9-th IEEE Int. Conference on Methods and Models in Automation and Robotics, Special Session on Analysis and Control of Discrete/Hybrid Systems; Miedzyzdroje, Poland, 25–28 August 2003, vol.2, pp.877-882.

STRUCTURAL METHODS IN PERFORMANCE ANALYSIS OF DISCRETE-EVENT SYSTEMS

W.M. ZUBEREK

Department of Computer Science, Memorial University, St.John's, Canada A1B 3X5, wlodek@cs.mun.ca

Abstract. Timed Petri nets, a formalism developed specifically for representation of concurrent activities, are popular models of discrete-event system. For net models which can be decomposed into simple cyclic subnets, the performance can be determined on the basis of components, without the exhaustive analysis of the behavior of the whole system; such an approach is known as structural analysis. The paper overviews the formal basic concepts of structural methods and uses manufacturing cells as an illustration of structural analysis.

Keywords. Timed Petri nets, structural methods, performance analysis.

1. INTRODUCTION

Discrete–event concurrent systems are often represented by Petri nets [10, 8], a formalism developed specifically for representation of concurrent activities and their coordination, i.e., for ordering specific actions or for synchronization of actions performed by different components of the system.

In order to analyze the performance of such models, the durations of activities must also be taken into account. Several types of nets "with time" have been proposed by associating "time delays" with places [11], or occurrence durations with transitions [9, 14] of net models. Also, the introduced temporal properties can be deterministic [9, 11, 14], or can be random variables described by probability distribution functions (the negative exponential distribution being probably the most popular choice) [1, 14].

Two basic approaches to analysis of net models are known as reachability analysis and structural analysis. Reachability analysis is based on the behavior of models, represented by the set of states and transitions between the states. For complex models, the exhaustive reachability analysis can easily become difficult because of a very large number of states (for some models, the number of states grows exponentially with model parameters which is known as the 'state explosion problem'). An alternative approach to reachability analysis is known as structural analysis, which predicts the performance of net models on the basis of model structure, without analyzing the state–based behavior. The most popular example of this approach is analysis based on place–invariants for models covered by families of simple cyclic subnets (implied by the place–invariants).

The paper uses models of manufacturing cells [2, 3] as an illustration of the use of structural approach. Manufacturing cells are simple manufacturing systems in which a collection of (flexible) machines and transportation systems are serviced by a robot which moves the parts from one machine to another, from the incoming transportation system to the first machine, and from the last machine to the outgoing transportation system. The (cyclic) sequence of the operations performed by the robot determines the performance (e.g., the throughput) of the cell.

Section 2 recalls basic concepts of timed Petri nets. Section 3 introduces Petri net models of manufacturing cells and uses the structural approach to find the cycle time of a cell as its basic performance indicator. Several concluding remarks are given in Section 4.

2. PETRI NETS

Petri nets are known as a simple and convenient formalism for modeling systems that exhibit parallel and concurrent activities [8], [10]. In Petri

nets, these activities are represented by the so called tokens which can move within a (static) graph–like structure of the net. More formally, a marked place/transition Petri net $\mathcal{M}$ is defined as $\mathcal{M} = (\mathcal{N}, m_0)$, where the structure $\mathcal{N}$ is a bipartite directed graph, $\mathcal{N} = (P, T, A)$, with a set of places P, a set of transitions T, a set of directed arcs A connecting places with transitions and transitions with places, $A \subseteq T \times P \cup P \times T$; m_0 is the initial marking function which assigns nonnegative numbers of tokens to places of the net, $m_0 : P \to \{0, 1, ...\}$.

A place is shared if it is connected to more than one transition. A shared place p is free–choice if the sets of places connected by directed arcs to all transitions sharing p are identical. A net is free-choice if all its shared places are free–choice. A net is structurally (or statically) conflict–free if it does not contain shared places. A marked net is dynamically conflict–free if for any marking reachable from the initial marking, and for any shared place, at most one of transitions sharing this place is enabled. The models of manufacturing cells discussed in this paper are (statically or dynamically) conflict–free nets.

