Petri Nets for the Design and Operation of Manufacturing Systems *

Manuel Silva and Enrique Teruel
Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior, Universidad de Zaragoza
María de Luna 3, E-50015 Zaragoza, Spain
E-mail: {silva,eteruel}@posta.unizar.es

Abstract

Modern manufacturing systems pose a diversity of problems all along their life cycle which are often treated with particular independent formalisms and techniques. Petri nets are a family of formalisms which can be used, sometimes with advantage, for many of these problems, with the additional benefit of improving the communication between stages of the life cycle.

The utilization of Petri nets in several of these stages is illustrated and surveyed in this paper, mainly addressed to manufacturing systems engineers with a basic knowledge on Petri nets.

1 Introduction

A wide range of *manufacturing systems* (MS) can be viewed as concurrent asynchronous *discrete event dynamic systems* (DEDS). Their behavior is often extremely complex and subtle or even paradoxical phenomena appear. Formal methods usually improve the understanding of systems, allow their analysis (validation, verification wrt. a specification, performance, performability, etc.), and help in the implementation. Moreover, some formal methods facilitate the dialog between the different people involved in the design and operation, specially when graphical/intuitive representations are provided.

Among the many diverse problems concerning MS we concentrate here in:

- The *design* phase, where a new plant is conceived or extensive changes are accomplished. This starts at the preliminary conception of the plant, when both its basic structure and dimension are determined, and then comes into more and more details until the implementation level is reached.
- The *operation*, concerning aspects such as allocation of resources, scheduling, control, supervision, etc.

The range of problems and solutions encountered is extremely broad. As an example, in [53], where the performance evaluation (a kind of analysis) of flow lines (a kind of MS) is surveyed, the authors devote a section to review the reviews!

We assume the reader is familiar with the MS domain. Some books on the topic from a *systems theory* perspective, are [61], [194], [134], [82]. There are several different kinds of MS (production lines, assembly/disassembly systems, flow-shops, job-shops, flexible manufacturing systems (FMS), etc.) depending mainly on the volume, diversity, and life cycle of products (e.g., production lines are adequate for high volumes of identical or very similar products — mass manufacturing, economy of *scale* — while in job-shops or FMS versatile machines and complex transportation and storage systems allow the production of small lots of a diversity of rapidly changing products — economy of *scope*). Within each type a wide range of problems are found (e.g., balancing, sequencing, part selection, machine grouping, resource allocation, dispatching of parts, reaction to disruption, etc.).

*This work was supported in part by Project TIC-94-0242 from the Spanish CICYT, and Contract CHRX-CT94-0452 (MATCH) within the HCM Programme of the EU.

Hierarchy is a usual way to cope with complexity, where inter-related views of a system with different degrees of abstraction are used to cope with different problems, depending on *time horizons* and/or *locations*. In this sense, it is conventional to have a hierarchical structure in the control of MS (see for instance [81], [164]). In the lowest (local) level programmable logic controllers (PLC) direct the sequencing of operations performed by machines, robots, transportation and storage systems, etc. The execution of the tasks by the local controllers is managed and supervised at coordination(s) level(s). Some decisions, e.g., which of two conflicting tasks will be carried out, when a task will be started, etc., are taken at higher levels (scheduling, planning, etc.) on request of the coordinator.

A time horizon driven usual classification of control problems distinguishes *planning* (also in different levels, from long-term decisions, where the whole plant is considered with an estimated demand, to medium-term, where the availability of raw material and due dates are taken into account); *scheduling* (the date — or just the order — and machine of each operation on each part is considered individually); and *real-time control* of the individual devices and subsystems in order to perform the scheduled operations.

The diversity, specificity, and difficulty of the problems leads to the development of particular formalisms and techniques for each problem or class of problems, e.g.: Markov chains, queueing networks, or discrete simulation for performance evaluation; mathematical programming, or MRP for planning; PERT/CPM, or artificial intelligence techniques for scheduling; relay ladder logic diagrams, state diagrams, or algorithmic state machines for local controllers implementation, etc. Of course each kind of problems requires an adequate solution but this may appear as a Babel Tower where the different people at each stage of the design and operation of the MS can seldom communicate, at least formally.

The suggested solution is to use a *family* of formalisms sharing basic principles in a consistent way. Although for each purpose or degree of detail the adequate formalism would be chosen from the family, the transformation from one formalism to another could be sound, if not formal or even automatic. Desirable characteristics for such a family of formalisms include:

- *Generality*, or descriptive power, both wrt. MS types and problems.
- *Adequacy* for dealing with real systems. There is an inevitable modeling tradeoff between *fidelity* to reality and *tractability*. If one wants to perform some analysis to gain insight into the basic structure and relevant parameters of a system, their influences, the cause of problems, etc. this will sometimes be paid for by the necessity of making strong assumptions.
- *Ease of use*. (This is naturally a matter of taste to some extent.)

Two kinds of DEDS formalisms are *denotational* and *operational* [115]. Denotational formalisms state *what* the system is meant to do, its functionalities. They are sometimes too abstract for some purposes such as plant control implementation. Operational formalisms are somehow closer to the habitual manufacturing engineer's culture. They state *how* the system is meant to work, usually in terms of states and state changes, and are *executable* to some extent, what can be used for simulation, prototyping, or code generation.

Petri nets (PN) are a family of operational formalisms which provides a framework or working paradigm for MS design and operation. The application of PN to MS is an active research field where many questions still remain open. The goal of this survey is to illustrate the usability of PN models along the life cycle of MS. By no means we intend to be exhaustive, but just quickly overview the contribution of PN to some classes of problems. In this sense the paper is meant to be a kind of *impressionist picture*, where main strokes are more important than details (for which the reader is referred to the literature).

The adequacy of PN along the life cycle is discussed in Section 2, making special emphasis on the abstraction levels and different interpretations, and on the construction and validation of models. In Section 3 the use of PN for the modeling and implementation of the local and coordination levels of the control is illustrated. In Section 4 the use of PN for the evaluation and optimization of the plant performance from the early stages of the design is discussed. Finally, Section 5 deals with the modeling and diverse approaches to the solution of scheduling problems using PN.

2 The Formal Framework Provided by Petri Nets

We assume the reader is familiar with basic PN concepts. Some textbooks on PN are [136], [23] covering basic modeling and analysis issues, [157] with emphasis on control implementation, [100] on colored PN, and [3] devoted to performance evaluation. Books on or containing chapters about PN in (diverse aspects of) MS are [194] on performance modeling, [55] on PN and Grafcet, [66] on modeling, validation, and performance evaluation, [201] on control modeling, [62] on performance evaluation and control implementation, and [139] on modeling and analysis. Some surveys are [125] on basic PN concepts and techniques, [164] on basic concepts of MS and the applicability of PN, or [57] concentrating on modeling issues. Many invited sessions in conferences, special issues of journals, or collections have been devoted to PN and MS, such as [162], [77], [61], [104], [64], [203], [200], [22], [165].

A *system* can be viewed as a collection of *objects* or entities with some attached attributes, some of them *fixed* and some of them *variable* (that define the *state*), and the relations between them. In PN the state of a DEDS is represented in a *distributed* fashion: places are local state variables whose value, depending on the chosen abstraction level, ranges from boolean (which is adequate for local control modeling, for instance), to typed and structured (which may be adequate for concise system descriptions). The possible *state changes* are locally defined by way of transitions, their connection to places expressed by arcs, and the firing or occurrence rule which can be enriched with suitable interpretations, e.g., associating external signals or timing to the firing of transitions.

Appealing characteristics of PN include:

- Ability to represent in a natural way concurrency, causality, synchronization, resource sharing, conflicts, bulk or lot movements, etc.
- Locality of states and actions, allowing both top-down and bottom-up model construction, i.e., refinement, modularity, reusability, etc.
- Compactness due to the distributed state representation compared to a centralized sequential representation.
- Adequacy to represent the essential features of a given system by way of the selection of an appropriate abstraction level.
- Interpretability. It is possible to associate a wide range of different meanings and/or connections to the external world to the model objects; different interpretations are appropriate for the different purposes of the model: validation, performance evaluation, scheduling, control implementation, etc.
- Graphic representation facilitating their use in the documentation and monitoring of the system.
- Formal/precise semantics allowing to undertake rigorous analysis or to automate the implementation or code generation either for control or simulation.

2.1 Petri Nets and the Life Cycle

The different interrelated *abstraction levels* of PN models, and the different *interpretations* that can be associated to a PN model, make them specially usable along the life cycle (design, preliminary and detailed, implementation, and operation). Both define a space of formalisms adequate for different purposes (see Figure 1). Broadly speaking, in the abstraction levels axis we have *elementary net systems* where local states are boolean (conditions), *place/transition net systems* where local state variables are counters and bulk processing is allowed, and diverse *high level* formalisms where tokens are distinguishable typed items (e.g., they have attached attributes) and transition enabling in a given mode requires specific tokens present in the input places. In addition one should consider in this axis some *extensions* such as *inhibitor arcs* or *priorities* which affect not only the conciseness but also the expressive power under some circumstances, or some *subclasses* which aim at improving the tractability at the price of losing modeling power. In the interpretation axis we have the *non-interpreted* or *autonomous* or basic PN model, different *timed interpretations* (stochastic, deterministic, firing times specified by intervals or fuzzy sets, etc., each one with several possible corresponding modifications of the firing rule) which are adequate for performance modeling, scheduling, etc., and addition of *external input and output signals* expressing the connection to the plant which is adequate for real-time control, coordination, diagnosis, etc. These interpretations can be seen as different abstractions of the environment of the model, ranging from total abstraction in the autonomous model (which is totally non-deterministic in the sense that only logical pre-requisites for firings are given, without information on what/when to fire) to total explicitation in

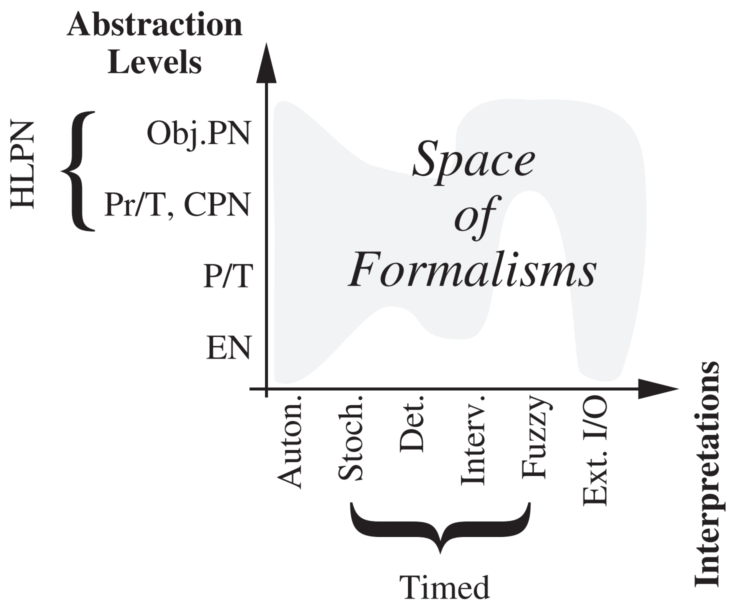

Figure 1: A pictorial view of the space of PN formalisms.

the models considering the actual signals (which should be totally deterministic). In between diverse assumptions are made (e.g., describing the time of occurrence of an enabled transition as a random variable or constraining it to be in some interval, associating probabilities or fairness constraints to conflicting transitions, etc.).

