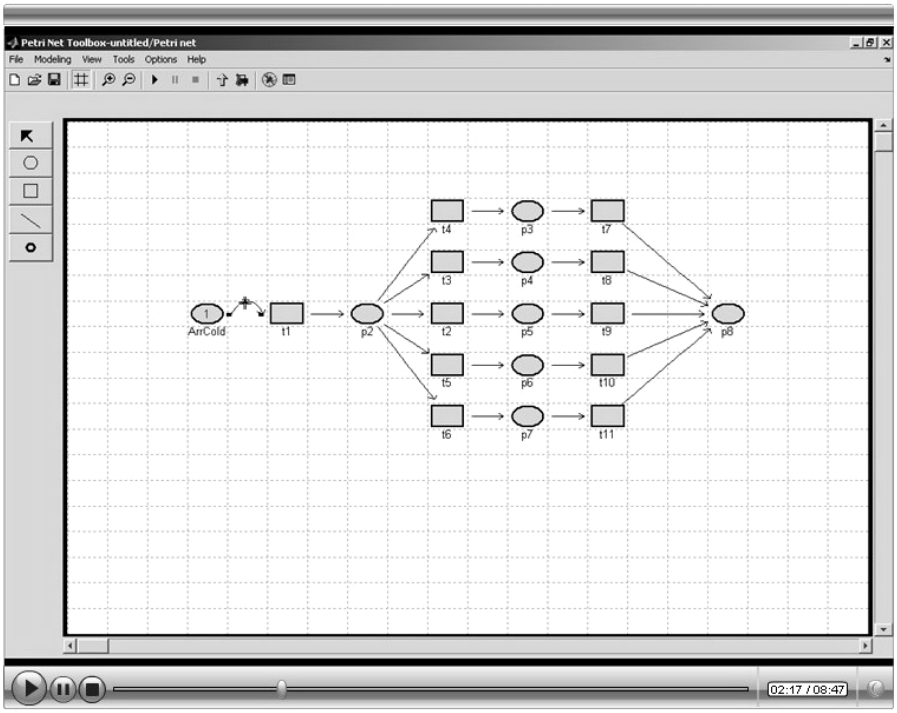
10. For each place of a P-timed model set the probability distribution and its parameters



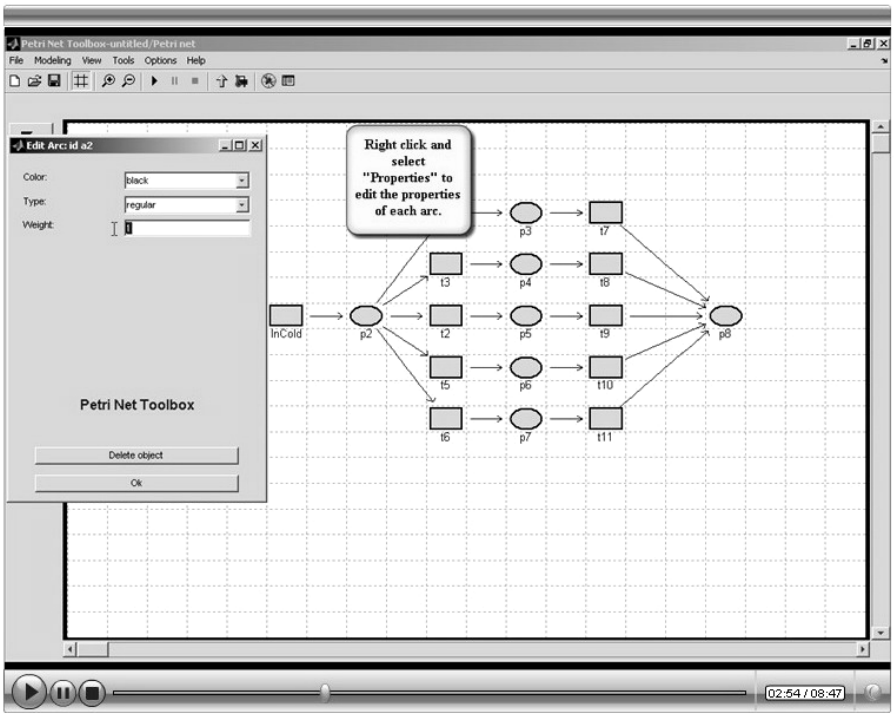
11. If necessary, represent the arcs in the model as cubic spline curves



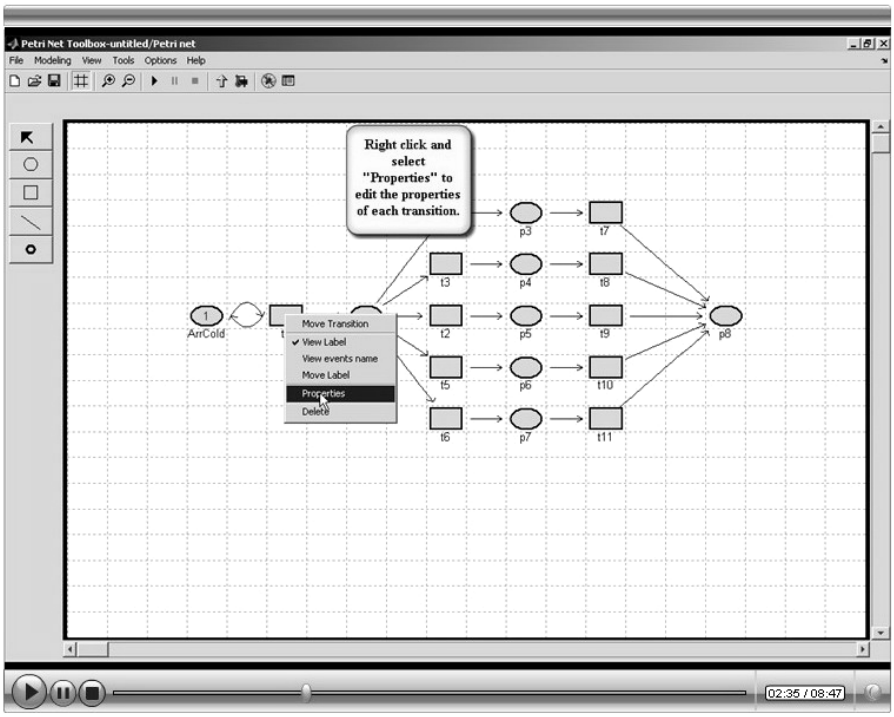
12. Drag the marks of the arc in the desired position



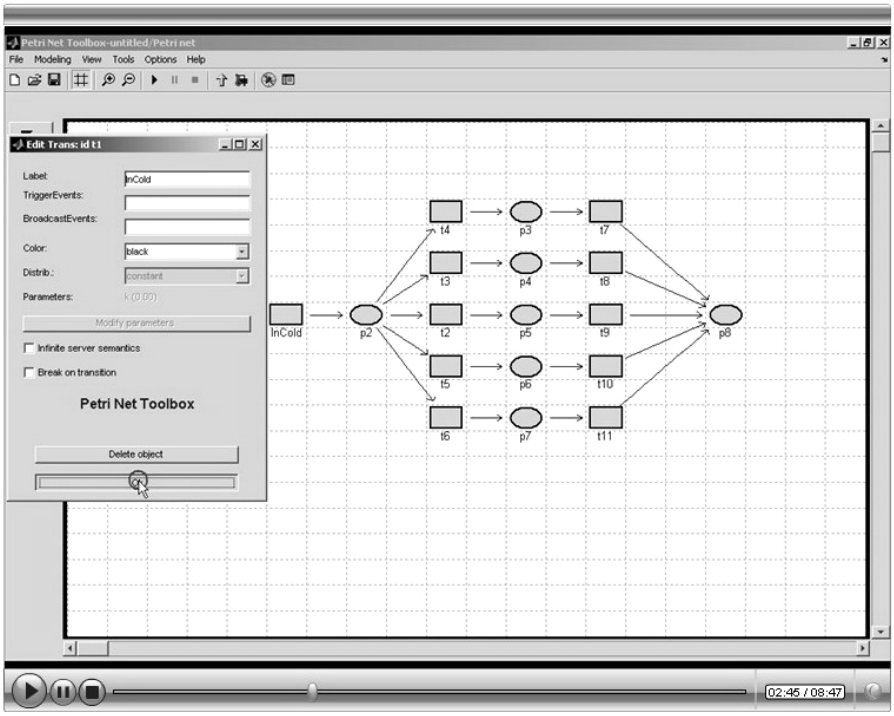
13. Set the weights of the arcs in the model, if different from 1