In order to study performance aspects of Petri net models, the duration of activities must also be taken into account and included into model specifications. In timed nets [14], occurrence times are associated with transitions, and transition occurrences are real–time events, i.e., tokens are removed from input places at the beginning of the occurrence period, and they are deposited to the output places at the end of this period (sometimes this is also called a three–phase firing mechanism as opposed to one–phase instantaneous occurrences of transitions in stochastic nets [1] and time nets [7]). All occurrences of enabled transitions are initiated in the same instants of time in which the transitions become enabled (although some enabled transitions cannot initiate their occurrences). If, during the occurrence period of a transition, the transition becomes enabled again, a new, independent occurrence can be initiated, which will overlap with the other occurrence(s). There is no limit on the number of simultaneous occurrences of the same transition. Similarly, if a transition is enabled "several times" (i.e., it remains enabled after initiating an occurrence), it may start several independent occurrences in the same time instant.

More formally, a conflict–free timed Petri net is a pair, $\mathcal{T} = (\mathcal{M}, f)$, where $\mathcal{M}$ is a marked net and f is a timing function which assigns a (constant or randomly distributed) occurrence time to each transition of the net, $f : T \to \mathbf{R}^+$, where $\mathbf{R}^+$ is the set of nonnegative real numbers.

The occurrence times of transitions can be either deterministic or stochastic (i.e., described by some probability distribution function); in the first case, the corresponding timed nets are referred to as D–timed nets, in the second, for the (negative) exponential distribution of firing times, the nets are called M–timed nets (Markovian nets). In both cases, the concepts of state and state transitions have been formally defined and used in the derivation of different performance characteristics of the model [14]. Only D–timed Petri nets are used in this paper.

Each place/transition net $\mathcal{N} = (P, T, A)$ can be conveniently represented by a connectivity (or incidence) matrix $\mathbf{C} : P \times T \to \{-1, 0, 1\}$ in which places correspond to rows, transitions to columns, and the entries $\mathbf{C}[p, t]$, $p \in P$, $t \in T$, are defined as:

$$\mathbf{C}[p,t] = \begin{cases} -1, & \text{if } (p,t) \in A \wedge (t,p) \notin A, \\ +1, & \text{if } (t,p) \in A \wedge (p,t) \notin A, \\ 0, & \text{otherwise.} \end{cases}$$

Connectivity matrices disregard 'selfloops', that is, pairs of arcs (p, t) and (t, p). Pure nets are defined as nets without selfloops [10].

A P–invariant (place invariant, sometimes also called S–invariant) of a net $\mathcal{N}$ is any nonnegative, nonzero integer (column) vector I which is a solution of the matrix equation

$$\mathbf{C}^T \times I = 0,$$

where $\mathbf{C}^T$ denotes the transpose of matrix $\mathbf{C}$. It follows immediately from this definition that if I_1 and I_2 are P–invariants of $\mathcal{N}$, then any linear (positive) combination of I_1 and I_2 is also a P–invariant of $\mathcal{N}$. A basic P–invariant of a net is defined as a P–invariant which does not contain simpler invariants.

Similarly, a T–invariant (transition invariant) of a net $\mathcal{N}$ is any nonnegative, nonzero integer (column) vector J which is a solution of the matrix equation

$$\mathbf{C} \times J = 0.$$

As before, a basic T–invariant of a net is defined as a T–invariant which does not contain simpler invariants.

Moreover, a net $\mathcal{N}_i = (P_i, T_i, A_i)$ is a P_i-implied subnet of a net $\mathcal{N} = (P, T, A)$, $P_i \subset P$, if:

(1) $A_i = A \cap (P_i \times T \cup T \times P_i)$;

(2) $T_i = \{t \in T \mid \exists\, p \in P_i : (p,t) \in A_i \vee (t,p) \in A_i\}$.

It should be observed that in a pure net $\mathcal{N}$, each P–invariant I of a net $\mathcal{N}$ determines a P_I-implied (invariant) subnet of $\mathcal{N}$, where $P_I = \{p \in$

$P \mid I(p) > 0\}$; P_I is sometimes called the support of the invariant I. All nonzero elements of I select rows of $\mathbf{C}$, and each selected row i corresponds to a place p_i with all its input and all output arcs associated with it.

