A[Q1] Tool for the Educational Study of Manufacturing Systems

MARÍA MARTÍNEZ, MARIO CAÑADAS

Escuela[Q3] *Tecnica Superior de Ingenieros Industriales, Plaza El Ejido, s/n Málaga, Málaga* 20013, *Spain*

Received 7 May 2007; accepted 4 July 2007

ABSTRACT: In this article, a new application for teaching manufacturing systems is presented. A manufacturing system can be considered as a discrete-event system constructed from devices that have a finite speed of operation. This application automatically simulates the manufacturing systems modelled by timed and stochastic Petri nets (PNs) with additional extensions that easily make possible to get specific information. The graphical model generated provides the necessary data to execute an event-driven language programme that simulates the model and provides results. From the student's point of view, the problem is only to draw the PN accordingly with the characteristics of the system to be modelled and to get the information without noticing the underlying event-driven language programme. © 2009 Wiley Periodicals, Inc. Comput Appl Eng Educ 14: 1–14, 2009; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.20189

Keywords: manufacturing systems; Petri nets; discrete-event simulation; educational tool

INTRODUCTION

Manufacturing can be classified as discrete parts or continuous processing. Discrete parts manufacturing is characterised by individual parts that are clearly distinguishable such as engine blocks. Continuous processing industries operate on product that is continually flowing like oil refineries or chemical industries. This application has been developed from the perspective of discrete parts manufacturing and the operations can be either a fabrication or assembly nature.

Continuous processing industries are specified by differential equations, and some specific tools have been designed to support the students in the task of modelling [1]. The modelling of discrete parts manufacturing systems is characterised by concurrent and asynchronous events that are typical for discrete-event dynamic systems.

The simulation of a manufacturing system has several purposes. One is to debug the design. Another is to assess its performance. Simulation is needed in order to validate the requirements and to evaluate the performance of various alternatives before any detailed design and implementation can take place [2]. In some cases, simulations are carried out with discrete-event languages that automatically collect

Correspondence to M. Martínez (mams@ctima.uma.es)

many standard statistics like average queue lengths or average waiting times [3]. Other authors prefer a general-purpose language [4].

Petri nets (PNs) have been used as a powerful tool for the tasks of specification, the modelling and the evaluation of discrete-event systems like manufacturing systems [5,6]. In ordinary PNs, state changes occur instantaneously. The introduction of time and stochastic hypotheses leads to timed and stochastic PNs, which are of use in performance evaluation. Applications of PNs to model manufacturing systems require the introduction of time into the model.

The concept of time applied to PNs may be associated with transitions [7,8], places [9], tokens [10] and arcs [11].

The main difficulties involved in the study of manufacturing systems lie in the large number of elements, which are involved in their functioning. Although generalised PNs permit the description of manufacturing systems, composed by a large number of elements, the final models are huge, with a corresponding drastic reduction in their legibility and manageability. So the use of the proposed tool for modelling a real big size manufacturing problem is very difficult while it is an interesting academic tool like other academic tools developed to help the students in different subjects like object-oriented programming [12] or control engineering teaching [13].

The tool presented in this article makes possible the following:

- To study the functioning of typical examples found in the literature in a course about manufacturing systems.
- To analyse the performance of the overall manufacturing system in terms of production rates, machine utilisation, average-in-process inventory and other measures.
- To use the ability of PNs to detect deadlock, conflict and other properties related to resource management in a manufacturing system.

This work proposes the use of a graphical PN-based software tool to help the design of manufacturing systems. The time is associated with transitions in the way proposed by Ramchandani [7]. In the model of Ramchandani, the transitions are allowed to hold tokens for a given range of times. Some additional facilities have been included in order to make the modelling task easier with the main objective of understanding the way of work of the modelled system as directly as possible. The model is then simulated in the event-driven language Simscript II.5, not being necessary for any knowledge of this language by the user. The tool implements the overall study of the system that includes design, simulation and analysis of results.

The organisation of this article is as follows: The first section is an overview of PN modelling techniques for manufacturing systems taking into account the new facilities incorporated. Then, a description of the tool is presented. In the next section, some details concerning the results are analysed. In the following section, an example is proposed to illustrate how the tool works. The article ends with some comments and conclusions about the use of the tool in a course about manufacturing systems at the University of Málaga.

PROPOSED PETRI NETS TO MODEL MANUFACTURING SYSTEMS

PNs are a graphical and mathematical model for representing information and controlling flow in an event-driven system. They have two nodes, transitions and places. Direct arcs link places to transitions and transitions to places.

The graphical nature of PNs makes them self-documenting and a powerful design tool. In the usual PN graphical representation, a circle represents a place, a rectangle or a bar represents a transition and arcs are represented as directed arrows. PN places can hold tokens. A token distribution is called a marking and represents a state of the system. On the other hand, they are based on a strong mathematical formalism, which makes it possible to set up mathematical models describing the behaviour of the system [14].

An ordinary PN can be defined as a 5-tuple $PN = (P, T, A, W, M_0)$, where $P = \{p_i: i = 1, \ldots, |p|\}$ is a finite set of places, $T = \{t_j: j = 1, \ldots, |t|\}$ is a finite set of transitions, $P \cap T = \emptyset$, $A \subseteq (P \times T) \cup (T \times P)$ is the set of directed arcs representing flow relations, joining places and transitions together, $W \rightarrow \{1, 2, 3, \ldots\}$ is a weight function on arcs, and it is assumed to be 1 in the absence of an explicit weight function, M_0: $P \rightarrow \{0, 1, 2, \ldots\}$ is the initial marking.

Starting from an initial marking M_0 a new marking M_k is reachable if it can be reached by means of a change to the state of the system. The changes are modelled by the firing of transitions. When a transition fires, it consumes the number of tokens given by the weight function on the arc from each of its input places and produces the tokens corresponding to the weight function on the arc in each of its output places.

A PN can be represented by a pair of incidence matrixes where the rows represent the places and the columns are the transitions. In the previous incidence matrix, the elements of the matrix indicate the weight of the arc from place p_i to transition t_j. In the posterior incidence matrix, the elements of the matrix represent the weight of the arc from transition t_j to place p_i.

Formally, a timed Petri net is a couple (PN, τ) where PN is an ordinary Petri net and τ is a function assigning a real non-negative number, τ_j, to each transition of the net. This number is the firing time of a transition. Transitions are enabled according to the same rules as applied to ordinary PNs. Firing an enabled transition t_j causes a two-step marking change. Instantaneous with the enabling of t_j, the marking of its input places is decreased by one token per input arc. After τ_j time units the marking of its output places is increased by one token per output arc. A simulation model of a timed net requires a clock reference. In a timed net, the transitions firing times may be either deterministic or stochastic. It may also contain instantaneous transitions.