These formalisms can be related to other non PN formalisms which are used in common application domains. For instance, PN performance models add synchronizations to classical queueing networks, PN scheduling models add resource constraints and cyclic behaviors to PERTs, PN controllers allow to deal with concurrent systems for which state diagrams are impractical, PN in artificial intelligence, compared to classical rule systems, add the ability to handle resources, etc.

PN models — selecting the appropriate formalism — have been used for diverse purposes within the field of MS, including:

- *Documentation* of a design, to be used in the communication between the different people involved (in the desing and also in the subsequent operation).
- *Verification* wrt. a specification and *validation* of some required logical properties. This analysis can be done by enumeration and model checking techniques. The state space explosion problem justifies the interest of transformation and structural techniques; in some cases relevant information about the behavior can be obtained reasoning solely on the net structure and initial marking. This will be further discussed in Subsection 2.2.
- *Control implementation.* PN models can be regarded as executable specifications of logic controllers. Both the modeling of the control and diverse techniques for its programmed implementation, including fault-tolerance issues, will be discussed in Section 3.
- *Performance and performability analysis.* This analysis can be done by simulation or analytically when suitable assumptions on the timing are made, e.g., exponential or phase-type distributions. A brute force approach to this analysis is generating the underlying Markov chain. Significant computational improvements can be obtained when information on the net structure is carefully considered. In some subclasses enumeration can be avoided at all and net-level analysis is done instead. This will be further discussed in Section 4.
- *Scheduling.* The PN description of the system comprises both the functional precedence and resource constraints. As we shall discuss in Section 5, diverse techniques have been applied to compute

optimal or suboptimal schedules combining information from the net, mathematical programming, heuristic search, etc. In some subclasses theoretical results are available.

- Monitoring and diagnosis, aiming at finding, as soon as possible, when a process is diverging from its intended behavior in a non random way [150], [181], [192].

The use of a single family of formalisms for such a diverse range of problems is not only beneficial from the point of view of communication and reutilization of results. It has proven to lead also to a synergy, where the concepts and techniques developed in one area help in the solution of open problems in another one [159, 160].

2.2 Modeling and Validation Issues

A (the most?) critical issue in the use of formal methods for problem solving is the construction of a "good" model, in particular a model that is *correct* with respect to some kind of logical specification.

Different concepts of correctness exist. Basically, a system is said to be correct when two models, namely the *specification* and the *implementation*, are *equivalent* [27], [137], or when the system exhibits a set of *desirable properties*, either expressed as formulas of a *logic* [47], [116], or selected from a given *kit* (e.g., boundedness, liveness, etc. See the already cited textbooks or surveys on basic PN theory and applications).

Broadly speaking, analysis methods of autonomous Petri net models can be classified as *enumeration* or *net-driven*. The first step in enumeration techniques is the computation of the *reachability graph*. If the system is bounded, this can be used as the computational model for a proof system, which verifies formulas from a given (temporal) logic, or for decision procedures and tools for automatic verification [47], [116]. The *state space explosion problem* can be alleviated by disregarding markings which are not interesting for a given analysis [185], taking advantage from symmetries [151], [100], or analyzing components of the model and combining the results [186], [130], [30], [45].

It is also possible to deal with unbounded systems by using the *coverability graph* [75], but this particular construction does not allow to decide important properties such as liveness and reachability [136].

The basic idea in net-driven approaches is to obtain useful information about the behavior reasoning on the structure of the net and the initial marking. Crucial advantages of this approach are the deep understanding of the system behavior that is gained, and the eventual efficiency of the algorithms. Two different approaches are [125], [66, Ch. 1]: (1) Net *transformations* (typically reductions), preserving the properties to be analyzed [17]. It is expected that the transformed system is easier to analyze. (2) *Structure theory* results, based on graph theory and/or linear algebra. Typically results for general PN systems are obtained some of which become specially powerful when restricted to particular subclasses (see [163] for a recent survey).

It must be noticed that the interpretations concerning timing or synchronization with external events restrict the behavior of the underlying autonomous model, so they should be taken into account for the analysis. On the one hand this may become extremely complicated in some cases because the notion of state must be enlarged, e.g., time PN [18]. On the other hand performing analysis of the autonomous system only may not be conclusive except for some particular properties and subclasses of systems. For instance, the autonomous PN in Figure 2 (a) is not bounded unless the interpretation ensures that t' fires as often as t; the autonomous PN in (b) is not live (fire t twice) unless the interpretation states that the conflict is resolved in alternating fashion; in (c), if t takes always more time than t' to fire then the system will not return to the initial marking and t'' will die, although the autonomous model is live and reversible. In general it can be said that *safety* properties of the autonomous system are preserved under any interpretation while *liveness* properties are neither necessary nor sufficient [157].

Model building is mainly a creative, thus difficult to automatize, task. Nevertheless for a given kind of systems and problems it may prove useful to somehow limit creativity, either to automatically generate the model from a (graphical high level) description of the plant layout, machines, material handling, storage and retrieval systems, etc. [191], or to facilitate the subsequent analysis.

An adequate formalism should allow for both *top-down* (refinement) and *bottom-up* (composition) design methodologies. Moreover it is important to cope with hierarchical and layered systems, where a collection of inter-related models are used to describe with different degrees of detail the same system or parts of it. This has been a major topic in PN research [27], [97], [16], [73], [66, Ch. 3].

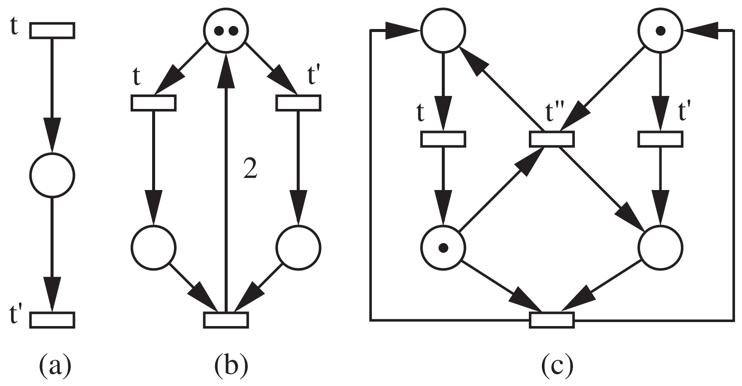

Figure 2: The interpretation affects qualitative properties.

A desirable feature of such methodologies is property preservation, i.e., the fact that the properties of the *refined* or *composed* system are guaranteed by, or can be more easily derived from, those of the coarse system or the components, respectively. To date this has been more successfully achieved for top-down methodologies ([177], [170], reversed reduction rules of [17]).

It has been a common trend in systems theory (PN theory in particular) to study restricted *subclasses* where the greater tractability is paid for by the lower expressive power. In several cases these subclasses allow for, or are meant to, systematic construction. Some well studied subclasses are marked graphs and other deterministic systems [50], [125], [66, Ch. 2 and 4], [173], free choice models [88], [71], [60], [174], macroplace/macrotransition [63], modules synchronized via rendez-vous [59], or restrictedly sharing resources [201], [72], [99], or cooperating via message passing [168], [144], [175], etc.

On the other hand in some applications within MS, e.g., coordination, monitoring, diagnosis, information systems modeling, etc., the major issue is expressive power and user-friendliness rather than tractability. In such cases high level formalisms may be preferred, integrating modern software engineering concepts both regarding methodological and data/knowledge representation aspects, like *algebraic specification* and *object-orientation* [13], [19], [67], [11], [31], [145], [66, Ch.5] [155], [188], [192].

3 Control Modeling and Implementation

3.1 Modeling the Plant and Control

Generally speaking, the control model at a given level reflects the (potential behavior of the) plant plus some constraints that must be satisfied at the degree of detail that is precised at that level. For instance, the local controller attached to the production cell depicted in Figure 3 can be described by the given PN model. The places *wait load*, *load*, op_1, *wait dep.*, and *deposit* represent the possible states of *MACH 1*; The place R is marked when the robot is available; The places *empty* and *parts* contain as many tokens as empty slots or parts are available in the temporary buffer, etc. In this model actions are associated to places, e.g., *MACH 2* performs its operations while place op_2 is marked, and transitions represent atomic instantaneous changes of state. External inputs (from plant sensors) condition these possible changes of state, e.g., a load operation is initiated (transition t_1 is fired) when *MACH 1* is waiting for a raw part (*wait load* marked), the robot is available (R marked), and a raw part is detected by the sensor Π_1 (Π_1 is true). As an example of constraint that is reflected in the model, a deposit operation cannot be initiated unless an empty slot is available. If the self-loops between *empty* and t_4 and between *parts* and t_9 were deleted the system could reach a deadlock situation, e.g., *MACH 2* is "withdrawing" a part when there are none available but *MACH 1* cannot deposit any because the robot is busy.

If the operation of *MACH 1* was directed by a lower level local controller we may have another PN model for this, which would be *started* when place op_1 becomes marked. (If the same PLC was used also to direct this operation then we would *refine* place op_1 instead.)

