Petri Nets and Industrial Applications: A Tutorial

Richard Zurawski and MengChu Zhou

Abstract—**This is a tutorial paper on Petri nets. Petri nets, as a graphical and mathematical tool, provide a uniform environment for modelling, formal analysis, and design of discrete event systems.**

The main objective of this paper is to introduce the fundamental concepts of Petri nets to the researchers and practitioners, both from academia and industry, who are involved in the work in the areas of modelling and analysis of industrial types of systems, as well as those who may potentially be involved in these areas.

The paper begins with an overview of applications of Petri nets, mostly industrial ones. Then, it proceeds with a description of Petri nets, properties, and analysis methods. The discussion of properties is put in the context of industrial applications. The analysis methods are illustrated using an example of a simple robotic assembly system. The performance analysis, using Petri nets, is discussed for deterministic and stochastic Petri nets. The presented techniques are illustrated by examples representing simple production systems. In addition, the paper introduces high-level Petri nets, fuzzy Petri nets, and temporal Petri nets. This is done in the context of application prospects. The paper also briefly discusses some of the reasons restricting the use of Petri nets, mostly, to academic institutions.

I. Introduction

THE growth in the complexity of modern industrial systems, such as production, process control, communication systems, etc., creates numerous problems for their developers. In the planning stage, one is confronted with increased capabilities of these systems due to the unique combination of hardware and software, which operate under a large number of constraints arising from the limited system resources. In view of the capital intensive and complex nature of modern industrial systems, the design and operation of these systems require modeling and analysis in order to select the optimal design alternative, and operational policy. It is well-known that flaws in the modeling process can substantially contribute to the development time and cost. The operational efficiency may be affected as well. Therefore special attention should be paid to the correctness of the models that are used at all planning levels.

Petri nets, as graphical and mathematical tools, provide a uniform environment for modeling, formal analysis, and design of discrete event systems. One of the major advantages of using Petri net models is that the same model is used for the analysis of behavioral properties and performance evaluation, as well as for systematic construction of discrete-event simulators and controllers. Petri nets were named after Carl A. Petri who created in 1962 a net-like mathematical tool for the study of communication with automata. Their further development was facilitated by the fact that Petri nets can be used to model properties such as process synchronization, asynchronous events, concurrent operations, and conflicts or resource sharing. These properties characterize discrete-event systems whose examples include industrial automated systems, communication systems, and computer-based systems. These, and other factors discussed in this paper, make Petri nets a promising tool and technology for application to Industrial Automation.

Petri nets as graphical tools provide a powerful communication medium between the user, typically requirements engineer, and the customer. Complex requirements specifications, instead of using ambiguous textual descriptions or mathematical notations difficult to understand by the customer, can be represented graphically using Petri nets. This combined with the existence of computer tools allowing for interactive graphical simulation of Petri nets, puts in hands of the development engineers a powerful tool assisting in the development process of complex systems.

As a mathematical tool, a Petri net model can be described by a set of linear algebraic equations, or other mathematical models reflecting the behavior of the system. This opens a possibility for the formal analysis of the model. This allows one to perform a formal check of the properties related to the behavior of the underlying system, e.g., precedence relations amongst events, concurrent operations, appropriate synchronization, freedom from deadlock, repetitive activities, and mutual exclusion of shared resources, to mention some. The simulation based model validation can only produce a limited set of states of the modeled system, and thus can only show presence (but not absence) of errors in the model, and its underlying requirements specification. The ability of Petri nets to verify the model formally is especially important for real-time safety-critical systems such as air-traffic control systems, rail-traffic control systems, nuclear reactor control systems, etc. Petri nets were used to model real-time fault tolerant and safety-critical systems in [11]–[12], [65]. Fault detection and in-process monitoring were modeled and analyzed in [27], [96], [101].

One of the most successful application areas of Petri nets has been modeling and analysis of communication protocols [14–15], [22], [28], [37], [42], [53], [84]. The work in this area can be dated back to the early 1970s. In the past few years, a number of approaches have been proposed which allow for the construction of Petri net models of protocols

Manuscript received December 12, 1993; revised July 22, 1994. Partial financial support was given by the Center for Manufacturing Systems at the New Jersey Institute of Technology.

R. Zurawski is with Laboratory for Robotics & Intelligent Systems, Swiburne University of Technology, Melbourne, Vic. 3122, Australia.

M. C. Zhou is with the Laboratory for Discrete Event Systems, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA.

IEEE Log Number 9405099.

from specifications written in a relatively skill-free languages [62], [99]. Methods were also proposed for transforming SDL [33], Lotos [9], and Estelle [106] based protocol specifications into Petri nets for performance and reliability analysis.

Petri nets have been used extensively to model and analyze manufacturing systems. In this area, Petri nets were used to represent simple production lines with buffers, machine shops, automotive production systems, flexible manufacturing systems, automated assembly lines, resource-sharing systems, and recently just-in-time and kanban manufacturing systems. Some of the most recent developments involving modeling and qualitative analysis were reported in [1], [5], [10], [110]–[113]. The deadlock avoidance was studied in [7], [52], [103].

The application of Petri nets to modeling sequence controllers is another success story. Programmable Logic Controllers are commonly used for the sequence control in automated systems. They are designed using ladder logic diagrams, which are known to be very difficult to debug and modify. Petri net based sequence controllers, on the other hand, are easy to design, implement, and maintain. In the early 80's, Hitachi Ltd. developed a Petri net based sequence controller [78] which was successfully used in real applications to control parts assembly system, and automatic warehouse load/unload system. The use of Petri nets, as reported, substantially reduced the development time compared with the traditional approach. Numerous approaches to the synthesis and implementation of Petri net based sequence controllers have been reported in the past few years [35], [41], [54], [79], [109]–[111].

Petri nets have been extensively used in software development. The work in this area focused on modeling and analysis of software systems using Petri nets [88]. The most mature developments involve the use of colored Petri nets. Colored Petri nets have been demonstrated in [57] to be a useful language for the design, specification, simulation, validation and implementation of large software systems. An integrated software development methodology based on hierarchical colored Petri nets was described in [82]. This approach allows for automatic translation of SADT diagrams into colored Petri nets for formal analysis, and for converting the nets into executable code. The design and analysis of Ada systems have also attracted a considerable attention [64], [77], [93].

Petri nets, as a mathematical tool, allow for the performance evaluation of the modeled systems. Both deterministic and stochastic performance measures can be evaluated by using a broad class of Petri net models incorporating in their definitions deterministic and/or probabilistic time functions. The performance evaluation can be conducted using either analytical techniques, based on solving the underlying (semi)-Markov processes, or discrete event simulation. The use of models which incorporate time functions having probabilistic distributions allows one to obtain production rates for manufacturing system models, throughput, delays, capacity for communication and microprocessor system models, as well as critical resource utilization and reliability measures for these and other systems. In recent years, this class of Petri net models has been extensively used to model and study analytically performance of multiprocessor systems [31], [50], [68], multiprocessor system buses [46], [59]–[60], DSP communication channels [48], parallel computer architectures [24], [105], as well as parallel distributed algorithms [6].

Another area of applications was communication networks. Work was conducted on Fiber Optics Local Area Networks such as Expressnet, Fastnet, D-Net, U-Net, Token Ring [67]. Fieldbuses, such as FIP and ISA-SP50, have attracted lots of attention in the last two years [13], [26], [58]. This is not surprising, since they are very important networks for factory automation. The interest steadily grows in modeling and evaluation of High Speed Networks, crucial for the successful development of Multimedia Systems [23], [25].

The performance of production systems, involving simple production lines, job shops, robotic assembly cells, flexible manufacturing systems, etc., was studied in [4], [19], [43], [49], [61], [71], [110]. When a state explosion problem arises, or the underlying stochastic models are not amenable for tractable mathematical analysis, simulation may be conducted for the analysis of both qualitative and quantitative properties [39], [74], [104]. The discrete-event simulation can be driven from the model, sometimes using complex algorithmic strategies representing real-time scheduling and control policies of production systems [70], [89].

Petri nets with time extensions, combined with heuristic search techniques, were used to model and study scheduling problems involving manufacturing systems [3], [63], [91], as well as robotic systems [90], [108]. The robotic assembly and trajectory planning using Petri nets were presented in [72].

Petri nets with time extensions were also used to model and analyze dynamics of continuous chemical processes [36]. The continuous time and discrete-event process control was modeled and analyzed in [17], [18], [47], [51], [107].

This brief overview of applications of Petri nets focused mainly on selected industrial areas. The references used were either the most representative in the area, or the most recent ones. The bibliography of Petri nets [83], published in 1991, contains 4099 entries dealing with Petri net theory and applications.

The main objective of this paper is to introduce the fundamental concepts of Petri nets to the researchers and practitioners, both from academia and industry, who are actively involved in the work in the areas of modeling and analysis of industrial type of systems, as well as those who may potentially become involved in these areas in the future. The presentation focuses on the ordinary Petri nets, although other types of Petri nets are also introduced in the context of the application driven developments. Additional tutorial material on Petri nets may be found in [38], [43], [76], [81], [86]. This paper is organized as follows. The Petri net description and definitions are presented in Section 2. This section, also, includes an example illustrating the use of Petri nets in the modeling of a simple multirobot assembly system. Some of the most fundamental properties of Petri nets, such as reachability, boundedness, conservativeness, and liveness, are discussed in Section 3. The analysis methods are presented in Section 4. The methods covered in this section are based on the coverability tree, and the incidence matrix and state equations. The two methods are, then, used

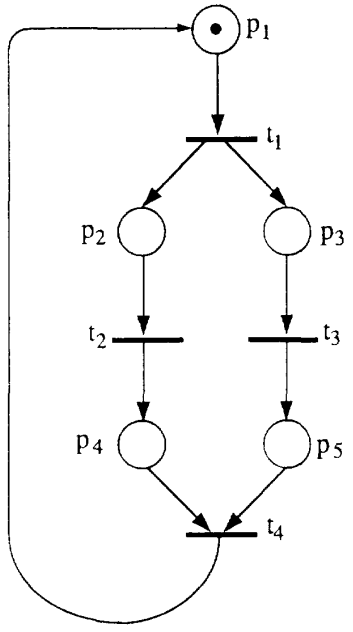

Fig. 1. Example of graphical representation of a Petri net.

to analyze the model of the multirobot assembly system. The performance analysis of Petri nets is presented in Section 5. Section 5 introduces the fundamental concepts instrumental in performance evaluation of timed, and stochastic timed Petri net models. The application prospects of Petri nets are discussed in Section 6, including the application driven Petri nets development.

II. Description of Petri Nets

A Petri net may be identified as a particular kind of bipartite directed graph populated by three types of objects. These objects are places, transitions, and directed arcs connecting places to transitions and transitions to places. Pictorially, places are depicted by circles and transitions as bars or boxes. A place is an input place to a transition if there exists a directed arc connecting this place to the transition. A place is an output place of a transition if there exists a directed arc connecting the transition to the place. In its simplest form, a Petri net may be represented by a transition together with its input and output places. This elementary net may be used to represent various aspects of the modeled systems. For instance, input (output) places may represent preconditions (postconditions), the transition an event. Input places may represent the availability of resources, the transition their utilization, output places the release of the resources. An example of a Petri net is shown in Fig. 1. This net consists of five places, represented by circles, four transitions, depicted by bars, and directed arcs connecting places to transitions and transitions to places. In this net, place p_1 is an input place of transition t_1. Places p_2, and p_3 are output places of transition t_1.

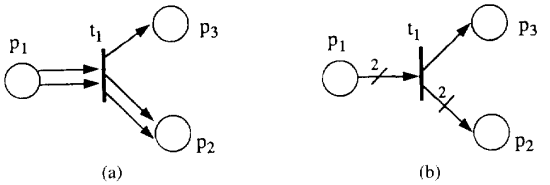

Fig. 2. (a) Multiple arcs. (b) Compact representation of multiple arcs.