To construct a model of a manufacturing system, it is necessary to make an abstraction of its functional aspects. In a manufacturing environment, *places* usually represent *resources* such as machines, automatically guided vehicles or parts in a buffer. The existence of one or more *tokens* in a place represents the *availability* of a particular resource, while no tokens indicate that the resource is unavailable. A *transition* firing represents an *activity* that can need some time to be carried out or to be instantaneous. Also, places and transitions together represent conditions and precedence relations in the system's operation. For example, a token in a place can imply that the condition is true, and no token, that it is false.

Some additional facilities have been included in the couple (PN, τ) in order to make easier the modelling of manufacturing systems:

First: It is possible to distinguish in the set of directed arcs A between:

- *Standard arcs* are arcs that link transitions to places or places to transitions, and they are characterised by a weight function W that controls the number of tokens necessary to fire the transitions.
- *Inhibitor arcs* are arcs that can only link places to transitions and they enable the transition only when its input places have no marks. These arcs are represented with a little circle at the end instead of an arrow.
- *Stochastic arcs* are arcs that only link transitions to places and they are characterised by a probability number w (from 0 to 1). Only $(100 \times w)\%$ of the fires in the transition change the marks in the output place. When the fire happens, only one mark is moved to the place, this means that it behaves like a standard arc with the weight of the arc equal to one. The stochastic arcs can model junctions in a manufacturing system, for instance, if several pieces arrive at the beginning of a process and must be separated in a given percentage in order to follow separated ways. In the example of Figure 1, the 20% of the arriving jobs must follow process 1, the 50% must follow process 2 and the 30% the process 3. The stochastic arcs must accomplish

$$\sum_{i=1}^{n} w_i(t_j) \leq 1$$

where n is the number or stochastic arcs belonging to transition t_j. A transition with stochastic arcs can also have standard arcs that work in their own way independent of stochastic arcs.

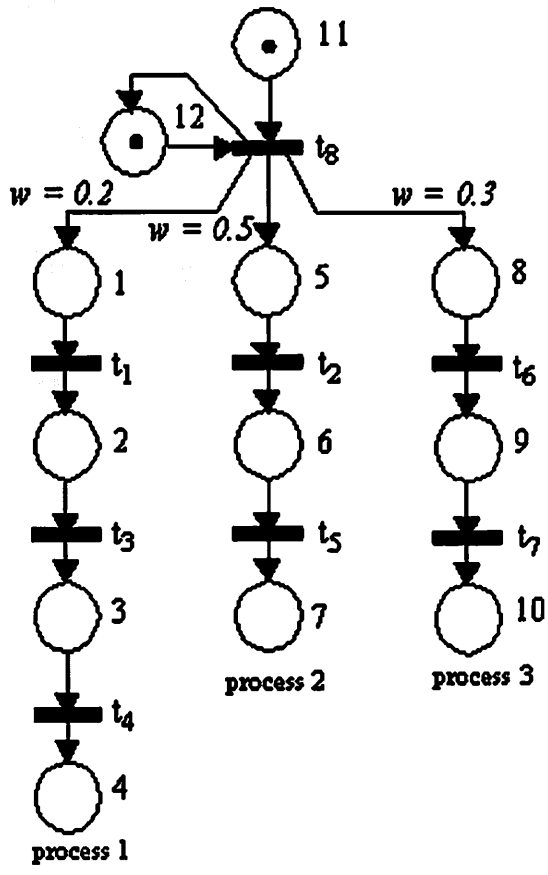

Figure 1 Petri net with stochastic arcs.

Second: The subset work centre WKC, where WKC $\subset T$ and is used when there are resources shared by several processes. An ordinary PN can model this situation as shown in Figure 2, nevertheless if the number of processes that share the resource increases, the net becomes more difficult to draft and to understand. The use of WKC makes easier to model this, for instance, the equivalent net of Figure 2 using WKC is shown in Figure 3.

So, there is an equivalent ordinary PN to the PN with WKCs. The rules for firing transitions included in any WKC are:

i. A transition t_j included in a WKC is enabled when it accomplishes the conditions of ordinary PNs and there are available resources in the corresponding WKC.
ii. Every time when the fire of a transition t_j included in a WKC begins, the number of available resources in the corresponding WKC decreases one unit.
iii. When the fire of transition t_j finished, the number of available resources in the corresponding WKC increases one unit.

Third: The subset work in progress WIP, where WIP $\subseteq (P \cup T)$ and it is a part of the PN. This tool is very useful when there are several processes modelled by the net and it is necessary to collect independent information of them. The definition of WIPs in the net does not disturb the behaviour of the net. In the example of Figure 3, it is possible to include places 1, 2, 3 and 4 and transitions t_1, t_3 and t_4 in a WIP1 and places 5, 6 and 7 and transitions t_2 and t_5 in a WIP2.

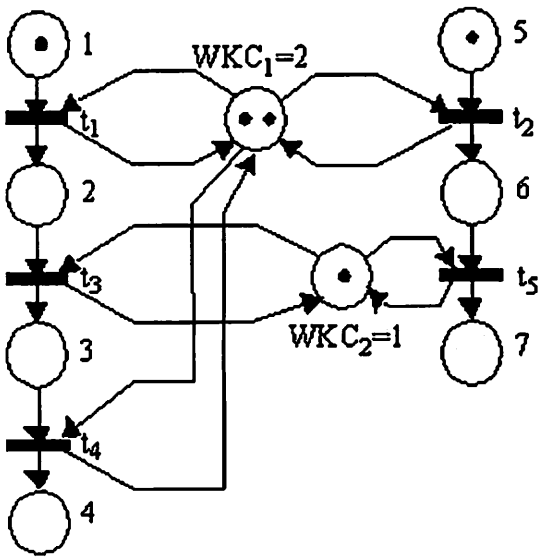

Figure 2 Ordinary Petri net with shared resources.

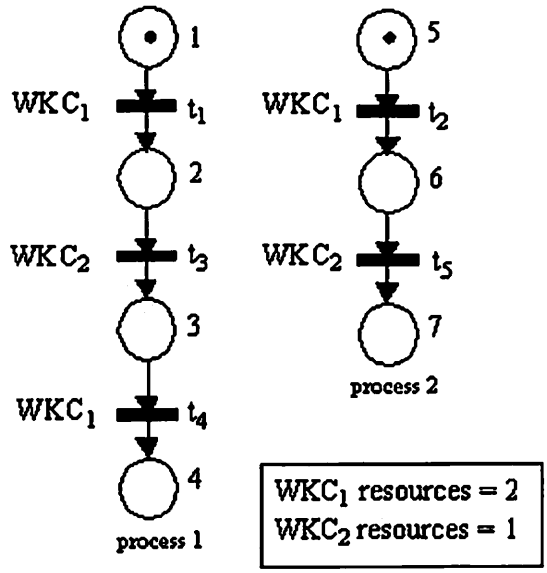

Figure 3 The WKC ability applied to Petri net in Figure 2.