Finding basic invariants is a 'classical' problem of linear algebra, and there are known algorithms to solve this problem efficiently [5], [6].

Net invariants can be very useful in performance evaluation of net models; if a net is covered by a family of conflict–free cyclic subnets, the cycle time of the net, τ_0, is equal to the maximum cycle time of the covering subnets [9], [11]:

$$\tau_0 \ = \ \max(\tau_1, \tau_2, ..., \tau_k)$$

where k is the number of subnets covering the original net, and each τ_i, $i = 1, ..., k$, is the cycle time of the subnet i, which is equal to the sum of occurrence times associated with the transitions, divided by the total number of tokens assigned to the subnet:

$$\tau_i \ = \ \frac{\sum_{t \in T_i} f(t)}{\sum_{p \in P_i} m(p)}.$$

For net models which are conflict–free (or deterministic), but are covered by a family of non–conflict–free, the cycle time of the net, τ_0, takes into account the frequencies of transition firings, determined by the T–invariant (deterministic models have only one T–invariant):

$$\tau_i \ = \ \frac{\sum_{t \in T_i} v(t) * f(t)}{\sum_{p \in P_i} m(p)}$$

where $v(t)$ is the element of the T–invariant corresponding to t (these elements are often called "visit rates").

In many cases, the number of basic P–invariants can be reduced by removing from the analyzed net the elements which do not affect the performance of models [13]. Fig.1 shows one of such transformations; it eliminates the so called "implicit place" [4] which creates a parallel path that has has no influence on the behavior of a timed net, but which can increase the number of inessential (basic) P–invariants.

Fig.1 part (a) shows a simple case of parallel paths, while part (b) shows a more intricate case, which still can be simplified without affecting the performance of the model (in fact, the state space in both cases is not affected by the transformation). It should be noted that the reduction of simple parallel paths can be performed only if either all paths are unmarked, as in Fig.1 (a), or all are marked, as in Fig.1(b); the paths to be reduced cannot be "mixed", i.e., one path marked and the other unmarked.

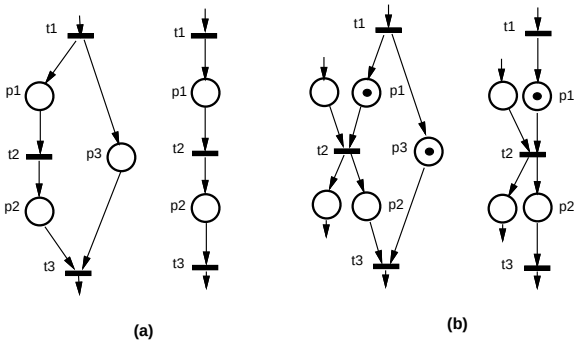

Fig.1. Reduction of implicit places.

3. MODELS OF MANUFACTURING CELLS

A simple manufacturing cell with three machines, M1, M2 and M3, an input conveyor In, an output conveyor Out, and a robot, is outlined in Fig.2. For simplicity, it is assumed that all part are processed by consecutive machines of the cell, first M1, then M2, and so on.

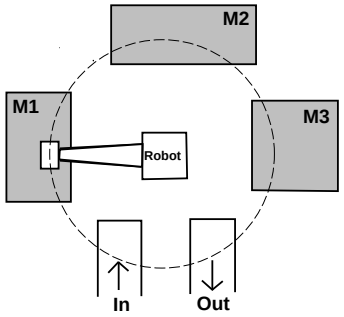

Fig.2. An outline of a 3–machine cell.

