Applied Mathematics, 2016, 7, 829-839
Published Online May 2016 in SciRes. http://www.scirp.org/journal/am
http://dx.doi.org/10.4236/am.2016.79074

Scientific Research Publishing

Petri Nets—A Versatile Modeling Structure

Miryam Barad

Department of Industrial Engineering, Faculty of Engineering, Tel Aviv University, Tel Aviv, Israel
Email: barad@post.tau.ac.il

Received 22 March 2016; accepted 23 May 2016; published 26 May 2016

Abstract

Petri Nets (PNs) are an effective structure for modeling and analyzing asynchronous systems with concurrent and parallel activities. A Petri net models the static properties of a discrete event system concentrating on two basic concepts: events and conditions. Most of the theoretical work on Petri nets is a formal definition of Petri nets structures, which consist of a set of places, representing conditions, a set of transitions, representing events, an input function and an output function. For practical purposes, a graphical representation is more useful. Two types of nodes portray places and transitions. A circle is a place and a bar is a transition. There is no inherent measure of time in a classical Petri net. To approach time-based evaluation of system performances, Timed Petri Nets (TPNs) were introduced. Modeling the notion of time is not straightforward. There are several possibilities for introducing time in PNs, among them timed transitions and timed places. This paper reviews several published examples where Petri Nets were used in different circumstances such as estimating expected utilization of processing resources at steady state in open queueing networks, verifying computerized simulations and batch planning in textile industry.

Keywords

Petri Nets, Timed Petri Nets, Open Queueing Networks, Simulation Verification, Textile Industry

1. Introduction

The fundamental concepts and characteristics of Petri Nets (PNs) made them a significant tool for modeling and analyzing asynchronous systems with concurrent and parallel activities. A Petri net is both a graphical and an analytical modeling tool. Carl Adam Petri from Bonn University, Germany, developed it as a special class of generalized graphs or nets. The chief attraction of a Petri net is the way in which it identifies the basic aspects of various systems, conceptually, through a graphical representation, and mathematically, eventually supported by formal programming languages. As a graphical tool, PNs are a visual-communication aid similar to flow charts, block diagrams, and networks. In addition, tokens (or markers) in these nets can simulate the dynamic activities

of systems. As a mathematical tool, it allows setting up algebraic equations for formally analyzing the mathematical models governing the systems' behavior, similarly to other approaches to formal analysis, e.g. occurrences nets and reachability trees. Using Petri nets to model and analyze asynchronous systems with concurrent and parallel activities is simple and straightforward making them a valuable technique for analyzing such complex real time systems. They are capable of modeling material and information flow phenomena, detecting the existence of deadlock states and inconsistency and are well suited for the study of Discrete Event Systems (DES). As such, they have been applied in manufacturing and logistics, hardware design and business processes, as well as in distributed databases and DES simulation [1]-[3].

There is no time measure in the original PNs. Hence, their modeling capability was limited and models often became excessively large, because all data manipulation had to be represented directly into the net structure. To overcome these drawbacks, over the years PNs have been extended in many directions including, time, data and hierarchy modeling. To adapt Petri nets to the study of systems' performances, Timed Petri Nets (TPNs), allowing for delays in the system, were introduced. There are several possibilities for introducing time in PNs, thus transforming them into TPNs, among them timed transitions and timed places (see e.g. [4] [5] for detailed presentations).

The objective of this paper is to show different ways to utilize this modeling frame. The paper reviews several published papers where the framework was used in different circumstances such as, estimating expected utilization of resources at steady state in open queueing networks, verifying computerized simulations and batch planning in textile industries.

The paper is organized as follows. Section 2, entitled "From Petri Nets to Timed Petri Nets", presents basic PN modeling and describes a technique for modeling TPNs, which associates time with places. Section 3 entitled "Decomposing TPNs" reviews a method for decomposing Timed Petri Nets of Open Queuing Networks. Section 4 describes the decomposition application for verifying computerized simulation. Section 5 entitled "Application of TPNs to textile processing" describes an additional published example. Section 6 concludes the paper.

2. From Petri Nets to Timed Petri Nets

The guiding principles of net theories concern states/conditions and changes/events, two fundamentally intertwined concepts. In a Petri Net the relationships between these two concepts are both graphically and analytically described. A PN consists of two parts: 1) its structure and 2) its marking representing the system overall state on the defined structure. The net structure describes its possible states (places) and events (transitions). When one or more markers (tokens) reside in a place, those tokens mark it and thus determine its local state. The state of the entire system is determined by the net marking, which is the distribution of markers to the net places. Tokens move according to firing rules. Firing of transitions and flows of tokens, model the dynamic behavior of the net. Let us first present PNs as a graphical tool and then as an analytical tool.