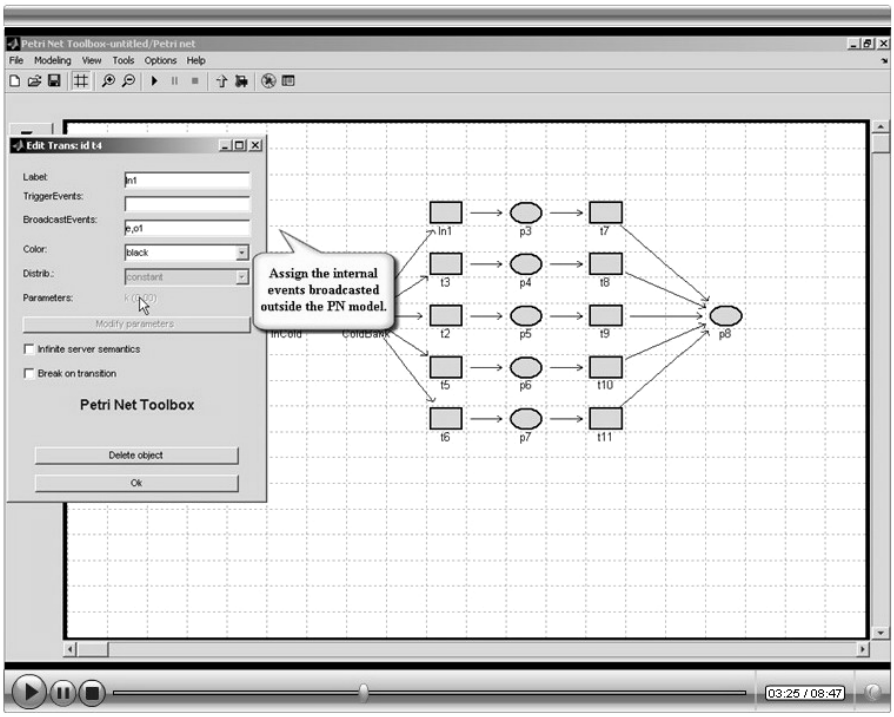
14. Edit the properties of the transitions in the model



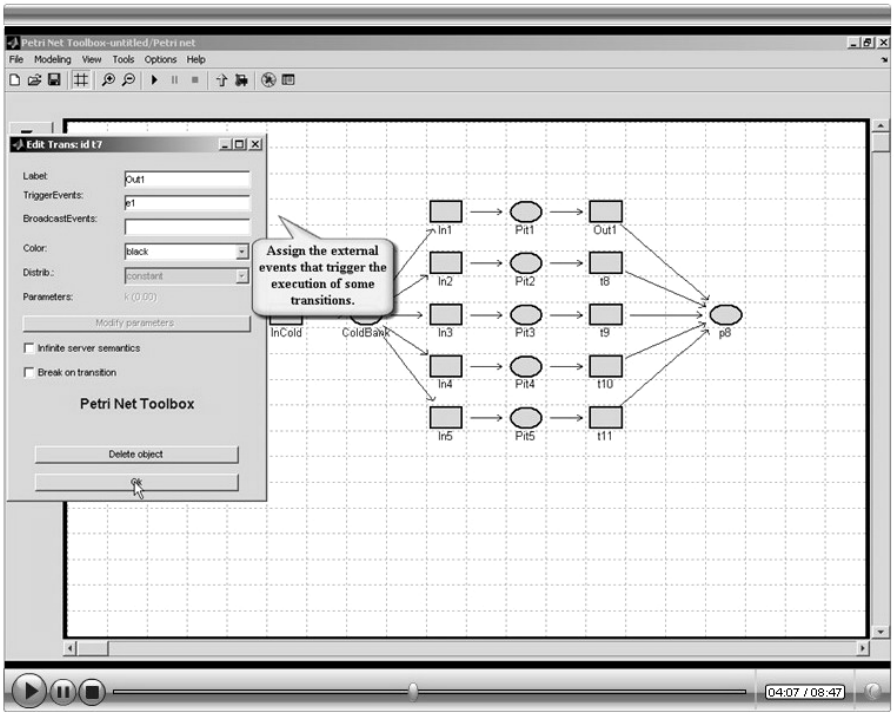
15. Change the default name of a transition so as to reflect its physical significance



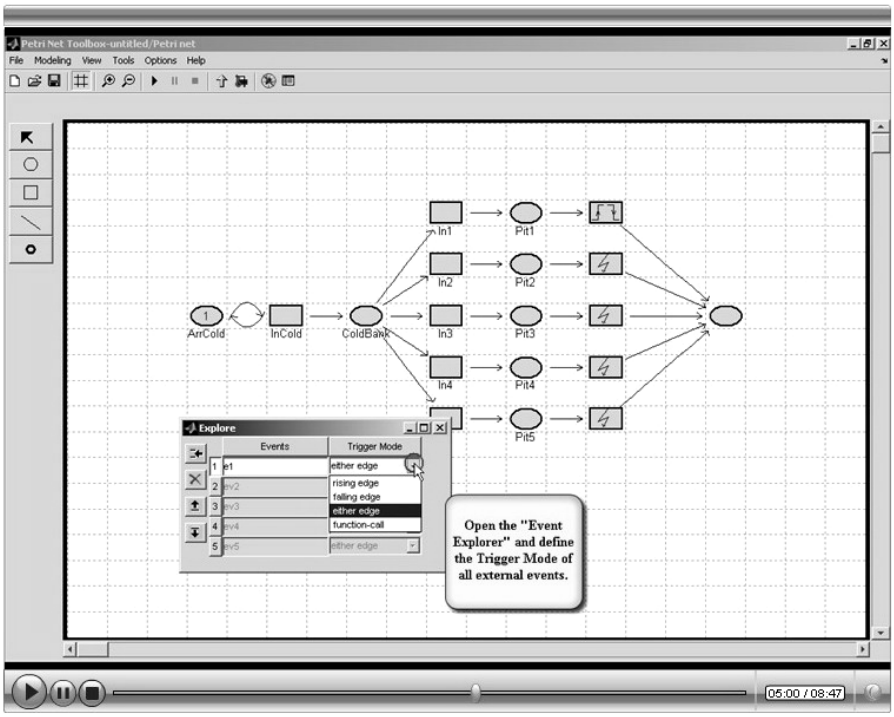
16. Assign the internal events to be broadcasted outside the PN model



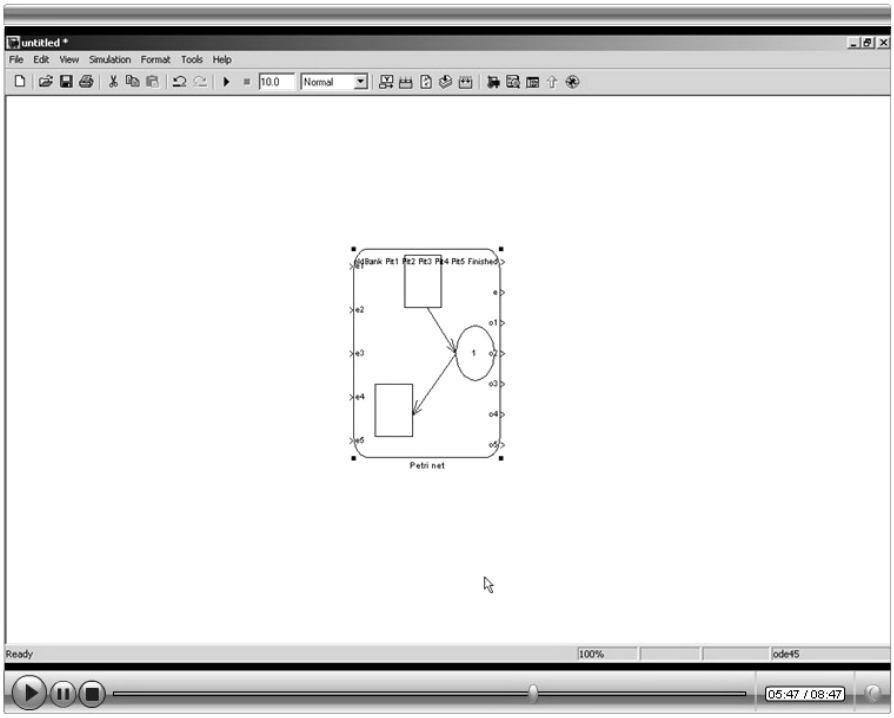
17. Assign the external events that trigger the synchronized transitions in the model