In order to study dynamic behavior of the modeled system, in terms of its states and their changes, each place may potentially hold either none or a positive number of tokens, pictured by small solid dots, as shown in Fig. 1. The presence or absence of a token in a place can indicate whether a condition associated with this place is true or false, for instance. For a place representing the availability of resources, the number of tokens in this place indicates the number of available resources. At any given time instance, the distribution of tokens on places, called Petri net marking, defines the current state of the modeled system. A marking of a Petri net with m places is represented by an $(m \times 1)$ vector M, elements of which, denoted as $M(p)$, are nonnegative integers representing the number of tokens in the corresponding places. A Petri net containing tokens is called a marked Petri net. For example, in the Petri net model shown in Fig. 1, $M = (1, 0, 0, 0, 0)^T$.

Formally, a Petri net can be defined as follows:

PN $= (P, T, I. O, M_0)$; where

1. $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places,
2. $T = \{t_1, t_2. ...., t_n\}$ is a finite set of transitions, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$,
3. $I : (P \times T) \mapsto N$ is an input function that defines directed arcs from places to transitions, where N is a set of nonnegative integers,
4. $O : (P \times T) \mapsto N$ is an output function which defines directed arcs from transitions to places, and
5. $M_0 : P \mapsto N$ is the initial marking.

If $I(p, t) = k$ $(O(p, t) = k)$, then there exist k directed (parallel) arcs connecting place p to transition t (transition t to place p). If $I(p, t) = 0$ $(O(p, t) = 0)$, then there exist no directed arcs connecting p to t (t to p). Frequently, in the graphical representation, parallel arcs connecting a place (transition) to a transition (place) are represented by a single directed arc labeled with its multiplicity, or weight k. This compact representation of multiple arcs is shown in Fig. 2.

By changing distribution of tokens on places, which may reflect the occurrence of events or execution of operations, for instance, one can study dynamic behavior of the modeled system. The following rules are used to govern the flow of tokens.

Enabling Rule: A transition t is said to be enabled if each input place p of t contains at least the number of tokens equal to the weight of the directed arc connecting p to t.

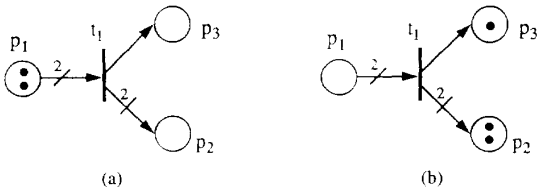

Fig. 3. (a) Transition t_1 enabled. (b) Enabled transition t_1 fires.

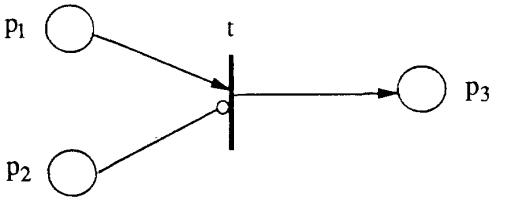

Fig. 4. Petri net with an inhibitor arc.

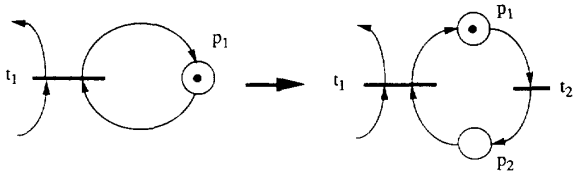

Fig. 5. Self-loop removal.

Firing Rule:

(a) An enabled transition t may or may not fire depending on the additional interpretation, and

(b) A firing of an enabled transition t removes from each input place p the number of tokens equal to the weight of the directed arc connecting p to t. It also deposits in each output place p the number of tokens equal to the weight of the directed arc connecting t to p.

The enabling and firing rules are illustrated in Fig. 3. In Fig. 3(a), transition t_1 is enabled as the input place p_1 of transition t_1 contains two tokens, and $I(p_1, t_1) = 2$. The firing of the enabled transition t_1 removes from the input place p_1 two tokens as $I(p_1, t_1) = 2$, and deposits one token in the output place p_3, $O(p_3, t_1) = 1$, and two tokens in the output place p_2, $O(p_2, t_1) = 2$. This is shown in Fig. 3(b).

The modeling power of Petri nets can be increased by adding the zero testing ability, i.e., the ability to test whether a place has no token. This is achieved by introducing an inhibitor arc. The inhibitor arc connects an input place to a transition, and is pictorially represented by an arc terminated with a small circle. A Petri net with an inhibitor arc is shown in Fig. 4. The presence of an inhibitor arc connecting an input place to a transition changes the transition enabling conditions. In the presence of the inhibitor arc, a transition is regarded as enabled if each input place, connected to the transition by a normal arc (an arc terminated with an arrow), contains at least the number of tokens equal to the weight of the arc, and no tokens are present on each input place connected to the transition by the inhibitor arc. The transition firing rules are the same as for normally connected places. The firing, however, does not change the marking in the inhibitor arc connected places.

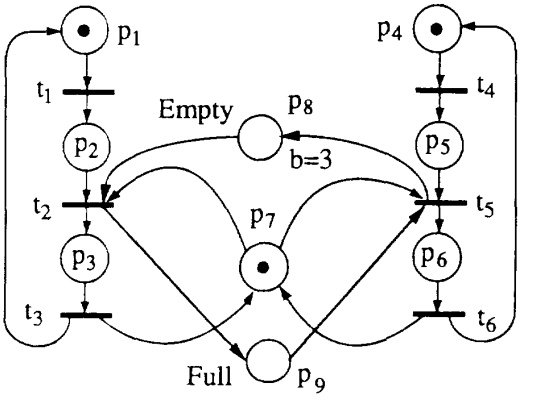

Fig. 6. Petri net model of a multirobot system.

TABLE I
INTERPRETATION OF PLACES AND TRANSITIONS OF THE PETRI NET MODEL OF THE MULTIROBOT ASSEMBLY SYSTEM

Place (with tokens)	Interpretation
$p_1(p_4)$	Robot R1 (R2) performs tasks outside the common workspace
$p_2(p_5)$	Robot R1(R2) waits for the access to the common workspace
$p_3(p_6)$	Robot R1 (R2) performs in the common workspace
p_7	mutual exclusion
$p_8(p_9)$	number of empty (full) positions in buffer
Transition	Interpretation
$t_1(t_4)$	Robot R1 (R2) requests access to the common workspace
$t_2(t_5)$	Robot R1(R2) enters the common workspace
$t_3(t_6)$	Robot R1 (R2) leaves the common workspace

A Petri net is said to be pure or self-loop free if no place is an input place to and output place of the same transition. A Petri net that contains self-loops can always be converted to a pure Petri net as shown in Fig. 5.

In order to illustrate how Petri nets can be used to model properties such as concurrent activities, synchronization, mutual exclusion etc., we consider a simple example of a multirobot system. This system is represented by a Petri net model shown in Fig. 6, and Table I. In this model, two robot arms perform pick-and-place operations accessing a common workspace at times to obtain or transfer parts. In order to avoid collision. it is assumed that only one robot can access the workspace at a time. In addition, it is assumed that the common workspace contains a buffer with a limited space for products. This could represent an operation of two robot arms servicing two different machining tools, with one robot arm transferring semiproducts from one machining tool to the buffer, and the other robot arm transferring semiproducts from the buffer to the other machining tool.

In this model, places p_1, p_2, p_3 and transitions t_1, t_2, t_3 model activities of robot arm $R1$. Places p_4, p_5, p_6 and transitions t_4, t_5, t_6 model activities of robot arm $R2$. Transitions t_1 and t_4 represent concurrent activities of $R1$ and $R2$. Either

of these transitions can fire before or after, or in parallel with the other one. The access to the common workspace requires synchronization of the activities of the arms in order to avoid collision. Only one robot arm can access the common workspace at a time. This synchronization is accomplished by the mutual exclusion mechanism implemented by a subnet involving places p_7, p_3, p_6 and transitions t_2, t_3, t_5, t_6. Firing transition t_2 disables t_5, assuming t_5 is enabled, and vice versa. Thus only one robot arm can access the common workspace at a time. In addition, it is assumed that the buffer space is "b." Thus, for instance, if p_8 is empty, then t_2 cannot be enabled. This prevents $R1$ from attempting to transfer to the buffer a part when there is no space in the buffer. Also, $R2$ cannot access the buffer if there is no part in the buffer, or place p_9 is empty.

III. Properties of Petri Nets

Petri nets as mathematical tools possess a number of properties. These properties, when interpreted in the context of the modeled system, allow the system designer to identify the presence or absence of the application domain specific functional properties of the system under design. Two types of properties can be distinguished: behavioral and structural properties. The behavioral properties are these which depend on the initial state, or marking, of a Petri net. The structural properties, on the other hand, do not depend on the initial marking of a Petri net. These properties depend on the topology, or net structure, of a Petri net. In this section, we provide an overview of some of the most important, from practical point of view, behavioral properties. The focus on the behavioral properties is dictated by the space limitations of this tutorial. An extensive description of the structural properties, and the analysis methods can be found in [76]. The behavioral properties discussed in this section are reachability, boundedness, conservativeness, liveness, reversibility and home state. Descriptions of other properties such as coverability, persistence, synchronic distance, and fairness can also be found in [76], [86].

A. Reachability

An important issue in designing distributed systems is whether a system can reach a specific state, or exhibit a particular functional behavior. In general, the question is whether the system modeled with Petri nets exhibits all desirable properties, as specified in the requirements specification, and no undesirable ones.

In order to find out whether the modeled system can reach a specific state as a result of a required functional behavior, it is necessary to find such a sequence of firings of transitions which would result in transforming a marking M_0 to M_i, where M_i represents the specific state, and the sequence of firings represents the required functional behavior. It should be noted that real systems may reach a given state as a result of exhibiting different permissible patterns of functional behavior. In a Petri net model, this should be reflected in the existence of specific sequences of transitions firings, representing the required functional behavior, which would transform a marking M_0 to the required marking M_i. The existence in the Petri net model of additional sequences of transitions firings which transform M_0 to M_i indicates that the Petri net model may not be reflecting exactly the structure and dynamics of the underlying system. This may also indicate the presence of unanticipated facets of the functional behavior of the real system, provided that the Petri net model accurately reflects the underlying system requirements specification. A marking M_i is said to be reachable from a marking M_0 if there exists a sequence of transitions firings which transforms a marking M_0 to M_i. A marking M_1 is said to be immediately reachable from a marking M_0 if a firing of an enabled transition in M_0 results in marking M_1. For instance, in the Petri net model of the multirobot assembly system shown in Fig. 6, the state in which robot arm $R1$ performs tasks in the common workspace, with robot arm $R2$ waiting outside, is represented by the marking vector $M_i = (0,0,1,0,1,0,0,2,1)^T$. M_i can be reached from the initial marking M_0, where $M_0 = (1,0,0,1,0,0,1,3,0)^T$, by the following sequence of transitions firings—$t_1t_2t_4$. The marking $M_1 = (0,1,0,1,0,0,1,3,0)^T$, which represents the state of the system in which robot arm $R1$ waits for the access to the common workspace and robot arm $R2$ performs tasks outside the commom workspace, is immediately reachable from the initial marking M_0 when transition t_1 fires. It should be noted that in M_0 transitions t_1, and t_4 are both enabled. The set of all possible markings reachable from M_0 is called the reachability set, and denoted by $R(M_0)$. This will be discussed in more detail in Section 4.1. The set of all possible firing sequences from M_0 is denoted by $L(M_0)$. Thus the problem of identifying the existence of a specific state M_i, the system can take on, can be redefined as the problem of finding if $M_i \in R(M_0)$.