2.1. PNs as a Graphical Tool

A Petri net is a graph, $N=(T,P,A)$ where:

T is a set of transitions $(T_1,T_2,\cdots,T_n)$, portrayed by bars, representing instantaneous or primitive events.

P is a set of places $(P_1,P_2,\cdots,P_m)$, portrayed by circles, representing conditions.

I(T) denotes the set of all input places of transition T. Similarly, O(T) denotes the set of all output places of transition T.

A is a set of directed arcs (T*P) U (P*T) derived through an incidence matrix (see Section 2.2). A "marking", M of a Petri net is a distribution of tokens (or markers) to the places of a Petri net.

A transition can "fire" (meaning the respective event occurs) if there is at least one marker in each of its input places (the necessary conditions for its occurrence). Following the firing of a transition, the number of tokens in the corresponding input places is depleted by one.

Figure 1 illustrates a simple example of an ordinary PN, PN1.

Its structure is as follows:

$$T=\{T_1,T_2,T_3,T_4,T_5,T_6\},\quad P=\{P_0,P_1,P_2,P_3,P_4\}$$

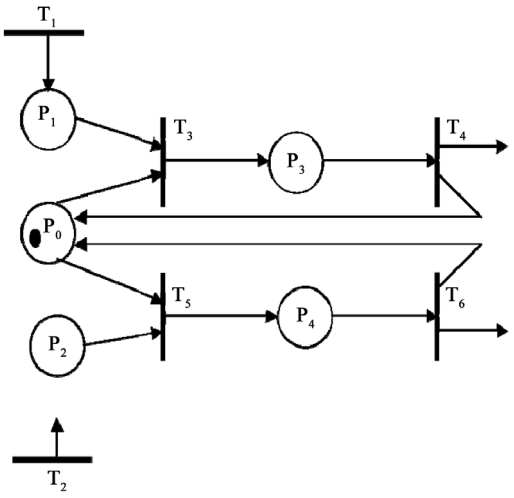

Figure 1. An ordinary Petri Net, PN1.

We see that the original marking of the net contains one token in place P_0. The firing of transition T_1 will insert one token in P_1. As P_0 and P_1 are the two input places of transition T_3, they will enable its firing. Firing of transition T_3 will remove the tokens in P_0 and P_1 and insert a token in P_3. Then, transition T_4 will fire and will thus remove the token in P_3 and insert a token in P_0. This process describes a firing sequence. Another firing sequence will take place if following the current initial state of the system, transition T_2 fires introducing a token in place P_2. The respective tokens in places P_0 and P_2 will enable the firing of transition T_5 that will remove the tokens in P_0 and P_2 and will add a token to place P_4. This will be followed by the firing of transition T_6 removing the token in P_4 and inserting a token in P_0.

Assume now that the initial state of the net contains not one token but three tokens, one in each of the three places P_0, P_1 and P_2. This situation represents a conflict as both transitions T_3 and T_5 are enabled. Which transition should fire, T_3 or T_5? A rule for solving the conflict should be introduced, or the system should be redesigned to eliminate the conflict.

2.2. PNs as an Analytical Tool

Formal analysis of the model can be carried out when a PN is described by linear algebraic equations.

Incidence Matrix

The structure of a PN is an integer matrix C, called an incidence or flow matrix. C is an $m \times n$ matrix whose m rows correspond to the places of the net and whose n columns correspond to the transitions. In ordinary PNs an element of this matrix is defined as follows:

$$C_{ij} = \begin{cases} -1 & \text{for } P_i \in I\left(T_j\right) \\ +1 & \text{for } P_i \in O\left(T_j\right) \\ 0 & \end{cases}$$

It means that upon firing transition T_j one token is removed from each input place P_i and one token is added to each output place. The incidence matrix $C1$ of the PN1 in **Figure 1** is given below.

$$C1 = \begin{pmatrix} 0 & 0 & -1 & 1 & -1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

By convention, the flow of tokens through transition T_j is denoted by I_j, j = 1, 2, 3, 4, 5, 6.

2.3. Time modeling of PNs

According to the version, introduced in [4] and adopted by others e.g. [6], Timed Petri Nets (TPNs) associate a positive number s(T) to each transition T. In [5] a different approach to model time in Petri nets was developed, relying on the classical Petri nets approach to a non-primitive event (with time greater than zero) which prescribes its decomposition into two transitions representing two primitive events. These are respectively a commencement and a termination of the original non-primitive event. The author associates time duration (delays) with the places describing the occurrence of the non-primitive events. Other scientists also adopted this approach. The delay is modelled either as a deterministic or as a stochastic variable. However, if a stochastic model is used, calculations only consider the expected values.