Fourth: It is possible to study the queues in all the WKCs and in the specific places enabled by the user in the corresponding menu.

DESCRIPTION OF THE TOOL

From the description of the net by a graphical tool built in Visual Basic 6, a file with the information of this specific net *data.dat* is generated, with a specific structure, as a source of data necessary for the execution of the Simscript II.5 programme that simulates the behaviour of the original net (see Fig. 4).

As the simulation core is coded in Simscript II.5 language and this programme is independent from the graphical model coded in VisualBasic, it is necessary to set up a synchronisation way at the level of operating system in order to exchange information between the two files *VisualRPT* (graphical interface) and *SimRPT* (simulation core). The choice has been to use intermediate files stored in the folder *Temp/*.

The information included in the file *data.dat* is:

- The number of places and transitions.
- The last WKC number.
- The previous incidence matrix, the posterior incidence matrix and the inhibitor arc matrix.
- The information related to every transition (stochastic or deterministic, temporal distribu-

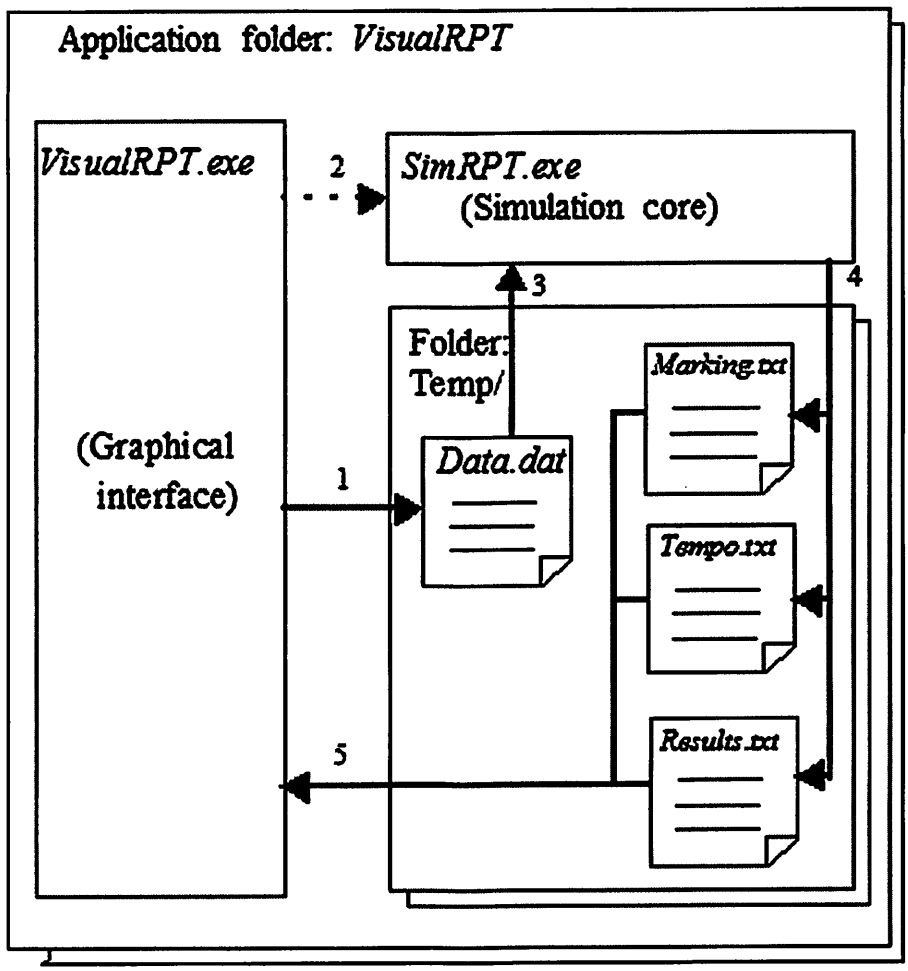

Figure 4 VisualTPN architecture.

tion function and its parameters, whether the transition belongs to a WKC or WIP, etc.). This information is provided by the user in the *VisualRPT* tool as a pop menu.

- The information related to every place (number of tokens, whether the study of queues is enabled or not, the ownership to a *WIP*, etc.). This information is also provided by the user in the *VisualRPT* application.

Once the *data.dat* file has been originated, *VisualRPT* calls the programme *SimRPT* and enables the timer. After that it is necessary to wait until the simulation finishes in order to open the generated files *marking.txt*, *tempo.txt* and *results.txt* (see Fig. 4).

While the simulation is happening, every time a transition is fired, a new line with the update marking vector M is written in the file *marking.txt*. The user specifies in the *VisualRPT* application the number of steps of simulation between the storage of temporary performance information in the file *tempo.txt*. When the simulation finishes, all the statistical results are stored in *results.txt*.

As a Simscript II.5 programme, *SimRPT* begins with a *preamble* whose statements are declarative in nature. The preamble includes the definition of the desired statistical measures of system performance with the sentences *tally* and *accumulate*, for instance, the mean number of tokens in a place or the mean occupation of a transition.

The simulation actually begins with the execution of the *start simulation* statement included in the *main* programme, since this is a call to the *timing routine* which advances the simulation clock to the time when the corresponding *event* or *process* defined in the preamble are due to occur. There are an event routine and a process routine for each type of event or process.

The code of the programme includes several routines, which can be called by the main programme or other routines like the initialise routine that generates the *LINK_TRANS* matrix in charge of controlling the simulation process. This matrix, sized NT × NT where NT is the number of transitions in the net, stores the connections between transitions. The maximum number of necessary operations to carry out in order to generate this matrix is NP × NT^2, where NP is the number of places in the net. As the number of transitions in the net is very similar to the number of places, approximately the amount of operations is NT^3. In spite of this amount of operations, it is better to do this work only once than to check every one of the transitions in every step of the simulation, that is, a number of operations equal to N_s × NT, where N_s is the number of steps. Generally, it is better to work with the *LINK_TRANS* matrix when $N_s > NT^2$ as usually happens.

The different sorts of links are:

- *Type 0*: When *LINK_TRANS*$(m, n) = 0$, there is no connection between transitions m and n.
- *Type 1*: When *LINK_TRANS*$(m, n) = 1$, the transition n can be enabled when the fire of transition m finishes (the tokens have been appended to the output places).
- *Type 2*: When *LINK_TRANS*$(m, n) = 2$, the transition n can be enabled when the fire of transition m begins (the tokens have been subtracted from the input places), for example, in case of inhibitor arcs.
- *Type 3*: When LINK_TRANS$(m, n) = 3$, includes types 1 and 2, so the transition n can be enabled both at the beginning or at the end of the fire of transition m.

