PERFORMANCE MODELS BASED ON PETRI NETS

Manuel Silva and Javier Campos

Departamento de Ingeniería Eléctrica e Informática*
Centro Politécnico Superior, Universidad de Zaragoza
María de Luna, 3, 50015 Zaragoza, Spain

Petri Nets (PNs) is a well suited formalism to model and analyze parallel and distributed discrete event dynamic systems. Provided with adequate timed interpretation, PNs allow to build performance models in which true concurrence can be described together with synchronization and sharing phenomena.
This survey paper gives a fast overview of some major topics related to net performance models, obtained through stochastically timed interpretation of PNs. Analysis of net models, pointing out the importance of the logical and performance perspectives, and some relations between queueing networks and PN models are briefly considered.

1. INTRODUCTION

A Petri net (PN), like a differential equation, is a mathematical formalism. PNs find their basics in a few simple objects, relations and rules, yet can represent very complex behaviours. More precisely, PNs can be considered as a graph theoretic tool specially suited to model and analyze *discrete event dynamic systems* (DEDS) which exhibit *parallel* evolutions and whose behaviours are characterized by *synchronization* and *sharing* phenomena. They have been used in the modelling of a wide range of fields. Examples of these are communication networks, computer systems [1] and discrete part manufacturing systems [2].

Logical (*qualitative*) analysis deals with properties like deadlock-freeness or the absence of (store) overflows. Its ultimate goal is the proof of the *correctness* of the modelled system. *Performance* (*quantitative*) analysis is concerned with measures like throughput or utilization rates, that is, with the evaluation of the *efficiency* of the modelled system. The complexity of the behaviours that can be described using PNs models makes essential that efficient performance analysis techniques are based on good logical properties.

PNs are built using two kinds of objects: *places* (circles), that substantiate *(distributed) state variables* and *transitions* (bars, boxes), that substantiate *(local) state transformers*. A weighted flow relation allows to easily represent *bulk services* and *arrivals*. In the sequel, basic concepts and notations on PNs are assumed to be known (see, for example, [3,4] as two recent survey works). Based on this, section 2 addresses the performance interpretation of PNs, while section 3 briefly mentions some of the basic modelling features of PN models. Sections 4 and 5 are devoted to classify analysis techniques and to mention some connections between PN-performance models and certain queueing network (QN) models.

* The work of the authors in this field is presently supported by the European ESPRIT-BRA Project 7269 QMIPS, the Spanish CICYT ROB91-0949, and the Aragonese CONAI-DGA P-IT-6/91.

2. PERFORMANCE INTERPRETATIONS OF NET MODELS

In the original definition, PNs did not include the notion of time and tried to model only the logical behaviour of systems by describing the causal relations existing among events. The introduction of a *timing specification* is essential if we want to get a more realistic model and, in particular, if we want to use this class of models for an evaluation of the performance of distributed systems (see, for example, [1,5,6,7]).