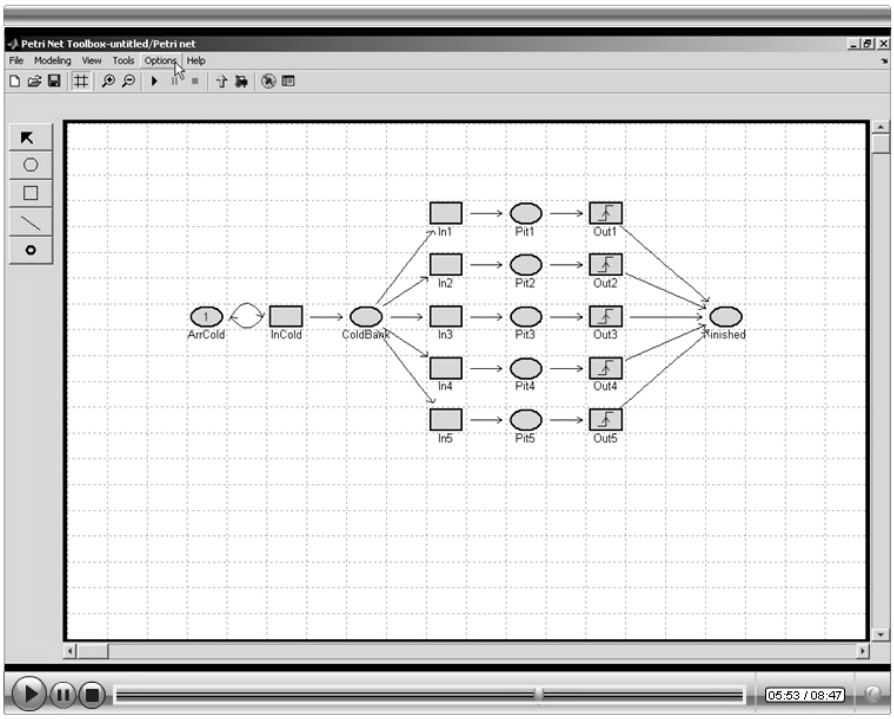
18. Open the ***Event Explorer*** to define the trigger mode of each external event



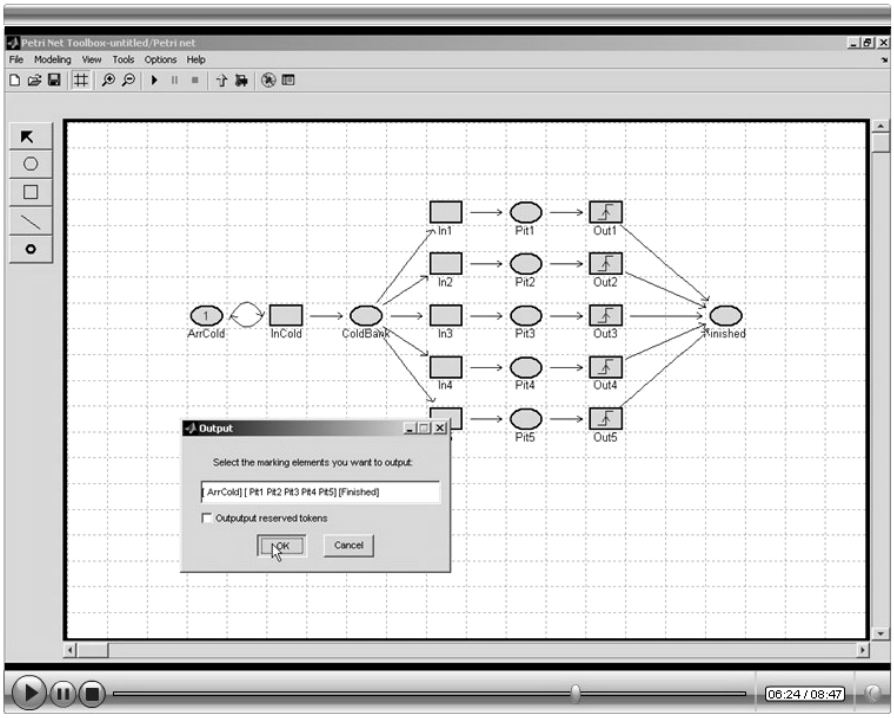
19. The default outputs of the PNSB are the internal events and the marking vector of the PN



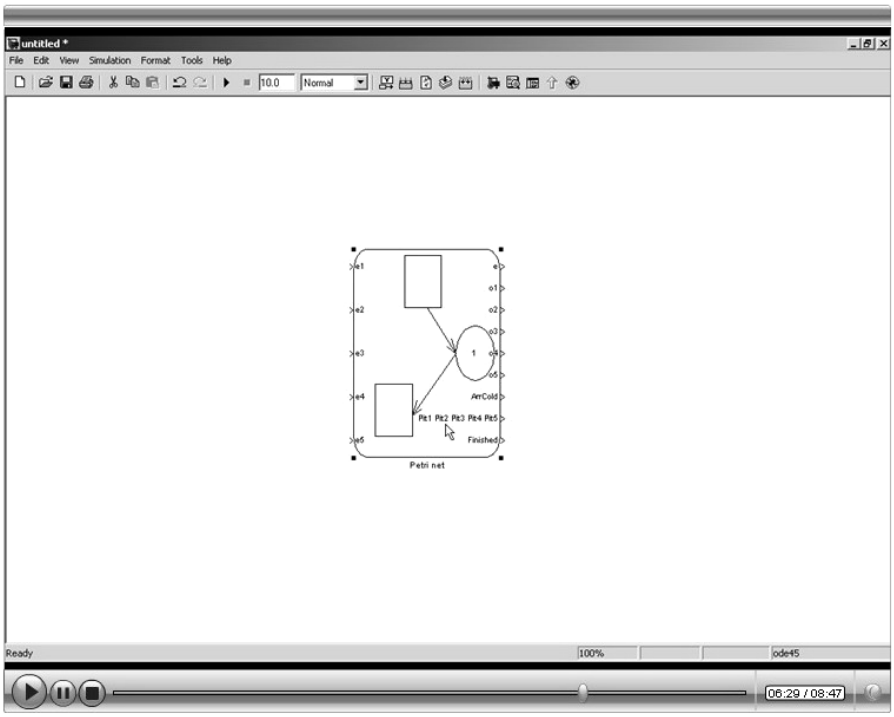
20. Select the ***Output*** command to define the places whose marking is sent out of the PNSB



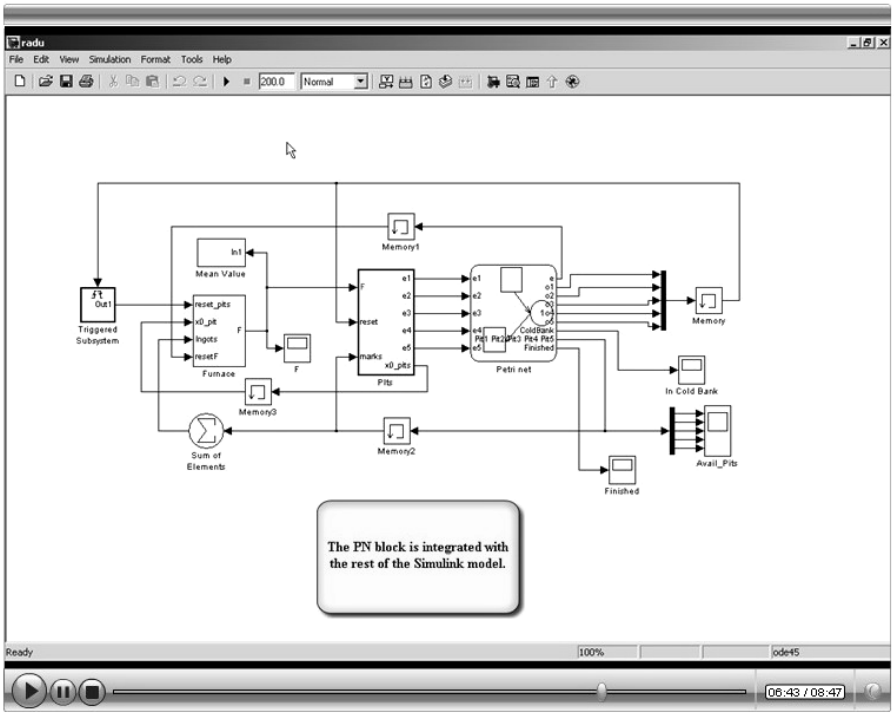
21. Select the places whose marking you consider as outputs of the PNSB