For the purpose of local controller modeling and implementation, place/transition nets with an state diagram like interpretation, such as the one depicted in Figure 3, are often the choice formalism since the major issue is the control flow rather than complex data representation [158], [66], [201]. This

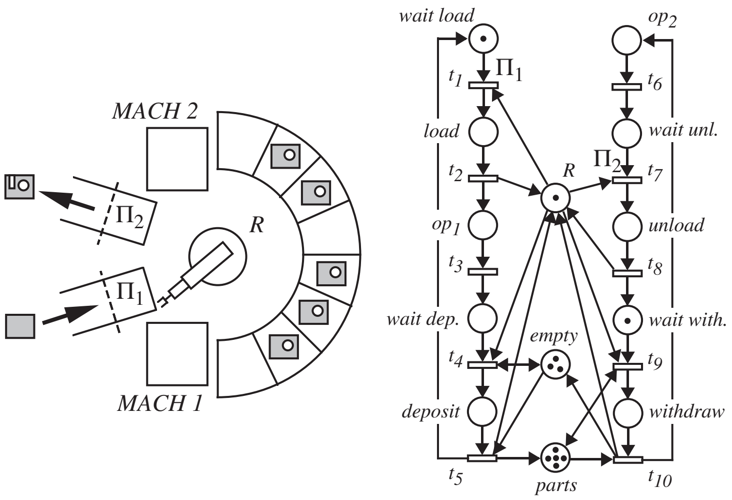

Figure 3: Layout of a production cell with two machines (*MACH 1* and *MACH 2*, a robot, R, and a buffer, and PN model of its local controller.

formalism allows for concise and natural representation of concurrency and sequencing compared to state diagrams and relay ladder logic diagrams, respectively. PN based PLCs are available [126]. *Grafcet*, an International Standard since 1987, is another tool for the specification of PLCs which is essentially a subclass of interpreted PN [55], [54]. *Marked Flow Graphs* (MFG) is another interpreted formalism derived from PN [90].

When the scope of the controller rises from local to coordination levels it becomes necessary to use an accordingly *higher level* formalism (colored PN, PN with objects, etc.), to be able to capture in a concise way complex information on the modeled system (i.e., different production processes, machines, etc.) [128], [117], [78], [79], [80], [118], [193], [191], [4], [98], [57], [192].

3.2 Programmed Implementations

Once a suitable PN model for a controller has been obtained it has to be *implemented*. Basically an implementation is a physical device, usually a programmed computer system, which emulates the behavior expressed by the model. The implementation is affected by the selected PN *formalism* (low or high level, different interpretations of the firing rule), the *algorithmic approach* (interpreted, where the PN model is a data structure, or compiled, where a program is obtained from the given PN; centralized or parallel/distributed schemas), and the *computer architecture* (high or low level programming language; single or multi processor).

For the case of local controllers specified by low level PN works on centralized implementations, either compiled or more frequently interpreted ("token players"), can be found in [51], [129], [166], [184], [157], [126], [52], [127]. A major issue for interpreted implementations is the efficient computation of enabled transitions. This topic is revisited in [49], where also colored PN are dealt with. Recently improvements have been achieved taking further advantage from the information provided by the net structure [28]. The inherent parallelism captured by a PN model is somehow dismissed in centralized implementations. Taking it into account to a large extent poses challenging questions [26]. Diverse parallel and distributed implementations have been proposed [49], [29], [172], [202].

In the implementation of higher control levels, some convergence has appeared between the fields of PN and artificial intelligence (AI) [180], [119], [183]. AI techniques allow for *automatic reasoning*, which is essential in complex decision making for instance, while PN are suitable for representing the system *dynamics*, specially concerning the resources. Going one step beyond merging techniques in the solution to a given problem, PN models that integrate modern software engineering concepts have been proposed as indicated in Subsection 2.2. It has been observed that some of these integrated PN models can be regarded as *knowledge representations*. Therefore suitable modifications of *rule-based* systems lead to possible implementations [14], [70], [179], [7], [183]. Transitions play the role of *rules* while the *working* *memory* can be split into several nodes corresponding to the respective input places. With respect to classical PN implementations the search for enabled transitions is efficiently carried out by the *matching phase* in the rule system which can take advantage from the partition into local working memories. For the selection phase transitions can be grouped into *conflict sets* by inspecting the net structure, and each one can be provided with a particular resolution strategy.

Besides the communication between the different levels, the control system communicates with the environment (plant and staff). Local controllers interact with the plant through sensors and actuators, and possibly with the staff in exceptional situations such as faults (low-level alarms, semi-manual operation, etc.). Coordination and higher levels report on their view of the plant for monitoring/statistics and accept different kinds of orders. Their graphic nature is a valuable feature of PN models for these communication/visualization/animation purposes.

3.3 Safety and Fault-tolerance

An important issue when designing a control system is that of *safety*. Formal modeling and analysis tools are needed to engineer safe computer-controlled systems. For this task it is necessary to consider both the control system and its environment, for which PN are a suitable formalism [109]. When faults can happen the controller should be able to detect them and even react appropriately degrading system's performance as little as possible.

Several techniques have been proposed to produce safe and/or *fault-tolerant* PN based controllers. Sometimes PN and coding theories are combined. The idea is to consider the marking as a *word* and the set of reachable markings as a *code*. Adding *implicit places* [156] or *test places* [167] the *Hamming distance* in the code can be increased leading to greater error detection/correction possibilities. In [15] a method to integrate fault-tolerance properties in the design combining Pr/T nets and *regular expressions* is presented.

As a totally different approach, *N-version* programming techniques [5] can be used. A less expensive schema is based on the idea of an *observer* [6] or *spy* [189], which accepts "normal" behaviors seen through some observable points. In [65] this observer concept is also employed for formal validation.

3.4 Synthesis of Feedback Controllers

PN models can be used in the so-called *supervisory control* theory, a control theory for DEDS originated by Ramadge and Wonham (see [140] for a tutorial survey). Supervisory control theory tries to use basic (continuous dynamic systems) control theory concepts and methodological aspects in the DEDS framework. Among these, the first one is a clear distinction between the *plant* (system to be controlled) and the *controller*, which can be viewed as two subsystems that are *parallelly composed*. The problem is stated as follows: given a plant (with some open-loop discrete event dynamics) and a specification of the desired final dynamic behavior (a subset of the possible behaviors of the plant) synthetize a set of constraints (controller) so that the overall system (closed-loop) meets the specification.

Following [24] two important advantages of this approach are that: (1) the obtained supervisors and control laws are correct by construction (i.e., they do not contradict the specification and are *non-blocking*), and (2) all events which do not contradict the specification can be allowed to happen (in *maximally permissive* controllers). Actually, this conceptually interesting features have often appeared in the PN literature, even out of the supervisory control framework. As some examples, (1) PN based (local) controllers can be formally validated or constructed in such a way that they enjoy desirable properties. For instance, using a top-down model building (synthesis) methodology the specifications can be preserved and/or incorporated during the synthesis process, leading to a model that is correct by construction; (2) Deadlock-avoidance problems in [12], [195], [72], [199] are stated in terms of a given model (which possibly exhibits deadlock problems) for which a control policy is to be derived to forbid reaching a deadlock (without being too restrictive).

Observability and *controllability* are two key concepts in control theory. Observability has been extensively used for logical analysis of DEDS in the context of *process algebras* (see [123]). It has been also studied in the PN framework (see [137] for a survey). In this context transitions (or places) are partitioned into observable or not, and problems like *equivalence* or *(bi)simulation* of systems are relevant to many practical purposes (e.g., model abstraction, model comparison, correctness analysis, etc.) Observable (or *check*) points in *recovery blocks* fault-tolerant techniques use also notions related to observability. The dual notion of observability is controllability, a notion that requires classifying events (transitions) into controllable or not. The integration into DEDS theory of this notion is classical in the context of *on-line scheduling*, where the schedule is computed within a state-feedback mechanism [82]. Controllable points are those at which the decision maker or scheduler influences the behavior of the system. From this controllability notion, the derivation of some *controllability criteria* is one of the major attempts of supervisory control theory [140]. Also classical systems engineering methodological concepts as modularity and hierarchy have been integrated into the control theory of DEDS [140], [197].

Most of these concepts are independent of the formalism chosen to model the plant and to describe the desired closed-loop behavior. One possible formalism to model the plant is PN, as it has been done in a number of works: [85], [87], [95], [103], [110], [111], [112] (the last three consider *vector DES*, which are place/transition net systems in PN terminology). In [96] a rather complete overview is given. The expected benefit from the use of PN comes from its clear and sound definition of the state space through a structure that reflects the locality principle and that can be exploited using (integer) linear algebraic or graph theoretic techniques. The desired closed-loop behavior is frequently described by sets of logical constraints concerning the flow of events (transition firings). In some cases the constraints are expressed in terms of undesirable markings: *forbidden state problems* [140], [103], [21]. This problem is somehow related to Patil's early attempts of increasing the modeling convenience (compactness) incorporating the so called *constraints* (i.e., a set of places that cannot be all marked simultaneously) [136]. Sometimes the controller can be described with the same PN formalism leading to a compiled implementation [86]. Following a general principle, the study of net subclasses has deserved special attention hoping to find computationally efficient algorithms: (controlled) marked graphs [103], [86]; (controlled) state machines [21]; elementary composed state machines [85], etc.

The supervisory control of timed DEDS has recently deserved attention [25], [48]. The synthesis of optimal controllers in this case is essentially a closed-loop scheduling problem [83], [81].

The application of supervisory control concepts and techniques to two MS is documented in [24], [41].

4 Performance Evaluation and Optimization at Design

The magnitude of the investments involved in the construction or significant modification of a manufacturing plant obviously justifies the need for a careful evaluation of the *performance* and *dependability* (i.e., *performability*) in order to guide the design, improvements, and operation (planning and scheduling at different levels).

At the preliminary (i.e., basic, non detailed) design the focus is on the relationships between subsystems and their quantitative effects on the overall behavior. This analysis is a *must* for decisions like the grouping of resources in cells, equipment selection (how much is gained by substituting a machine by a faster one / a buffer by a larger one?), preliminary product-mix selection (i.e., definition of transition visit ratios), buffers and machines allocation, transportation system layout, basic routing strategies, etc. One factor that has a major influence on system performance is the unscheduled down-time due to failures. Traditionally down-time management is done via dependability analysis (reliability, availability, etc.) but it should be done in combination with performance analysis (defining appropriate *rewards* associated to the different states of the system) [187], [194], [122].