The cyclic behavior of a cell can be described by a sequence of operations performed by the robot, such as picking a part from the input conveyor, transporting it to a machine and loading it, then (when the machine's operation is finished) unloading the part, transporting it to another machine, and so on. A systematic approach to the derivation of all possible schedules is presented in [13]. For the case when the time of the operation performed by M3 is comparable with the sum of the times of operations performed by M1 and M2, the following schedule can be appropriate (it is assumed that each cycle of operations uniformly begins by picking a new part from In, and that Pi denotes the operation of picking a new part from the input conveyor, Do - the operation of dropping a part on the output conveyor, that Lj and Uj denote the operations of loading a part on machine Mj and unloading the part from Mj, and, finally, Mjk denotes the "moves" of the robot from position "j" to position "k", so $M13$ denotes the move from machine M1 to machine M3, $M3o$ – the move from machine M3 to the output conveyor,

and $Mi1$ – the move from the input conveyor to machine M1):

$$Pi, Mi1, L1, U1, M12, L2, M23, U3, M3o, Do, Mo2, U2, M23, L3, M3i$$

A timing diagram for this schedule is shown in Fig.3.

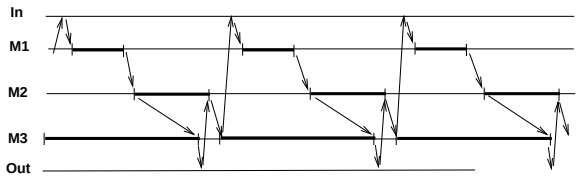

Fig.3. Example timing diagram for schedule 1.

A different schedule may be needed if the relations between the durations of operations performed by the machines of the cell are different. Fig.4 illustrates the case when the time of M2's operation is comparable with the sum of the times of operations performed by M1 and M3. In this case, the detailed (cyclic) sequence of robot operations is:

$$Pi, Mi1, L1, M12, U2, M23, L3, M31, U1, M12, L2, M23, U3, M3o, Do, Moi$$

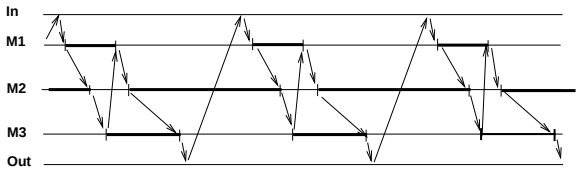

Fig.4. Example timing diagram for schedule 2.

Petri net models are direct representations of the detailed schedules, with each machine represented by a "standard" model shown in Fig.5, in which transition t_{ia} represents the "load" operation, transition t_{ib} the "unload" operation, and transition t_i the operation performed by machine M_i.

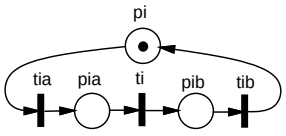

Fig.5. Petri net model of a single machine.

The complete model is shown in Fig.6. It contains three machine subnets (as in Fig.5) and the remaining part is the robot subnet which follows the sequence of robot steps. The transitions (and times associated with them) represent the following operations:

transition	*operation*
t_1	M1 operation
t_2	M2 operation
t_3	M3 operation
t_{in}	picking a part from Input
t_{out}	dropping a part in Output
t_{01}	moving from Input to M1
t_{02}	moving from Output to M2
t_{1a}	loading M1
t_{1b}	unloading M1
t_{12}	moving from M1 to M2
t_{2a}	loading M2
t_{2b}	unloading M2
t_{23}	moving from M2 to M3
t_{3a}	loading M3
t_{3b}	unloading M3
t_{3i}	moving from M3 to Input
t_{30}	moving from M3 to Output
t_{32}	moving from M2 to M3

After eliminating places p_1, p_2, p_3 and p_{11} with their arcs, the model has 3 P–invariants, with the following transitions in the implied subnets:

invariant	*set of transitions*
1	t_{1a}, t_1, t_{1b}, t_{12}, t_{2a}, t_2, t_{2b}, t_{23}, t_{3a}, t_{3i}, t_{in}, t_{01}
2	t_{1a}, t_1, t_{1b}, t_{12}, t_{2a}, t_{32}, t_{3b}, t_{30}, t_{out}, t_{02}, t_{2b}, t_{23}, t_{3a}, t_{3i}, t_{in}, t_{01}
3	t_{2b}, t_{23}, t_{3a}, t_3, t_{3b}, t_{30}, t_{out}, t_{02}