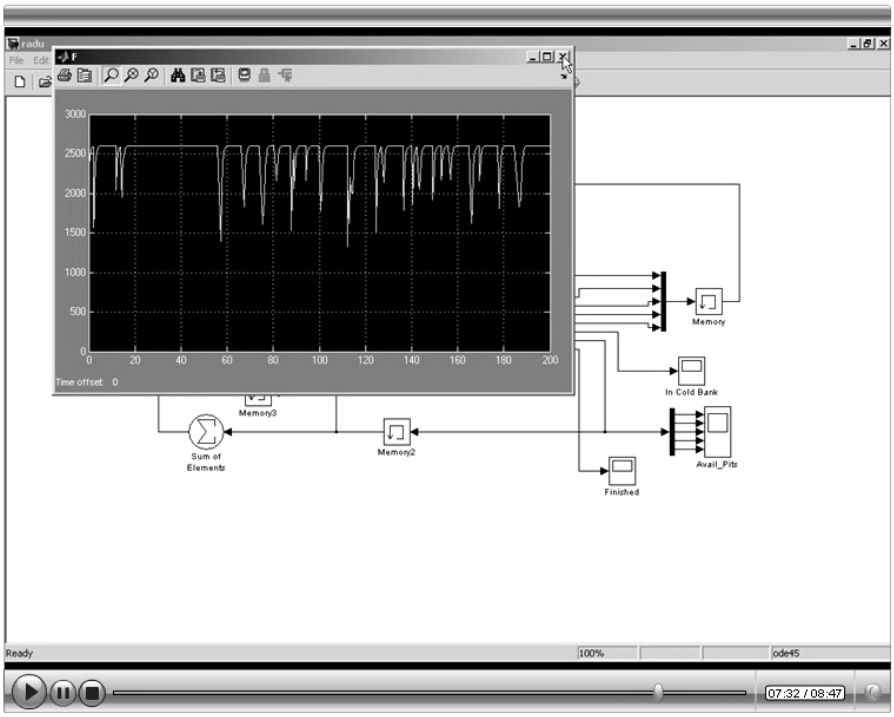
22. Notice the change in the number and significance of the outputs of the PNSB



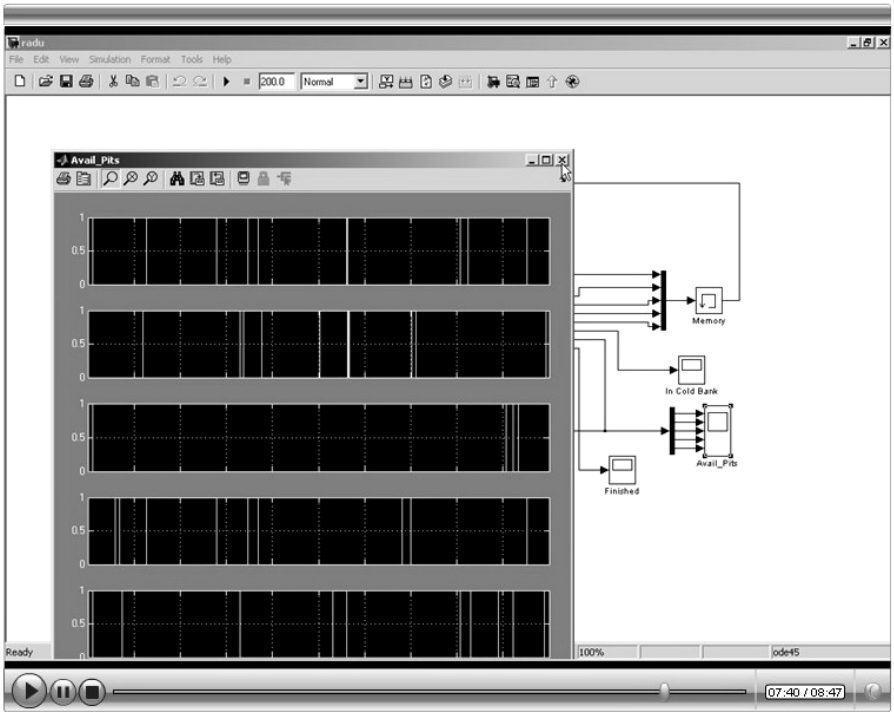
23. Integrate the PN block with the whole Simulink model



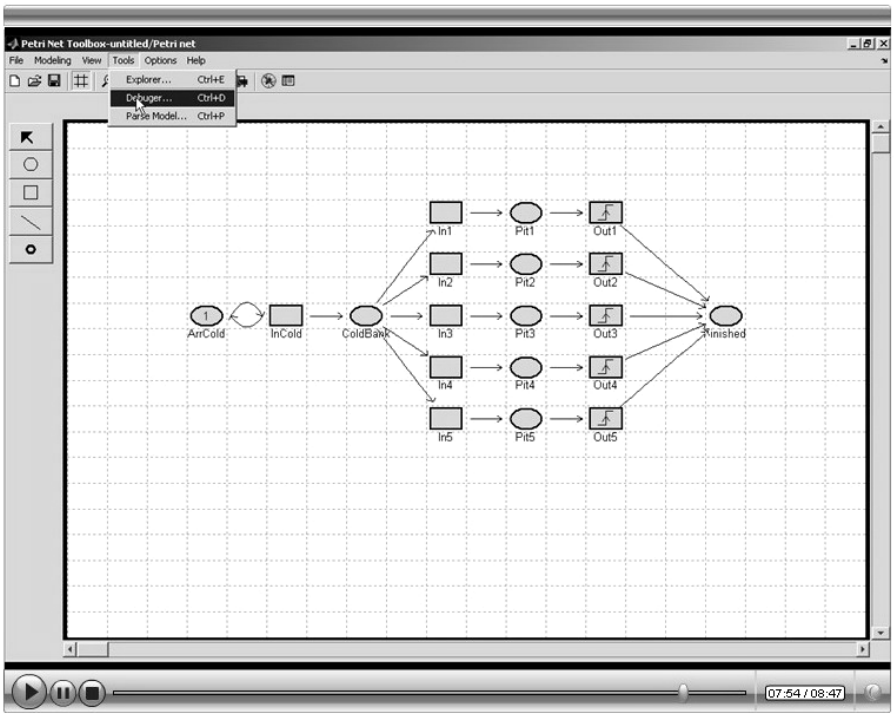
24. Simulate the Simulink model and visualize the evolution of some variables



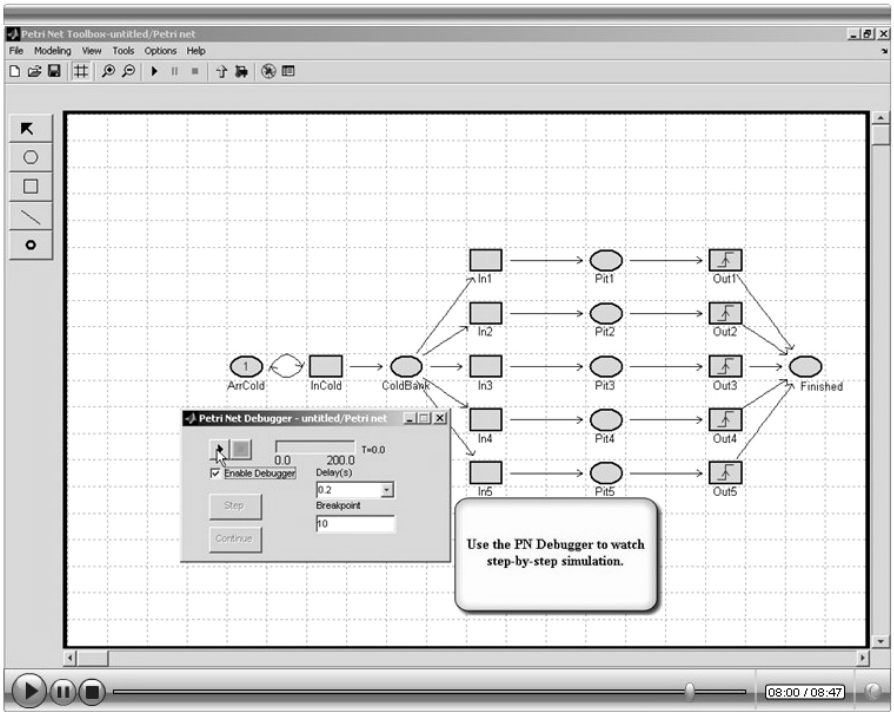
25. Visualize the evolution of the markings of some relevant places of the PN model