For an example of the different types, see Figure 5.

When the simulation starts, the programme offers two possibilities, to stop the simulation after a given number of fires in the transitions; or execute the simulation for a length of time T_s. To choose T_s it is necessary to take into account:

1. The bias in the results can be severe if T_s is too short because of unrealistic initial conditions.
2. The desired precision of the results; its variability increases as the run time simulation T_s decreases.

In order to reduce the impact of initial conditions, the time T_0 is specified. That is, the simulation begins at time $t = 0$ but data collection on the response variables of interest does not begin until time T_0 and continues until time $T_0 + T_s$.

The software verification has been carried out in two stages. Firstly, it is necessary to check whether the user tries to introduce a wrong PN, for example, *VisualRPT* detects the following mistakes while the net is designed (supplying information to the user):

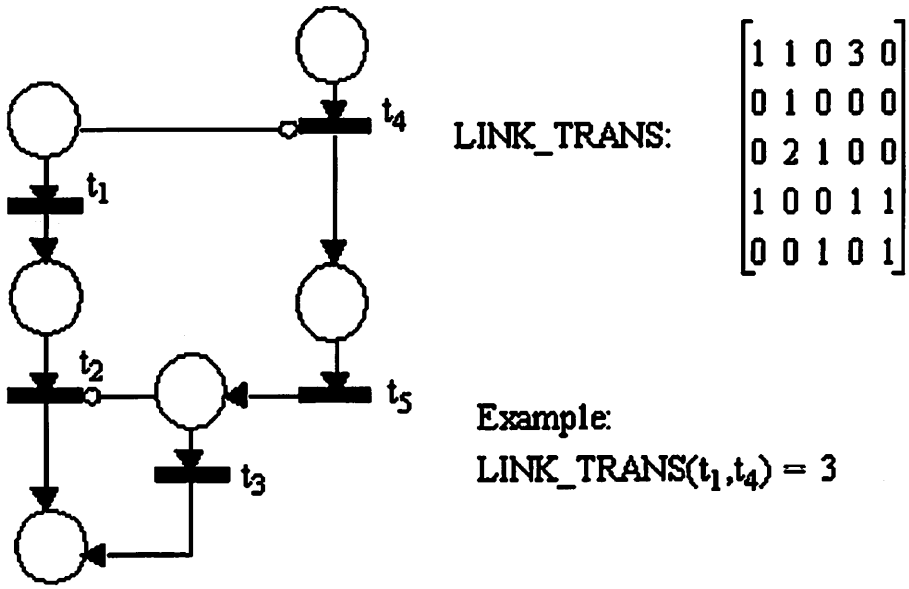

Figure 5 Example of the LINK_TRANS matrix.

- An arc cannot directly link two transitions or two places.
- To check the numerical information, whether the sort is correct and the value is within the limits.
- To verify that the correct conditions for the simulation are accomplished, for instance, that $T_s > T_0$.

Secondly, it is time to check whether the simulation core *SimRPT* works correctly with the information supplied by the file *data.dat* so, it is necessary that the simulation:

- Accomplishes the rules for firing transitions, taking into account the inhibitor arcs, the stochastic arcs, the limited capacity of some places and the WKC's characteristics.
- Solves a conflict when it happens accordingly with the decision given by the user.
- Finishes correctly, mainly when a deadlock is detected.
- Carries out the statistical operations accurately, mainly regarding queue and WIP analysis.

ANALYSIS OF RESULTS

The tool has been used to assert the performance of the modelled system, in this way, the implications of PN properties in modelled manufacturing systems have been studied [4]. The reachability property provides information whether a particular state can be reached from some other state by firing a sequence of transitions. The *k*-bounded of a place such as a buffer or queue ensures that there is not an overflow. The liveliness implies the absence of deadlocks and guarantees that a system can successfully produce under all specified circumstances. The last property taken into account, reversibility, means the system can be reinitialised from any reachable state.

For studying the previous properties, the *Marking.txt* file provides a trace of the overall simulation. In general, a trace is a detailed computer printout, which gives the value of every marking vector in a computer programme, every time that the marking changes in value. Thus, a simulation trace is nothing more than a detailed printout of the state of the simulation model as it changes over time. Optionally, *VisualRPT* shows the changes of the marking by an animation offline with tokens moving between the places of the net.

Other analysis provided by the tool is regarding the data generated by a simulation. Its purpose is to predict the performance of a system, or to compare the performance of two or more alternative system designs. So, it is possible to evaluate the performance of the shop for a particular workload, measuring the delays experienced by jobs at each WKC, and overall delays in the shop. Also, it is possible, for instance, to evaluate the impact of reconfiguring the machines or to decide whether the system could handle additional job types.

The information registered in the *results.txt* file is shown by some graphics and reports. In Figure 6 the main window of analysis of results appears with some statistical results about places and transitions, as well as some histograms. A more detailed information about queues in places and WKCs can be seen in Figure 7. In Figure 8 is represented one of the options for additional information about transitions like its evolution with time. Figure 9 shows the histogram of probability for a selected WIP.

The method used to know the precision of the results of the simulation is the *method of independent replications* (see Fig. 10). The simulation is repeated a total of R times, each using a different random number stream and independently chosen initial conditions which includes the case that all runs have identical initial conditions.

Let X_r be the result of parameter X in the r replication with $r = 1, 2, \ldots, R$. The R samples $X_1, X_2, \ldots, X_R$ are statistically independent and identically distributed, and are unbiased estimators of X. The parameter to estimate according to the classical method of confidence interval estimation is

$$E(X) = \bar{X} = \frac{1}{R}\sum_{r=1}^{R} X_r$$

and the variance of $E(X)$ by

$$\sigma^2(\overline{X}) = \frac{1}{R(R-1)}\sum_{r=1}^{R}(X_r - \overline{X})^2$$

and a confidence interval of $100(1-\alpha)\%$ by expression

$$\overline{X} \pm \left(t_{(\alpha/2),R-1}\sqrt{\sigma(\overline{X})}\right)$$

where $t_{(\alpha/2)R-1}$ is the percentage point $100(1-\alpha)$ of the Student's t distribution with $R-1$ degrees of freedom.