B. Boundedness and Safeness

Places are frequently used to represent information storage areas in communication and computer systems, product and tool storage areas in manufacturing systems, etc. It is important to be able to determine whether proposed control strategies prevent from the overflows of these storage areas. The information storage areas can hold, without corruption, only a restricted number of pieces of data. In manufacturing systems, attempts to store more tools, for instance, in the tool storage area may result in the equipment damage. The Petri net property which helps to identify in the modeled system the existence of overflows is the concept of boundedness. A Petri net is said to be k-bounded if the number of tokens in any place p, where $p \in P$, is always less or equal to k (k is a nonnegative integer number) for every marking M reachable from the initial marking $M_0, M \in R(M_0)$. A Petri net is safe if it is 1-bounded. A Petri net shown in Fig. 7 is safe. In this net no place can contain more then one token. An example of a Petri net which is unbounded is shown in Fig. 8. This net is unbounded because place p_4 can hold an arbitrarily large number of tokens.

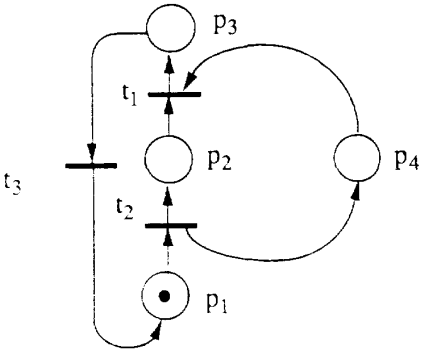

Fig. 7. Petri net that is safe.

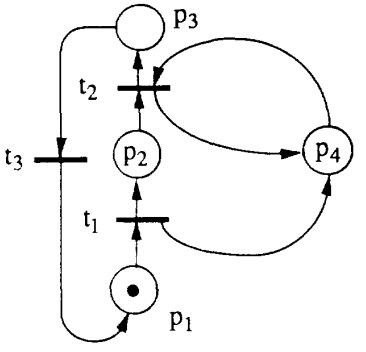

Fig. 8. Petri net that is unbounded.

C. Conservativeness

In real systems, the number of resources in use is typically restricted by the financial as well as other constraints. If tokens are used to represent resources, the number of which in a system is typically fixed, then the number of tokens in a Petri net model of this system should remain unchanged irrespective of the marking the net takes on. This follows from the fact that resources are neither created nor destroyed, unless there is a provision for this to happen. For instance, a broken tool may be removed from the manufacturing cell, thus reducing the number of tools available by one.

A Petri net is conservative if the number of tokens is conserved. From the net structural point of view, this can only happen if the number of input arcs to each transition is equal to the number of output arcs. However, in real systems resources are frequently combined together so that certain tasks can be executed, then separated after the task is completed. For instance, in a flexible manufacturing system an automatic guided vehicle collects a pallet carrying products from a machining cell, and subsequently delivers it to the unload station where the vehicle and pallet are separated. This scenario is illustrated in Fig. 9. Transition t_1 models loading a pallet onto a vehicle; transition t_2 represents the pallet being delivered to the unload station and subsequently removed from the vehicle. Although the number of tokens in the net changes from two to one when t_1 fires, and then back to two tokens when t_2 fires, the number of resources in the system does not change. In order to overcome this problem, weights may be

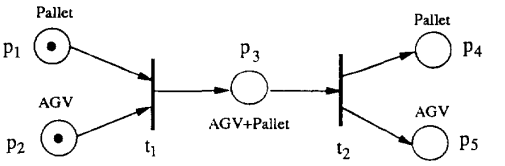

Fig. 9. Petri net conservative with respect to $w = [1, 1, 2, 1, 1]$.

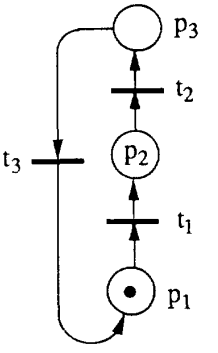

Fig. 10. Petri net that is strictly conservative.

associated with places allowing for the weighted sum of tokens in a net to be constant. A Petri net is said to be conservative if there exists a vector $w, w = [w_1, w_2, ..., w_m]$, where m is the number of places, and $w(p) > 0$ for each $p \in P$, such that the weighted sum of tokens remains the same for each marking M reachable from the initial marking M_0. A Petri net is said to be strictly conservative if all entries of vector w are unity. The Petri net shown in Fig. 9 is conservative with respect to vector $w = [1, 1, 2, 1, 1]$ as the weighted sum of tokens in each marking is two. An example of a Petri net which is not conservative is shown in Fig. 8; place p_4 can hold an arbitrarily large number of tokens. If a Petri net is conservative with respect to a vector with all elements equal to one, then the net is said to be strictly conservative. An example of a Petri net which is strictly conservative is shown in Fig. 10.

D. Liveness

The concept of liveness is closely related to the deadlock situation, which has been studied extensively in the context of operating systems. Coffman *et al.* [34] showed that four conditions must hold for a deadlock to occur. These four conditions are:

1. Mutual exclusion: a resource is either available or allocated to a process which has an exclusive access to this resource.
2. Hold and wait: a process is allowed to hold a resource(s) while requesting more resources.
3. No preemption: a resource(s) allocated to a process cannot be removed from the process, until it is released by the process itself.
4. Circular wait: two or more processes are arranged in a chain in which each process waits for resources held by the process next in the chain.

For instance, in a flexible manufacturing system, a deadlock occurs when the input/output buffer of a machining tool holds

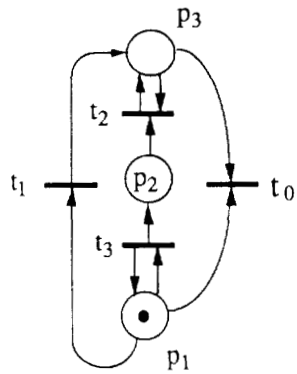

Fig. 11. Petri net with different levels of liveness of transitions.

a pallet with already machined products, and another pallet with products to be machined has been delivered to the buffer. Assuming that the buffer can hold one pallet only at a time, and an automated guided vehicle (AGV), for instance, has a space for one pallet, a deadlock occurs. The pallet with machined parts cannot be moved from the buffer to the AGV. The pallet with parts to be machined cannot be moved from the AGV to the buffer. In this example, all four conditions hold, with the buffer and AGV space for pallets regarded as resources. Unless there is a provision in the control software for deadlock detection and recovery, a deadlock situation, initially confined to a small subsystem, may propagate to affect a large portion of a system. This frequently results in a complete standstill of a system. A Petri net modeling a deadlock free system must be live. This implies that for all markings M, which are reachable from the initial marking M_0, it is ultimately possible to fire any transition in the net by progressing through some firing sequence. The Petri net shown in Fig. 10 is live. This requirement, however, might be too strict to represent some real systems or scenarios which exhibit deadlock-free behavior. For instance, the initialization of a system can be modeled by a transition (or transitions) which fires a finite number of times. After initialization, the system may exhibit a deadlock free behavior, although the Petri net representing this system is no longer live as specified above. For this reason, different levels of liveness for transition t, and marking M_0, were introduced. A transition t in a Petri net is said to be:

- $L0$-live (or dead) if there is no firing sequence in $L(M_0)$ in which t can fire,
- $L1$-live (potentially firable) if t can be fired at least once in some firing sequence in $L(M_0)$,
- $L2$-live if t can be fired at least k times in some firing sequence in $L(M_0)$ given any positive integer k,
- $L3$-live if t can be fired infinitely often in some firing sequence in $L(M_0)$, and
- $L4$-live (or live) if t is $L1$-live (potentially firable) in every marking in $R(M_0)$.

Following this classification, a Petri net is said to be Li-live, for marking M_0, if every transition in the net is Li-live. Different levels of liveness of transitions are shown in Fig. 11.

In this example, transitions t_0, t_1, t_2, and t_3 are $L0$, $L1$, $L2$, and $L3$-live, respectively, and strictly.

E. Reversibility and Home State

An important issue in the operation of real systems, such as manufacturing systems, process control systems, etc., is the ability of these systems for an error recovery. These systems are required to return from the failure states to the preceding correct states. This requirement is closely related to the reversibility and home state properties of a Petri net. A Petri net, for the initial marking M_0, is said to be reversible if for each marking M in $R(M_0)$, M_0 is reachable from M. The home state property is less restrictive, and more practical, then the reversibility property of a Petri net. A Petri net state M_i is said to be a home state if for each marking M in $R(M_0)$, M_i is reachable from M. The Petri net shown in Fig. 7 is reversible. The Petri net shown in Fig. 8 is nonreversible.

IV. Analysis Methods

In the previous section, we defined a number of properties of Petri nets which are useful for analyzing modeled systems. An important issue to be considered during analysis is whether there exists one-to-one functional correspondence between the Petri net model and the original requirements specification; typically expressed in an informal way. The construction of Petri net models from informal requirements specifications is not a trivial task, which requires a great deal of modeling experience, as well as the knowledge of the techniques assisting in the model construction. As a result, a Petri net model may differ considerably from its original specification. This is especially true when large Petri net models of complex systems are involved. The existence of the one-to-one functional correspondence between an original requirements specification and its Petri net representation allows projection of the analysis results, obtained for the Petri net model, onto the original description. This provides feedback to the customers which can, in many instances, help the customers clarify their perception of the system. Another important issue to be addressed during the analysis stage is the completeness of the requirements specification. In most cases, the requirements specification defines the external functional behavior of a system. This is typically expressed in terms of the system input output relationships. Inputs are generated by the environment of the system. Outputs are responses of the system to these inputs. If some inputs, generated by the environment of the system, are not included in the requirements specification, then the system will be unable to respond to these inputs properly when they occur during the system normal operation. The completeness of the requirements is especially important in the case of safety-critical systems. In these systems, the incompleteness of the requirements specification may lead to catastrophic events to occur in the environment of the system. For instance, the occurrence of unanticipated states in the operation of a nuclear reactor may result in the failure of the control system to respond to them properly, or at all, thus potentially leading to the reactor system failure. The consistency of the requirement specification is another issue to be considered during analysis. The inconsistency occurs when for a given permissible, temporal combination of inputs, a requirements

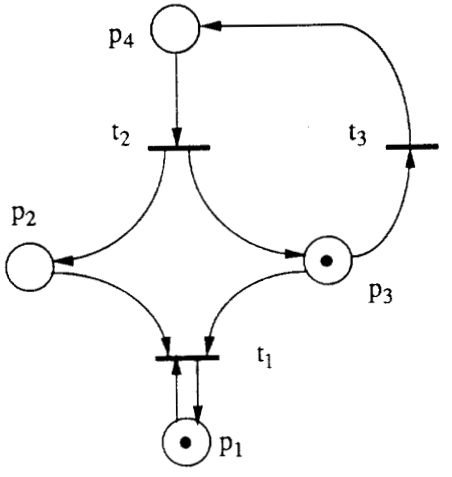

Fig. 12. A Petri net model.

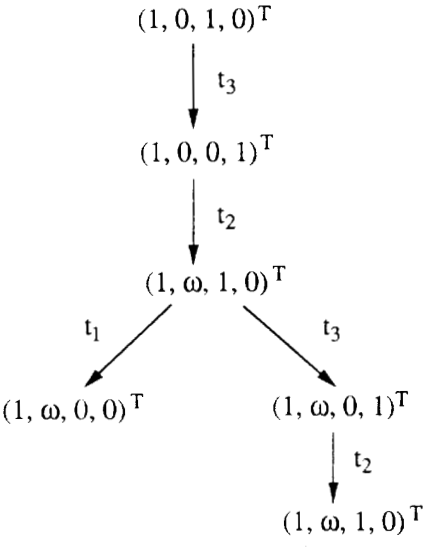

Fig. 13. The coverability tree of the Petri net model shown in Fig. 9.

specification allows for two or more different permissible temporal combinations of outputs. It is mainly due to a vague, incomplete, and frequently incorrect perception of the system functionality. In this paper, we are going to present an overview of two fundamental methods of analysis. One is based on the reachability tree, and the other on the matrix equation representation of a net. In addition to the two methods, a number of techniques were proposed to assist in the analysis of Petri net models. These approaches allow for a systematic transformation of a Petri net, by reducing the number of places and transitions in a net, and at the same time preserving the properties such as boundedness, conservativeness, liveness, etc. Smaller nets are easier to analyze. Some of these techniques were discussed in [76].