Since each of these subnets contains just one token, the cycle time is determined by the maximum sum of times associated with transitions in each subnet:

$$\tau_0 = \max(\tau_1, \tau_2, \tau_3)$$

where:

$$\tau_1 = f(t_{1a}) + f(t_1) + f(t_{1b}) + f(t_{12}) + f(t_{2a}) + f(t_2) + f(t_{2b}) + ... + f(t_{01}),$$
$$\tau_2 = f(t_{1a}) + f(t_1) + f(t_{1b}) + f(t_{12}) + f(t_{2a}) + f(t_{32}) + f(t_{3b}) + ... + f(t_{01}),$$
$$\tau_3 = f(t_{2b}) + f(t_{23}) + f(t_{3a}) + ... + f(t_{02}).$$

After eliminating p_1, p_2 and p_3, the net shown in Fig.7 has 5 P–invariants with the following sets of transitions in the implied subnets:

invariant	*set of transitions*
1	t_{1a}, t_1, t_{1b}, t_{12}, t_{2a}, t_2, t_{2b}, t_{23}, t_{3a}, t_3, t_{3b}, t_{30}, t_{out}, t_0, t_{in}, t_{01}
2	t_{1a}, t_1, t_{1b}, t_{12}, t_{2a}, t_{32}, t_{3b}, t_{30}, t_{out}, t_{3i}, t_{in}, t_{01}
3	t_{1a}, t_{21}, t_{2b}, t_{23}, t_{3a}, t_3, t_{3b}, t_{30}, t_{out}, t_{3i}, t_{in}, t_{01}
4	t_{1a}, t_{21}, t_{2b}, t_{23}, t_{3a}, t_{31}, t_{1b}, t_{12}, t_{2a}, t_{32}, t_{3b}, t_{30}, t_{out}, t_{3i}, t_{in}, t_{01}
5	t_{2a}, t_2, t_{2b}, t_{23}, t_{3a}, t_{31}, t_{1b}, t_{12}

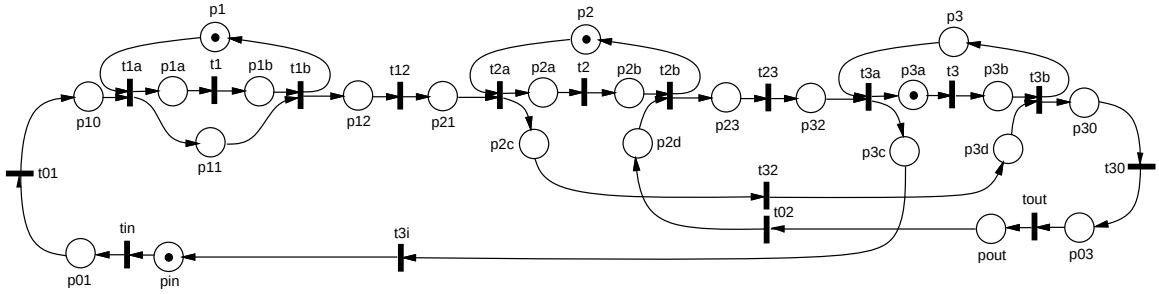

Fig.6. Petri net model of schedule 1.

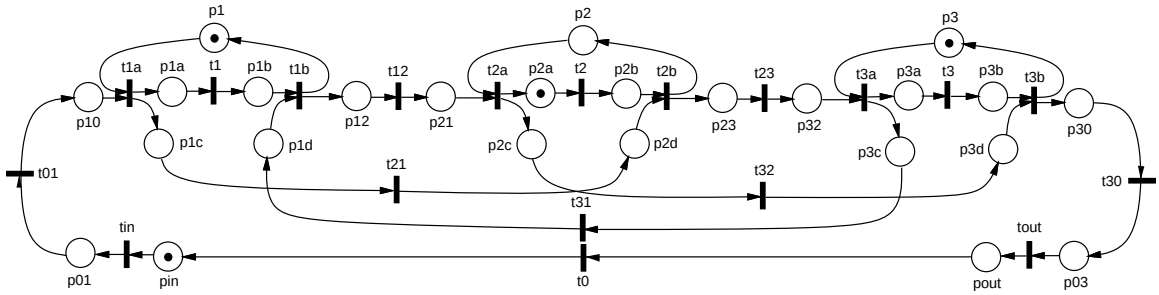

Fig.7. Petri net model of schedule 2.