For carrying out the checkup, it has been necessary firstly the observation of the results in several simulations. They must reasonably vary when the initial conditions are modified (marking, seeds, etc.). In some cases, it is possible to directly verify the

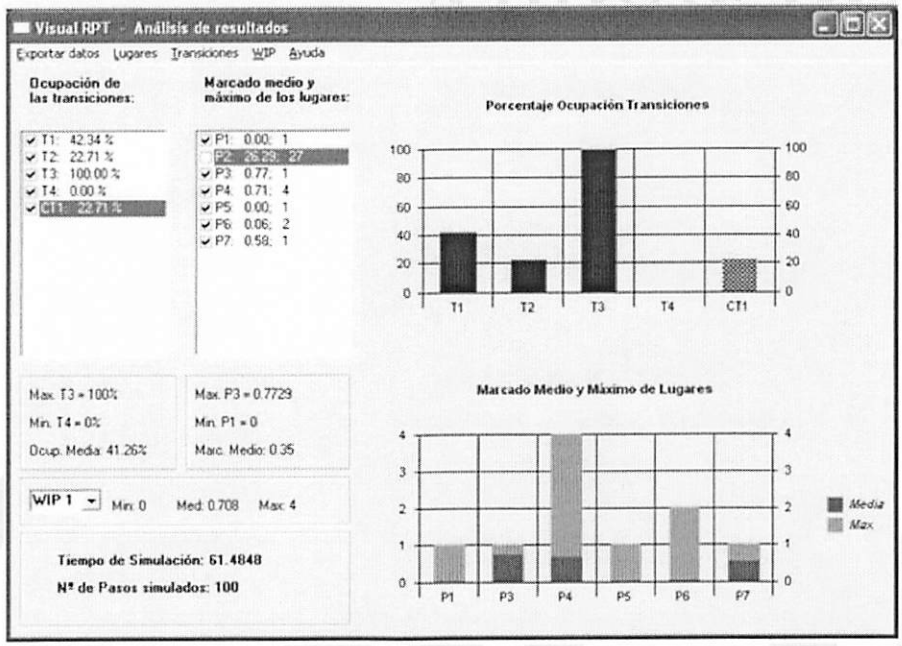

Figure 6 Window of analysis of results.

simulation results with the analytical ones calculated for the same model. So, any measure of performance that can be computed analytically and then compared to its simulated counterpart provides a valuable tool for verification. Presumably, the objective of the simulation is to estimate some measure of performance, such as mean response time, which cannot be computed analytically. But as illustrated by the formulas in Figure 11, for a number of special queues ($M/M/c$, $M/G/c$, etc., c being the number of machines

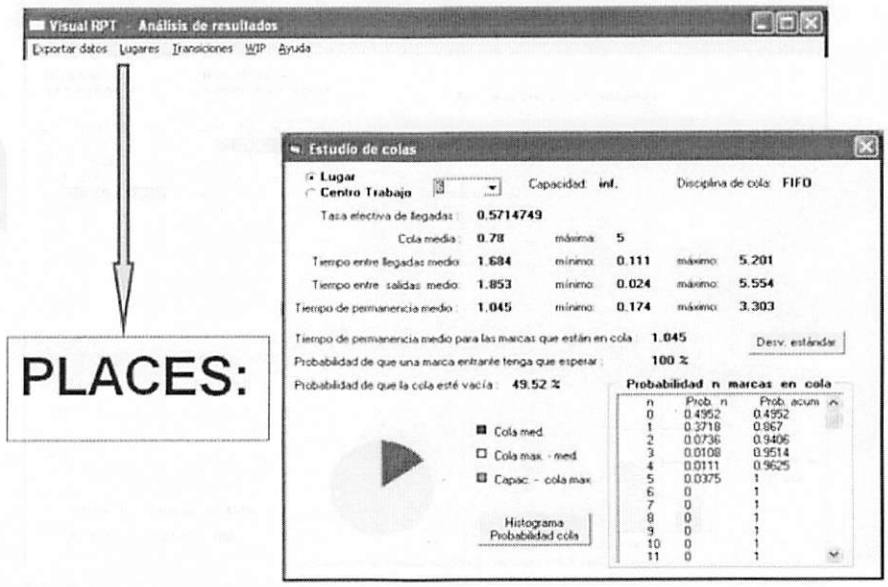

Figure 7 Statistical information about places and WKCs.

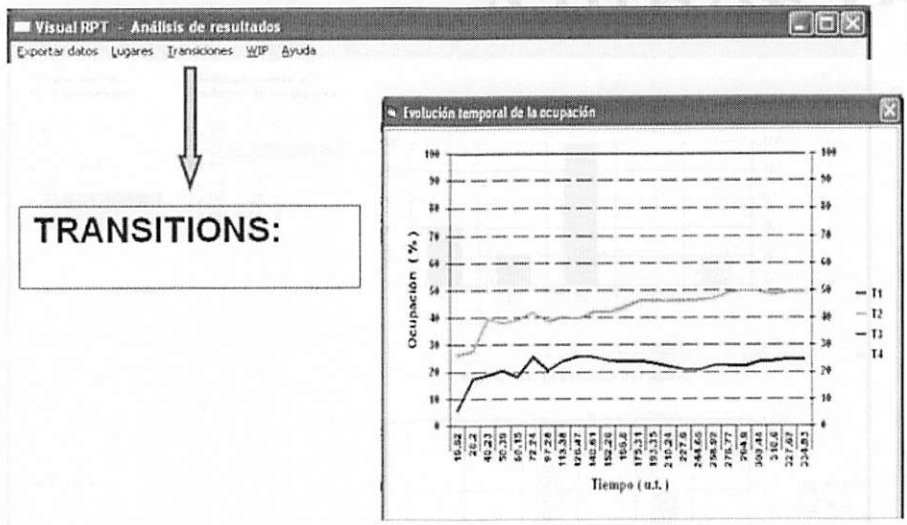

Figure 8 Evolution with the time of activity in the transitions.

or servers), all the measures of performance in a queuing system are interrelated.

Thus, if a simulation model is predicting one measure (such as utilisation) correctly, the confidence in the predictive ability of the model for other related measures (such as response time) is increased (even though the exact relation between the two measures is, of course, unknown in general and varies from model

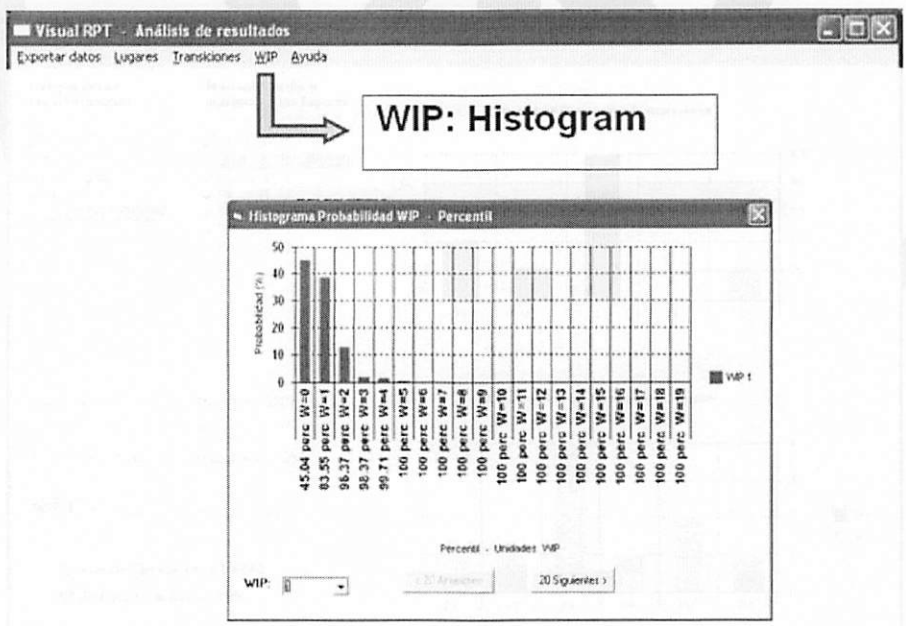

Figure 9 Histogram for a selected WIP.