Our approach uses *timed places* to model processing of multiple part classes within a mix, repairs following machine breakdowns and eventually set-ups ensuing a change in mix. Since most of these variables are stochastic by nature, the actual values are their respective expected values. For instance, the input of parts to the system and the machine breakdowns are respectively the expected time between arrivals and the expected time between failures. It is worth mentioning that the current method is not Stochastic Petri Nets, which deal with non-primitive events whose duration is assumed to be exponentially distributed [7]. A Timed Petri Net (TPN) graph looks exactly like a PN graph, except that tokens in timed places become available only after z_i time units ($z_i > 0$). This delay is independent of the token's arrival instant. A TPN is described by its incidence matrix, which is exactly like that of its equivalent PN, to which a diagonal matrix Z, representing time delays, is added.

Let us now turn our attention back to **Figure 1**. Examining PN1 we observe that places P_3 or P_4 are as described above, meaning that a token in P_3 or in P_4 may well represent concurrent, time elapsing activities. We may now interpret PN1 as a PN describing a resource that provides service to two types of customers. The firing of transitions T_1 or T_2, respectively represents input of type 1 or type 2 customers to the system. A token in place P_1 or P_2, respectively, represents a type 1 or type 2 customers waiting to be served by the resource. A token in place P_0 shows that the resource is available. Firing of transition T_3 or T_5, respectively, expresses the beginning of servicing a type 1 or type 2 customers. Firing of transition T_4 or T_6 expresses the end of service. This occurs after $\mathbf{z}_3$ or $\mathbf{z}_4$, time units have respectively elapsed. Then, one token returns to P_0, meaning the resource is again available.

To convert PN1, illustrated in **Figure 1**, into TPN1, we use incidence matrix $C1$ and delay matrix $Z1$. $Z1$ is defined as a square diagonal matrix with elements $Z1 = z_i$, i = 0, 1, 2, 3, 4 for $i = j$ and zero otherwise. The timed places are $\mathbf{z}_3$ and $\mathbf{z}_4$ respectively representing service time of type 1 and type 2.

The graphical representation of TPN1 is also **Figure 1** to which $\mathbf{z}_3$ and $\mathbf{z}_4$ are respectively added (eventually in parentheses) below places 3 and 4.

$$Z1 = \begin{pmatrix} z_0 & 0 & 0 & 0 & 0 \\ 0 & z_1 & 0 & 0 & 0 \\ 0 & 0 & z_2 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{z}_3 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{z}_4 \end{pmatrix}, \quad I = \begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{pmatrix}$$

The flow of tokens through transition T_j, j = 1, 2, 3, 4, 5, 6 is vector I.

3. Decomposing Timed Petri Nets

The author of [8] provides a detailed illustration and proof for decomposing a TPN model of n-part type open queuing network with M workstations, into M independent TPNs. A part (customer) type is a composite of specific jobs to be performed by workstations (resources) in a predetermined order. In open queueing networks, customers enter and leave the network in an unconstrained manner.

The essence of the approach consists in proving that, under steady state conditions as defined in [5] and applied to open systems, the arrival flow of any such job to its assigned workstation equals the input flow to the system of its parent part type.

Figure 2 graphically illustrates an example of an open queueing network at steady state, as presented in [10]. It consists of $M = 3$ workstations and $n = 4$ part types. Parts of type j, (exogenous customers comprising the mix served by the queueing network) arrive at the system with given flow rate I_j^0, $j = 1, 2, 3, 4$. Processing of a part is composed of a number of operations (jobs) to be performed in a given order by previously assigned stations (deterministic routing). Upon arrival, each part is routed to its initial station. Part types 1 and 2 enter the system at workstation 1, while part types 3 and 4 enter the system at workstation 2. Part types 1 and 4 exit the system at workstation 3, while part types 2 and 3 exit at workstation 2. The flow labeling in **Figure 2** indicates the preservation of the exogenous input flows throughout the system. According to the conditions detailed in Section 3.1, this means the queueing network operates at steady state.

3.1. TPN Decomposition Rules

An important strategy for managing complex systems is breaking them down into appropriate subsystems and providing means for integrating them.

A TPN model of an n-part type open queueing network with M workstations can be decomposed into M independent TPNs, provided steady state conditions are fulfilled at each work station m, $m = 1, 2, \cdots, M$.

As stated in [9] and repeated in [5], given a timed Petri net by its incidence matrix C, the net functions at its steady-state rate for a given flow vector I^0 iff I^0 satisfies the equations:

$$CI^0 = 0 \quad \left(I^0 > 0\right) \tag{1}$$

$$J_s Q(0) = J_s Z\left(C^+\right) I^0 \tag{2}$$

$\left[J_s\right]$ $s = 1, 2, \cdots, k$ is a generator of the set of solutions of Equation (1) $\left[J_s C = 0\right]$.