In industrial practice the most usual approach is to analyze performance by "ad hoc" *simulation*, using either specific tools for manufacturing systems simulation or a more or less general DEDS simulator (see for instance [138], [134]). Although simulation is a valuable and very general approach, specially suited for the last stages in the performance evaluation and for the tuning of parameters, it has several drawbacks when used alone: (1) The obtained numerical results can be meaningless when some kind of ergodicity is not guaranteed or only a part of the state space has been properly explored; (2) Little insight is gained into the key issues that determine the system's behavior, specially when the simulation model is not built using an adequate formalism; (3) Simulation is highly time-consuming, both in terms of model developing and running time. Nevertheless, simulation appears as a key technique when arbitrary *probability density functions* (pdf) are used or when the structure and size of the model precludes more analytical methods. A major issue is that of distributed simulation (see, for instance, [44], [74]).

Performance evaluation and optimization should start instead by modeling and manipulating the problem using well-founded formalisms and techniques (e.g., Markov chains, queueing networks, mathematical programming). Some of these techniques are *generative* (given a set of criteria and constraints

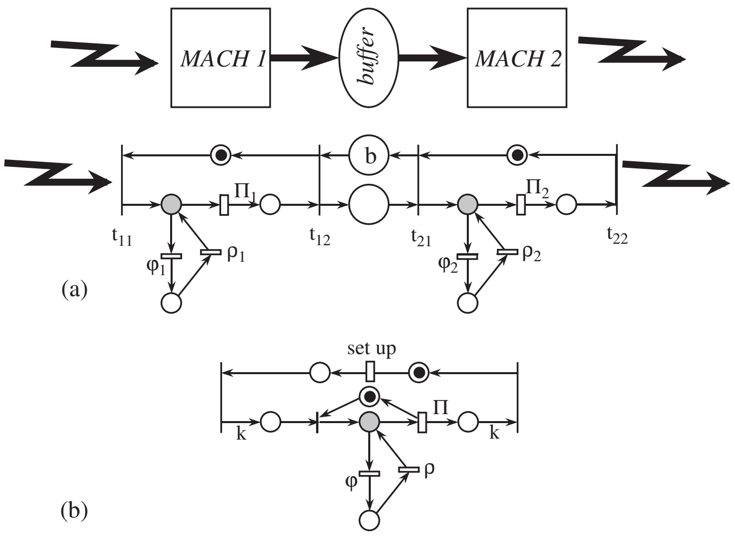

Figure 4: (a) Net model of a two-machines flow line. (b) Model of machine accepting batches.

generate a set of decisions) while others are *evaluative* (given a set of decisions evaluate the performance of the system). For some optimization problems (e.g., buffer size optimization, product mix optimization) evaluative techniques can be used in a iterative schema with some alternatives generation strategy.

Two general requirements for the formalism to use are *adequacy* to model the class of problems and *proximity* to the models to be used in subsequent design phases. Both are satisfied by PN provided with some stochastic and timed interpretation. When this interpretation associates a stochastic timing to transitions (firing delay) and a probabilistic routing, a clear correspondence appears between net *transitions* and *queueing network (QN) stations* and between net *places* and *queues* [33], [190]. In this sense *stochastic PN* are evaluative formalisms that generalize well-known QN formalisms either from the Gordon-Newell type (OR/OR logic) or from the fork-join type (AND/AND logic). Stochastic PN can deal with resource contention, synchronization, batch movements, and can model *functional* descriptions of (parts of) the system besides the *customer/server* point of view. Other interpretations of stochastic PN have been proposed in the literature (e.g., associating time to places or arcs, using different pdf's for time delays, incorporating immediate transitions and priorities, including different queueing and service disciplines, etc.) some of which affect the relationship between the behavior of the interpreted model and the autonomous one [2]. Sound relationships are important to interleave concepts and techniques for the correctness (qualitative) and performance (quantitative) analysis of net models [159], [160].

We shall not insist on basic modeling issues, which are essentially those already mentioned in Section 2. For instance, the *locality* of states and actions has proved very helpful in the *modular* construction of models, as can be easily observed in the literature [194], [66], [62], [3]. As an extremely basic modeling example, Figure 4 (a) shows a net model for a two-machines manufacturing flow line with an intermediate buffer acting as a *low-pass filter* against the possible disruptions due to failures at machines. It is not difficult to recognize the sub-models describing the machines (operation dependent failures) and the buffer. The firing of transition t_{11} represents the arrival of a part to the first machine while parts exit through t_{22} (merging these transitions with others from PN models of other subsystems could be done to analyze this flow line embedded in its environment; here we assume saturation, i.e., *MACH 1* is never starved and *MACH 2* is never blocked). The buffer is modeled by place B (parts processed by *MACH 1* waiting for *MACH 2*) and B' (unused locations in the buffer). Regarding the rates associated to transitions (1) $1/\Pi_i$ is the average processing time, (2) $1/\varphi_i$ is the mean time between failures, and (3) $1/\rho_i$ is the average repair time of machine i. Assume now machines accept *batches* of k parts (e.g., due to the pallet size), although they process parts one by one, and that there is a *set-up time* between two successive batches. Clearly we can substitute the PN models for the machines by the subnet shown in Figure 4 (b), while the buffer places now contain pallets instead of parts.

Next table shows how the throughput (Γ) varies with the buffer size, b (size zero is obtained merging t_{12} and t_{21} and removing B and B') for the model if Figure 4 (a) with $\Pi_i = 1$, $\varphi_i = 0.01$, and $\rho_i = 0.2$:

b	0	1	.	6	7	8	.	15
Γ	.627	.701	.	.828	.839	.848	.	.885
$\overline{M}(B)$	0	0.5	.	3.0	3.5	4.0	.	7.5
Profit	314	345	.	383	**384**	383	.	366

In this case the performance monotonically increases with the buffer size (something that is not true in general [164] although it happens in some systems like flow lines [53] and in some net subclasses like marked graphs [8], [9]). On the other hand the cost also increases with the buffer size both due to the needed investment, proportional to the buffer size, b, and the operational cost related to the work-in-progress (WIP), proportional to the average number of parts in the buffer, $\overline{M}(B)$. In this case the profit, $50\Gamma - b - 10\overline{M}(B)$, is optimized for a buffer size of seven. This example is a basic instance of the so called *minimal initial marking* problem, which has unfortunately been shown to be NP-complete even for marked graphs with deterministic firing time [196]. In [35] an efficient heuristic approach is suggested. An analogous problem from a quite different perspective is considered in [105].

We refer to the literature (books already cited, and also papers like [10], [93], [35], [121], [63]) for models of different manufacturing systems (production lines, job-shops, assembly systems, Kanban systems, etc.).

There exists a significant number of concepts, techniques, and tools for the computation of performance indices, although major efforts are still required to cope with the computational complexity. Most of the available analysis techniques consider exponential (or phase-type) pdf's. Thus the direct computation of the underlying Markov chain is the basic (brute force) approach [3], [62], [194]. In order to reduce the computational complexity, net-structure driven Markov chain generation methods allow significant improvements. In [43] the *symmetries* in the behavior of token color classes (a concept close to that of customer classes in QNs) is exploited. In [68], [102] net systems are structurally decomposed into modules that synchronize at certain transitions. Instead of computing the full Markov chain, the probability distribution of the markings is computed through the Markov chains of subsystems using *Kronnecker algebra*. Purely net structure based approaches have also been proposed. In [92], [153] structural and stochastic conditions at the net level are explored to derive *product form* solutions for performance evaluation, generalizing those of QNs. Product-form analysis techniques have been also used for the net level approximate computation of performance indices [152]. In [33], [36], [42] performance *bounds* are derived from an *operational* approach. In order to improve the quality of the bounds, [37], [113] incorporate (in different ways) higher order moments of the stochastic process. In [32] marked graphs are studied in detail. The obtained bounds are tight (i.e., pdf's always exist such that the bounds are reachable) and the upper bound for the throughput, computed by linear programming (thus in polynomial time) is the exact performance for deterministic timing. Another approach to cope with computational complexity is to use "divide and conquer" techniques to obtain *approximate* values [46], [176], [34], [198], [135]. A quite different approach is to transform the discrete state space into a *continuous* space, as in [55], [56], where examples from manufacturing systems are also presented. This state relaxation turns out to be analogous to the relaxation of integer programming into linear programming for the computation of some qualitative/quantitative properties of net models [161]. Last but not least, even if continuous PN were introduced to cope with computational complexity now they make possible the modeling of continuous or *hybrid* systems [106], [57].

For non Markovian models *recursive equations*, involving firing epochs and/or arrival and departure times, are very promising in order to compute performance indices or to derive qualitative performance properties (e.g., throughput monotonicity, ergodicity, etc.) [9], [8].

5 Scheduling: Performance Control

Scheduling is a fundamental problem in the control of any organization. It concerns the allocation of resources over time to perform a set of tasks, where by *resources* we mean the necessary means to perform the tasks (e.g., machines, tools, energy, information, manpower, etc.) [76], [39], [20].

While *performance evaluation* concerns the *analysis* of time-dependent characteristics of a system as its throughput or its WIP, scheduling can be regarded as a *performance control* activity: given a plant or system apply a control strategy (i.e., decide which *tasks* to perform with the available resources, in which *order*, at which *time*) so that, for instance, a given product-mix is produced optimizing the makespan, the throughput, and/or the WIP. The control strategy can be either *static* (off-line / open-loop) or *dynamic*

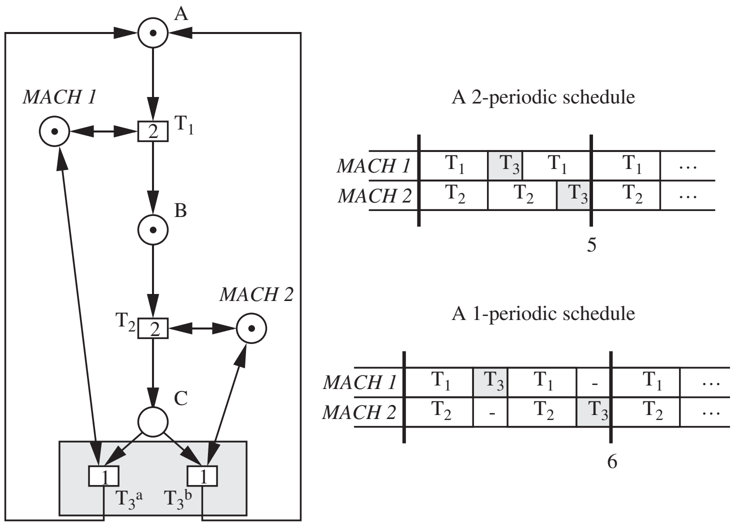

Figure 5: The optimal schedule is k-periodic with $k = 2$ and mean period $5/2$. The initial marking is reproduced in each period at time epochs 3 and 5. The strict periodic (i.e., $k = 1$) schedule's mean period is 3.