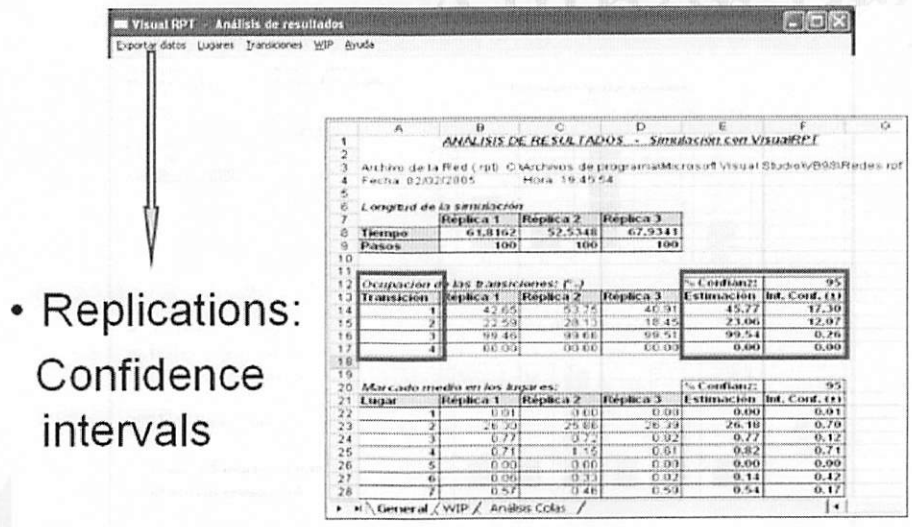

Figure 10 Result of applying the method of independent replications.

to model). Conversely, if a model incorrectly predicts utilisation, its prediction of other quantities, such as mean response time, is highly suspect.

The values of parameters in Figure 11 are for $c = 2$. Effectively, supposing that there are two machines operating in parallel. Each machine has an independent and identical exponential process time distribution with mean $1/\mu$. Jobs are entering the system at a rate of λ per time unit in a Poisson distribution and are leaving the system at a maximum rate of 2μ per time unit. The proportion of time the machine is busy is called ρ. P_0 is the probability that the system is empty while $P(L(\infty) \geq 2)$ is the probability that all machines are busy. The expected number of jobs is L and the average length of the queue is L_Q. The mean time spent in system is ω while the waiting time in queue is ω_Q.

ρ	$\frac{\lambda}{2\mu}$
P_0	$\left\{\left[\sum_{n=0}^{1}\frac{(2\rho)^n}{n!}\right]+\left[(2\rho)^2\frac{\left(\frac{1}{2\mu}\right)}{1-\rho}\right]\right\}^{-1}$
$P(L(\infty)\geq 2)$	$\frac{(2\rho)^2 P_0}{2!(1-\rho)}$
L	$2\rho+\frac{\rho P(L(\infty)\geq 2)}{1-\rho}$
ω	$\frac{L}{\lambda}$
ω_Q	$\omega-\frac{1}{\mu}$
L_Q	$\frac{\rho P(L(\infty)\geq 2)}{1-\rho}$

Figure 11 The steady-state parameters for a M/M/2 queue.

The net of Figure 12 models a *M/M/2/∞/∞/* FIFO, which includes two operations represented by transitions t_2 and t_3. The operations are carried out with two machines that work in a parallel way. The processing time required in both machines is stochastic, specifically an exponential distribution with a mean value of 40 s, which corresponds with a value of $\mu = 1.5$ jobs/min. The arrival of parts is according to a Poisson distribution with a value of $\lambda = 2$ jobs/min and it is represented by transition t_1.

In order to catch the performance of this system, it has been necessary to enable the queue study for place number 1. Additionally, this place together with transitions t_2 and t_3 compose the subset WIP1 providing the updated information of all the jobs currently included in the system.

Table 1 shows the results supplied by *VisualRPT* which are very close to the calculated for *M/M/2* queue in a theoretical way by the mathematic expressions of Figure 11. An amount of 10 replications have been carried out with different seeds and 50,000 steps of simulation running where a 10% has been assigned to the parameter T_0.

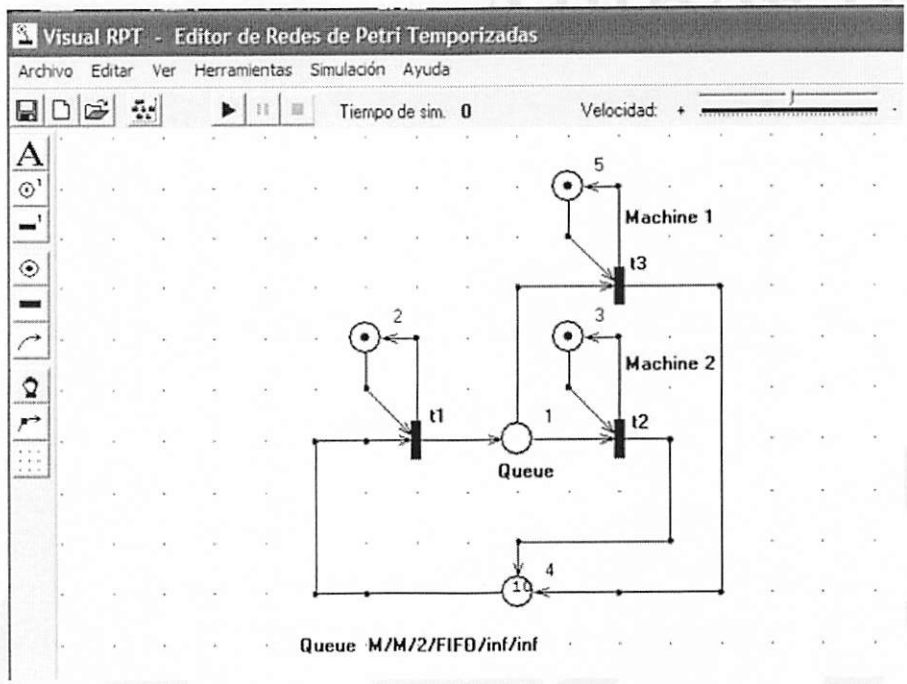

Figure 12 Example of Petri net for analysing queues.