Z is the delay matrix (on places) defined as a square diagonal matrix with elements $Z_{ij} = z_i$ $i = 1, 2, \cdots, m$ for $i = j$ and zero otherwise (see detailed matrix in Section 2.3).

$Q(0)$ is a vector representing the initial marking of the net.

(C^+) is obtained by replacing the "−1" elements in the incidence matrix C by zeros.

Intuitively Equations (1) and (2) can be respectively associated with conservation of flow and conservation of tokens.

3.2. Application of the Decomposition Rules

Let us now apply the decomposition rules to the queueing network in **Figure 2**. The two steady state conditions,

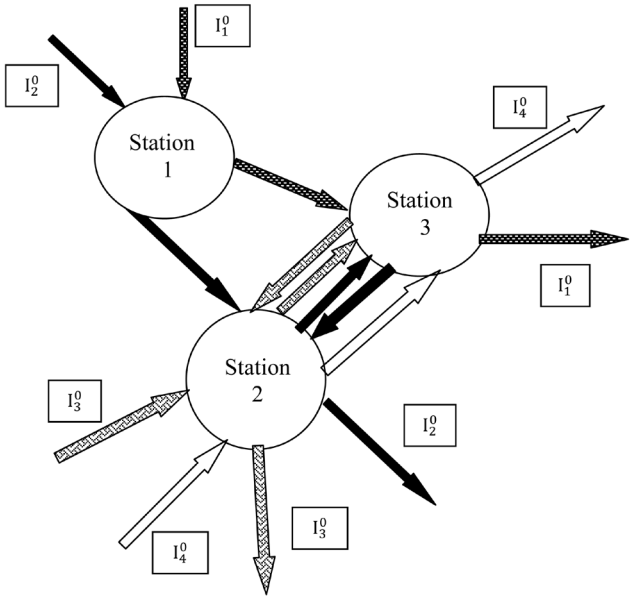

Figure 2. An open queueing network.

Equations (1) and (2), should be fulfilled by each of the three stations.

Here, we shall only analyze workstation 1. This station processes two part types, type 1 and type 2, thus matching TPN1 in **Figure 1**. We may notice that station 1 is the entering station of these two part types to the open queueing network. Hence, their arrival flow at this station is their respective exogenous flow to the system, I_1^0 and I_2^0. According to **Figure 2** when part type 1 leave station 1, they proceed to station 3, while part type 2 proceed to station 2.

Applying Equation (1) to the incidence matrix of **Figure 1**, $C1I^0 = 0 \quad (I^0 > 0)$, we obtain, respectively, from rows 2 and 3: $I_1^0 = I_3$ and $I_3 = I_4$. Rows 3 and 5, respectively, yield $I_2^0 = I_5$ and $I_5 = I_6$. The above results comply with the equation in row 1. Combining the equations, we obtain,

$$I_1^0 = I_3 = I_4 \quad I_2^0 = I_5 = I_6 \tag{3}$$

It is thus seen that under steady state conditions the entering (exogenous) flow of type 1 parts to the system, I_1^0 equals I_4, the firing rate of transition T_4 representing the exit flow of type 1 parts from station 1 and also their entering flow to station 3. Similarly, at steady state the entering (exogenous) flow of type 2 parts to the system, I_2^0 equals I_6, the exit flow of type 2 parts from station 1, which is also their entering flow to station 2.

In this example, one generator of the set of solutions of Equation (1) is J_1,

$$J_1 = (1,0,0,1,1) \quad J_1 C1 = 0 \tag{4}$$

The "1" values in J_1 represent all the possible steady states of workstation 1: P_0-idle, P_3-processing part type 1, P_4-processing part type 2.

Substituting Equation (4) in Equation (2) and using Equation (3) leads to:

$$Q_0 + Q_3 + Q_4 = z_0\left(I_1^0 + I_2^0\right) + \mathbf{z}_3 I_1^0 + \mathbf{z}_4 I_2^0 \tag{5}$$

Q_0, Q_3, Q_4 represent the token content of places P_0, P_3, P_4 for any firing sequence. Hence, the Left Hand Side (LHS) of Equation (5) represents the sum of tokens over the steady states of station 1. Since these three states represent all the mutually exclusive states of workstation 1, LHS equals 1. The three summation components of the Right Hand Side (RHS) represent the respective proportion of time station 1 spends in each state under steady-state conditions or, in probabilistic terms, it represents the steady state distribution over the three states. After substitution, Equation (5) becomes:

$$1 = z_0\left(I_1^0 + I_2^0\right) + \mathbf{z}_3 I_1^0 + \mathbf{z}_4 I_2^0 \tag{6}$$

$\mathbf{z}_3 I_1^0$ and $\mathbf{z}_4 I_2^0$ are the respective contributions of part type 1 and part type 2 to the utilization of station 1. As I_1^0 and I_2^0 represent the total entering flow to station 1, intuitively, z_0 is the station expected idle time between any two consecutive arrivals of parts. Any feasible solution has to satisfy $z_0 > 0$; leading to:

$$\mathbf{z_3} I_1^0 + \mathbf{z_4} I_2^0 < 1 \tag{7}$$

We may now use the above results to estimate the utilization of a station.