A. The Coverability Tree

This approach is based on the enumeration of all possible markings reachable from the initial marking M_0. Starting with an initial marking M_0, one can construct the reachability set by firing all possible transitions enabled in all possible markings reachable from the initial marking M_0. In the reachability tree, each node is labeled with a marking; arcs are labeled with transitions. The root node of the tree is labeled with an initial marking M_0. The reachability set becomes unbounded for either of two reasons: The existence of duplicate markings, and a net is unbounded. In order to prevent the reachability tree from growing indefinitely large, two steps need to be taken when a tree is constructed. The first step involves eliminating duplicate markings. If on the path from the initial marking M_0 to a current marking M there is a marking M', which is identical to the marking M, then the marking M, as a duplicate marking, becomes a terminal node. The occurrence of a duplicate marking implies that all possible markings reachable from M have been already added to the tree. For unbounded nets, in order to keep the tree finite, the symbol ω is introduced. The symbol ω can be thought of as the infinity. Thus, for any integer n, $\omega + n = \omega$, $\omega - n = \omega$, $n < \omega$. In this case, if on the path from the initial marking M_0 to a current marking M there is a marking M', with its entries less or equal to the corresponding entries in the marking M, then the entries of M, which are strictly greater than the corresponding entries of M', should be replaced by the symbol ω. In some paths the existence of markings with the corresponding entries equal or increasing (as we move away from the root node) indicates that the firing sequence which transforms M' to M can be repeated indefinitely. Each time this sequence is repeated, the number of tokens on places labeled by the symbol ω increases. The coverability tree is constructed according to the following algorithm:

1.0) Let the initial marking M_0 be the root of the tree and tag it "new."
2.0) While "new" markings exist do the following:
3.0) Select a "new" marking M.
3.1) If M is identical to another marking in the tree, then tag M "old," and go to another "new" marking.
3.2) If no transitions are enabled in M, tag M "terminal."
4.0) For every transition t enabled in marking M do the following:
4.1) Obtain the marking M' which results from firing t in M.
4.2) If on the path from the root to M, there exists a marking M'' such that $M'(p) \geq M''(p)$ for each place p, and $M' \neq M''$, then replace $M'(p)$ by ω for each p. wherever $M'(p) > M''(p)$.
4.3) Introduce M' as a node, draw an arc from M to M' labeled t, and tag M' "new."

The following example will illustrate the approach. Consider the net shown in Fig. 12, and its coverability tree in Fig. 13. For the given initial marking, the root node is $M_0 = (1, 0, 1, 0)^T$. In this marking, transition t_3 is enabled.

When t_3 fires a new marking is obtained: $M_1 = (1,0,0,1)^T$. This is a "new" marking in which transition t_2 is enabled. Firing t_2 in M_1 results in $M_2 = (1,1,1,0)^T$. Since $M_2 = (1,1,1,0)^T \geq M_0 = (1,0,1,0)^T$, the second component should be replaced by the symbol ω. This reflects the fact that the firing sequence t_3t_2 may be repeated arbitrarily large number of times. In marking $M_2 = (1,\omega,1,0)^T$ two transitions are enabled: transition t_1 and transition t_3. Firing t_1 results in marking $M_3 = (1,\omega,0,0)^T$, which is a "terminal" node. Firing t_3 results in a "new" marking $M_4 = (1,\omega,0,1)^T$, which enables transition t_2. Firing t_2 in M_4 results in an "old" node: $M_5 = (1,\omega,1,0)^T$ which is identical to M_2.

A number of properties can be studied by using the coverability tree. For instance, if any node in the tree contains the symbol ω, then the net is unbounded since the symbol ω can become arbitrarily large. Otherwise, the net is bounded. If each node of the tree contains only zeros and ones, then the net is safe. A transition is dead if it does not appear as an arc label in the tree. If a marking M is reachable from a marking M_0, then there exists a node M', such that $M \leq M'$. However, since the symbol ω can become arbitrarily large, certain problems, such as coverability and liveness, cannot be solved by studying the coverability tree only. For a bounded Petri net, the coverability tree contains, as nodes, all possible markings reachable from the initial marking M_0. In this case, the coverability tree is called the reachability tree. For a reachability tree any analysis question can be solved by inspection.

B. The Incidence Matrix and State Equation

An alternative approach to the representation and analysis of Petri nets is based on matrix equations. In this approach matrix equations are used to represent dynamic behavior of Petri nets. The fundamental to this approach is the incidence matrix which defines all possible interconnections between places and transitions in a Petri net. The incidence matrix of a pure Petri net is an integer $n \times m$ matrix A, where n is the number of transitions, and m is the number of places. The entries of the incidence matrix are defined as follows: $a_{ij} = a_{ij}^+ - a_{ij}^-$, where a_{ij}^+ is equal to the number of arcs connecting transition t_i to its output place p_j ($a_{ij}^+ = O(p_j,t_i)$), and a_{ij}^- is equal to the number of arcs connecting transition t_i to its input place p_j ($a_{ij}^- = I(p_j,t_i)$). When transition t_i fires, a_{ij}^+ represents the number of tokens deposited on its output place; p_j, a_{ij}^- represents the number of tokens removed from its input place; p_j, a_{ij} represents the change in the number of tokens in place p_j. Therefore, transition t_i is said to be enabled in marking M if

$$a_{ij}^- \leq M(p_j), i = 1,2,\ldots,m.$$

For Petri nets with self-loops, $a_{ij} = 0$ for a place p_j and transition t_i which belong to a self-loop. For this reason, in order to make sure that the incidence matrix properly reflects the structure of a Petri net, the net is assumed to be pure, or made pure by introducing two additional places (see Fig. 5). The state equation for a Petri net represents a change in the distribution of tokens on places (marking) as a result of a transition firing. This equation is defined as follows:

$$M_k = M_{k-1} + A^T u_k, k = 1,2,\ldots$$

M_k is an $m \times 1$ column vector representing a marking M_k immediately reachable from a marking M_{k-1} after firing transition t_i. The k-th firing vector u_k, an $n \times 1$ column vector, has only one nonzero entry. This entry, a 1 in the i-th position, represents a transition t_i firing in the k-th firing of the net firing sequence starting with an initial marking M_0. This entry corresponds to the i-th row of the incidence matrix A which represents a change of a marking as a result of a firing transition t_i. The matrix equation is useful in studying the reachability problem. However, this issue is outside the scope of this tutorial. For details see [76], [81], [86].

Two concepts related to the incidence matrix are especially useful in studing properties of Petri net models. They are T-invariant, and P-invariant.

An integer solution x of $A^T x = 0$ is called a T-invariant. The nonzero entries in a T-invariant represent the firing counts of the corresponding transitions which belong to a firing sequence transforming a marking M_0 back to M_0. Although a T-invariant states the transitions comprising the firing sequence transforming a marking M_0 into M_0, and the number of times these transitions appear in this sequence, it does not specify the order of transitions firings.

An integer solution y of $Ay = 0$ is called a P-invariant. The P-invariants can be explained intuitively in the following way. The nonzero entries in a P-invariant represent weights associated with the corresponding places so that the weighted sum of tokens on these places is constant for all markings reachable from an initial marking.

The subset of places (transitions) corresponding to the nonzero entries of a T-invariant (P-invariant) is called the support of an invariant, and denoted by $||x||$ ($||y||$). A support is said to be minimal if no proper nonempty subset of the support is also a support.

C. An Example

In this section, we demonstrate how the coverability tree and invariant based techniques can be used to analyze the Petri net model of the multirobot system which is shown in Fig. 6. Without losing the discussion generality, we assume $b = 1$. The coverability tree, in this case a reachability tree, is shown in Fig. 14. The incidence matrix of this net is shown in Fig. 15.

The P-invariants obtained for this net are as follows:

$$y_1 = (1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0)^T$$
$$y_2 = (0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0)^T$$
$$y_3 = (0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0)^T$$
$$y_4 = (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1)^T.$$

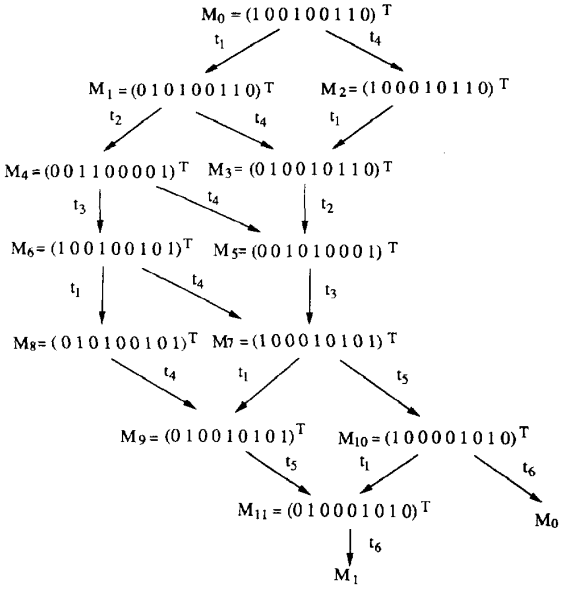

Fig. 14. Coverability tree of the Petri net model shown in Fig. 3.

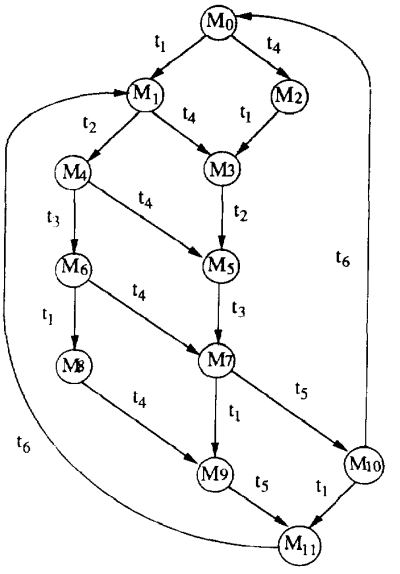

Fig. 16. Reachability graph of the Petri net shown in Fig. 3.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
t_1	-1	1	0	0	0	0	0	0	0
t_2	0	-1	1	0	0	0	-1	-1	1
t_3	1	0	-1	0	0	0	1	0	0
t_4	0	0	0	-1	1	0	0	0	0
t_5	0	0	0	0	-1	1	-1	1	-1
t_6	0	0	0	1	0	-1	1	0	0

Fig. 15. Incidence matrix of the Petri net model of the multirobot assembly cell.

The following are the corresponding invariant supports:

$$\|y_1\| = \{p_1, p_2, p_3\}$$
$$\|y_2\| = \{p_4, p_5, p_6\}$$
$$\|y_3\| = \{p_3, p_6, p_7\}$$
$$\|y_4\| = \{p_8, p_9\}.$$

Boundedness and Safeness: The Petri net shown in Fig. 6 is bounded. This is evident from the reachability tree; no marking reachable from the initial marking M_0 contains the ω symbol. In addition, since for each marking, no entry is greater than one, the net is safe. These properties can be also easily established using P-invariants. Since each place in the net belongs to some invariant support, and the net starts from a bounded initial marking, the net is bounded. In addition, since the token count in each invariant support in the initial marking is one, the net is also safe. Two properties related to the operation of the actual system can be deduced from the boundedness property of the Petri net model. There is no buffer overflow, no provision for $R1$ to access the buffer area when it is full. Also, there is no buffer underflow, no provision for $R2$ to access the buffer area when it is empty. When using the reachability tree, these properties follow from the net safeness. The entries in each marking, which represent the number of tokens in places p_8 and p_9, are either zero or one. Using invariants: The invariant support $\|y_4\|$ covers places p_8, and p_9. Since the token content in $\|y_4\|$ in the initial marking is one, there is only one token either on p_8, or p_9 at a time. Therefore there is neither buffer overflow nor underflow.