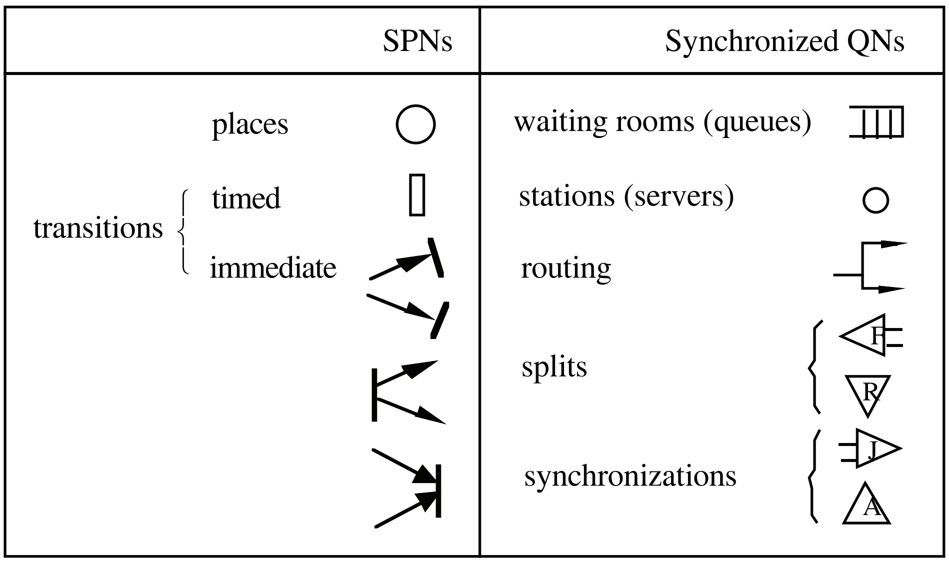
Since PNs are bipartite graphs, historically there have been two ways of introducing the concept of time, namely, associating a time interpretation (*deterministic* or *stochastic*) with either transitions or places. Since transitions represent activities that change the state (marking) of the system, it seems natural to associate a duration with these activities (transitions). This has been the preferred choice in the literature [1].

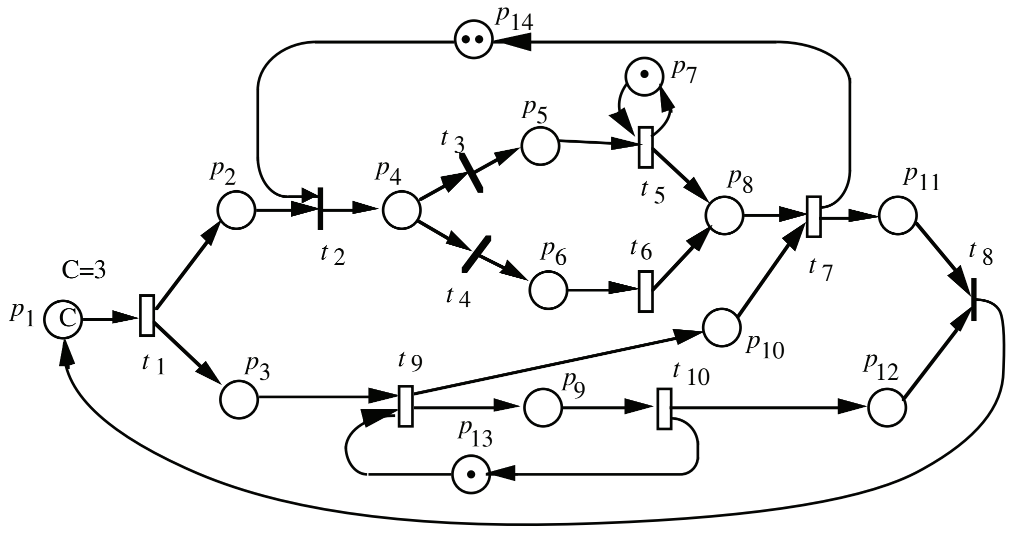
In case of stochastic timing, the interpretation requires the specification of the *probability distribution function of the random firing delays* and the *execution policy*. By execution policy it is understood "the way in which the next transition to fire is chosen, and how the model keeps track of its past history" [8]. Historically, the first stochastic interpretation associated *exponentially distributed* random variables to the transition firing delays, while the usual execution policy was very simple: *race* (transitions compete for firing; the competition is won by the transition that samples the shortest delay). This basic model was completed in Generalized Stochastic Petri Nets, GSPN [1], adding *immediate* transitions (which fire in zero time with priority over timed transitions) and *inhibitor* arcs. Weights are associated with immediate transitions for the computation of firing probabilities in case of conflict.