3.3. Estimating a Station Utilization Using TPN

Based on the application of the decomposition rules presented in the previous section, the procedure for computing the utilization of a station is summarized as follows:

The total expected steady-state utilization r_m of station m, $m = 1,2,\cdots,M$ is the sum of its expected utilization by each class of customers. These comprise the n-part types as determined by a given deterministic routing and eventual machine failures or other disturbances including eventual set-ups.

$$r_m = \sum_{j=1}^{n} r_j^{(m)} + r_f^{(m)} \qquad r_f^{(m)} = t_f^{(m)} * I_f \tag{8}$$

where $r_j^{(m)}$ and $r_f^{(m)}$ represent the respective contributions of I_j^0, the input flow of part types j, and that of I_f, the flow of disturbances, to the m station utilization, $j = 1,2,\cdots,n$, $m = 1,2,\cdots,M$

$$r_j^{(m)} = t_j^{(m)} * I_j^0 \tag{9}$$

$t_j^{(m)}$ is the expected operation duration of part j at station m.

$t_f^{(m)}$ is the expected duration of a disturbance.

Provided the constraint $r_m < 1$ is satisfied for each $m = 1, 2, \cdots, M$, and all buffers are of unlimited size, Equation (8) is a valid expression of the TPN based expected steady state utilization r_m of any processing station m, $m = 1, 2, \cdots, M$ in the system.

In the next section, we shall use Equation (8) for verifying simulation models results of computerized open queueing network at steady state.

4. Using Timed Petri Nets as a Simulation Verification Tool

This section is a simplified version of a paper describing the usage of decomposed Timed Petri Nets as an analytical approach for verification of simulation models [11].

An important and difficult task in any simulation project is the validation and verification of the simulation model. Validation is commonly defined as the process intended to substantiate that the conceptual simulation model, within its domain of applicability, is an accurate representation of the real world system it describes. The verification process seeks to determine that the computerized simulation model was built right, and operates as intended. There is a rich literature on these topics, see e.g. [12]-[14].

From an output analysis perspective, system simulation is typically classified into *terminating* or *steady state*. While terminating simulations aim to estimate system parameters for periods well defined in terms of starting and ending conditions, steady state simulations aim to estimate system parameters as limits obtained for infinite simulation time. Before reaching *steady state* conditions when the system parameters attain stable values, the system behavior undergoes transition periods, reflecting the initial system operating conditions (see e.g. [15]). In the analysis of such systems, an important objective is to avoid bias in parameter estimation, eventually introduced by data collected during transition periods. One among the many approaches to simulation verification is comparing simulation outputs with analytical results [16]. In many simulation studies dealing with manufacturing systems or computers and communication systems, the networks of queues represents the reality modelled by the system simulation. Hence, for simulation verification of such systems analytical approaches describing *queueing networks* may be appropriate.

The TPN approach puts into evidence the flow realization along a variety of network paths and may consider different network structures. Such a decomposed TPN model of a resource enables a quick calculation of its expected utilization at steady state thus providing a stick yard against which to verify the long run simulation based utilization estimate of the same resource. In other words, it represents a verification method.

4.1. A Numerical Example

There are three stations ($M = 3$) and four part types ($n = 4$).

1) The inter-arrival time for part j, $j = 1, 2, 3, 4$ is Erlang distributed with mean A_j (time units)

J	1	2	3	4
A_j	10	100	25	20

The reciprocal of the inter-arrival mean serves to estimate the average input flow of parts. Hence the input flows (in parts per time unit, pptu) are:

I_j^0	1/10	1/100	1/25	1/20

2) The service times (in time units) of parts j, $j = 1, 2, 3, 4$ at stations $m = 1$ and $m = 3$ are normally distributed, while the service time at station $m = 2$ is Uniformly distributed, each with their respective mean, $t_j^{(m)}$, $m = 1, 2, 3$.