(on-line / closed-loop, the feedback of the actual state of the system allows to react to possible disruptive events that affect the system under control). In many cases a kind of *periodic* behavior is also desired [89], [141]. In summary, scheduling determines the time epochs of future controllable events in a sort of multi-objective decision making.

PN with an appropriate timed interpretation are very well suited to the modeling of scheduling problems in parallel and distributed DEDS. PN allow to model within a single formalism the *functional*, *temporal*, and *resource* constraints. These determine the enabled transitions, and then the scheduling problem is reducing the undeterminism by deciding *when* to fire *which* transitions among the enabled ones. As a simple example, consider the net system in Figure 5, which models a system with a *working plan* consisting on the repeated execution of three tasks in sequence (T_1 and T_2 taking two time units, and T_3 taking one). These tasks are performed on two machines (T_1 on *MACH 1*, T_2 on *MACH 2*, and T_3 on any of them). A possible scheduling problem is deciding the firing epoch for each transition so that the throughput is maximized.

A scheduling problem can be *deterministic* (i.e., all timings are of known deterministic duration) or *stochastic*. We can restrict the analysis to single server semantics because multiple server semantics scheduling problems can be transformed to single server in a very general setting [132].

In scheduling theory it is conventionally assumed that tasks are to be executed *only once*. Periodic or cyclic schedules [89] are seldom treated by the theory despite they abound in practice. PN are also well suited to model periodic behaviors, as the case of Figure 5 illustrates. Periodic schedules can be *strict* (i.e., the firing of each transition happens at fixed and regular time intervals) or *not* (i.e., some time regularity exists between successive periods although individual transition firings may not be periodic). Periodic schedules are conceptually characterized by the notion of k-periodicity [39], [40]. Strict periodic schedules (i.e., with $k = 1$) are not optimal in general, as illustrated by Figure 5, but they can be easier to implement. For deterministically timed bounded net systems optimal schedules are k-periodic. These correspond to critical circuits of the so called *earliest state graph* [40], which is frequently too large. Actually the computation of earliest schedules is usually an NP-problem even for subclasses such as live and bounded free choice systems [114], [142].

Approaches to scheduling of PN models range from quantitative algorithmic (using *dynamic programming*, *branch and bound*, A^*, etc.) to rather qualitative knowledge-based approaches. In between, heuristics incorporated to algorithmic approaches aim at reducing the computational complexity at the price of finding just suboptimal or "acceptable" solutions.

In [154] a modified branch and bound procedure is used to schedule some robot control computations to minimize the makespan. A net decomposition heuristic is applied to reduce the complexity, so global optimality is not guaranteed. In [120] a modified beam search is used on a class of net models. Heuristic approaches obtained modifying standard algorithms like A^* are used in [107, 108], [169] for non-repetitive behaviors. In [132] the throughput bound in [33] is proposed to inform a search for repetitive behaviors. In [131], [38] systematic approaches to schedule ratio-driven FMS modeled with PN are presented. The idea in [38] is to transform the original PN model into a marked graph in which all allocations and sequencings of resources are solved. The optimization is done hierarchically, first looking at the tradeoff between throughput and cycle horizon (using *flow* optimization) and later reducing the WIP using a *discrete* optimization procedure.

In [91] a rule-based dispatching heuristic is evaluated for on-line scheduling on a stochastic PN model dealing with uncertain events (as failure of machine tools and repair time). *Simulated annealing*, an efficient meta-heuristic search approach needing very little information on the model and objectives, has also been used to schedule MS [139], after solving the short-term planning allocating resources to processes by a linear constraint problem. In [171] two heuristic algorithms for *priority-list* scheduling in deterministically timed PN are presented and compared. In [133] a survey of scheduling rules is presented. They can be used with PN models of the plant. More generally, the integration of PN models and AI techniques is used in many works (see for instance [119], [182]). Scheduling is not only solved using heuristic rules but also considering the essential problem of conflict solving as a constraint satisfaction problem when production requirements are expressed in quantitative terms, time constraints, etc. (see for instance [119]). In some cases no solution is possible if the production requirements are too rigid, thus some constraints have to be relaxed until a solution is found. In this sense it is interesting to consider *fuzzy* production requirements to compute a schedule [69].

Finally, although presently not too much has been achieved, a major hope in using PN for scheduling problems is the potential use of its structure theory to reduce the practical computational complexity. For instance, it is known that in choice-free systems (every place has at most one output transition; event graphs [40], [124] are particular cases) earliest firing maximizes throughput which is not true in general because local optimality does not guarantee global optimality [164]. Attempts concerning some subclasses of live and bounded free choice systems are considered in [142]. For large models the practical complexity can be reduced when certain *local* properties are satisfied [143]. For instance, if a subnet is a single-input/single-output *macrotransition*, then earliest firing inside it (local optimum) does not prevent finding a global optimum.

6 Concluding Remarks

The adequacy of PN to deal with a diversity of problems in MS design and operation, including modeling, validation, implementation, performability analysis, and scheduling has been discussed. The formal framework provided by PN, and particularly their representation of the structure of systems, has proved helpful in the treatment of hard problems in this and other domains. Moreover, the notion of a *family of formalisms* allows to adapt to particular problems and domains without losing the possibility of mutual communication and reutilization of results. A number of PN computer tools have been developed to assist in the modeling and qualitative or quantitative analysis. Several tool descriptions can be accessed through `http://www.daimi.aau.dk/PetriNets/`, the Web page on PN maintained by DAIMI, Aarhus University.

Despite the great amount of work and achievements, much work remains to be done to meet industrial requirements.

References

[1] M. Ajmone Marsan, editor. *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*. Springer, 1993.

[2] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Trans. on Software Engineering*, 15(7):832–846, 1989.

[3] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.

[4] S. Amar, E. Craye, and J. C. Gentina. A method for hierarchical specification and prototyping of flexible manufacturing systems. In *Proc. IEEE Workshop on Emerging Technologies and Factory Automation*, pages 44–59, Melbourne, Australia, 1992.

[5] A. Avizenis and J. P. Kelly. Fault tolerance by design diversity: Concepts and experiments. *Computer*, 17(8):67–80, 1984.

[6] J. M. Ayache, P. Azema, and M. Diaz. Observer, a concept for on line detection for control errors in concurrent systems. In *Proc. 9^{th} IEEE Int. Sypm. Fault-Tolerant Computing*, pages 79–86, Madison, WI, USA, June 1992.

[7] J. A. Bañares, P. Muro, and J. L. Villarroel. Taking advantages of temporal redundancy in high level Petri nets implementations. In Ajmone Marsan [1], pages 32–48.

[8] F. Baccelli, G. Cohen, and B. Gaujal. Recursive equations and basic properties of timed Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 1:415–439, 1992.

[9] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat. *Synchronization and Linearity.* Wiley, 1992.

[10] G. Balbo, G. Chiola, G. Franceschinis, and G. Molinar. Generalized stochastic Petri nets for the performance evaluation of FMS. In Silva and Murata [162], pages 1013–1018.

[11] M. Baldassari and G. Bruno. Protob: An object oriented methodology for developing discrete event dynamic systems. *Computer Languages*, 16(1):39–63, 1991.

[12] Z. A. Banaszak and B. H. Krogh. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Trans. on Robotics and Automation*, 6(6):724–734, 1990.

[13] E. Battiston, F. DeCindio, and G. Mauri. OBJSA nets: A class of high-level Petri nets having objects as domains. In Rozenberg [146], pages 20–43.

[14] R. Bauman and T. A. Turano. Production based language simulation of Petri nets. *Simulation*, 47(5):191–198, 1986.

[15] F. Belli and K. E. Grosspietsch. Specification of fault-tolerant systems issues by predicate/transition nets and regular expressions: Approach and case study. *IEEE Trans. on Software Engineering*, 17(6):513–526, 1991.

[16] L. Bernardinello and F. DeCindio. A survey of basic net models and modular net classes. In Rozenberg [148], pages 304–351.

[17] G. Berthelot. Checking properties of nets using transformations. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1986.

[18] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. on Software Engineering*, 17(3):259–273, 1991.

[19] J. Billington. Many-sorted high-level nets. In *Procs. of the 3^{th} Int. Workshop on Petri Nets and Performance Models (PNPM89)*, pages 166–179. IEEE Computer Society Press, 1989.

[20] J. Blazewicz, K. H. Ecker, G. Schmidt, and J. Weglarz. *Scheduling in Computer and Manufacturing Systems.* Springer, 1994.

[21] R. K. Boel, L. Ben-Naoum, and V. Van Breusegem. On forbidden state problems for a class of controlled Petri nets. *IEEE Trans. on Automatic Control*, 40(10):1717–1731, 1995.

[22] P. Borne, J. C. Gentina, E. Craye, and S. El Khattabi, editors. *Symposium on Discrete Events and Manufacturing Systems. CESA '96 IMACS Multiconference*, Lille, France, July 1996.

[23] G. W. BRAMS. *Réseaux de Petri: Théorie et Pratique.* Masson, 1983.

[24] B. A. Brandin. The real-time supervisory control of an experimental manufacturing cell. *IEEE Trans. on Robotics and Automation*, 12(1):1–14, 1996.

[25] B. A. Brandin and W. M. Wonham. Supervisory control of timed discrete event systems. *IEEE Trans. on Automatic Control*, 39(2):329–341, 1994.

[26] W. Brauer. How to play the token game? *Petri Net Newsletter*, 16:3–13, 1984.

[27] W. Brauer, R. Gold, and W. Vogler. A survey of behaviour and equivalence preserving refinements of Petri nets. In Rozenberg [147], pages 1–46.

[28] J. L. Briz and J. M. Colom. Implementation of weighted P/T nets based on linear enabling functions. In Valette [178], pages 99–118.

[29] G. Bruno and G. Marchetto. Process-translatable Petri nets for the rapid prototyping of process control systems. *IEEE Trans. on Software Engineering*, 12(2):346–357, 1986.

[30] G. Bucci and E. Vicario. Compositional validation of time-critical system using communicating time Petri nets. *IEEE Trans. on Software Engineering*, 21(12):969–992, 1995.