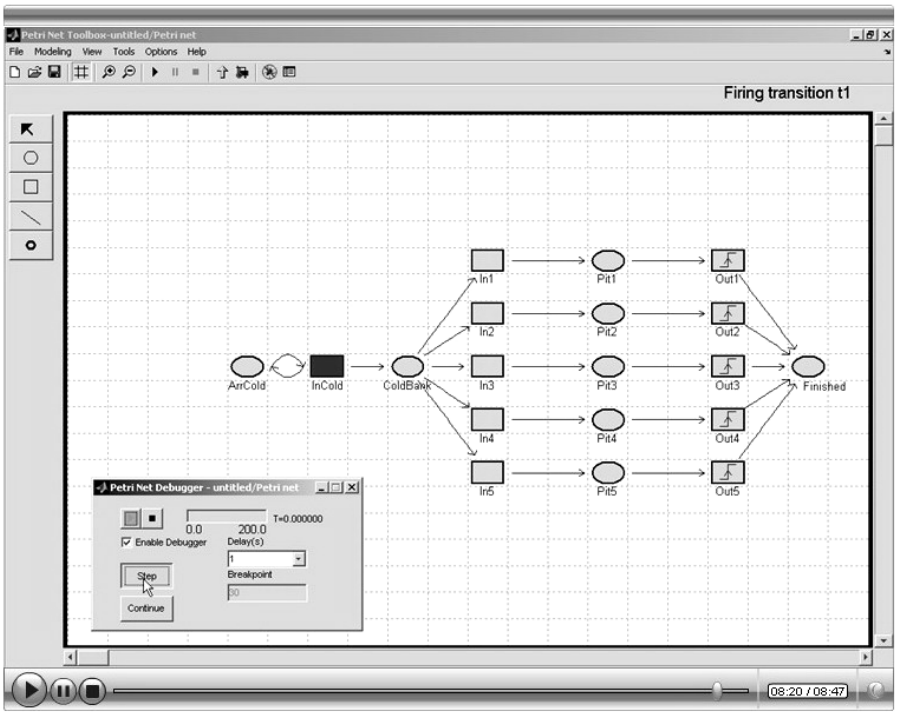
26. Open the ***Debugger***

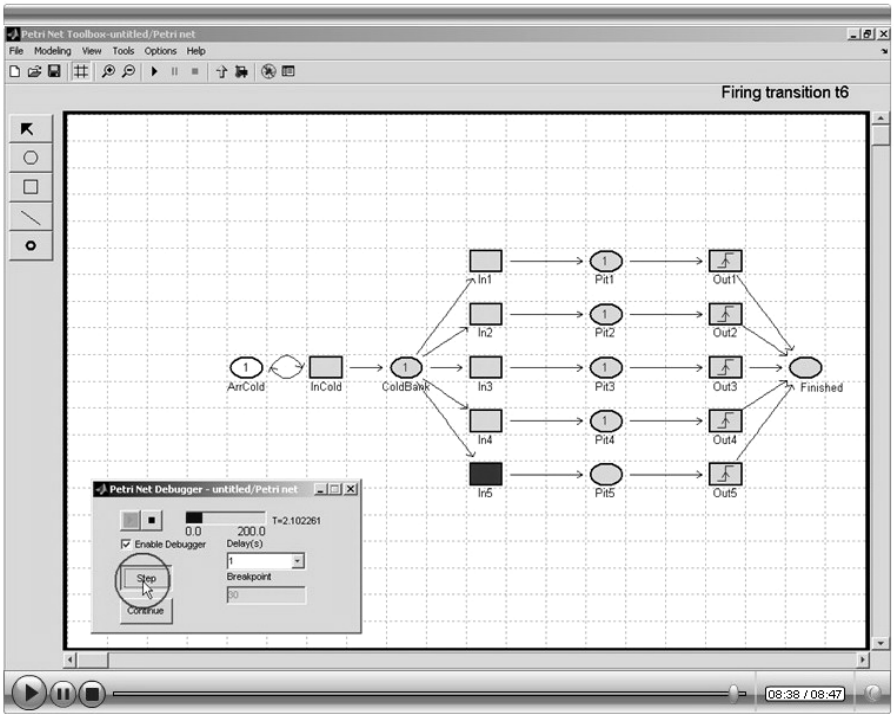


27. Enable the usage of the debugger



28. Visualize a step by step simulation of the Simulink model





Appendices

Appendix A

XML FILE-FORMAT OF A PN MODEL

A.1. XML File-format Used in the PN Toolbox

A.1.1. Description of Tags

The ***PN Toolbox*** uses an XML file for saving a model. This file is automatically generated by the ***PN Toolbox*** when the ***Save*** or ***Save As…*** commands are used. It can also be edited by using any text editor, if the format described below is respected.

The root tag for the PN model is **PNToolbox**. The tags for global information are:

- **Model_name** – the name of the file containing the model, including the extension ".xml";
- **Type** – a number that represents the PN type which can take one of the following values: 1 – untimed PN, 2 – T-timed PN, 3 – P-timed PN, 4 – stochastic PN, 5 – generalized stochastic PN;
- **Seed** – positive integer used to initialize the random number generator of MATLAB;
- **Place** – the necessary information related to a place of the model, depending on the PN type;
- **Transition** – the necessary information related to a transition of the model, depending on the PN type;
- **Arc** – the information related to an arc of the model;
- **Probability** – the information related to the firing probabilities set for a group of conflicting transitions, if any;
- **Priority** – the information related to the firing priorities set for a group of conflicting transitions, if any.

The **Place** tag corresponding to a place in the PN model has the following structure:

- **Id** – the unique identifier of the place; a string in the form "px" with "x" a distinct positive integer;
- **Value** – the coordinates of the cell of the ***Drawing Area*** that contains the place (a couple of integers between 0 and 49);
- **Color** – a string representing the MATLAB color used for drawing the place in the ***Drawing Area***;
- **Label** – additional information used for the graphical representation of the place:
  - **Name** – label of the place that is shown in the ***Drawing Area*** (any string character);
  - **Offset** – a couple of real values representing the offset from the default position of the label (bottom of place);
  - **Visible** – boolean variable corresponding to the status of the label visibility ("yes" or "no").
- **InitialMarking** – initial marking of the place;
- **Capacity** – capacity of the place;
- **Time** – information corresponding to the probability distribution of the time duration assigned to the place; used only in the case of P-timed PNs:
  - **Distribution** – name of the MATLAB distribution function;
  - **Parameters** – values for the parameters of the selected distribution function.

The **Transition** tag corresponding to a transition in the PN model has the following structure:

- **Id** – the unique identifier of the transition; a string in the form "ty" with "y" a distinct positive integer;
- **Value** – the coordinates of the cell of the ***Drawing Area*** that contains the transition (a couple of integers between 0 and 49);
- **Color** – a string representing the MATLAB color used for drawing the transition in the ***Drawing Area***;
- **Message** – a string representing the message displayed by the ***PN Toolbox*** in the ***Message Box*** when the transition is fired;
- **Label** – additional information used for the graphical representation of the transition:
  - **Name** – label of the transition that is shown in the ***Drawing Area*** (any string character);
  - **Offset** – a couple of real values representing the offset from the default position of the label (bottom of transition);
  - **Visible** – boolean variable corresponding to the status of the label visibility ("yes" or "no").
- **Time** – information corresponding to the probability distribution of the time duration assigned to the transition; used only in the case of T-timed, stochastic and generalized stochastic PNs:
  - **Distribution** – name of the MATLAB distribution function;
  - **Parameters** – values for the parameters of the selected distribution function;

- **Marking_Dependent** – boolean variable showing the dependence of the firing rate of the transition on the marking of its input places ("yes" or "no"); used only in the case of stochastic and generalized stochastic PNs.