A step further has been trying to increase the modelling power of GSPN and alternative models, allowing arbitrary distribution functions for the random variables representing the firing delays. Under this circumstance the precise definition of the execution policy becomes crucial because the memoryless property of exponential distribution is lost (hence the model can keep track of its past history). In [8] the topic is considered in detail, defining some possible execution policies and their modelling and analysis consequences.

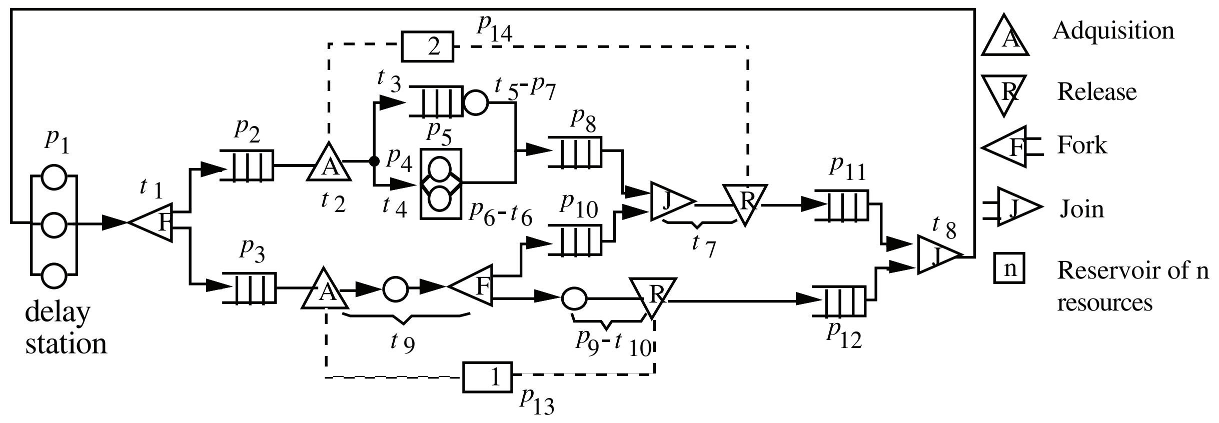
Another possible source of confusion in the definition of the timed interpretation of a PN model is the concept of *degree of enabling* of a transition (or re-entrance) [9]. If *delay server semantics* is assumed, whenever there exist enough tokens in the input places of a transition, several concurrent firings of it can be performed. If this is not adequate to the system because the number of servers should be constrained to k, then a place self-loop around the transition with k tokens will guarantee the k-server semantics. In particular, with *single-server semantics*, the number of concurrent enablings at each transition is limited to 1 (i.e., there is no self-concurrence).

Once a timing interpretation is assumed for a PN model, the usual performance indices can be defined in net terms. For instance, "mean number of firings of a transition per time unit" (throughput) is a *productivity* measure, "residence time of a token at a place" (waiting time) is a *responsiveness* measure, while "mean marking of a place" (queue length) and "probability of a transition for being enabled" (utilization) are *utilization* measures.





(a) Stochastic Petri net system representation.



(b) Extended queueing network representation.

Figure 1: A stochastic PN and the corresponding synchronized queueing network.

Figure 1 illustrates a stochastic PN and how it could be mapped onto a QN extended with synchronization primitives [10]. In this correspondence, places represent waiting rooms, timed transitions represent stations (servers), routing is modelled by means of immediate transitions with branching probabilities, and transitions can be seen as a simple and clean general synchronization primitive.

3. SOME BASIC MODELLING FEATURES

The first modelling feature to point out is that PN systems are defined by a structure (the *net*) plus a (distributed) initial state (the *marking*). This separation makes possible the existence of some powerful analysis techniques obtaining conclusions for any marking or just on the existence of markings leading to certain behaviours. The structure/state separation is a basic property of net models on which the *structure analysis theory* (i.e., the branch of net theory devoted to investigate the relationship between the structure and the behaviour of net system models) is founded (see, for example, [11, 12, 13]). Structure theory of net models leads to *graph-based* and *convex geometry-based* efficient analysis algorithms for restricted net subclasses in which choices (conflicts) and concurrence interfere in a somehow "controlled" way.