Conservativeness: The Petri net shown in Fig. 6 is conservative. From the reachability tree, the net is conservative with respect to vector $w = [1, 1, 2, 1, 1, 2, 1, 1, 1]$. The weighted sum of tokens remains the same for each marking reachable from the initial marking, and equals four. Using invariants: The token content in each invariant support in the initial marking is one. The invariant supports $\|y_1\|$, $\|y_2\|$, and $\|y_4\|$ are mutually exclusive. The invariant supports $\|y_1\|$ and $\|y_3\|$ contain place p_3 as a common element. The invariant supports $\|y_2\|$ and $\|y_3\|$ contain place p_6 as a common element. Thus the weight of places p_3 and p_6 should be two for the net to be conservative. The implication of this property is that the number of robot arms operating in the assembly system is two and does not change. Also, the number of the space resources in the buffer area is one and does not change.

Liveness: The Petri net shown in Fig. 6 is live; all transitions are live.

Fig. 16 shows a reachability graph of the Petri net of Fig. 6. The reachability graph shown in Fig. 16 is a directed graph consisting of a set of nodes, and a set of directed arcs. The set of nodes represents all distinct labeled nodes in the reachability tree. The set of directed arcs, where each arc is labeled with a transition, represents all possible transitions between all distinct labeled nodes in the reachability tree.

By inspection, the net is $L4$-live, since for any marking reachable from making M_0, it is possible to ultimately fire any transition by executing some firing sequence. The invariants could be used to demonstrate "manually" that the net is live.

However, for the net of this size, this would be a tedious procedure. As the net is live, the system cannot come to a standstill where no operation is possible.

Reversibility: The Petri net shown in Fig. 6 is reversible. Also by inspection, using the reachability graph, M_0 is reachable from any marking $M \in R(M_0)$.

V. Performance Analysis

The development of man-made systems requires that both functional and performance requirements are met. Depending on the development stage of a system, the knowledge of either an approximate or exact (or both) performance may be required. For example, at the design stage, the approximate performance of the alternative design models is required in order to eliminate these proposals which are highly unlikely to meet the performance requirements when fully developed and implemented. Analytical techniques play an important role at this stage. They allow the designer to obtain the required performance measures, involving a relatively small time investment needed for the model construction and its solution. The selected design alternatives are then refined by increasing the level of details in order to include in the model the actual operational policies and time characteristics. As a result, the model complexity or the presence of heuristic algorithms may prohibit the use of the analytical techniques. The discrete-event simulation is, then, the only viable alternative for the performance evaluation. Although, it is an expensive and time consuming technique.

The ordinary Petri nets do not include any concept of time. With this class of nets, it is possible only to describe the logical structure of the modeled system, but not its time evolution. Responding to the need for the temporal performance analysis of discrete-event systems, time has been introduced into Petri nets in a variety of ways. In this section two fundamental types of timed Petri nets are discussed in the context of the performance evaluation. These are deterministic timed Petri nets, and stochastic ones. An example of a simple production systems is used to illustrate the basic solution techniques for these classes of nets. The focus of the discussion in this section is on the analytical performance evaluation of Petri net models. This choice was motivated mainly by the recognition of the importance of using the analytical techniques for performance evaluation.

A. Deterministic Timed Petri Nets

Time may be associated with transitions (timed transition Petri net-TTPN). TTPN's can be classified into three-phase firing TTPN's, and atomic firing TTPN's. In the former class, tokens are removed from the input places when a transition becomes enabled. The transition fires after a certain period of time (time delay), depositing tokens on the output places. In the latter class, tokens remain on the input places of a transition which is enabled. After a certain period of time (time delay), the transition fires removing tokens from the input places, and deposits tokens on the output places.

The presence of the conflict structures (a structure involving a place having two, or more output transitions) in a Petri net, execution of which leads to conflicts (in the conflict structure only one transition can fire, others become disabled), requires a conflict resolution mechanism to be introduced. Since this mechanism is, typically, based on a probabilistic function, the net becomes stochastic. For this very reason, the use of the deterministic timed Petri nets for the performance evaluation, as reported to date, has been restricted to the choice-free or conflict-free nets, which can be further modeled as marked graphs, or event-graphs.

A marked graph is a PN (P, T, I, O, M_0) such that $\forall p \in P, t \in T$, $I(p,t) \leq 1$, $O(p,t) \leq 1$, and given any $p \in P$, $|\{t \in T : O(p,t) = 1\}| = 1$, and $|\{t \in T : I(p,t) = 1\}| = 1$.

In a marked graph, each place has exactly one input transition and exactly one output transition. Thus no conflict is possible. The marked graph models are suitable to represent and study performance of the asynchronous concurrent systems which do not involve conflicts [85], [95]. This class of nets has been extensively used to model and analyze performance of the industrial automated systems such as job-shop production systems, robotic assembly cells, and flexible manufacturing cells [43], [49], [110].

A timed marked graph is a marked graph $(P, T, I, O, M_0, \gamma, \tau)$, where γ: $P \mapsto R^+$ is the delay function, R^+ is the set of nonnegative real numbers, and $\tau : T \mapsto R^+$ is the firing time function.

In a timed marked graph, deterministic time delays are associated with places and transitions. A token in a place, with an associated delay time, can be available or not-available to enable its output transition. The previous firing rules for ordinary PN's are expanded to include the following:

1. At any time instance, a transition t becomes enabled if each input place p of t contains at least one available token,
2. Once transition t is enabled, it starts firing by removing one token from each input place. It completes firing after the time delay $\tau(t)$, and deposits one token into each output place p. These tokens become available after time $\gamma(p)$.

A marked graph is strongly connected if there is a directed path from any node to any other node, where a node can be either a transition or place.

An elementary path is a sequence of nodes: $x_1 x_2 \cdots x_n$, $n \geq 1$, such that there is an arc from x_i to x_{i+1}, where $1 \leq i < n$. If $n > 1$, $x_i = x_j$ implies that $i = j, 1 \leq i, j \leq n$.

An elementary loop, or circuit, is a sequence of nodes: $x_1 x_2 \cdots x_n$, $n > 1$, such that $x_i = x_j$, where $1 \leq i < j \leq n$, implies that $i = 1$ and $j = n$.

The fundamental approach to studying timed marked graphs is based on two concepts: The total time delay in a loop, and the total number of of tokens in a loop. A strongly-connected marked graph consists of a number of elementary loops. The total time delay in a loop is obtained as a sum of delays introduced by all transition and places comprising the loop. The total number of tokens in a loop is obtained as a sum of tokens present in the places which belong to the

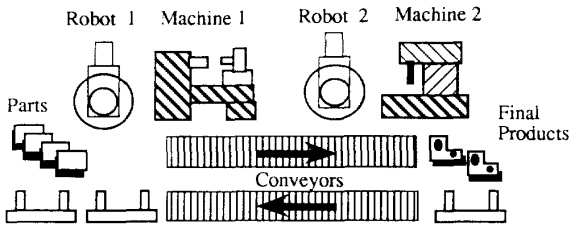

Fig. 17. A production line consisting of two machines, two robots, and two conveyers.

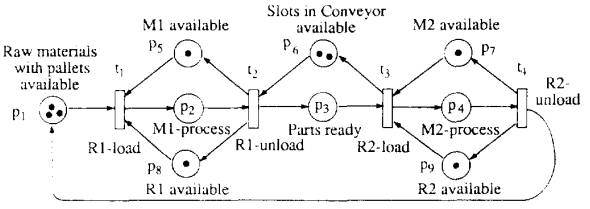

Fig. 18. Deterministic timed Petri net model for the production line.

loop. Then, the minimum cycle time of the marked graph is [85], [95]:

$$\pi = \max_i \{D_i/N_i\}$$

where D_i is the time delay associated with the i-th loop, N_i is the total number of tokens present in the places of the i-th loop. The ratio D_i/N_i is called the cycle time of loop i.

The upper and lower cycle times can be derived for the timed (deterministic or random) marked graphs using the methods proposed in [20] and [43]. Linear programming was demonstrated to be a computationally efficient method for computing the cycle times [20], [49], [75].

An example of a simple production line will be used to demonstrate the method for obtaining cycle times of a marked graph.

The production line consists of two machining tools ($M1$ and $M2$), two robot arms, and two conveyors. Each machining tool is serviced by a dedicated robot arm which performs load and unload tasks. One conveyor is used to transport workpieces, a maximum of two at a time. The other conveyor is used to transport empty pallets. There are three pallets available in the system. The arrangement is shown in Fig. 17. Each workpiece is machined on $M1$ and $M2$, in this order. The machining on $M1$ takes 10 time units. The machining on $M2$ takes 16 time units. The load and unload tasks require 1 time unit. The conveyors transportation times are negligible.

The deterministic timed Petri net model of this system is shown in Fig. 18. Table II provides a description of the places and transitions involved, as well as the associated time delays. The initial marking of the net is $(300012111)^T$. The Petri net model contains four loops. The time delays associated with these loops, as well as their token contents, are shown in Table III. Then, the minimum cycle time is 18 time units, as determined by the loops $t_3p_4t_4p_7$(or p_9)t_3. This means that it takes a minimum of 18 time units to transform a raw workpiece into a final product.

TABLE II
INTERPRETATION AND DELAYS OF PLACES AND TRANSITIONS IN FIG. 18.

Place	Interpretation	Time Delay
p_1	Workpieces and pallets available	0
p_2	M1 processing a workpiece	10
p_3	Workpiece available for processing at M2	0
p_4	M2 processing a workpiece	16
p_5	M1 available	0
p_6	Conveyor slots available	0
p_7	M2 available	0
p_8	R1 available	0
p_9	R2 available	0
Transition	Interpretation	Time Delay
t_1	R1 moves a workpiece to M1	1
t_2	R1 moves a workpiece from M1 to Conveyor	1
t_3	R2 moves a workpiece to M2	1
t_4	R2 removes a final workpiece from M2	1

TABLE III
LOOPS AND THEIR DELAYS, TOKEN SUMS, AND CYCLE TIMES FOR FIG. 18.

Loop	Loop Delay	Token Sum	Cycle Time
$t_1p_2t_2p_3t_3p_4t_4p_1t_1$	30	3	10
$t_1p_2t_2p_6$(or p_8)t_1	12	1	12
$t_2p_3t_3p_6t_2$	2	2	1
$t_3p_4t_4p_7$(or p_9)t_3	18	1	18

It should be noted that the increase in the number of available pallets is not going to make the minimum cycle time of the system any shorter. This is because of the time involved in machining at $M2$. Thus, Petri net analysis can also help decide the appropriate number of pallets and/or fixtures.

B. Stochastic Timed Petri Nets

When time delays are modeled as random variables, or probabilistic distributions are added to the deterministic timed Petri net models for the conflict resolution, stochastic timed Petri net models are yielded. In such models, it has become a convention to associate time delays with the transitions only. When the random variables are of general distribution or both deterministic and random variables are involved, the resulting net models cannot be solved analytically for general cases. Thus simulation or approximation methods are required. The stochastic timed Petri nets in which the time delay for each transition is assumed to be stochastic and exponentially distributed are called stochastic Petri nets (SPN) [73]. The SPN models which allow for immediate transitions, i.e., with zero time delay, are called generalized SPN (GSPN) [68], [69]. Both models, including extensions such as priority transitions, inhibitor arcs, and probabilistic arcs can be converted into their equivalent Markov process representations, and analyzed analytically.

To demonstrate the methodology, we consider the example of the system shown in Fig. 17. Suppose that the machining tool $M1$ is faster than $M2$, however subject to failures. $M2$ and the two robots are failure-free. Given the processing rates of the machining tools and robots, as well as the failure and repair rates of $M1$, we are interested in obtaining the average utilization of $M1$, and the production rate of the system (throughput), assuming that only one pallet is available.