The **Arc** tag corresponding to an arc in the PN model has the following structure:

- **Id** – the unique identifier of the arc; a string in the form "az" with "z" a distinct positive integer;
- **From** – the **Id** of the departure node;
- **To** – the **Id** of the arrival node;
- **Style** – arc style (1 – regular, 2 – bidirectional, 3 – inhibitor);
- **Type** – type of graphical representation of the arc (1 – line, 2 – cubic spline);
- **Cubic** – the coordinates in the ***Drawing Area*** of the four points defining the spline curve; used only for the arcs with type = 2;
- **Color** – a string representing the MATLAB color used for drawing the arc in the ***Drawing Area***;
- **Weight** – the weight of the arc.

The **Probability** tag has the following structure:

- **Transitions** – the Ids of the conflicting transitions (separated by comma) subject to the probability assignment;
- **Values** – the corresponding values of the firing probabilities (separated by comma); nonnegative values whose sum equals 1.

The **Priority** tag has the following structure:

- **Transitions** – the **Id**s of the conflicting transitions (separated by comma) subject to the priority assignment;
- **Values** – the corresponding values of the firing priorities (separated by comma); nonnegative values, 0 meaning the highest priority.

A.1.2. Example

The XML file corresponding to the PN model in figure A.1 is given below.

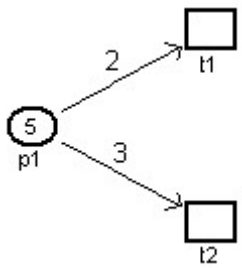


Fig. A.1. *ThePN model used for illustrating the XML file-format used by the **PN Toolbox**.*

```
<?xml version="1.0"?>
<PNToolbox>
	<Model_name>XMLexample.xml</Model_name>
	<Type>2</Type>  <!-- T-timed PN -->
	<Seed>66</Seed> <!-- initial seed -->
	<Place> <!-- place definition -->
		<Id>p1</Id> <!-- place's id -->
		<Value>5,43</Value> <!-- coordinates in the Drawing Area -->
		<Color>black</Color> <!-- drawing color -->
		<Label> <!-- information of the label -->
			<Name>p1</Name> <!-- string label -->
			<Offset>0.50,-0.20</Offset> <!-- the offset of the representation –>
			<Visible>yes</Visible> <!-- the visibility of the label -->
		</Label>
		<InitialMarking>5</InitialMarking> <!-- initial marking of the place-->
		<Capacity>Inf</Capacity> <!-- capacity of the place-->
	</Place>
	<Transition> <!—transition's definition-->
		<Id>t1</Id> <!-- transition's id-->
		<Value>8,45</Value> <!-- coordinates in the Drawing Area-->
		<Color>black</Color> <!-- drawing color-->
		<Message>Firing transition t1</Message> <!-- message displayed when it is fired-->
		<Label> <!-- information of the label-->
			<Name>t1</Name> <!-- string label-->
			<Offset>0.50,-0.20</Offset> <!-- the offset of the representation-->
			<Visible>yes</Visible> <!-- the visibility of the label-->
		</Label>
		<Time> <!-- information for time distribution function>
			<Distribution>constant</Distribution> <!-- constant distribution-->
			<Parameters>3</Parameters><!-- duration equal to 3-->
		</Time>
	</Transition>
	<Transition> <!-- second transition's definition -->
		<Id>t2</Id>
		<Value>8,41</Value>
		<Color>black</Color>
		<Message>Firing transition t2</Message>
		<Label>
			<Name>t2</Name>
			<Offset>0.50,-0.20</Offset>
			<Visible>yes</Visible>
		</Label>
		<Time>
			<Distribution>cont. uniform</Distribution> <!-- uniform distribution between 1 and 5-->
			<Parameters>1,5</Parameters>
		</Time>
```

```
        </Transition>
        <Arc> <!-- arc definition>
                <Id>a1</Id> <!-- arc's ID-->
                <From>p1</From> <!-- departure node -->
                <To>t1</To> <!-- arrival node -->
                <Style>1</Style> <!-- regular arc -->
                <Type>1</Type> <!—the graphical representation is a line -->
                <Color>black</Color> <!-- drawing color -->
                <Weight>2</Weight> <!-- arc's weight -->
        </Arc>
        <Arc>
                <Id>a2</Id>
                <From>p1</From>
                <To>t2</To>
                <Style>1</Style>
                <Type>1</Type>
                <Color>black</Color>
                <Weight>3</Weight>
        </Arc>
        <Probability> <!-- probability for conflicting transitions t1 and t2; 25% for t1 and 75% for t2 -->
                <Transitions>t1,t2</Transitions>
                <Values>0.25,0.75</Values>
        </Probability>
</PNToolbox>
```

A.2. XML File-format Used in the PNSB

A.2.1. Description of Tags

The ***PN Toolbox*** uses an XML file for saving a model. This file is automatically generated by the ***PN Toolbox*** when the ***Save*** or ***Save As…*** commands are used. It can also be edited by using any text editor, if the format described below is respected.

The root tag for the PN model is **PNToolbox**. The tags for global information are:

- **Model_name** – the name of the file containing the model, including the extension ".xml";
- **Type** – a number that represents the PN type which can take one of the following values: 1 – untimed PN, 2 – T-timed PN, 3 – P-timed PN;
- **Seed** – a positive integer used to initialize the random number generator of MATLAB;
- **Place** – the necessary information related to a place of the model, depending on the PN type;
- **Transition** – the necessary information related to a transition of the model, depending on the PN type;
- **Arc** – the information related to an arc of the model;

- **Probability** – the information related to the firing probabilities set for a group of conflicting transitions, if any;
- **Priority** – the information related to the firing priorities set for a group of conflicting transitions, if any;
- **Event** – the information related to the events that trigger different transitions in the PN model.

The **Place** tag corresponding to a place in the PN model has the following structure:

- **Id** – the unique identifier of the place; a string in the form "px" with "x" a distinct positive integer;
- **Value** – the coordinates of the cell of the ***Drawing Area*** that contains the place (a couple of integers between 0 and 49);
- **Color** – a string representing the MATLAB color used for drawing the place in the ***Drawing Area***;
- **Label** – additional information used for the graphical representation of the place:
  - **Name** – label of the place that is shown in the ***Drawing Area*** (any string character);
  - **Offset** – a couple of real values representing the offset from the default position of the label (bottom of place);
  - **Visible** – boolean variable corresponding to the status of the label visibility ("yes" or "no").
- **InitialMarking** – initial marking of the place;
- **Capacity** – capacity of the place;
- **Time** – information corresponding to the probability distribution of the time duration assigned to the place; used only in the case of P-timed PNs:
  - **Distribution** – name of the MATLAB distribution function;
  - **Parameters** – values for the parameters of the selected distribution function.

The **Transition** tag corresponding to a transition in the PN model has the following structure:

- **Id** – the unique identifier of the transition; a string in the form "ty" with "y" a distinct positive integer;
- **Value** – the coordinates of the cell of the ***Drawing Area*** that contains the transition (a couple of integers between 0 and 49);
- **Color** – a string representing the MATLAB color used for drawing the transition in the ***Drawing Area***;
- **Message** – a string representing the message displayed by the ***PN Toolbox*** in the ***Message Box*** when the transition is fired;
- **BroadcastEvent** – string containing the names of the associated broadcasted events, separated by commas.
- **TriggerEvent** – string containing the names of the triggering events, separated by commas.
- **ViewEvent** –boolean variable corresponding to the status of events' names visibility ("yes" or "no").

- **Label** – additional information used for the graphical representation of the transition:
  - **Name** – label of the transition that is shown in the ***Drawing Area*** (any string character);
  - **Offset** – a couple of real values representing the offset from the default position of the label (bottom of transition);
  - **Visible** – boolean variable corresponding to the status of the label visibility ("yes" or "no").
- **Time** – information corresponding to the probability distribution of the time duration assigned to the transition; used only in the case of T-timed, stochastic and generalized stochastic PNs:
  - **Distribution** – name of the MATLAB distribution function;
  - **Parameters** – values for the parameters of the selected distribution function;
  - **Marking_Dependent** – boolean variable showing the dependence of the firing rate of the transition on the marking of its input places ("yes" or "no"); used only in the case of stochastic and generalized stochastic PNs.