The separation of net structure and distributed state in net systems leads also to the following features of PN models:

(1) Both the net structure and the state can be easily described in *graphical* terms, being this a powerful communication support ("a picture is worth a thousand words"). Nevertheless, big and not well structured net models are difficult to understand and analyze. This means in practice that good *modelling disciplines* are very important.

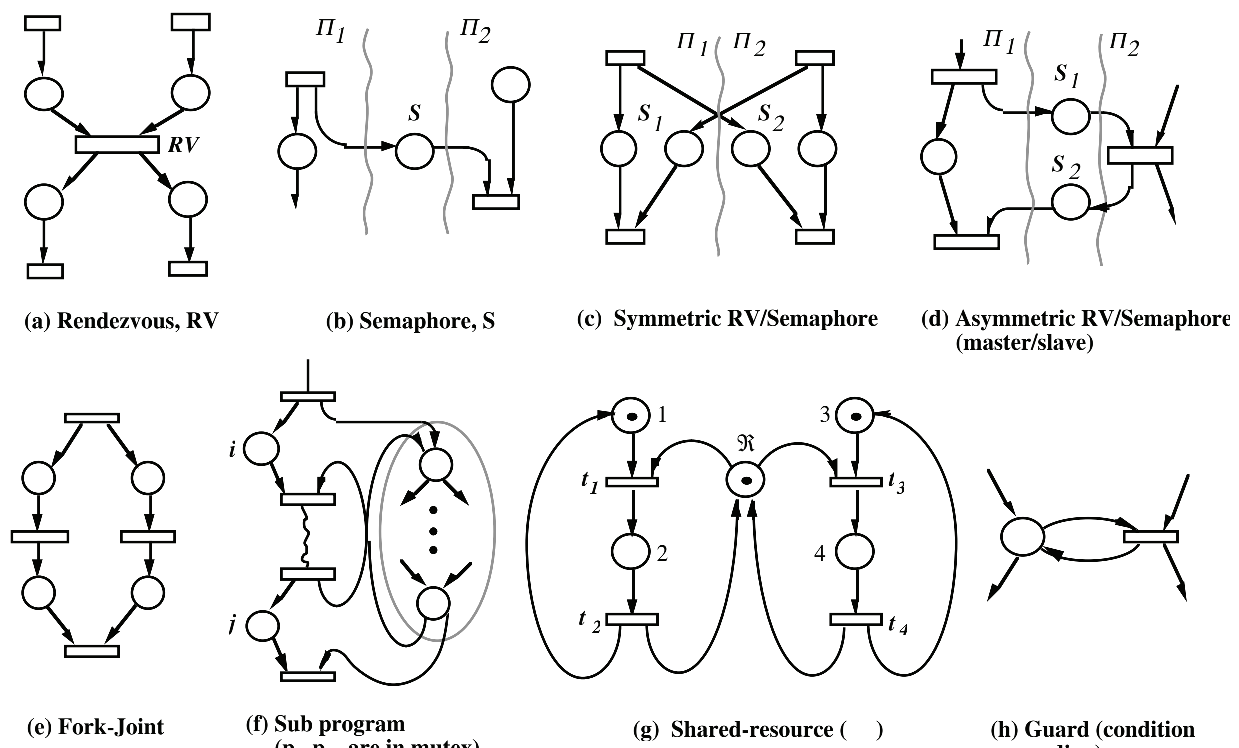

RV
Π1
Π2
S
S1
S2
1
2
3
4
ℜ
t1
t2
t3
t4
i
j
(a) Rendezvous, RV
(b) Semaphore, S
(c) Symmetric RV/Semaphore
(d) Asymmetric RV/Semaphore (master/slave)
(e) Fork-Joint
(f) Sub program (pi ,pj are in mutex)
(g) Shared-resource ( )
(h) Guard (condition reading)

Figure 2: Some typical synchronization schemes [4].