4.1.1. The Simulation Verification Procedure

We shall first use Equation (9) to calculate $r_j^{(m)}$, for $j = 1, 2, 3, 4$, the contribution of each part type to the station's utilization, and then Equation (8) to calculate r_m, the total expected utilization of station m ($m = 1, 2, 3$).

	j	$t_j^{(m)}$	I_j	$r_j^{(m)}$
$m = 1$	1	5.0	1/10	0.500
	2			
	3	2.0	1/25	0.080
	4	4.0	1/20	0.200
			$r_1 = 0.780$	
$m = 2$	1	1.7	1/10	0.170
	2	2.7	1/100	0.027
	3	2.9	1/25	0.116
	4	4.4	1/20	0.220
			$r_2 = 0.533$	
$m = 3$	1	3.0	1/10	0.30
	2	4.0	1/100	0.04
	3	2.5	1/25	0.10
	4	7.0	1/20	0.35
			$r_3 = 0.790$	

4.1.2. Simulation Results and Conclusions

The simulation project used Siman V software and the Arena framework. As the published example investigated the effect of several factors such as the planned utilization of the stations and the queue discipline, forty-eight simulation runs were carried out. Here we do not investigate such effects. We solely compared the TPN calculated estimates with the simulation average results after we deleted the transient period (about 400 minutes). The simulation average estimating the station utilizations were close to those calculated using TPN and their differences were no higher than 2%.

The relatively low differences obtained between the numerical results of the two methods provide evidence that the TPN technique is able to verify simulation models. Observed discrepancies between the TPN based and the results of the computerized simulation model can thus detect eventual flaws in the simulation model.

5. Application of TPNs to Textile Processing

We shall briefly present an additional example dealing with the application of TPNs to textile batch planning [17].

The need for advanced methods to model and characterize the performance of planning and scheduling is especially apparent in the textile dyeing and finishing operations. There is a huge number of process combinations that a fiber, yarn or fabric may go through on its way to the final product. Well organized planning and control of the entering materials, flowing through the intricate manufacturing operations to reach their final state is critical for meeting customers due dates and thus provide on time delivery. The TPN decomposition approach has been developed and applied to calculate long run resource utilization for manufacturing systems of discrete items. The textile manufacturing conditions are different. Workstations (machines) are of two types: batch processing (especially for dyeing) and continuous processing (for finishing operations such as squeezing or drying). The discrete items entering the system are orders. Orders vary in size, fabric type, fiber blend, color and the kinds of required operations. For batches, an order “size” is measured in weight units while for continuous processing it is measured in length units. An order is characterized by its specific processing requirements and by its specific attributes such as fiber blend and Machine Direction (MD) weight (kg/m). Incoming orders are stored and wait for processing. There are two main operation categories: equipment oriented operations and work force oriented operations such as lot transporting, loading and unloading operations. As the relative duration of the latter is insignificant, here these operations are disregarded.

To apply the TPN decomposition approach for calculating the expected (long run) utilization at steady state of the equipment used in the textile processes, we shall adapt the basic assumptions in [8], detailed in Section 3, to a textile dyeing workstation.

5.1. Manufacturing Assumptions of the Textile Dyeing Operations

This section is an extension of Section 3.3, which estimates a station utilization of discrete systems.

Orders of type *j*, arrive at the system with given flow of weights, F_j^0, $j=1,2,\cdots,n$ (Kg/hour). Processing of an order is composed of a number of operations (jobs) to be performed in a given order by previously assigned stations (deterministic routing). Upon arrival, each order is routed to its initial station, where it is stored and waits for processing. Typically, the initial station is a dyeing workstation, the example here.

5.1.1. Dyeing Operations

- The dyeing operations are performed in *batches*, not exceeding the given manufacturing capacity of the dyeing machines. We assume that all dyeing machines have the same manufacturing capacity, *G* (kg/batch).
- A batch of type *j* is the maximal accumulated weight of type *j* orders in store, since the starting of the last batch, not exceeding the machine capacity, *G*.
- The average input flow of type *j* weights, F_j^0, is transformed into I_j^0, the average input flow of type *j* batches, and is calculated as, $I_j^0 = F_j^0 / G$ (batches/hour).
- A dyeing machine, *m*, has a batch processing duration, $t_{j,k}^{(m)}$, for processing operation *k* of type *j* order, $k=1,2,\cdots,v_j$, $j=1,2,\cdots,n$, $m=1,2,\cdots,M$.

The duration of this operation is dependent on the required "basic dyeing operation", such as washing, bleaching or dyeing to a specific shade as well as on the specific order attribute in terms of "fiber blend".

5.1.2. Evaluating the Utilization of the Dyeing Machines at Steady State Using TPN

Let $r_j^{(m)}$ be the contribution of I_j^0, the input flow of type *j* batches, for possible multiple visits to station *m*.

$$r_j^{(m)} = \sum_{k=1}^{v_j} R_{j,k}^{(m)} t_{j,k}^{(m)} I_j^0 \tag{10}$$

for $j=1,2,\cdots,n$, $m=1,2,\cdots,M$.