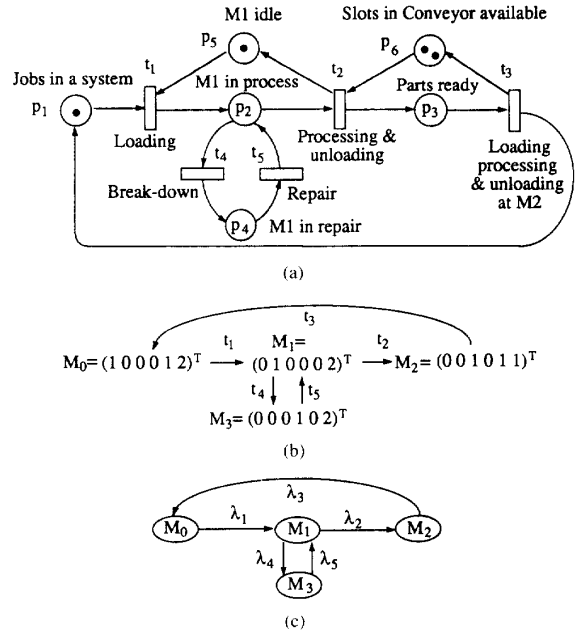

Fig. 19. (a) Stochastic Petri net model for the production line. (b) Reachability graph. (c) Markov chain.

The SPN model of the system is shown in Fig. 19(a). A number of modeling assumptions were made for the convenience of the presentation:

1. The tasks performed by $M2$ and $R2$ were aggregated into a single transition representing *loading, processing and unloading at* $M2$.
2. The time associated with the unloading tasks performed by $R1$ is incorporated into transition t_2.
3. The failure-repair loop is added to $M1$.
4. The time delays are associated with transitions only.

In the model, λ_i is used to represent the firing rate of a transition t_i. A firing rate of a transition equals to the reciprocal of the average firing delay time of the corresponding event or operation. The description of places and transitions is given in Table IV, together with the adopted values of the firing rates.

The reachability graph of the model is shown in Fig. 19(b). The graphical representation of the corresponding embedded continuous time Markov chain is shown in Fig. 19(c). In this representation, the links between states are labeled with the firing rates of the corresponding transitions.

The transition rate matrix $(a_{ij})(i,j = 0,1,2,3)$ for the embedded continuous time Markov chain, which is shown in Fig. 19(c), is as follows:

$$C = \begin{bmatrix} -\lambda_1 & \lambda_1 & 0 & 0 \\ 0 & -\lambda_2 - \lambda_4 & \lambda_2 & \lambda_4 \\ \lambda_3 & 0 & -\lambda_3 & 0 \\ 0 & \lambda_5 & 0 & -\lambda_5 \end{bmatrix}.$$

The steady-state probabilities, denoted as π_i, can be obtained by solving the following equations:

$$(\pi_0\ \pi_1\ \pi_2\ \pi_3)C = 0$$
$$\pi_0 + \pi_1 + \pi_2 + \pi_3 = 1.$$

TABLE IV
INTERPRETATION AND FIRING RATES OF PLACES AND TRANSITIONS IN FIG. 19(a).

Place	Interpretation	–
p_1	Workpieces and pallets available	–
p_2	M1 processing a workpiece	–
p_3	Workpiece ready for processing at M2	–
p_4	M1 in repair	–
p_5	M1 available	–
p_6	Conveyor slots available	–
Transition	Interpretation	Firing Rate
t_1	Loading involving R1, and M1	$\lambda_1 = 40$
t_2	Processing and unloading involving M1, and R1	$\lambda_2 = 5$
t_3	Loading, processing, and unloading involving R2, and M2	$\lambda_3 = 4$
t_4	M1 breaks down	$\lambda_4 = 0.5$
t_5	M1 is repaired	$\lambda_5 = 4$

This yields

$$\pi_0 = \lambda_2\lambda_3\lambda_5/\lambda, \qquad \pi_1 = \lambda_1\lambda_3\lambda_5/\lambda,$$
$$\pi_2 = \lambda_1\lambda_2\lambda_5/\lambda, \qquad \pi_3 = \lambda_1\lambda_3\lambda_4/\lambda$$

where $\lambda = \lambda_2\lambda_3\lambda_5 + \lambda_1\lambda_3\lambda_5 + \lambda_1\lambda_2\lambda_5 + \lambda_1\lambda_3\lambda_4$.

When using the firing rates from Table IV, and the derived formulas for the steady-state probabilities, the following values are obtained $\pi_0 = 0.05$, $\pi_1 = 0.40$, $\pi_2 = 0.50$, and $\pi_3 = 0.05$.

1. The $M1$ executing a task is modeled by the presence of a token on place p_2. This corresponds to the marking M_1, and state π_1. The expected utilization of $M1$ is then 40%.
2. The completion of a product is modeled by firing transition t_3. This transition is enabled in marking M_2. Thus, the system production average rate is given by the product of the probability that t_3 is enabled and the firing rate of t_3, i.e., $\pi_2\lambda_3 = 2.0$ workpieces/time unit.

The analysis of a Markov model may involve the steady-state (as shown in the example) or transient analysis (the study of the system evolution from its initial state to the steady state), or both depending on the requirements and the Markov process itself. For example, if the net is not deadlock-free, i.e., the underlying Markov process has an absorbing state(s), then the steady-state analysis makes no sense. General-purpose software packages are available for solving the above stochastic net models, including GreatSPN [30] and SPNP [32].

VI. PETRI NETS: APPLICATIONS PROSPECTS

The development of Petri nets has been, to a large extent, motivated by the need to model the industrial systems. Ordinary Petri nets are not always sufficient to represent and analyze complex industrial and other systems. This prompted the development of new classes of nets, some of which are introduced briefly in this section.

Tokens in the ordinary nets have no identity. This poses some problems for modeling systems such as manufacturing and communication systems, which require the physical resources or messages, if represented by tokens, to have identity. Without this identity, it is impossible to trace the flow of different resources or messages in the system. A potential solution is to construct a model in such a way that the flow of each resource or message is associated with a dedicated subnet. Since the resources or messages share, in most cases, the same system, all these subnets are identical. This approach increases

the graphical complexity of the model. In order to address this issue, Petri nets which allow for tokens to have distinct identity were proposed. These nets, referred to as high-level Petri nets, include predicate-transition nets [45], colored nets [56], and nets with individual tokens [87]. In high-level Petri nets, a token can be a compound object carrying data. This data can be of arbitrary complexity involving integers, reals, text strings, records, lists and tuples. A detailed discussion of this class of nets is beyond the scope of this paper. However, it should be noted that ordinary and high-level Petri nets have the same descriptive power. High-level Petri nets provide much better structuring facilities than ordinary nets. The colored and predicate-transition nets are almost identical with respect to their description and simulation. However, there are some considerable differences with respect to formal analysis. The colored Petri nets were used in numerous application areas. These areas include communication protocols [53], production systems [70], VLSI [92], etc. An important development in the area of high-level Petri nets was the introduction of object-oriented Petri nets, which are described in detail in [94]. Object-oriented Petri nets can be considered as a special kind of high level Petri nets which allow for the representation and manipulation of an object class [16]. In this class of nets, tokens are considered as instances or tuples of instances of object classes which are defined as lists of attributes [102]. This type of net was used to model and analyze FMS systems [2], [10], assembly tasks [8], and assembly systems [1].

The recognition of the need for the qualitative specification of the industrial control, as well as the need for representing approximate and uncertain information has led to the development of various types of fuzzy Petri nets. The definitions of these nets, to a large extent, were influenced by the various application areas. Fuzzy Petri nets have been used for knowledge representation and reasoning [29], [44], [66]. Fuzzy Petri nets (Petri nets with objects [94]) have been also used to model monitoring and control of a FMS system [101]. Fuzzy Programmable Logic Controllers were modeled in [80]. Task sequence planning in robotic assembly systems using fuzzy Petri nets was discussed in [21].

Ordinary Petri nets are not powerful enough for representing and studying some of the important properties of concurrent systems, such as eventuality (certain transitions must eventually fire; certain places must eventually have tokens), and fairness (if a transition becomes firable infinitely often, then it must fire infinitely often), for instance. In order to address these issues, temporal Petri nets were introduced. In this class of nets, the timing constraints are represented by the operators of a variant of the propositional temporal logic of linear time. Typical operators used in this class of nets are next, henceforth, eventually, until, etc. Temporal Petri nets were used to model and analyze a handshake daisy chain arbiter [97], and the alternating bit protocol [98]. The ability of temporal Petri nets to express eventuality makes this model suitable to represent and study the external functional behavior of systems. This functionality is expressed in terms of the input-output relationship, i.e., if a certain input pattern has been established, than eventually a certain output pattern will be generated. In [112], a method was proposed which allows for the construction of nets realizing the external functionality using less places and transitions then the original models. This allows for the construction of the verification models of complex systems using less places and transitions, and thus reducing the difficulty of the formal verification tasks.

Although attempts to combine Petri nets with other techniques, such as neural networks, fuzzy logic, etc., seem to be on the increase, it appears that the use of Petri nets is still restricted to research laboratories and academic institutions. This situation, to a large extent, results from the lack of widely available inexpensive software tools suitable for the development of industrial type of systems. These types of tools would be required to provide facilities for dealing with the application domain specific problems at a relatively skill-free level which would not require the knowledge of Petri nets, and the analysis methods. The facility for translating Petri net models to the executable code will be also essential. They will allow for rapid prototyping of the developed systems in the operational environment. In the past few years, a large number of tools have been reported in the Petri net literature. However, a majority of these tools are used mostly for research and educational purposes. An overview of some of the Petri net tools can be found in [40].

Another reason why the use of Petri nets is largely confined to academic and research institutions is the difficulty involved in constructing Petri net models. Constructing Petri net models of systems, especially large scale systems, is not a trivial task. It requires a great deal of experience. No methodology is available yet, which would allow for a fully automatic construction of Petri net models. From our observations, in most cases, Petri net models are constructed in an ad hoc manner. However, attempts have been recently made to make this particular approach more systematic. These were reported in [38], [111]. In the past two decades, numerous approaches to the systematic construction of Petri net models have been proposed, and the work in this area still continues. These approaches, using the terms of software engineering, can be broadly classified into bottom-up, top-down, and hybrid approaches [109], [111]. A comprehensive discussion of these approaches can be found in [55]. The reuse of Petri net models is also restricted. This is mainly due to the fact that Petri net models are, typically, constructed on a one-off basis. The development is, in most cases, not supported by proper documentation. It is clear that if Petri nets were to be widely used, especially by the industry people, methods, and the supporting tools, allowing for an automatic or semi-automatic construction of Petri net models from requirements specifications would have to be developed. In the past few years, a number of approaches have been reported which allow for the automatic construction of restricted classes of Petri net models from requirements specifications expressed using production rules, flow diagrams, state machines, temporal logic, application domain dependent semi-formal languages, etc.

ACKNOWLEDGMENT

The authors greatly appreciate the comments given by A. D. Robbi, K. Venkatesh, and H. H. Xiong.

REFERENCES

[1] M. Adamou, A. Bourljault, and S. Zerhouni, "Modelling and control of flexible manufacturing assembly systems using object oriented Petri nets," in *Proc. IEEE Int. Workshop on Emerging Technol. and Factory Automation*, Palm-Cove, Cairns, Australia, 1993, pp. 164–168.

[2] S. A. Ben, M. Moalla, M. Courvoisier, and R. Valette, "Flexible manufacturing production system modelling using object Petri nets and their analysis," *IMACS Symp. MCTS*, Lille, France, 1991, pp. 553–560.