(2) The system can be easily *parametrized* through the initial marking (for high level nets [7] the structure can also be parametrized).

(3) *Sequence*, *choice* and *true concurrence* are well captured.

(4) A simple, appealing, and powerful *synchronization* mechanism facilitates the introduction of mutual exclusion and fork-join constraints. Figure 2 shows net models of a few typical synchronization schemes.

(5) The marking representing a distributed state, *hierarchical* and *modular* construction of large models is quite natural.

All the above features correspond to the uninterpreted net models, and all interpreted models inherit them.

A basic element to model flow-lines is described in Fig. 3 [14]: a producer (*M1*)–consumer (*M2*) system composed by two machines, a buffer storage, and a device working on the storage to place parts produced by *M1* and to pick parts to feed *M2*. The pick and the place operations, performed by a robot, are assumed to the be in mutual exclusion (place s_r represents the rest state for the robot). This simple example illustrates sequence, conflict, concurrence, synchronization, and mutual exclusion.

4. STOCHASTIC PETRI NETS AND ANALYSIS TECHNIQUES

An outstanding aspect of PN models is their ability to be analyzed. Both *qualitative* or logical and *quantitative* or performance-oriented properties can be defined in net terms and different analysis techniques exist for their *validation* or *computation*.

Concerning the logical validation of the model, *safety* and *liveness* properties (in temporal logic sense) can be studied. Safety properties express that “some bad thing” never happens in the system (e.g., boundedness of a place, deadlock freeness, synchronic properties [15]), while liveness properties express that “some good thing” will eventually happen in the system (e.g. transition

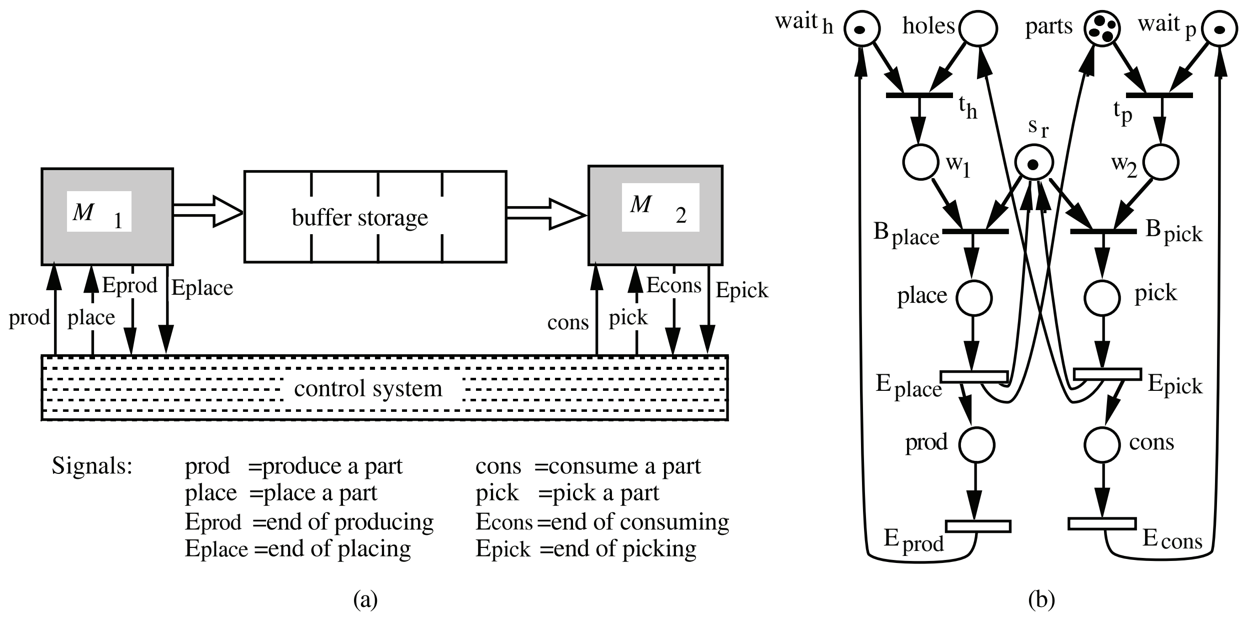


Figure 3: A producer-consumer system.