[31] D. Buchs and N. Guelfi. CO-OPN: A concurrent object oriented Petri net approach. In *Proc.* 12^{th} *Intern. Conference on Application and Theory of Petri Nets*, pages 432–454, Aarhus, Denmark, June 1991.

[32] J. Campos, G. Chiola, J. M. Colom, and M. Silva. Properties and performance bounds for timed marked graphs. *IEEE Trans. on Circuits and Systems-I: Fundamental Theory and Applications*, 39(5):386–401, 1992.

[33] J. Campos, G. Chiola, and M. Silva. Properties and performance bounds for closed free choice synchronized monoclass queueing networks. *IEEE Trans. on Automatic Control*, 36(12):1368–1382, 1991.

[34] J. Campos, J. M. Colom, H. Jungnitz, and M. Silva. Approximate throughput computation of stochastic marked graphs. *IEEE Trans. on Software Engineering*, 20(7):526–535, 1994.

[35] J. Campos, J. M. Colom, and M. Silva. Performance evaluation of repetitive automated manufacturing systems. In *Procs.* 2^{nd} *Int. Conf. on Computer Integrated Manufacturing and Automation Technology (CIMAT '90)*, pages 74–81. IEEE-Computer Society Press, 1990.

[36] J. Campos and M. Silva. Structural techniques and performance bounds of stochastic Petri net models. In Rozenberg [148], pages 352–391.

[37] J. Campos and M. Silva. Embedded product-form queueing networks and the improvement of performance bounds for Petri net systems. *Performance Evaluation*, 18(1):3–19, 1993.

[38] H. Camus, H. Ohl, O. Korbaa, and J. C. Gentina. Cyclic schedules in FMS with flexibilities in operating sequences. In Silva et al. [165], pages 97–116.

[39] J. Carlier and P. Chretienne. *Problèmes d'Ordonnancement. Modélisation, complexité et algorithmes.* Masson, 1988.

[40] J. Carlier and P. Chretienne. Timed Petri net schedules. In Rozenberg [146], pages 62–84.

[41] F. Charbonnier. *Commande Supervisée des Systèmes à Événements Discrets.* PhD thesis, INPG, Grenoble, France, January 1996.

[42] G. Chiola, C. Anglano, J. Campos, J. M. Colom, and M. Silva. Operational analysis of timed Petri nets and application to the computation of performance bounds. In F. Baccelli, A. Jean-Marie, and I. Mitrani, editors, *Quantitative Methods in Parallel Systems*, pages 161–174. Springer, 1995. Also appears in Procs. PNPM93.

[43] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Trans. on Computers*, 42(11), 1993.

[44] G. Chiola and A. Ferscha. Distributed simulation of Petri nets. *IEEE Trans. on Parallel and Distributed Technology*, 3(1):33–50, 1993.

[45] S. Christensen and L. Petrucci. Modular state space analysis of coloured Petri nets. In De Michelis and Diaz [58], pages 201–217.

[46] G. Ciardo and K. S. Trivedi. A decomposition approach for stochastic reward net models. *Performance Evaluation*, 18(1):37–59, 1993.

[47] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems*, 8:244–263, 1986.

[48] D. D. Cofer and V. K. Garg. Supervisory control of real-time discrete-event systems using lattice theory. *IEEE Trans. on Automatic Control*, 41(2):199–209, 1996.

[49] J. M. Colom, M. Silva, and J. L. Villarroel. On software implementation of Petri nets and colored Petri nets using high-level concurrent languages. In *Proc. 7^{th} European Workshop on Application and Theory of Petri Nets*, pages 207–241, Oxford, England, July 1986.

[50] F. Commoner, A. W. Holt, S. Even, and A. Pnueli. Marked directed graphs. *Journal on Computer Systems Science*, 5:72–79, 1971.

[51] M. Courvoisier, R. Valette, J. M. Bigou, and P. Esteban. A programmable logic controller based on a high level specification tool. In *Proc. IEEE Annual Conference on Industrial Electronics (IECON 83)*, pages 174–179, San Francisco, USA, November 1983.

[52] D. Crockett, A. Desrochers, F. DiCesare, and T. Ward. Implementation of a Petri net controller for a machining workstation. In Silva and Murata [162], pages 1861–1867.

[53] Y. Dallery and S. B. Gershwin. Manufacturing flow line systems: A review of models and analytical results. *Queueing Systems: Theory and Applications*, 12:3–94, 1992.

[54] R. David. Grafcet: A powerful tool for specification of logic controllers. *IEEE Trans. on Control Systems Technology*, 3(3):253–268, 1995.

[55] R. David and H. Alla. *Petri Nets and Grafcet.* Prentice-Hall, 1992.

[56] R. David and H. Alla. Autonomous and timed continuous Petri nets. In Rozenberg [149], pages 71–90.

[57] R. David and H. Alla. Petri nets for modeling of dynamic systems — a survey. *Automatica*, 30(2):175–202, 1994.

[58] G. De Michelis and M. Diaz, editors. *Application and Theory of Petri Nets 1995*, volume 935 of *Lecture Notes in Computer Science.* Springer, 1995.

[59] F. DeCindio, G. DeMichelis, L. Pomello, and C. Simone. Superposed automata nets. In Girault and Reisig [84].

[60] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 1995.

[61] A. Desrochers, editor. *Modeling and Control of Automated Manufacturing Systems.* IEEE Computer Society Press, 1989.

[62] A. Desrochers and R. Y. Al-Jaar. *Applications of Petri Nets in Manufacturing Systems.* IEEE Press, 1994.

[63] A. Desrochers, H. Jungnitz, and M. Silva. An approximation method for the performance analysis of manufacturing systems based on GSPNs. In *Procs. 3^{rd} Int. Conf. on Computer Integrated Manufacturing and Automation Technology (CIMAT '92)*, pages 46–55. IEEE Computer Society Press, 1992.

[64] A. Desrochers and M. Silva, editors, Special issue on computer integrated manufacturing. *IEEE Trans. on Robotics and Automation*, 10(2), 1994.

[65] M. Diaz, G. Juanole, and J. P. Courtiat. Observer — a concept for formal on-line validation of distributes systems. *IEEE Trans. on Software Engineering*, 20(12):900–913, 1994.

[66] F. Dicesare, G. Harhalakis, J. M. Proth, M. Silva, and F. B. Vernadat. *Practice of Petri Nets in Manufacturing*. Chapman & Hall, 1993.

[67] R. DiGiovanni. Petri nets and software engineering: HOOD nets. In *Proc.* 11^{th} *Int. Conference on Application and Theory of Petri Nets*, pages 123–138, Paris, France, June 1990.

[68] S. Donatelli. Superposed generalized stochastic Petri nets: Definition and efficient solution. In Valette [178], pages 258–277.

[69] D. Dubois, J. C. Pascal, and R. Valette. Fuzzy decision and control in manufacturing automation. In R. C. Dorf and A. Kusiak, editors, *Handbook of Design, Manufacturing and Automation*, pages 955–975. Wiley, 1994.

[70] J. Duggan and J. Browne. ESPNET: Expert-system-based simulator of Petri nets. *IEE Proceedings*, 135(4):239–247, 1988.

[71] J. Esparza and M. Silva. On the analysis and synthesis of free choice systems. In Rozenberg [147], pages 243–286.

[72] J. Ezpeleta, J. M. Colom, and J. Martínez. A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Trans. on Robotics and Automation*, 11(2):173–184, 1995.

[73] R. Fehling. A concept of hierarchical Petri nets with building blocks. In Rozenberg [149], pages 148–168.

[74] A. Ferscha. Parallel and distributed simulation of discrete event systems. In A. Y. Zomaya, editor, *Parallel and Distributed Computing Handbook*, pages 1003 – 1041. McGraw-Hill, 1996.

[75] A. Finkel. The minimal coverability graph for Petri nets. In Rozenberg [149], pages 210–243.

[76] S. French. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Ellis-Horwood, 1982.

[77] J. C. Gentina, organizer. *Invited sessions: Application of Petri Nets to Flexible Manufacturing.* 12^{th} *IMACS World Congress*, Paris, France, 1988.

[78] J. C. Gentina, J. P. Bourey, and M. Kapusta. Coloured adaptive structured Petri nets. *Computer-Integrated Manufacturing*, 1(1):39–47, 1988.

[79] J. C. Gentina, J. P. Bourey, and M. Kapusta. Coloured adaptive structured Petri nets (II). *Computer-Integrated Manufacturing*, 1(2):103–109, 1988.

[80] J. C. Gentina and D. Corbeel. Hierarchical control of flexible manufacturing systems. In Silva and Murata [162], pages 1166–1173.

[81] S. B. Gershwin. Hierarchical flow control: a framework for scheduling and planning discrete events in manufacturing systems. In *Proceedings of the IEEE* [94], pages 195–209.

[82] S. B. Gershwin. *Manufacturing Systems Engineering*. Prentice-Hall, 1994.

[83] S. B. Gershwin, R. R. Hildebrant, R. Suri, and S. K. Mitter. A control perspective on recent trends in manufacturing systems. *Control Systems Magazine*, 6(2):3–15, 1986. Reprinted in [61].

[84] C. Girault and W. Reisig, editors. *Application and Theory of Petri Nets*. Springer, 1982.

[85] A. Giua and F. DiCesare. Petri net incidence matrix analysis for supervisory control. *IEEE Trans. on Robotics and Automation*, 10(2):185–195, 1994.

[86] A. Giua, F. DiCesare, and M. Silva. Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *IEEE Int. Conf. on Systems, Man, and Cybernetics*, Chicago, IL, USA, October 1992.

[87] A. Giua, F. DiCesare, and M. Silva. Petri net supervisors for generalized mutual exclusion constraints. In 12^{th} *IFAC World Congress*, volume 1, pages 267–270, Sidney, Australia, July 1993.

[88] M. H. T. Hack. Analysis of production schemata by Petri nets. Master's thesis, M.I.T., Cambridge, MA, USA, 1972. (Corrections in *Computation Structures Note* 17, 1974).

[89] C. Hanen and A. Munier. Cyclic scheduling problems: An overview. In P. Chretienne, E. G. Coffman, J. K. Lengstra, and Z. Liu, editors, *Scheduling Theory and its Applications*. Wiley, 1995.

[90] K. Hasegawa, K. Takahashi, and P. E. Miyagi. Application of the mark flow graph to represent discrete event production systems and system control. *Trans. of the Society of Intrument and Control Engineers (SICE Japan)*, 24(1):69–75, 1988.