EXAMPLE OF USE OF THE TOOL

The tool has been used to model several systems that represent typical configurations usually found in manufacturing systems which can be modelled by PNs.

In this example, the special focus has been the use of the specific tools, that is, the WKCs represent accurately the work centres in a manufacturing system and with the WIPs it is possible to achieve real statistical results about the units through the manufacturing process.

For some representative models, the results are available in the bibliography like the simple job shop that operates as follows [15]:

There are any number of machines clustered in groups called work centres. Each work centre has different machines, but within a centre, machines are identical. The number of centres and the number of machines within centres will not change during a simulation run. This information is available in Table 2. Every work centre has an assigned colour in order to easily distinguish the transitions that use the same work centre. Besides, a legend indicating the work centre has been added to the transitions.

Jobs are orders that come into the shop from outside at times provided as input data. Each job consists of a sequence of tasks to be performed by specified machines. The routing for jobs through the shop and the mean processing time at each machine, are specified as input data as shown in Table 3. The Erlang distribution has been used and so has the hour unit time.

Table 1 Simulation Versus Theoretic Results

	VisualRPT	*M/M/2* parameters
P_0	0.201 ± 0.004	0.200
$P\,(L(\infty) \geq 2)$	0.533 ± 0.009	0.533
L_Q	1.084 ± 0.088	1.067
ω_Q	0.541 ± 0.043	0.533

Table 2 Number of Identical Machines in Each Centre

	WKC1	WKC2	WKC3	WKC4	WKC5
Index	1	2	3	4	5
Number of machines	3	2	4	3	1
Type	Shaper	Welder	Lather	Saw	Drill

Table 3 Sequence of Tasks for Each Job

Job type	WKC routing (indexes)	Mean processing time
1	3-1-2-5	0.5-0.6-0.85-0.5
2	4-1-3	1.1-0.8-0.75
3	2-5-1-4-3	1.2-0.25-0.7-0.9-1

The PN of this system is represented in Figure 13. The jobs arrive at the model according to an exponential distribution of mean 0.25 h. The stochastic arc tool has been used in transition 1 to model a ratio of 30% for type 1 jobs, 50% for type 2 jobs and the last 20% for type 3 jobs. The place number 4 has a big enough number of marks in order to finish the simulation when a time of 2,920 h is accomplished. This time represents 1 year (365 days with 8 h per day).

Transitions t_{14} to t_{19} happen instantaneously while the rest have a WKC assigned according to the routing given in Table 3 and represent the different operations to be carried out. There are more than one transition that belong to the same WKC, so conflicts are possible and the FIFO solution has been adopted. The marks in the places 1−3 provide information about the WIP for every type of job, so the study of queue has been enabled in these places. When the data to collect is the queue in a specific WKC, it is necessary to take into account the marks in all the input places to transitions that belong to this WKC. For instance, for knowing the queue in the WKC2, the information of marks in places 9 and 18 is necessary and it does not matter how many belong to job 1 of 3. Nevertheless, if the information to collect is the queue in a WKC for a specific job, the study of queues is enabled only in the specific place (9 or 18 in the example before).

The comparison between *VisualRPT* and the *SIMLAB* application written in Fortran language is shown in Tables 4 and 5.

To reach the confidence intervals with *VisualRPT*, an amount of 10 replications have been carried out changing the seeds for the service times, stochastic arcs and the time between incoming jobs generators (transition t_1). The differences are because the results provided by SIMLAB correspond to only one simulation instead of 10, so there is no information about the confidence intervals. In spite of that, the results provided by both methods are close enough to each other.

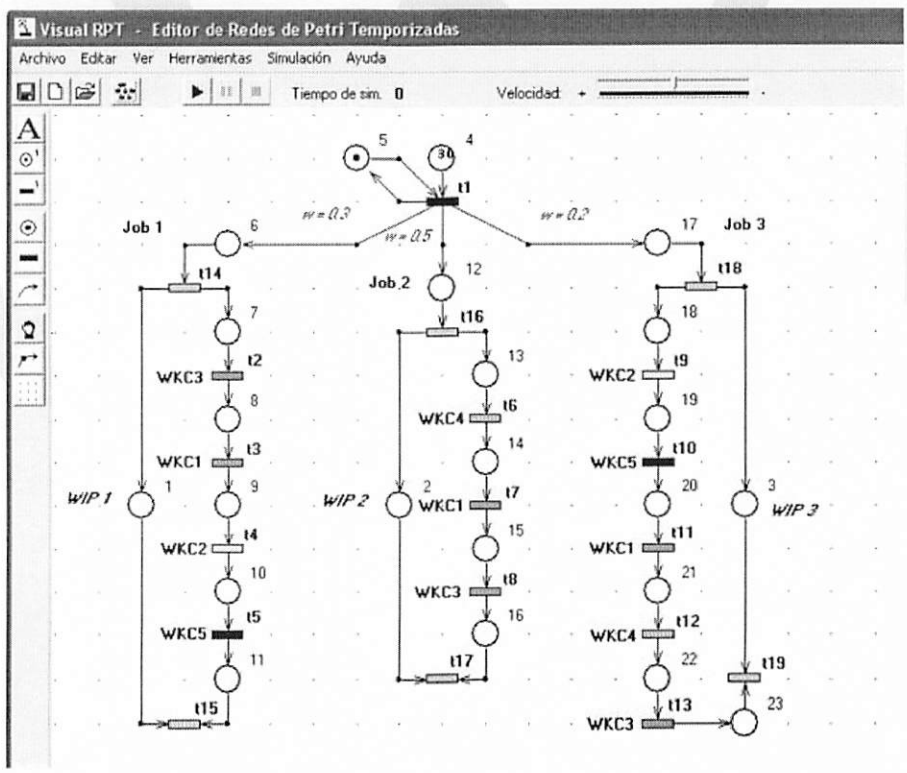

Figure 13 The Petri net for the job shop operation mode.

Table 4 Mean Process Time (h)

	VisualRPT	SIMLAB
Job 1	14.613 ± 1.878	12.472
Job 2	11.568 ± 1.569	12.053
Job 3	22.418 ± 2.799	19.858

THE EDUCATIONAL USE OF VisualRPT

VisualRPT has been used in a course of manufacturing systems in the study of Industrial Engineering at the University of Málaga, being a tool of valuable help for the students to analyse the differences between several options when a manufacturing system is studied. The students who attend the course have a previous knowledge of PNs that have been applied in previous courses to program PLCs. So, it has been possible to introduce the study of manufacturing systems without the teaching of specific event-driven languages like Arena or Simscript II.5. The proposed tool is based on Simscript II.5 language, but the students do not need to have any knowledge about it (these languages are studied in other courses about modelling and simulation).