The corresponding subnets can easily be traced in Fig.7, and the cycle time is obtained, as before, by adding times associated with transitions in the subnets.

The two models shown in Fig.6 and Fig.7 can be combined into a single model corresponding to processing two different products, one according to schedule shown in Fig.6 and the second according to Fig.7. More detailed discussion of conditions for such a "fusion" of schedules is given in [15]. For the combined model the representation of machines are slightly different to model processing of two different products, with possibly different load, unload and processing times, as shown in Fig.8. Each machine model is a free–choice structure with the number of (parallel) branches equal to the number of different products.

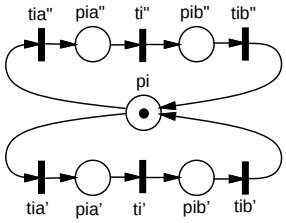

Fig.8. Extended Petri net model of a single machine.

Fig.9 shows a complete model representing a cell processing two different products, say A and B, in a sequence ABABAB...; only minor change is needed for implementation of other patterns such as AABAAB..., ABBAABBA... and so on (the subnet in upper right corner determines the pattern of products).

The three machine models (as in Fig.8) are in the central part of Fig.9, the upper part corresponds schedule shown in Fig.6 (product A), and the lower part to schedule shown in Fig.7 although some details are different because of coordination of the two schedules.

The model has one T–invariant which assigns the value 1 to all transitions except of t_{30} and t_{out}, for which the invariant values are equal to 2 (i.e., in each cycle, which is comprised of schedule A followed by schedule B, t_{30} and t_{out} occur twice, while all other transitions have a single occurrence). After elimination of place p_{11}, the model shown in Fig.9 has 4 essential P–invariant–implied subnets (all other subnets have sets of transitions which are subsets of the essential subnets).

4. CONCLUDING REMARKS

The paper illustrates the use of structural methods in performance analysis of manufacturing cells represented by timed Petri nets. The approach is applicable to conflict–free models (only such models have well–defined cyclic behavior) which are covered by families of simple cyclic subnets. The approach is based on invariant analysis and P–invariant–implied subnets of the model.

Although in general, the number of P–invariants can grow quickly with the size of the model, the number of invariants can be often reduced by simple performance–preserving transformations of the