liveness, home state, reversibility). For the analysis of such properties the standard evaluation (checking) techniques can be applied. *Simulation* of the system does not provide, in general, a proof of verification of a property, but can help in certain analysis.

Reachability analysis [3], based on the construction of the state space of the model (if bounded), provides a complete knowledge of all its properties. However, the exponential temporal and spatial computational complexity due to the *state explosion* reduces the applicability of this enumeration technique.

In order to avoid the state explosion, *transformation* (in most cases, *reduction*) techniques have been developed [16,17]. They are based on the application of local rules for the simplification of net systems, preserving the properties under analysis.

Finally, *structural* analysis techniques [3,4,11,12,13] allow to (semi)decide some properties of the model just from the net structure and using mathematical tools taken from graph theory, linear algebra, convex geometry, or linear programming.

Concerning quantitative or performance-oriented analysis of stochastic PN models, the objective is to compute the selected performance measures among those defined in Section 2 (throughput of transitions, mean marking of places…). The quantitative analysis techniques admit two independent classifications. One of them, similar to that presented in previous paragraphs for qualitative analysis, distinguishes between *enumerative* (or reachability-based) [1], *transformation* [18,19], and *structural* [9,20]. Another classification criterion, derived from the quantitative nature of performance measures, allows to differentiate between *exact*, *approximate*, and *bound* computation.

Discrete event simulation can be used to approximate the performance indices. Its major advantage is its generality and flexibility; almost any behaviour can be easily simulated. However, there are many important issues that must be considered: the level of detail, a careful experiment design, and the data analysis using statistical techniques. The significance of numerical data is in general an open problem (what does mean data of non-ergodic systems?…).

Exact performance results can be obtained, under certain timing assumptions, from the numerical solution of an underlying (embedded) *Markov chain* [1] (enumerative analysis). In some particular cases, exact analysis can be done at the structural level due to a *product-form solution* [21] (similar to classical results in queueing theory).

In most cases, only approximate analysis (by means of *flow equivalent aggregation* [22] of some subnets or by *iterative aggregation* technique based on *response time approximation* of subnets [23]) or *bounds* computation [9,24] is possible, due to the generality of the model and the explosion of the state space.

5. SOME CONCLUDING REMARKS: QNs AND SPNs

There exist many different classes of QN models. All of them are founded on the *client-server paradigm*: The input/output relation is represented through the concept of *customers* visiting different *service stations*. Two well-known QN classes of models are Gordon-Newell QNs (GNQNs [1]) and fork-join QNs with blocking (FJQNs/B [25]). Both are built on an underlying graph, provided with OR/OR interpretation (for GNQNs) or with AND/AND interpretation (for FJQN/B).

PNs are bipartite graphs, where places have OR/OR semantics and transitions have AND/AND semantics. Thus the following can be informally stated:

$$\text{SPNs} \approx \text{GNQNs} + \text{Forks} + \text{Joins}$$

$$\text{SPNs} \approx \text{FJQNs/B} + \text{Choices} + \text{Attributions}$$

Thus SPNs allow the modelling of stochastic systems with choices and synchronizations. SPNs "generalize" QNs identifying *places* as *queues* and *transitions* as *stations*. (Bulk) services [resp., arrivals] are modelled with (weighted) arcs from places to transitions [resp., transitions to places]. The number of servers at a given station (firing re-entrance of a transition) can be formalized in pure net terms: t-enabling bound concept (see, for example, [9,24]). Tokens in a place can be seen, for example, as customers in a queue or as servers in a station. Fork and join primitives change usually the "identity" of customers in the systems (splitting one unit into parts, getting a compound customer from parts).