In the classroom, some books about the applications of PNs in manufacturing systems are used [4,6,15] and some examples included in these books are studied but it was necessary to have a tool to simulate the models, to see how the tokens move, the different markings reached or to analyse the results and to compare them with the other ones given in the books. Another interesting point is to analyse how the amount of resources can modify the results in order to get the optimal value. So, the tool has been designed to achieve several educational goals:

1. To supply a screen-integrated environment with a friendly and extremely easy to use graphical user interface, to help the students to define the PN and to assign the initial set of tokens that correspond with the initial state of the system.
2. When the simulation of the net starts, the animation on the screen shows the tokens moving over the net and it is possible to detect some problems like deadlocks. Also, the students can edit the file *marking.txt* and study all the markings generated in the simulation.
3. The students can directly understand the implications that PNs properties bring to the modelled manufacturing system, for instance, liveness implies the absence of deadlocks and guarantees that the system can successfully produce under all specified circumstances.
4. Save in time to study the manufacturing system. The tool makes possible that the student only has to focus on the correct modelling of the manufacturing system with the PN and to define the set of results that the simulation will supply at the end.

CONCLUSIONS

In this article, the problem addressed has been the design of a new computerised tool that automatically simulates the production systems modelled by specific timed and stochastic PNs.

One of the objectives of this work has been the easy construction of the models guaranteed by the graphical nature of PNs that practically self-document the models. This property remains in spite of the new specific abilities included in the basic timed and stochastic PNs. The mathematical model associated with the PN draw has made possible the easy transfer of information for its automatic simulation in Simscript II.5.

When compared to other PN-based tools like Design/CPN [16] or GreatSPN [17], *VisualRPT* has some common points like the graphical interface and the simulation of the net. On the other hand, the main differences are that this tool focuses specifically for manufacturing systems, providing directly the way to model with the PN production concepts like the WIP or the way of group the machines in a work centre. Other interesting point is the aim of build a friendly

Table 5 Statistical Results for Each WKC

	Mean length queue		Mean occupation (%)		Mean queuing time (h)	
	VisualRPT	SIMLAB	VisualRPT	SIMLAB	VisualRPT	SIMLAB
WKC_1	10.075 ± 1.631	12.31	94.74 ± 0.67	96.9	2.536 ± 0.400	3.055
WKC_2	17.016 ± 3.370	11.40	97.02 ± 1.10	97.8	6.421 ± 1.717	5.677
WKC_3	0.703 ± 0.051	0.711	71.76 ± 0.98	71.9	0.177 ± 0.012	0.177
WKC_4	17.378 ± 3.786	17.098	96.20 ± 0.63	96.1	6.232 ± 1.333	6.110
WKC_5	1.751 ± 0.153	2.095	79.00 ± 1.09	79.7	0.889 ± 0.077	1.043

and easy to use tool limiting the properties of the PN to stochastic and timed, avoiding other extensions like coloured PN that make more difficult to understand the modelled manufacturing system.

The verification and validation of this tool has been successfully carried out by the modelling of several systems that represent typical configurations usually found in manufacturing systems, and some of the results have been presented. Our effort in future will be focused on extending the features of this tool as well as its application in the optimisation of actual small size automated material handling systems based on overhead trolley conveyors available in our laboratories.

REFERENCES

[1] H. Van Der Schaff, J. Tramper, M. Vermüe, and R. Hartog, Support of modelling in process-engineering education, Comput Appl Eng Educ 14(2006), 161–168.

[2] K. Sörensen and G. K. Janssens, Automatic Petri net simulation model generation for a continuous flow transfer line with unreliable machines, Qual Reliab Eng Int 20(2004), 343–362.

[3] R. Suárez and J. Rosell, Feeding sequence selection in a manufacturing cell with four parallel machines, Robot Comput Integr Manuf 21(2005), 185–195.

[4] M. C. Zhou and F. DiCesare, Petri net synthesis for discrete event control of manufacturing systems, Kluwer Academic Publishers, Boston, 1993.

[5] M. Silva, E. Teruel, R. Valette, and H. Pingaud, Petri[Q4] nets and production systems. In: G. Rozenberg and W. Reisig (Eds.), Lectures in Petri nets II: Applications, Lecture Notes in Computer Science, Vol. 1492, Springer, 1998, pp 85–124.

[6] A. A. Desrochers and R. Y. Al-Jaar, Applications of Petri nets in manufacturing systems, IEEE Press, New York, USA, 1995.

[7] C. Ramchandani, Performance evaluation of asynchronous concurrent systems by timed Petri nets, PhD Thesis, Massachusetts Institute of Technology, Cambridge, 1973.

[8] P. Merlin and D. Farber, Recoverability[Q5] of communication protocols—Implications of a theoretical study, IEEE Trans Commun 24(1976).

[9] J. Sifakis, Use of Petri nets for performance evaluation. Measuring, modelling and evaluating computer systems. Elsevier Science Publishers, Amsterdam, 1977, pp 75–93.

[10] J. Celko, Time token design methodology, Softw Pract Exp 12(1982), 889–895.

[11] M. Nielsen, V. Sassone, and J. Srba, Towards a notion of distributed time for Petri nets. Proceedings of the 22nd International Conference on Application and Theory of Petri Nets, Vol. 2075, Springer-Verlag, 2001, pp 23–31.

[12] J. B. García Perez-Schofield, F. Ortín Soler, E. García Roselló, and M. Pérez Cota, Towards an object-oriented programming system for education, Comput Appl Eng Educ 14(2006), 243–255.

[13] J. Albino Méndez, C. Lorenzo, L. Acosta, S. Torres, and E. González, A Web-based tool for control engineering teaching, Comput Appl Eng Educ 14(2006), 178–187.

[14] T. Murata, Petri nets: Properties, analysis, and applications, Proc IEEE 71(1989), 541–580.

[15] A. M. Law and W. D. Kelton, Simulation, modelling and analysis, 2nd edition, McGraw-Hill, New York, USA, 1991.

[16] K. Jensen, Coloured Petri nets. Basic concepts, analysis methods and practical use. EATCS Monographs on Theoretical Computer Science, Vol. 1, Springer-Verlag, Basic Concepts, 1992.

[17] R. Gaeta, Efficient[Q6] discrete-event simulation of coloured Petri nets, IEEE Trans Softw Eng 22(1996).

Q1: Please provide authors biographies with photos.
Q2: Please check the suitability of the short title on the odd-numbered pages. It has been formatted to fit the journal's 45-character (including spaces) limit.
Q3: Please check the authors names and affiliation.
Q4: Please provide the publisher location.
Q5: Please provide the page range.
Q6: Please provide the page range.