The **Arc** tag corresponding to an arc in the PN model has the following structure:

- **Id** – the unique identifier of the arc; a string in the form "az" with "z" a distinct positive integer;
- **From** – the **Id** of the departure node;
- **To** – the **Id** of the arrival node;
- **Style** – arc style (1 – regular, 2 – bidirectional, 3 – inhibitor);
- **Type** – type of graphical representation of the arc (1 – line, 2 – cubic spline);
- **Cubic** – the coordinates in the ***Drawing Area*** of the four points defining the spline curve; used only for the arcs with type = 2;
- **Color** – a string representing the MATLAB color used for drawing the arc in the ***Drawing Area***;
- **Weight** – the weight of the arc.

The **Probability** tag has the following structure:

- **Transitions** – the Ids of the conflicting transitions (separated by comma) subject to the probability assignment;
- **Values** – the corresponding values of the firing probabilities (separated by comma); nonnegative values whose sum equals 1.

The **Priority** tag has the following structure:

- **Transitions** – the **Id**s of the conflicting transitions (separated by comma) subject to the priority assignment;
- **Values** – the corresponding values of the firing priorities (separated by comma); nonnegative values, 0 meaning the highest priority.

The **Event** tag has the following structure:

- **Name** – the name of the event;

- **Trig** – the corresponding type of triggering, 1 – rising edge, 2 – falling edge and 3 – either edge.
- **Out** – boolean variable corresponding to the type of event: 0 – triggering event, 1 – broadcasted (output) event;

The **Output** tag contains the vectors with the labels of the places whose marking is supplied outside the PN Simulink Block

A.2.2. Example

The XML file corresponding to the PN model in figure A.2 is given below.

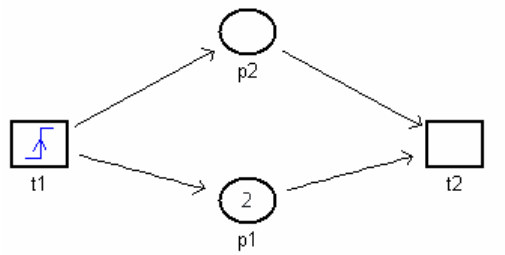


Fig. A.2. *ThePN model used for illustrating the XML file-format used by the **PN Simulink Block**.*

```
<?xml version="1.0"?>
<PNToolbox>
        <Model_name>pnsb_test_Petri net</Model_name>
        <Type>3</Type>
        <Seed>66</Seed>
        <Place>
                <Id>p1</Id>
                <Value>4,43</Value>
                <Color>black</Color>
                <Label>
                        <Name>p1</Name>
                        <Offset>0.50,-0.20</Offset>
                        <Visible>yes</Visible>
                </Label>
                <InitialMarking>2</InitialMarking>
                <Capacity>Inf</Capacity>
                <Time>
                        <Distribution>exponential</Distribution>
                        <Parameters>1</Parameters>
                </Time>
        </Place>
        <Place>
                <Id>p2</Id>
                <Value>4,45</Value>
```

```
            <Color>black</Color>
            <Label>
                  <Name>p2</Name>
                  <Offset>0.50,-0.20</Offset>
                  <Visible>yes</Visible>
            </Label>
            <InitialMarking>0</InitialMarking>
            <Capacity>Inf</Capacity>
            <Time>
                  <Distribution>constant</Distribution>
                  <Parameters>2</Parameters>
            </Time>
</Place>
<Transition>
            <Id>t1</Id>
            <Value>1,43</Value>
            <Color>black</Color>
            <Message>Firing transition t1</Message>
            <TriggerEvent>e1</TriggerEvent>
            <ViewEvent>no</ViewEvent>
            <Label>
                  <Name>t1</Name>
                  <Offset>0.50,-0.20</Offset>
                  <Visible>yes</Visible>
            </Label>
</Transition>
<Transition>
            <Id>t2</Id>
            <Value>8,43</Value>
            <Color>black</Color>
            <Message>Firing transition t2</Message>
            <BroadcastEvent>o1</BroadcastEvent>
            <ViewEvent>no</ViewEvent>
            <Label>
                  <Name>t2</Name>
                  <Offset>0.50,-0.20</Offset>
                  <Visible>yes</Visible>
            </Label>
</Transition>
<Arc>
            <Id>a1</Id>
            <From>t1</From>
            <To>p1</To>
            <Style>1</Style>
            <Type>1</Type>
            <Color>black</Color>
```

```
            <Weight>1</Weight>
        </Arc>
        <Arc>
            <Id>a2</Id>
            <From>p1</From>
            <To>t2</To>
            <Style>1</Style>
            <Type>1</Type>
            <Color>black</Color>
            <Weight>1</Weight>
        </Arc>
            <Arc>
            <Id>a3</Id>
            <From>t1</From>
            <To>p2</To>
            <Style>1</Style>
            <Type>1</Type>
            <Color>black</Color>
            <Weight>1</Weight>
        </Arc>
        <Arc>
            <Id>a4</Id>
            <From>p2</From>
            <To>t2</To>
            <Style>1</Style>
            <Type>1</Type>
            <Color>black</Color>
            <Weight>1</Weight>
        </Arc>
        <Event>
            <Name>e1</Name>
            <Trig>1</Trig>
            <Out>0</Out>
        </Event>
        <Event>
            <Name>o1</Name>
            <Trig>3</Trig>
            <Out>1</Out>
        </Event>
        <Output>[ p1] [ p2] </Output>
</PNToolbox>
```

Appendix B

CONFIGURATION FILE FOR THE PN TOOLBOX AND THE PNSB

The ***PN Toolbox*** and the ***PNSB*** use configuration files to store the global information needed for initialization. This files, called *PNTconfig.txt* and, respectively, *PNSBconfig.txt*, are stored in the folders where the ***PN Toolbox*** / ***PNSB*** are installed. The user can modify the values of all the parameters; the modifications being effective the next time when the graphical interface of is started.

The options used in the *PNTconfig.txt* / *PNSBconfig.txt* file are:

- **Seed** – the initial state of the *random number generator* version 5 from MATLAB. This parameter is called "Seed" for history reasons; first version of ***PN Toolbox*** used version 4 of the *random number generator* where the 'seed' option of the MATLAB **rand** function was used. The version 5 uses 'state' parameters of **rand** function. (this parameter can have any integer value);
- **DisplayMessages** – boolean value that gives the status of the transitions' messages visibility in the ***Message Box*** in ***Step*** and ***Run Slow*** simulation modes (1 – the messages are visible, 0 – the messages are invisible);
- **Background** – the background color for all MATLAB figures opened by the ***PN Toolbox*** (a vector with three elements ranging from 0 to 1);
- **HTMLBackground** –the background color for all HTM files opened by the ***PN Toolbox*** (this parameter is given in hexadecimal);
- **HTMLText** – the text color for all HTML files opened by the ***PN Toolbox*** (given in hexadecimal);

- **Distributions** – probability distributions, with positive support, that can be used for timed PNs (constant, continuous uniform, exponential, lognormal, gamma, beta, Weibull);
- **Distributions_Parameters** – the number of parameters used by each probability distribution listed in **Distributions** tag (1, 2, 1, 2, 2, 2, 2);
- **Distributions_Key** – one letter symbols used to represent each probability distribution listed in **Distributions** tag (k, u, e, l, g, b, w);
- **Breakpoint** – the default breakpoint for ***Run Fast*** simulation mode (1 – Number of events, 2 – Simulation time, 3 – Firings for transition, 4 – Tokens arrived in place);
- **Breakpoint_Event** – default number of events, if breakpoint is selected on *Number of events*;
- **Breakpoint_Time** – default simulation time, if breakpoint is selected on *Simulation time*;
- **Breakpoint_Transition** – default index of the transition in the transitions array, if breakpoint is selected on *Firings for transition*;
- **Breakpoint_ServiceSum** – the number of firings for the **Breakpoint_Transition**;
- **Breakpoint_Place** – default index of the place in the places array, if breakpoint is selected on *Tokens arrived in place*;
- **Breakpoint_ArrivalSum** – the number of tokens arrived in the **Breakpoint_Place**;
- **Color_Wait_Token** – MATLAB color used for the graphical representation of places that contain reserved tokens (only in the case of *P*-timed PNs);
- **Color_Fire_Transition** – MATLAB color used for the graphical representation of firing transitions;
- **ColorPlaces** – MATLAB color used for drawing the places;
- **ColorTransitions** – MATLAB color used for drawing the transitions;
- **ColorArcs** – MATLAB color used for drawing the arcs.
- **NewFileName** – the default name of the new model, including extension (.xml);
- **Projects_Save** – string with a MATLAB expression (constructed with the **fullfile** command) that, after execution returns, the whole path of the folder where the ***PN Toolbox*** saves the new models;

References

R.1. Works on Petri Net Fundamentals

M. Ajmone Marsan, G.Balbo, G.Conte, S.Donatelli, G.Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, John Wiley & Sons, 1995.