[3] M. Akaza, D.-I. Lee, S. Kumagai, and S. Kodama, "Application of timed marked graphs to a scheduling problem of production systems including repetitive processes with set-up times," in *Modern Tools for Manufacturing Systems*, R. Zurawski, T. Dillon, Eds. Amsterdam: Elsevier, 1993, pp. 263–277.

[4] R. Y. Al-Jaar and A. A. Desrochers, "Performance evaluation of automated manufacturing systems using generalized stochastic Petri nets," *IEEE Trans. Robotics and Automat.*, vol. 6, no. 6, pp. 621–639, 1990.

[5] 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 Technol. and Factory Automation*, Melbourne, Australia, 1992, pp. 44–59.

[6] H. H. Ammar, S. M. Islam and S. Deng, "Performability analysis of parallel and distributed algorithms" in *Proc. IEEE Int. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, 1989.

[7] Z. A. Banaszak and B. H. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrency competing process flows," *IEEE Trans. Robotics Automat.*, vol. 6, no. 6, pp. 724–734, 1990.

[8] O. Barakat, L. Paris, J. Bourrieres, and F. Lhote, "Object oriented modelling of assembly tasks," *IMACS Symp. MCTS*, Lille, France, 1991, pp. 638–643.

[9] M. Barbeau and G. Bochman, "A subset of Lotos with the computational power of place/transition-nets," in *Advances and Theory of Petri Nets 1993, Lecture Notes in Computer Science*. A. Marsan, Ed. Berlin: Springer-Verlag, vol. 691, pp. 49–68.

[10] R. Bastide and C. Silbertin-Blanc, "Modelling of flexible manufacturing system by means of cooperative objects," in *Computer Applications in Production and Engineering*, G. Doumeingts, J. Browne, M. Tomljanovich, Eds. Amsterdam: Elsevier, 1991, pp. 593–600.

[11] J. Bechta-Dugan and K. S. Trivedi, "Coverage modelling for dependability analysis of fault-tolerant systems," *IEEE Trans. Comput.*, vol. 38, no. 6, pp. 775–787, 1989.

[12] 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. Software Eng.*, vol. 17, no. 6, pp. 513–526, 1991.

[13] N. Berge, M. Samaan, G. Juanole, and J. Atamna, "Methodology for LAN modelling and analysis using Petri net based models," in *Proc. Int. Workshop on Modelling, Analysis and Simulation in Telecomm. Syst., MASCOT'94*, Durham, NC, 1994, pp. 335–342.

[14] B. Berthomieu and M. Diaz, "Modelling and verification of time dependent systems," *IEEE Trans. Software Eng.*, vol. 17, no. 3, pp. 259–273, 1991.

[15] J. Billington, G. R. Wheeler, and M. C. Wilbur-Ham, "PROTEAN: A high-level Petri net tool for the specification and verification of communication protocols," *IEEE Trans. Software Eng.*, vol. 14, no. 3, pp. 301–31, 1988.

[16] G. Booch, "Object-oriented development," *IEEE Trans. Software Eng.*, vol. 12, no. 2, pp. 211–221, 1986.

[17] K. Brand and J. Kopainsky, "Principles and engineering of process control with Petri nets," *IEEE Trans. Automat. Cont.*, vol. 33, no. 2, pp. 138–149, 1988.

[18] G. Bruno and G. Marchetto, "Process-translatable Petri nets for the rapid prototyping of process control systems," *IEEE Trans. Software Eng.*, vol. 12, no. 2, pp. 346–357, 1986.

[19] G. Bundell, "Multi-facet modelling of flexible manufacturing systems using a minimax algebraic Petri net representation," in *Proc. IEEE Int. Workshop on Emerging Technol. and Factory Automat.*, Palm-Cove, Cairns, Australia, 1993, pp. 178–187.

[20] J. Campos, G. Chiola, J. M. Colom, and M. Silva, "Properties and performance bounds for timed marked graphs," *IEEE Trans. Circuits and Syst.—I: Fundamental Theory and Applicat.*, vol. 39, no. 5, pp. 386–401, 1992.

[21] T. Cao and A. C. Sanderson, "Task decomposition and analysis of assembly sequence plans using Petri nets," *Proc. 3rd Int. Conf. on Comput. Integrated Manufacturing*, Troy, NY, 1992, pp. 138–147.

[22] C. Capellmann and H. Dibold, "Petri net based specifications of services in an intelligent network-experiences gained from a test case application," in *Applications and Theory of Petri Nets 1993, Lecture Notes in Computer Science*, A. Marsan, Ed. Berlin: Springer-Verlag, 1993, vol. 691, pp. 543–551.

[23] L. R. Carmo and G. Juanole, "Modelling and evaluating the DQDB protocol with stochastic timed Petri nets," in *Proc. Int. Workshop on Modelling, Analysis and Simulation in Telecomm. Syst., MASCOT'94*, Durham, NC, 1994, pp. 269–275.

[24] S. Caselli and G. Conte, "GSPN models of concurrent architectures with mesh topology," in *Proc. IEEE International Workshop on Petri Nets and Performance Models*, Melbourne, Australia, 1991, pp. 280–289.

[25] V. Catania, A. Puliafito, and L. Vita, "A modeling framework to evaluate performability parameters in gracefully degrading systems," *IEEE Trans. Ind. Electron.*, vol. 40, no. 5, pp. 461–472, 1993.

[26] S. Cavalieri, A. Di Stefano, and O. Mirabella, "Optimization of acyclic bandwith allocation exploiting the priority mechanism in the Fieldbus data link layer," *IEEE Trans. Ind. Electron.*, vol. 40, no. 3, pp. 297–306, 1993.

[27] A. Chaillet, M. Combacau, and M. Courvoisier, "Specification of FMS real-time control based on Petri nets with objects and process failure monitoring," in *Proc. IECON'93*, Hawaii, 1993, pp. 144–149.

[28] G. Chehaibar, "Validation of phase-executed protocols modelled with coloured Petri nets," in *Proc. 11th Int. Conf. on Applicat. and Theory of Petri Nets*, Paris, France, 1990, pp. 84–103.

[29] S. Chen, J. Ke, and J. Chang, "Knowledge representation using fuzzy Petri nets," *IEEE Trans. Knowledge and Data Eng.*, vol. 2, no. 3, pp. 311–319, 1990.

[30] G. Chiola, "A software package for the analysis of generalized stochastic Petri net models" in *Proc. IEEE Int. Workshop on Timed Petri Nets*, Torino, Italy, 1985.

[31] G. Chiola, C. Dutheillet, G. Franceschinis and S. Haddad, "Stochastic well-formed coloured nets and symmetric modelling applications," *IEEE Trans. Comput.*, vol. 42, no. 11, pp. 1343–1360, 1993.

[32] G. Ciardo, *Manual for the SPNP Package*, Duke University, 1989.

[33] F. Cindo, G. Lanzarone and A. Torgano, "A Petri net model of SDL," in *Proc. 5th European Workshop on Applicat. and Theory of Petri Nets*, Aarhus, Denmark, 1984, pp. 272–289.

[34] E. G. Coffman, M. J. Elphick, and A. Shoshani, "System deadlocks," *Computing Surveys*, vol. 3, pp. 67–78.

[35] D. Crockett, A. Desrochers, F. DiCesare, and T. Ward, "Implementation of a Petri net controller for a machining workstation," in *Proc. of IEEE Int. Conf. Robotics and Automat.*, Raleigh, NC, 1987, pp. 1861–1867.

[36] R. David and H. Alla, "Autonomous and timed continuous Petri nets," in *Advances in Petri Nets 1993, Lecture Notes in Computer Science*, G. Rozenberg Ed. Berlin: Springer-Verlag, 1993, vol. 674, pp. 71–90.

[37] M. Diaz, "Petri net based models in the specification and verification of protocols," in *Applications of Petri Nets 1986 Part II, Lecture Notes in Computer Science*, W. Brauer, W. Reisig and G. Rozenberg, Eds. Berlin: Springer-Verlag, vol. 255, pp. 135–170, 1987.

[38] F. DiCesare and A. A. Desrochers, "Modeling, control, and performance analysis of automated manufacturing systems using Petri nets," *Control and Dynamic Systems*, C. T. Leondes, Ed. New York: Academic, 1991, vol. 47, pp. 121–172.

[39] P. Dubois, "A flexible workshop design and optimization using SEDRIC: A Petri net simulator," in *Proc. 10th Int. Conf. on Applicat. and Theory of Petri Nets*, Bonn, Germany, 1989, pp. 378–393.

[40] F. Feldbrugge, "Petri net tools overview 1989," in *Advances in Petri Nets 1989, Lecture Notes In Computer Science*, G. Rozenberg, Ed. Berlin: Springer-Verlag, 1990, vol. 424, pp. 151–178.

[41] L. Ferrarini, "An incremental approach to logic controller design with Petri nets," *IEEE Trans. Syst., Man, and Cybern.*, vol. 22, no. 3, pp. 461–473, 1992.

[42] G. Florin, C. Keiser, and S. Natkin, "Petri net models of a distributed election protocol on unidirectional ring," in *Proc. 10th Int. Conf. on Applicat. and Theory of Petri Nets*, Bon, Germany, 1989, pp. 154–173.

[43] P. Freedman, "Time, Petri nets, and robotics," *IEEE Trans. Robotics and Automat.*, vol. 7, no. 4, pp. 417–433, 1991.

[44] M. L. Garg, S. I. Ahson, and P. V. Gupta, "A fuzzy Petri net for knowledge representation and reasoning," *Inform. Processing Lett.*, vol. 39, pp. 165–171, 1991.

[45] H. J. Genrich and K. Lautenbach, "System modelling with high-level Petri nets," *Theoret. Comp. Sci.*, vol. 13, pp. 109–136, 1991.

[46] H. J. Genrich and R. M. Shapiro, "Formal verification of an arbiter cascade," in *Advances and Theory of Petri Nets 1992, Lecture Notes in Computer Science*. Berlin: Springer-Verlag, vol. 616, pp. 205–223, 1992.

[47] H.-M. Hanisch, "Analysis of place/transition nets with timed arcs and its application to batch process control," in *Applications and Theory of*

Petri Nets 1993, Lecture Notes in Computer Science, M. A. Marsan, Ed. Berlin: Springer-Verlag, 1993, vol. 691, pp. 282–299.

[48] D. A. Hartley and D. M. Harvey, "Analysis of the TMS320C40 communication channels using timed Petri nets," in *Advances and Theory of Petri Nets 1993, Lecture Notes in Computer Science,* A. Marsan, Ed. Berlin: Springer-Verlag, 1993, vol. 691, pp. 562–571.

[49] H. P. Hillion and J. M. Proth, "Performance evaluation of job-shop systems using timed-event-graphs," *IEEE Trans. Automat. Cont.,* vol. 34, no. 1, pp. 3–9, 1989.

[50] M. A. Holliday and M. K. Vernon, "Exact performance estimates for multiprocessor memory and bus interface," *IEEE Trans. Comput.,* vol. 36, no. 1, pp. 76–85, 1987.

[51] L. E. Holloway and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets," *IEEE Trans. Automat. Cont.,* vol. 35, no. 5, pp. 514–523, 1990.

[52] D.-Y. Hsieh and S.-C. Chang, "Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems," *IEEE Trans. Robotics and Automat.,* vol. 10, no. 2, pp. 196–209, 1994.

[53] P. Huber and V. Pinci, "A formal executable specification of the ISDN basic rate interface," in *Proc. 12th Int. Conf. Applicat.and Theory of Petri Nets,* Aarhus, Denmark, 1991, pp. 1–21.

[54] M. A. Jafari, "An architecture for a shop-floor controller using colored Petri nets," *Int. J. Flexible Mfg. Syst.,* vol. 4, pp. 159–181, 1992.

[55] M. D. Jeng and F. DiCesare, "A review of synthesis techniques for Petri nets with applications to automated manufacturing systems," *IEEE Trans. Syst., Man, and Cybern.,* vol. 23, no. 1, pp. 301–312, 1993.