[91] I. Hatono, K. Yamagata, and H. Tamura. Modeling and on-line scheduling of FMS using stochastic Petri nets. *IEEE Trans. on Software Engineering*, 17(2):126–132, 1991.

[92] W. Henderson and P. G. Taylor. Embedded processes in stochastic Petri nets. *IEEE Trans. on Software Engineering*, 17(2):108–116, 1991.

[93] H. P. Hillion and J. M. Proth. Performance evaluation of job-shop systems using timed event-graphs. *IEEE Trans. on Automatic Control*, 34(1):3–9, 1989.

[94] L. Ho, editor, Special issue on discrete event systems. *Proceedings of the IEEE*, 77(1), 1989.

[95] L. E. Holloway and B. H. Krogh. On closed-loop liveness of discrete event systems under maximally permissive control. *IEEE Trans. on Automatic Control*, 37(5):692–697, 1992.

[96] L. E. Holloway, B. H. Krogh, and A. Giua. Petri nets for the control of discrete event systems: A tutorial survey. In *Supervisory Control of Discrete Event Systems*. Laboratoire d'Automatique de Grenoble, INPG, September 1995.

[97] P. Huber, K. Jensen, and R. M. Saphiro. Hierarchies in coloured Petri nets. In Rozenberg [147], pages 313–341. Collected in [101].

[98] M. A. Jafari. An architecture for a shop-floor controller using colored Petri nets. *Int. Jnl. Flexible Manufacturing Systems*, 4:159–181, 1992.

[99] M. D. Jeng and F. DiCesare. Synthesis using resource control nets for modeling shared resource systems. *IEEE Trans. on Robotics and Automation*, 11(3):317–327, 1995.

[100] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods, and Practical Use.* EATCS Monographs on Theoretical Computer Science. Springer, 1994.

[101] K. Jensen and G. Rozenberg, editors. *High-level Petri Nets.* Springer, 1991.

[102] P. Kemper. Numerical analysis of superposed GSPNs. In *Procs. of the* 6^{th} *Int. Workshop on Petri Nets and Performance Models (PNPM95)*, pages 52–61. IEEE Computer Society Press, 1995.

[103] B. H. Krogh and L. E. Holloway. Synthesis of feedback control logic for discrete manufacturing systems. *Automatica*, 27(4):641–651, 1991.

[104] S. Kumagai, editor, Special section on application of Petri nets to concurrent system design. *IEICE Trans. on Fundamentals of Electronics, Communication, and Computer Science*, E75-A(10), 1992.

[105] S. Laffit, J. M. Proth, and X. L. Xie. Optimization of invariant criteria for event graphs. *IEEE Trans. on Automatic Control*, 37(5):547–555, 1992.

[106] J. Le Bail, H. Alla, and R. David. Hybrid Petri nets. In *European Control Conference (ECC '91)*, pages 1472–1477, Grenoble, France, 1991.

[107] D. Y. Lee and F. DiCesare. Integrated scheduling of flexible manufacturing systems employing automated guided vehicles. *IEEE Trans. on Industrial Electronics*, 41(6):602–610, 1994.

[108] D. Y. Lee and F. DiCesare. Scheduling flexible manufacturing systems using Petri nets and heuristic search. *IEEE Trans. on Robotics and Automation*, 10(2):123–132, 1994.

[109] N. G. Leveson and J. L. Stolzy. Safety analysis using Petri nets. *IEEE Trans. on Software Engineering*, 13(3):386–397, 1987.

[110] Y. Li and W. M. Wonham. Control of vector discrete event systems I - the base model. *IEEE Trans. on Automatic Control*, 38(8):1214–1227, 1993.

[111] Y. Li and W. M. Wonham. Control of vector discrete event systems II - controller synthesis. *IEEE Trans. on Automatic Control*, 39(3):512–531, 1994.

[112] Y. Li and W. M. Wonham. Concurrent vector discrete event systems. *IEEE Trans. on Automatic Control*, 40(4):628–637, 1995.

[113] Z. Liu. Performance bounds for stochastic timed Petri nets. In De Michelis and Diaz [58], pages 316–334.

[114] J. Magott. New NP-complete problems in performance evaluation of concurrent systems using Petri nets. *IEEE Trans. on Software Engineering*, 13(5):578–581, 1987.

[115] D. Mandrioli and C. Ghezzi. *Theoretical Foundations of Computer Science*. John Wiley and Sons, 1987.

[116] Z. Manna and A. Pnueli. The anchored version of the temporal framework. In J. W. de Bakker et al., editors, *Linear Time, Branching Time and Partial Order in Logics and Models of Concurrency*, volume 354 of *Lecture Notes in Computer Science*, pages 201–284. Springer, 1989.

[117] J. Martínez, H. Alla, and M. Silva. Petri nets for the specification of FMSs. In A. Kusiak, editor, *Modelling and Design of Flexible Manufacturing Systems*, pages 389–406. Elsevier, 1986.

[118] J. Martínez, P. Muro, and M. Silva. Modeling, validation and software implementation of production systems using high level Petri nets. In Silva and Murata [162], pages 1180–1185.

[119] J. Martínez, P. Muro, M. Silva, S. F. Smith, and J. L. Villarroel. Merging artificial intelligence techniques and Petri nets for real time scheduling and control of production systems. In R. Huber et al., editors, *Artificial Intelligence in Scientific Computation*, pages 307–313. Scientific Publishing Co., 1989.

[120] H. Martins and T. Sekiguchi. A timed Petri net and beam search based on-line FMS scheduling system with routing flexibility. In *IEEE Int. Conf. on Robotics and Automation*, pages 2548–2553, Sacramento, CA, USA, April 1991.

[121] M. Mascolo, Y. Frein, Y. Dallery, and R. David. A unified modeling of Kanban systems using Petri nets. *Int. Jnl. Flexible Manufacturing Systems*, 3:275–307, 1991.

[122] J. F. Meyer. Performability evaluation: Where it is and what lies ahead. In *IEEE IPDS '95*, pages 334–343. IEEE Computer Society Press, 1995.

[123] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[124] A. Munier. Régime asymptotique d'un graphe d'événements temporisé généralisé: application à un problème d'assemblage. *RAIRO-APII*, 27(5):487–513, 1993.

[125] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[126] T. Murata, N. Komoda, K. Matsumoto, and K. Haruna. A Petri net based controller for flexible and maintainable sequence control and its applications in factory automation. *IEEE Trans. on Industrial Electronics*, 33(1):1–8, 1986. Reprinted in [61].

[127] Y. Nagao, H. Urabe, S. Nakano, and S. Kumagai. Application of Petri nets to sequence control. *IEICE Trans. on Fundamentals of Electronics, Communication, and Computer Science*, E76-A(10):1598–1606, 1993.

[128] Y. Narahari and N. Viswanadham. A Petri net approach to the modelling and analysis of flexible manufacturing systems. *Annals of Operations Research*, 3:449–472, 1985.

[129] R. A. Nelson, L. M. Haibt, and P. B. Sheridan. Casting Petri nets into programs. *IEEE Trans. on Software Engineering*, 9(5):590–602, 1983.

[130] M. Notomi and T. Murata. Hierarchical reachability graph of bounded Petri nets for concurrent software analysis. *IEEE Trans. on Software Engineering*, 20(5):325–336, 1994.

[131] H. Ohl, H. Camus, E. Castelain, and J. C. Gentina. A heuristic algorithm for the computation of cyclic schedules and the necessary WIP to obtain optimum cycle time. In *Procs.* 4^{th} *Int. Conf. on Computer Integrated Manufacturing and Automation Technology (CIMAT '94)*, pages 339–344. IEEE Computer Society Press, 1994.

[132] K. Onaga, M. Silva, and T. Watanabe. On periodic schedules for deterministically timed Petri nets. *IEICE Trans. on Fundamentals of Electronics, Communication, and Computer Science*, E76-A(4):580–592, 1993.

[133] S. S. Panwalkar and W. Iskandar. A survey of scheduling rules. *Operations Research*, 25(1):45–61, 1977.

[134] H. T. Papadopoulos, C. Heavey, and J. Browne. *Queueing Theory in Manufacturing Systems Analysis and Design*. Chapman & Hall, 1993.

[135] C. J. Pérez, J. Campos, and M. Silva. Approximate throughput computation of a class of cooperating sequential processes. In *Procs.* 5^{th} *Int. Conf. on Computer Integrated Manufacturing and Automation Technology (CIMAT '96)*, pages 382–389, 1996.

[136] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[137] L. Pomello, G. Rozenberg, and C. Simone. A survey of equivalence notions for net based systems. In Rozenberg [148], pages 410–472.

[138] A. A. B. Pritsker and C. D. Pegden. *Introduction to Simulation and SLAM*. Wiley, 1979.

[139] J. M. Proth and X. Xie. *Les Réseaux de Petri pour la Conception et la Gestion des Systèmes de Production*. Masson, 1995.

[140] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. In *Proceedings of the IEEE* [94], pages 81–98.

[141] K. Ramamritham. Allocation and scheduling of precedence-related periodic tasks. *IEEE Trans. on Parallel and Distributed Systems*, 6(4):412–420, 1995.

[142] A. Ramírez. *Scheduling en Redes de Petri*. PhD thesis, DIEI. Univ. Zaragoza, January 1994. In Spanish.

[143] A. Ramírez, J. Campos, and M. Silva. On optimal scheduling in DEDS. In *Procs. 1993 IEEE Int. Conf. on Robotics and Automation*, pages 821–826, Atlanta, GA, USA, May 1993.

[144] L. Recalde, E. Teruel, and M. Silva. On well-formedness analysis: The case of deterministic systems of sequential processes. In J. Desel, editor, *Proc. of the Int. Workshop on Structures in Concurrency Theory (STRICT)*, Workshops in Computing, pages 279–293. Springer, 1995.

[145] W. Reisig. Petri nets and algebraic specifications. *Theoretical Computer Science*, 80:1–34, 1991.

[146] G. Rozenberg, editor. *Advances in Petri Nets 1988*, volume 340 of *Lecture Notes in Computer Science*. Springer, 1988.

[147] G. Rozenberg, editor. *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*. Springer, 1991.

[148] G. Rozenberg, editor. *Advances in Petri Nets 1992*, volume 609 of *Lecture Notes in Computer Science*. Springer, 1992.

[149] G. Rozenberg, editor. *Advances in Petri Nets 1993*, volume 674 of *Lecture Notes in Computer Science*. Springer, 1993.