From the above perspective, SPNs deal also with the client-server paradigm. Moreover, with PNs we can merge client-server views of a system together with a more refined description in which complex synchronization schemas make explicit the underlying behaviour of certain customers and/or servers. In other words, SPNs allow the *interleaving of views with different levels of detail* on a single model. The total number of "customers" of a bounded PN is usually easy to identify through some P-invariants (a token conservation law) [3,17] of the net model.

Customers classes is also a well-known concept in QNs framework (for instance, it is used in BCMP networks [1]). At a first glance we may have the idea that classes of customers are in direct relationship with *colours* in coloured Petri nets (CPNs [7]). Nevertheless, CPNs can be unfolded into uncoloured PNs with equivalent behaviour. This means that customers classes already appear in uncoloured models. The reader can easily check that the net system in Figure 3 "has two classes of customers": the class *M1* and the class *M2*. In any case, recent important progress in the computation of performance indices of a broad subclass of stochastic CPNs has been achieved [7].

Well developed features in the context of QNs interpretations are *queueing* and *service disciplines*. In this respect, classical service disciplines are FIFO, LIFO, and processor sharing. For SPNs, the assumed discipline is *random choice*, unless otherwise stated.

The difficulty (impossibility) of computing exact performance figures for certain SPNs makes approximation techniques and the computation of performance bounds appealing and promising for a better understanding of the performance behaviour of models with fork-join-choices-attributions characteristics. The most promising performance analysis techniques typically use deep knowledge of the behaviour of the underlying PN model.

As a final remark, using PNs we can go through the different phases of a design project using models belonging to a *unique family*, what makes easier and more reliable the full process.

REFERENCES

[1] Ajmone Marsan, M., Balbo, G., and Conte, G., Performance Models of Multiprocessor Systems (MIT Press, Cambridge, 1986).

[2] Silva, M. and Valette, R., Petri Nets and Flexible Manufacturing, in: Rozenberg, G. (ed.), Advances in PNs'89, vol. 424 of LNCS (Springer-Verlag, Berlin, 1990) pp. 374–417.

[3] Murata, T., Petri Nets: Properties, Analysis, and Application, Proc. of the IEEE 77 (1989) pp. 541–580.

[4] Silva, M., Introducing Petri Nets, Technical Report GISI-RR-91-45, Dpto. Ingeniería Eléctrica e Informática, Univ. Zaragoza, Spain (1991), to appear as Chapter 1 of Practice of Petri Nets in Manufacturing (Chapman).

[5] Proceedings of the Third International Workshop on Petri Nets and Performance Models, Kyoto, Japan, December 1989 (IEEE-CS Press).

[6] Proceedings of the Fourth International Workshop on Petri Nets and Performance Models, Melbourne, Australia, December 1991 (IEEE-CS Press).

[7] Jensen, K. and Rozenberg, G. (eds.), High-Level PNs: Theory and Application (Springer-Verlag, Berlin, 1991).

[8] Ajmone Marsan, M., Balbo, G., Bobbio, A., Chiola, G., Conte, G., and Cumani, A., The Effect of Execution Policies on the Semantics and Analysis of Stochastic PNs, IEEE Trans. Soft. Eng. 15-7 (1989) pp. 832–846.

[9] Campos, J. and Silva, M., Structural Techniques and Performance Bounds of Stochastic Petri Net Models, in: Rozenberg, G. (ed.), Advances in PNs'92, vol. 609 of LNCS (Springer-Verlag, Berlin, 1992) pp. 352–391.

[10] Sauer, C.H., MacNair, E.A., and Kurose, J.F., The Research Queueing Package: Past, Present, and Future, in: Proc. of the National Computer Conference, AFIPS (1982).