$R_{j,k}^{(m)}$ is the given routing of batches *j* to machine m for performing operation *k*. It may assume a "0" or "1" value.

v_j is the number of operations of type *j* batches at machine *m*.

$t_{j,k}^{(m)}$ is the expected processing duration of operation *k* of batch *j* at station *m*.

The total expected steady-state utilization r_m of station *m*, $m=1,2,\cdots,M$ is the sum of its expected utilization by each class of customers to be served by the machine. Here these are the n different types of batches as determined by the given deterministic routing and eventually failures or other disturbances as an additional class of customers. As the batches replace the discrete part types in Section 3.3, the following equation is identical with Equation (8),

$$r_m = \sum_{j=1}^{n} r_j^{(m)} + r_f^{(m)} \qquad r_f^{(m)} = t_f^{(m)} * I_f .$$

Provided the constraint $r_m < 1$ is satisfied for each $m=1,2,\cdots,M$, and all buffers are of unlimited size, the above equation is a valid expression of the TPN based expected steady state utilization r_m, of any dyeing station *m*, $m=1,2,\cdots,M$ in the system.

5.2. An Example

The example is based on a real textile dyeing and finishing mill which performs about 25 - 30 specific processes. We have selected eight among these processes, whose accumulated input flow in kg/hour represents about 77% of the total input flow to the system.

5.2.1. Expected Utilization of the Dyeing Machines

The average overall input order rate is 2 orders per hour and the mean weight per order is 170 kg. Accordingly, the total input flow of weights of the selected processes to the first processing machine, which is a dyeing machine (DM) is 262 kg/hour. There are 10 DM machines assigned to the selected processes and the manufacturing capacity of each is *G* = 250 Kg.

A type *j order* arriving at a dyeing machine where its first operation (*k* = 1) will be performed, is characte-

rized by its "basic dyeing operation" (washing, bleaching, pale shade or dark shade, *i.e.* four possibilities) and by its specific fiber blend (cotton/viscose, cotton/lycra, cotton/polyester, polyamide or acetate, *i.e.* five possibilities). The batch processing duration at any dyeing machine is dependent on the above combinations.

A type *j batch* comprises fabrics requiring the same type of basic operation and the same dyeing duration. Following this rule, eight types of batches were defined, each with a given weight flow F_j^0 (kg/hour) and given batch duration, $t_{j,1}^{(m)}$, $j=1,2,\cdots,8$, $m=1,2,\cdots,10$, see **Table 1**. Assuming that only one batch type is assigned to a machine, the results in **Table 1** show that for batches of type *j*, *j* = 1, 2, 3, 4, 5, 6 and 7, one machine is needed for each, whereas for *j* = 8, four machines are needed. The total number of machines needed becomes 11, while the total number of machines available for these processes is only 10. Hence, we have to assign more than one type of batch to a machine.

To switch from one basic dyeing operation to another requires a complete cleaning operation of the machine whose duration is 4 hours. To switch between fiber blends within the same basic dyeing operation requires partial cleaning whose duration is 2 hours. It is to be noted that *j* = 2, 3 need the same basic dyeing operation and so do *j* = 4, 5, 6 and *j* = 7, 8.

Considering these set-ups as additional types of customers, we shall calculate their contribution to the machine utilizations using Equation (8) as well.

To obtain a feasible planning solution to our problem (less than 11 machines) we have to select at least two batch types to be processed by the same machine while keeping the expected utilization of each machine below 1. Batch types 5 and 6, respectively allocated to machines 5 and 6, contribute to a low utilization of these two machines and they need only partial cleaning for switching, 2 hours.

Hence we may select batch types *j* = 5 and *j* = 6, to be performed by machine 5. The set-up duration for cleaning, $t_f^{(m)}=2$, has to be added to the respective processing duration of both batch type 5 and batch type 6.

Accordingly, the expected utilization of machine 5 is calculated as follows:

$$I_5^0 * \left(t_{5,1}^{(m)} + t_f^{(m)}\right) + I_6^0 * \left(t_{6,1}^{(m)} + t_f^{(m)}\right) = 0.018(5+2) + 0.021(3.5+2)$$
$$= 0.126 + 0.121 = 0.247$$

In the same manner, we may further reduce the number of operating machines.

5.2.2. Conclusion

In this example, we extended the TPN decomposition approach as applied to manufacturing systems of discrete items to accommodate conditions existing in textile processing systems. Under these conditions, the input to the system is through flows of orders. These were transformed into flows of batches, which now replace the flows of discrete items. The procedure also assists in planning the allocation of different types of batches to machines.