[150] A. Sahraoui, H. Atabachche, M. Courvoisier, and R. Valette. Joining Petri nets and knowledge based systems for monitoring purposes. In Silva and Murata [162], pages 1160–1165.

[151] K. Schmidt. Symmetries of Petri nets. *Petri Net Newsletter*, 43:9–25, 1993.

[152] M. Sereno. Approximate mean value analysis technique for non-product form solution stochastic Petri nets: An application to stochastic marked graphs. In *Procs. of the* 6^{th} *Int. Workshop on Petri Nets and Performance Models (PNPM95)*, pages 42–51. IEEE Computer Society Press, 1995.

[153] M. Sereno and G. Balbo. Computational algorithms for product form solution stochastic Petri nets. In *Procs. of the* 5^{th} *Int. Workshop on Petri Nets and Performance Models (PNPM93)*. IEEE Computer Society Press, 1993.

[154] L. Shen, Q. Chen, and J. Y. S. Luh. Truncation of Petri net models for simplifying computation of optimum scheduling problems. *Computers in Industry*, 20:25–43, 1992.

[155] C. Sibertin-Blanc. Cooperative nets. In Valette [178], pages 377–396.

[156] J. Sifakis. Realization of fault-tolerant systems by coding Petri nets. *Design Automation and Fault-Tolerant Computing*, 3(2):93–107, 1979.

[157] M. Silva. *Las Redes de Petri: en la Automática y la Informática.* AC, 1985.

[158] M. Silva. Logic controllers. In *IFAC Symposium on Low Cost Automation (vol. II)*, pages 157–166, Milano, Italy, November 1989.

[159] M. Silva. Interleaving functional and performance structural analysis of net models. In Ajmone Marsan [1], pages 17–23.

[160] M. Silva and J. Campos. Structural performance analysis of stochastic Petri nets. In *IEEE IPDS '95*, pages 61–70. IEEE Computer Society Press, 1995.

[161] M. Silva and J. M. Colom. On the computation of structural synchronic invariants in P/T nets. In Rozenberg [146], pages 387–417.

[162] M. Silva and T. Murata, organizers. *Invited Sessions: Petri Nets and Flexible Manufacturing. IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, USA, April 1987.

[163] M. Silva and E. Teruel. Analysis of autonomous Petri nets with bulk services and arrivals. In G. Cohen and J. P. Quadrat, editors, 11^{th} *Int. Conf. on Analysis and Optimization of Systems. Discrete Event Systems*, volume 199 of *Lecture Notes in Control and Information Sciences*, pages 131–143. Springer, 1994.

[164] M. Silva and R. Valette. Petri nets and flexible manufacturing. In G. Rozenberg, editor, *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*, pages 374–417. Springer, 1989.

[165] M. Silva, R. Valette, and K. Takahashi, editors. *Procs.* 1^{st} *Int. Workshop on Manufacturing and Petri Nets*, Osaka, Japan, June 1996.

[166] M. Silva and S. Velilla. Programmable logic controllers and Petri nets: A comparative study. In E. A. Puente and G. Ferrate, editors, *Software for Computer Control,* 3^{rd} *IFAC/IFIP Symposium*, pages 83–88, Madrid, Spain, October 1983. Pergamon.

[167] M. Silva and S. Velilla. Error detection and correction on Petri net models of discrete event control systems. In *Proc. ISCAS 85*, pages 921–924, 1985.

[168] M. Y. Souissi. Deterministic systems of sequential processes: A class of structured Petri nets. In Rozenberg [149], pages 406–426.

[169] T. H. Sun, C. W. Chen, and L. C. Fu. A Petri net based approach to modeling and scheduling for an FMS and a case study. *IEEE Trans. on Industrial Electronics*, 41(6):593–601, 1994.

[170] I. Suzuki and T. Murata. Stepwise refinement of transitions and places. In Girault and Reisig [84], pages 136–141.

[171] T. Tanida, T. Watanabe, and K. Onaga. Priority-list scheduling in Petri nets. In *IEICE Trans. on Fundamentals of Electronics, Communication, and Computer Science* [104], pages 1394–1406.

[172] D. Taubner. On the implementation of Petri nets. In Rozenberg [146], pages 418–439.

[173] E. Teruel, J. M. Colom, and M. Silva. Choice-free Petri nets: A model for deterministic concurrent systems with bulk services and arrivals. *IEEE Trans. on Systems, Man, and Cybernetics*, 27(1):73–83, 1997.

[174] E. Teruel and M. Silva. Structure theory of equal conflict systems. *Theoretical Computer Science*, 153(1-2):271–300, 1996.

[175] E. Teruel, M. Silva, J. M. Colom, and J. Campos. Functional and performance analysis of cooperating sequential processes. In F. Baccelli, A. Jean-Marie, and I. Mitrani, editors, *Quantitative Methods in Parallel Systems*, pages 52–65. Springer, 1995.

[176] M. Tilgner and M. Silva. Approximate performance analysis on Petri net based models of manufacturing systems. Research report, Dep. Ingeniería Eléctrica e Informática, Universidad de Zaragoza, María de Luna, 3, 50015 Zaragoza, Spain, 1993. Presented in the 1994 IEEE Int. Conf. on Robotics and Automation.

[177] R. Valette. Analysis of Petri nets by stepwise refinements. *Journal of Computer and System Sciences*, 18(1):35–46, 1979.

[178] R. Valette, editor. *Application and Theory of Petri Nets 1994*, volume 815 of *Lecture Notes in Computer Science*. Springer, 1994.

[179] R. Valette and B. Bako. Software implementation of Petri nets and compilation of rule-based systems. In G. Rozenberg, editor, *Advances in Petri Nets 1991*, volume 524 of *Lecture Notes in Computer Science*, pages 296–316. Springer, 1991.

[180] R. Valette, J. Cardoso, H. Atabakhche, M. Courvoisier, and T. Lemaire. Petri nets and production rules for decision levels in FMS control. In *Procs. IMACS 1988,* 12^{th} *World Congress on Scientific Computation*, pages 522–524, 1988.

[181] R. Valette, J. Cardoso, and D. Dubois. Monitoring manufacturing systems by means of Petri nets with imprecise markings. In *Procs. IEEE Int. Symp. on Intelligent Control*, pages 233–238, Albany, NY, USA, 1989.

[182] R. Valette and M. Courvoisier. Petri nets and artificial intelligence. In *Procs. IEEE/SICE Int. Workshop on Emerging Technologies for Factory Automation*, pages 218–238, North Queensland, Australia, August 1992.

[183] R. Valette and M. Courvoisier. Petri nets and artificial intelligence. In R. Zurawski and T. Dillon, editors, *Modern Tools for Manufacturing Systems*, pages 385–405. Elsevier, 1993.

[184] R. Valette, M. Courvoisier, J. M. Bigou, and J. Albukerque. A Petri nets based programmable logic controller. In *IFIP* 1^{st} *Int. Conf. on Computer Applications in Production and Engineering*, Amsterdam, Holland, April 1983.

[185] A. Valmari. Stubborn sets for reduced state space generation. In Rozenberg [147], pages 491–515.

[186] A. Valmari. Compositional state space generation. In Rozenberg [149], pages 427–457.

[187] N. M. van Dijk, B. R. Haverkort, and I. G. Niemegeers, editors, Special issue on performability modelling of computer and communication systems. *Performance Evaluation*, 14(3–4), 1992.

[188] K. M. van Hee. *Information Systems Engineering: A Formal Approach.* Cambridge University Press, 1994.

[189] S. Velilla and M. Silva. The spy: A mechanism for safe implementation of highly concurrent systems. In *Real Time Programming 1988,* 15^{th} *IFAC/IFIP Workshop*, pages 95–102, Valencia, Spain, May 1988. Pergamon.

[190] M. Vernon, J. Zahorjan, and E. D. Lazowska. A comparison of performance Petri nets and queueing network models. In *3rd Int. Workshop on Modelling Techniques and Performance Evaluation*, Paris, France, March 1987.

[191] J. L. Villarroel, J. Martínez, and M. Silva. GRAMAN: A graphic system for manufacturing system design. In S. Tzafestas, A. Eisinberg, and L. Carotenuto, editors, *IMACS Symp. on System Modelling and Simulation*, pages 311–316. Elsevier, 1988.

[192] J. L. Villarroel and P. Muro. Using Petri net models at the coordination level for manufacturing systems control. *Robotics and Computer-Integrated Manufacturing*, 11(1):41–50, 1994.

[193] N. Viswanadham and Y. Narahari. Coloured Petri net models for automated manufacturing systems. In Silva and Murata [162], pages 1985–1990.

[194] N. Viswanadham and Y. Narahari. *Performance Modeling of Automated Manufacturing Systems.* Prentice-Hall, 1992.

[195] N. Viswanadham, Y. Narahari, and T. L. Johnson. Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models. *IEEE Trans. on Robotics and Automation*, 6(6):713–723, 1990.

[196] T. Watanabe, Y. Mizobata, and K. Onaga. Minimum initial marking problems of Petri nets. *Transactions of the IEICE*, E72(12):1390–1399, 1989.

[197] W. M. Wonham. Lecture notes on control of discrete-event systems. In *Supervisory Control of Discrete Event Systems.* Laboratoire d'Automatique de Grenoble, INPG, September 1995.

[198] C. M. Woodside, J. E. Neilson, D. C. Petriu, and S. Majumdar. The stochastic rendezvous network model for performance of synchronous client-server-like distributed software. *IEEE Trans. on Computers*, 44(1):20–34, 1995.

[199] K. Y. Xing, B. S. Hu, and H. X. Chen. Deadlock avoidance policy for Petri-net modeling of flexible manufacturing systems with shared resources. *IEEE Trans. on Automatic Control*, 41(2):289–294, 1996.

[200] M. C. Zhou, editor. *Petri Nets in Flexible and Agile Automation.* Kluwer Academic Publishers, 1995.

[201] M. C. Zhou and F. DiCesare. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems.* Kluwer Academic Publishers, 1993.

[202] M. C. Zhou, F. DiCesare, and D. L. Rudolph. Design and implementation of a Petri net based supervisor for a flexible manufacturing system. *Automatica*, 28(6):1199–1208, 1992.

[203] R. Zurawski and M. C. Zhou, editors, Special issue on Petri nets in manufacturing. *IEEE Trans. on Industrial Electronics*, 41(6), 1994.