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

G. Bolch, S. Greiner, H. de Meer, K. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, *Second edition*, John Wiley and Sons, Hoboken, New Jersey, 2006.

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

C.G. Cassandras, S. Lafortune, *Introduction to Discrete Event Systems*, Second Edition, Springer, New York, 2008.

R. David, H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*, Springer-Verlag, Berlin Heidelberg, 2005.

A.A. Desrochers, R.Y. Al-Jaar, *Petri Nets for Automated Manufacturing Systems: Modeling, Control and Performance Analysis*, Rensselaer Polytechnic Institute, 1993.

P.J. Haas, *Stochastic Petri Nets. Modelling, Stability, Simulation*, Springer-Verlag New York Berlin Heidelberg, 2002.

B. Hrúz, M.C. Zhou, *Modeling and Control of Discrete-event Dynamic Systems with Petri Nets and Other Tools*, Springer-Verlag, London, 2007.

K.H. Mortensen, *Petri Nets Tool Database*, http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/, 2006.

T. Murata, Petri nets: properties, analysis and applications. *Proc. of the IEEE*, vol. **77**, no. *4*, pp.541-580, 1989.

C.A Petri, *Kommunikation mit Automaten*, Institut für Instrumentelle Mathematik Bonn, Schriften des IIM Nr. 2, 1962.

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

R.2. Works on Petri Net Toolbox

C. Lefter, C. Mahulea, M. H. Matcovschi and O. Pastravanu, "Instruments for topology-based analysis in Petri Net Toolbox", *Periodica Politechnica Timisoara*, *Transactions on Automatic Control and Computer Science*, vol. 49 (63), pp. 153-158, 2004.

C. Lefter, M.H. Matcovschi, O. Pastravanu. "Computer-Aided Analysis and Design of Discrete-Event Systems with Petri Net Toolbox for MATLAB", În: Yagawa, Atanasiu G. and C. Brătianu (Eds.), *Performance Based Engineering for 21st Century*, Ed. Cermi, ISBN 973-667-063-5, pg. 222-227, 2004.

C. Lefter, C. Popescu, I. Podaru. "The integration of the Petri Net toolbox with Simulink", *SACCS 2004, 8th International Symposium on Automatic Control and Computer Science*, October 23 - 24, 2004, Iasi, Romania, CD-ROM, 6 pg, 2004.

C. Mahulea, *Petri Net Toolbox – a MATLAB library for discrete event systems*, MS thesis, The Sheffield University, Department of Automatic Control & System Engineering, UK (ERASMUS / SOCRATES grant), 2002.

C. Mahulea, L. Bârsan, and O. Pastravanu, "MATLAB tools for Petri-net-based approaches to flexible manufacturing systems". In: (F.G. Filip, I. Dumitrache, S. Iliescu, Eds.) *Large Scale Systems: Theory and Applications 2001, Proceedings volume from the 9th IFAC / IFORS / IMACS / IFIP Symposium on Large Scale Systems LSS 2001*, Bucharest, Romania, 18-20 July 2001, Elsevier Science, pp. 184-189, 2001.

C. Mahulea, M.H. Matcovschi, O. Pastravanu. "Role of animation în teachware for Control Engineering – A case study", *Annals of the University of Craiova*, nr.27, vol. I, pg. 296-301, 2003.

M.H. Matcovschi, C. Lefter, C. Mahulea, O. Pastravanu. "Petri Net Toolbox for MATLAB – Modern teachware for control engineering", În: I. Chitescu (Ed), *Software educațional. Conferința Națională de Învățământ Virtual CNIV 2004*, Ed. Universității Bucureşti, ISBN 973-575-947-0, pg. 384-391, 2004

M.H. Matcovschi and C. Mahulea, "Generalized Stochastic Petri Nets in Performance Evaluation for Queueing Networks", *Proc. of the Symposium on Intelligent Software and Control Systems SISCS 2002*, July 18-19, 2002, Iasi, Romania, CD-Rom, 2002.

M.H. Matcovschi and C. Mahulea, "Timing in Rule-Based Control of Flexible Manufacturing Systems", *Memoriile Secțiilor Ştiințifice ale Academiei Române*, seria IV, tom XXV/2002, pg. 229-248.

M.H. Matcovschi, C. Mahulea, C. Lefter, O. Pastravanu, "Petri Net Toolbox in Control Engineering Education", *2006 IEEE Conf. on Computer-Aided Control Systems Design* (*CACSD 2006*), Munchen, Germany, pp. 2298-2303, 2006.

M.H. Matcovschi, C. Mahulea, O. Pastravanu, "Exploring structural properties of Petri nets in MATLAB", Buletinul Institutului Politehnic din Iaşi, tom XLVII (LI), fasc. 1-4, secția AUTOMATICĂ ŞI CALCULATOARE, pg. 15-25, 2001

M.H. Matcovschi, C. Mahulea and O. Pastravanu. "Computer Tools for Linear Systems over Max-Plus Algebra", *Periodica Politechnica Timisoara*, *Transactions on Automatic Control and Computer Science*, vol. 47 (61), pp. 97-102, 2002.

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

M. H. Matcovschi, C. Mahulea and O. Pastravanu, "Modeling, Simulation and Analysis of Petri Nets in MATLAB", *The 14th International Conference On Control Systems And Computer Science - CSCS14*, July 2-5, 2003, Bucharest, Romania, vol. 1, pp. 106-111, 2003.

M.H. Matcovschi, C. Mahulea, O. Pastravanu, *Homepage of the Petri Net Toolbox, version 2.2*, http://www.ac.tuiasi.ro/pntool, 2007.

M. H. Matcovschi, C. Popescu, O. Pastravanu, "A new approach to hybrid system simulation: Development of a Simulink library for Petri Net models", *Control Engineering and Applied Informatics*, vol. 7, nr. 4, pp. 55-62, 2005.

M.H. Matcovschi, C. Lefter, O. Pastravanu, "Petri nets în hybrid system simulation under Simulink", *Proc. 15th Int. Conf. Control Systems and Computer Science CSCS 15*, Ed. Politehnica Press, Bucuresti, vol. I, ISBN: 973-8449-90-1, pg. 168-173, 2005.

M.H. Matcovschi, O. Pastravanu, "Qualitative versus quantitative techniques in manufacturing systems analysis and design – A Petri-net-based approach", *Proceedings of the 5th International Conference on Microelectronics and Computer Science ICMCS-2007*, September 19-21, 2007, Technical University of Moldova, Chisinau, Moldova, vol. I, pp. 351-359.

M.H. Matcovschi, O. Pastravanu, *Homepage of the Petri Net Simulink Block, version 1.1*, http://www.ac.tuiasi.ro/~pnsb, 2007.

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

O. Pastravanu, M.H. Matcovschi and C. Mahulea. "Petri Net Toolbox – Teaching Discrete Event Systems under MATLAB", In: *Advances in Automatic Control*, (Mihail Voicu, Ed.), Kluwer Academic Publishers, Boston/Dordrecht/London, pp. 247-258, 2004.

C. Popescu, *Implementation and Testing of Simulink Blocks for Petri Net Models*, Research Report Socrates-Erasmus Mobility Program, University of Sheffield, Department of Automatic Control and Systems Engineering; 2005.

3. Works on MATLAB – Simulink

* * *, *MATLAB Creating Graphical User Interfaces*, The MathWorks, Natick, Massachusets, 2000-2007.

* * *, *MATLAB Desktop Tools and Development Environment*, The MathWorks, Natick, Massachusets, 2004-2008.

* * *, *MATLAB Mathematics*, The MathWorks, Natick, Massachusets, 2004-2007.

* * *, *MATLAB Programming Funamentals*, The MathWorks, Natick, Massachusets, 2004-2007.

* * *, *Using Simulink*, The MathWorks, Natick, Massachusets, 2007.

* * *, *Writing S-Functions*, The MathWorks, Natick, Massachusets, 2004.

* * *, *Classes and Object-Oriented Programming*, The MathWorks, Natick, Massachusets, 2008.

Index