Table 1. Machine utilizations by different batch types.

j	F_j^0 kg/hour	$I_j^0 = F_j^0/250$ batches *j*/hour	$t_{j,1}^{(m)}$ machine hours/batch *j*	$I_j^0 * t_{j,1}^{(m)}$ machine hours/hour
1	26.2	0.105	1	0.105
2	30.0	0.12	3	0.36
3	51.0	0.204	4	0.816
4	17.0	0.068	6	0.408
5	4.6	0.018	5	0.09
6	5.4	0.021	3.5	0.077
7	26.0	0.104	4.5	0.468
8	102	0.408	8.0	3.264
All	262	1.048		

6. Discussion

The reviewed published papers showed that Petri nets are a versatile modeling structure. Section 2 of this paper presented the important extension of Petri nets to Timed Petri Nets (TPNs), which are able to time-base evaluate system performances. The section described in some details a technique for introducing time in Petri nets, by associating it with places. Section 3 showed how timed petri nets of open queuing networks were decomposed. Through decomposition, complex networks systems became easier to manage. Long run expected utilization of workstations at steady state, could be quickly calculated using decomposed TPNs. This enabled to use TPNs as a simulation verification method, as described in Section 4. The reviewed paper compared simulation results with TPN easily calculated estimates of stations utilizations. The low differences between the two techniques showed the advantages of TPNs as a simulation verification method. In the last presented paper, the decomposition approach to TPNs was applied to textile industry. Orders of different types are customary input to textile processing. The described procedure transformed the input flows of different types of orders into input flows of different types of batches, enabling to estimate the long run machines utilizations. The procedure also assisted in planning the allocation of different types of batches to operating machines.

References

[1] Billington, J., Diaz, M. and Rozenberg, G. (1999) Application of Petri Nets to Communication Networks. Springer-Verlag, Berlin. http://dx.doi.org/10.1007/BFb0097770

[2] Yakovlev, A., Gomes, L. and Lavagno, L. (2000) Hardware Design and Petri Nets. Kluwer, Boston. http://dx.doi.org/10.1007/978-1-4757-3143-9

[3] Girault, C. and Valk, R. (2002) Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications. Springer-Verlag, Berlin.

[4] Peterson, J.L. (1981) Petri Net Theory and the Modeling of Systems. Prentice Hall, Englewood Cliffs.

[5] Sifakis, J. (1980) Performance Evaluation of Systems Using Nets, Net Theory and Applications. Springer-Verlag, New York.

[6] Narahari, Y. and Viswanadham, N. (1985) A Petri Net Approach to the Modelling and Analysis of Flexible Manufacturing System. *Annals of Operations Research*, **3**, 449-472. http://dx.doi.org/10.1007/BF02023780

[7] Molloy, M.K. (1982) Performance Analysis Using Stochastic Petri Nets. *IEEE Transactions on Computers*, **31**, 913-917. http://dx.doi.org/10.1109/TC.1982.1676110

[8] Barad, M. (1994) Decomposing Timed Petri Nets of Open Queueing Networks. *Journal of the Operational Research Society*, **45**, 1385-1397. http://dx.doi.org/10.1057/jors.1994.215

[9] Sifakis, J. (1977) Use of Petri Nets for Performance Evaluation. *Acta Cybernetica*, **4**, 185-202.

[10] Barad, M. (2003) An Introduction to Petri Nets. *International Journal of General Systems*, **32**, 565-582. http://dx.doi.org/10.1080/03081070310001623366

[11] Barad, M. (1998) Timed Petri Nets as a Verification Tool. *Proceedings of Winter Simulation Conference*, Washington DC, 13-16 December 1998, Vol. 1, 547-554. http://dx.doi.org/10.1109/WSC.1998.745033

[12] Shannon, R.E. (1981) Tests for the Verification and Validation of Computer Simulation Models. *Proceedings of the Winter Simulation Conference*, 573-577.

[13] Sargent, R.G. (1991) Simulation Model Verification and Validation. *Proceedings of the Winter Simulation Conference*, Phoenix, 8-11 December 1991, 37-47. http://dx.doi.org/10.1109/wsc.1991.185589

[14] Balci, O. (1994) Validation, Verification and Testing Techniques throughout the Life Cycle of a Simulation Study. *Annals of Operations Research*, **53**, 121-173. http://dx.doi.org/10.1007/BF02136828

[15] Kelton, W.D. (1989) Random Initialization Methods in Simulation. *IIE Transactions*, **21**, 355-367. http://dx.doi.org/10.1080/07408178908966242

[16] Kleijnen, J.P.C. (1995) Verification and Validation of Simulation Models. *European Journal of Operational Research*, **82**, 145-162. http://dx.doi.org/10.1016/0377-2217(94)00016-6

[17] Barad, M. and Cherkassky, A. (2009) Timed Petri Nets for Textile Batch Processing under Varying Input Characteristics. *Proceedings of* 20*th ICPR*, Shanghai, China.