[11] Best, E., Structure Theory of Petri Nets: The Free Choice Hiatus, in: Rozenberg, G. (ed.), Advances in PNs'86-Part I, vol. 254 of LNCS (Springer-Verlag, Berlin, 1987) pp. 168–205.

[12] Esparza, J. and Silva, M., On the Analysis and Synthesis of Free Choice Systems, in: Rozenberg, G. (ed.), Advances in PNs'90, vol. 483 of LNCS (Springer-Verlag, Berlin, 1991) pp. 243–286.

[13] Teruel, E., Chrząstowski-Wachtel, P., Colom, J.M., and Silva, M., On Weighted T–Systems, in: Jensen, K. (ed.), Application and Theory of PNs 1992, vol. 616 of LNCS (Springer-Verlag, Berlin, 1992) pp. 348–367.

[14] Campos, J., Colom, J.M., and Silva, M., Performance Evaluation of Repetitive Automated Manufacturing Systems, in: Proc. of the Rensselaer's Second Int. Conf. on CIM, Troy, NY, May 1990 (IEEE-CS Press) pp. 74–81.

[15] Silva, M., Towards a Synchrony Theory for P/T Nets, in: Voss, K., Genrich, H., and Rozenberg, G. (eds.), Concurrency and Nets (Springer-Verlag, Berlin, 1987) pp. 435–460.

[16] Berthelot, G., Transformations and Decomposition of Nets, in: Brauer, W, Reisig, W., and Rozenberg, G. (eds.), Advances in PNs'86, vol. 254 of LNCS (Springer-Verlag, Berlin, 1987) pp. 359–376.

[17] Silva, M., Las Redes de Petri en la Automática y la Informática (Editorial AC, Madrid, 1985).

[18] Zuberek, W.M. and Zuberek, M.S., Transformations of Timed Petri Nets and Performance Analysis, in: Proc. of the Midwest Simp. on Circuits and Systems'90 (Special Session on Petri Net Models) 1990, pp. 1–5.

[19] Campos, J., Sánchez, B., and Silva, M., Throughput Lower Bounds for Markovian Petri Nets: Transformation Techniques, in: Proc. of the Fourth Int. Workshop on Petri Nets and Performance Models, Melbourne, Australia, December 1991 (IEEE-CS Press) pp. 322–331.

[20] Campos, J., Colom, J.M., Jungnitz, H., and Silva, M., A General Iterative Technique for Approximate Throughput Computation of Stochastic Marked Graphs, Technical Report GISI-RR-93-6, Dpto. Ingeniería Eléctrica e Informática, Univ. Zaragoza, Spain (1993).

[21] Donatelli, S. and Sereno, M., On the Product Form Solution for Stochastic Petri Nets, in: Jensen, K. (ed.), Application and Theory of PNs 1992, vol. 616 of LNCS (Springer-Verlag, Berlin, 1992) pp. 154–172.

[22] Jungnitz, H. and Desrochers, A.A., Flow Equivalent Nets for the Performance Analysis of Flexible Manufacturing Systems, in: Proc. of the IEEE Robotics and Automation Conference 1991, pp. 122–127.

[23] Jungnitz, H., Sánchez, B., and Silva, M., Approximate Throughput Computation of Stochastic Marked Graphs, Journal of Parallel and Distributed Computing 15 (1992) pp. 282–295.

[24] Campos, J., Chiola, G., and Silva, M., Properties and Performance Bounds for Closed Free Choice Synchronized Monoclass Queueing Networks, IEEE Trans. Autom. Cont. 36-12 (1991) pp. 1368–1382.

[25] Dallery, Y., Liu, Z., and Towsley, D., Equivalence, Reversibility, and Symmetry Properties in Fork/Join Queueing Networks with Blocking, Technical Report MASI 90-32, Univ. Paris 6, France (1990).