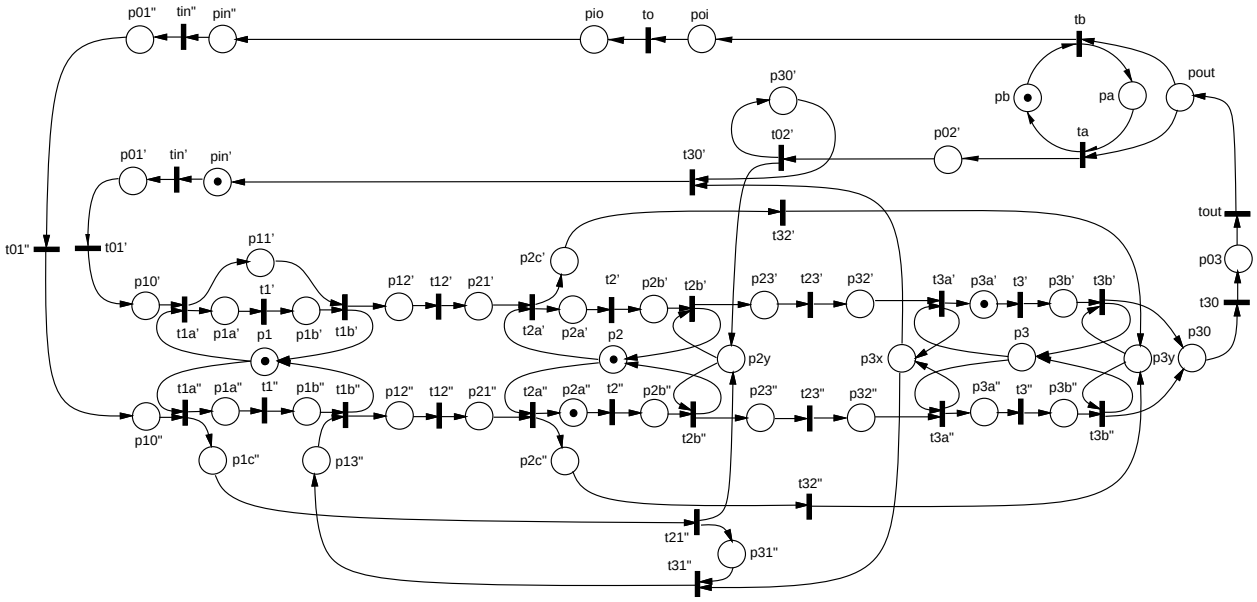

Fig.9. Petri net model of schedule 1+2.

net model, resulting in a linear relationship between the number of P–invariants and the size of the model; several such transformations are discussed in the literature [4, 13]. Moreover, P–invariants of complex models can be derived from the invariants of components, as shown in [16].

Acknowledgement

The Natural Sciences and Engineering Research Council of Canada partially supported this research through grant RGPIN-8222.

References

[1] Bause, F., Kritzinger, P.S., "Stochastic Petri nets – an introduction to the theory" (Academic Studies in Computer Science); Vieweg Publ. 1996.

[2] Desrochers, A.A., Al-Jaar, R.Y., 11Applications of Petri nets in manufacturing systems"; IEEE Pres 1995.

[3] Dixon, C., Hill, S.D., "Work–cell cycle–time analysis in a flexible manufacturing system"; Proc. Pacific Conf. on Manufacturing, Sydney–Melbourne, Australia, vol.1, pp.182-189, 1990.

[4] Garcia-Valles, F., Colom, J.M., "Implicit places in net systems"; Proc. 8-th Int. Workshop on Petri Nets and Performance Models (PNPM'99), Zaragoza, Spain, pp.104-113, 1999.

[5] Krueckeberg, F., Jaxy, M., "Mathematical methods for calculating invariants in Petri nets"; in *Advances in Petri Nets 1987* (Lecture Notes in Computer Science 266), pp.104–131, Springer–Verlag 1987.

[6] Martinez, J., Silva, M., "Simple and fast algorithm to obtain all invariants of a generalized Petri net"; in *Applications and Theory of Petri Nets* (Informatik Fachberichte 52); pp.301–310, Springer–Verlag 1982.

[7] Merlin, P.M., Farber, D.J., "Recoverability of communication protocols – implications of a theoretical study"; *IEEE Trans. on Communications*, vol.24, no.9, pp.1036–1049, 1976.

[8] Murata, T., "Petri nets: properties, analysis and applications"; Proceedings of IEEE, vol.77, no.4, pp.541-580, 1989.

[9] Ramamoorthy, C.V., Ho, G.S., "Performance evaluation of asynchronous concurrent systems using Petri nets"; IEEE Trans. on Software Engineering, vol.6, no.5, pp.440-449, 1980.

[10] Reisig, W., "Petri nets – an introduction" (EATCS Monographs on Theoretical Computer Science 4); Springer-Verlag 1985.

[11] Sifakis, J., "Use of Petri nets for performance evaluation"; in: "Measuring, modeling and evaluating computer systems", pp.75-93, North-Holland 1977.

[12] Wang, J., "Timed Petri nets"; Kluwer Academic Publ. 1998.

[13] Zuberek, W.M., Kubiak, W., "Timed Petri nets in modeling and analysis of simple schedules for manufacturing cells"; Computers and Mathematics with Applications, vol.37, no.12-12, pp.191–206, 1999.

[14] Zuberek, W.M., "Timed Petri nets – definitions, properties and applications"; Microelectronics and Reliability (Special Issue on Petri Nets and Related Graph Models), vol.31, no.4, pp.627-644, 1991.

[15] Zuberek, W.M., "Composite schedules of manufacturing cells and their timed Petri net models"; Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'96), Beijing, China, pp.2990-2995, 1996.

[16] Zuberek, W.M., "Stepwise refinements of net models and their place invariants"; Proc. 8-th Int. Workshop on Petri Nets and Performance Models (PNPM'99), Zaragoza, Spain, pp.92-101, 1999.