[56] K. Jensen, "Coloured Petri nets and the invariant method," *Theoret. Comp. Sci.,* vol 14, pp. 317–336, 1981.

[57] ——, "Coloured Petri nets: a high level language for system design and analysis," in *Advances in Petri Nets 1990, Lecture Notes in Computer Science,* G. Rozenberg, Ed. Berlin: Springer-Verlag, 1990, pp. 342–416.

[58] G. Juanole and J. Atamna, "Petri net based models and communication protocols," in *Modern Tools for Manufacturing Systems,* R. Zurawski and T. Dillon, Eds. Amsterdam: Elsevier, 1993, pp. 359–384.

[59] G. Klas and C. Wincheringer, "A generalized stochastic Petri net model of Multibus II," in *Proc. CompEuro 92,* The Hague, the Netherlands, 1992, pp. 406–411.

[60] G. Klas, "Protocol optimization for a packet-switched bus in case of burst traffic by means of GSPN," in *Advances and Theory of Petri Nets 1993, Lecture Notes in Computer Science,* A. Marsan, Ed. Berlin: Springer-Verlag, 1993, vol. 691, pp. 572–581.

[61] S. Lafitts, J-M. Proth, and X. L. Xie, "Marking optimization in timed event graphs," in *Advances in Petri Nets 1993, Lecture Notes in Computer Science,* G. Rozenberg, Ed. Berlin: Springer-Verlag, 1993, vol. 674, pp. 281–299.

[62] C. A. Lakos and C. D. Keen, "Modelling layered protocols in LOOP," in *Proc. 4th IEEE Int. Workshop on Petri Nets and Performance Models,* Melbourne, Australia, 1991, pp. 106–115.

[63] D. Y. Lee and F. DiCesare, "Scheduling flexible manufacturing systems using Petri nets and heuristic search," *IEEE Trans. Robotics and Automat.,* vol. 10, no. 2, pp. 123–132, 1994.

[64] W. W. Lendon, R. F. Vidale, "Analysis of an Ada system using coloured Petri nets and occurrence graphs," in *Applications and Theory of Petri Nets 1992, Lecture Notes in Computer Science.* Berlin: Springer-Verlag, 1992, vol. 616, pp. 384–388.

[65] N. G. Leveson and J. L. Stolzy, "Safety analysis using Petri nets," *IEEE Trans. Software Eng.,* vol. 13, no. 3, pp. 386–397, 1987.

[66] C. J. Loony, "Fuzzy Petri nets for rule-based decision making," *IEEE Trans. Syst., Man, and Cybern.,* vol. 18, no. 1, pp. 178–183, 1988.

[67] A. M. Marsan and V. Signore, "Timed Petri net performance of fibre optics LAN architectures," in *Proc. IEEE Int. Workshop on Petri Nets and Performance Models,* Madison, WI, 1987, pp. 66–74.

[68] A. M. Marsan, G. Balbo, and G. Conte, *Performance Evaluation of Multiprocessor Systems using Petri Nets.* Cambridge, MA: MIT, 1986.

[69] ——, "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems," *ACM Trans. Comput. Syst.,* vol. 2, no. 2, pp. 93–122, 1984.

[70] J. Martinez, P. Muro, and M. Silva, "Modeling, validation and software implementation of production systems using high level Petri nets," in *Proc. IEEE Int. Conf. Robotics and Automat.,* Raleigh, NC, 1987, pp. 1180–1185.

[71] B. Mazigh and F. Simon, "Well suited modelling and evaluation techniques based on GSPN for real production systems," in *Proc. IEEE Int. Workshop on Emerging Technol. and Factory Automat.,* Palm-Cove, Cairns, Australia, 1993, pp. 169–176.

[72] B. McCarragher, "Robotic assembly and trajectory planning using discrete-event modelling," in *Proc. IEEE Int. Workshop on Emerging Technol. and Factory Automat.,* Palm-Cove, Cairns, Australia, 1993, pp. 187–196.

[73] M. K. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Trans. Comput.,* vol. 3, no. 9, pp. 913–917, 1982.

[74] S. Moriguchi and G. S. Shedler, "Simulation methods for logistics systems using stochastic Petri nets," in *Modern Tools for Manufacturing Systems,* R. Zurawski and T. Dillon, Eds. Amsterdam: Elsevier, 1993, pp. 289–320.

[75] S. Morioka and T. Yamada, "Performance evaluation of marked graphs by linear programming," *Int. J. Syst. Sci.,* vol. 22, no. 9, pp. 1541–1552, 1991.

[76] T. Murata, "Petri nets: Properties, analysis, and applications," in *Proc. IEEE,* 1989, vol. 77, no. 4, pp. 541–580.

[77] T. Murata, B. Shenker, and S. Shatz, "Detection of ADA static deadlocks using Petri net invariants," *IEEE Trans. Software Eng.,* vol. 15, no. 3, pp. 314–326, 1989.

[78] T. Murata, N. Komoda, K. Matsumoto, and K. Haruna, "A Petri net based controller for flexible and maintainable sequence control and its application in factory automation," in *IEEE Trans. Ind. Electron.,* vol. 33, no. 1, pp. 1–8, 1986.

[79] A. Nketsa and M. Courvoisier, "A Petri net based single chip programmable controller for distributed local controls," in *Proc. IECON'90,* 1990, vol. 1, pp. 542–547.

[80] J-C. Pascal, R. Valette, and D. Andreu, "Fuzzy sequential control based on Petri nets," in *Proc. IEEE International Conference on Emerging Technology and Factory Automation,* Melbourne, Australia, 1992, pp. 140–145.

[81] J. L. Peterson, *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

[82] V. Pinci and R. M. Shapiro, "An integrated software development methodology based on hierarchical coloured Petri nets," in *Advances in Petri Nets 1991, Lecture Notes in Computer Science,* G. Rozenberg, Ed. Berlin: Springer-Verlag, 1991, vol. 524, pp. 227–252.

[83] H. Plunnecke and W. Reisig, "Bibliography of Petri nets 1990," in *Advances on Petri Nets 1991, Lecture Notes in Computer Science,* G. Rozenberg, Ed. Berlin: Springer-Verlag, 1991, vol. 524, pp. 317–572.

[84] C. V. Ramamoorthy, S. T. Dong, and Y. Usuda, "An implementation of anautomated protocol synthesizer (APS) and its application to the X.21 protocol," *IEEE Trans. Software Eng.,* vol. 11, no. 9, pp. 886–908, 1987.

[85] C. V. Ramamoorthy and G. S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Trans. Software Eng.,* vol. 6, no. 5, pp. 440–449, 1980.

[86] W. Reisig, *Petri Nets, EATCS Monographs on Theoretical Computer Science* New York: Springer-Verlag, 1985, vol. 4.

[87] ——, "Petri nets with individual tokens," *Informatik Fachberichte,* vol. 66, no. 21, pp. 229–249, 1983.

[88] ——, "Petri nets in software engineering," in *Advances in Petri Nets 1986,* Part II, *Lecture Notes in Computer Science,* W. Brauer, W. Reisig and G. Rozenberg, Eds. Berlin: Springer-Verlag, 1987, vol. 255, pp. 63–98.

[89] A. Sahraoui, H. Atabakche, M. Courvoisier, and R. Valette, "Joining Petri nets and knowledge based systems for monitoring purposes," in *Proc. IEEE Conf. Robotics and Automat.,* 1987, pp. 1160–1165.

[90] L. Shen, Q. Chen, J. Y. Luh, C. Chen, and Z. Zhang, "Truncation of Petri net models of scheduling problems for optimum solution," in *Proc. Japan/USA Symp. on Flexible Automation,* San Francisco, CA, 1992, pp. 1681–1688.

[91] H. Shih and T. Sehiguchi, "A time Petri net and beam search based on-line FMS scheduling system with routing flexibility," in *Proc. IEEE Conf. on Robotics and Automat.,* Sacramento, CA, 1991, pp. 2548–2553.

[92] R. M. Shapiro, "Validation of VLSI chip using hierarchical coloured Petri nets," in *Proc. 11th Int. Conf. Applicat. and Theory of Petri Nets,* Paris, France, 1990.

[93] S. Shatz, K. Mai, C. Black, and S. Tu, "Design and implementation of a Petri net based toolkit for Ada tasking analysis," *IEEE Trans. Parallel Dist. Syst.,* vol. 1, no. 4, pp. 424–441, 1990.

[94] C. Sibertin-Blanc, "High level Petri nets with data structures," in *Proc. 6th European Workshop on Applicat. and Theory of Petri Nets,* Helsinki, Finland, 1985.

[95] J. Sifakis, "Use of Petri nets for performance evaluation," in *Measuring, Modelling, and Evaluating Computer Systems.* Amsterdam: 1977, pp. 75–93.

[96] V. S. Srinivasan and M. A. Jafari, "Fault detection/monitoring using time Petri nets," *IEEE Trans. Syst., Man, and Cybern.,* vol. 23, no. 4, pp. 1155–1162, 1993.

[97] I. Suzuki and H. Lu, "Temporal Petri nets and their application to modelling and analysis of a handshake daisy chain arbiter," *IEEE Trans. Comput.,* vol. 38, no. 5, pp. 696–704, 1989.

[98] I. Suzuki, "Formal analysis of the alternating bit protocol by temporal Petri nets," *IEEE Trans. Software Eng.*, vol. 16, no. 11, pp. 1273–1281, 1990.
[99] T. Suzuki, S. Shatz, and T. Murata, "A protocol modelling and verification approach based on specification language and Petri nets," *IEEE Trans. Software Eng.*, vol. 16, no. 5, pp. 523–536, 1990.
[100] C-J. Tsai and L-C. Fu, "Modular approach for Petri net modelling of flexible manufacturing systems adaptable to various task-flow requirements," in *Proc. of 1992 IEEE Conf. Robotics and Automat.*, Nice, France, 1992, pp. 1043–1048.
[101] R. Valette, J. Cardoso, and D. Dubois, "Monitoring manufacturing systems by means of Petri nets with imprecise markings," in *Proc. IEEE Int. Symp. Intell. Cont.*, pp. 233–237, 1989.
[102] R. Valette and M. Courvoisier, "Petri nets and artificial intelligence," in *Modern Tools for Manufacturing Systems*,R. Zurawski and T. Dillon, Eds. Amsterdam: Elsevier, pp. 385–405, 1993.
[103] N. Viswanadham, Y. Narahari, and L. Johnson, "Deadlock prevention and avoidance in flexible manufacturing systems using Petri net models," *IEEE Trans. Robotics and Automat.*, vol. 6, no. 6, pp. 713–723, 1990.
[104] N. Viswanadham and S. Sundar, "Distributed simulation of flexible manufacturing systems," in *Proc. IEEE IECON'88*, Singapore, 1988, pp. 895–900.
[105] C. J. Wang and V. P. Nelson, "Petri net performance modelling of a modified mesh-connected parallel computer," *Parallel Computing*, vol. 17, no. 1, pp. 75–84, 1991.
[106] C.-Y. Wang and K. Trivedi, "Integration of specification for modelling and specification for system design," in *Advances and Theory of Petri Nets 1993, Lecture Notes in Computer Science*, A. Marsan, Ed. Berlin: Springer-Verlag, 1993, vol. 691, pp. 473–491.
[107] E. C. Yamalidou and J. Kantor, "Modelling and optimal control of discrete-event chemical processes using Petri nets," *Comput. and Chemical Eng.*, vol. 15, pp. 503–519, 1990.
[108] D. Zhang, "Planning using timed Pr/T Nets," in *Proc. Japan U.S.A. Symp. Flexible Automat.*, San Francisco, CA, 1992, pp. 1179–1184.
[109] M. C. Zhou, F. DiCesare, and A. A. Desrochers, "A hybrid methodology for synthesis of Petri nets for manufacturing systems," *IEEE Trans. Robotics and Automat.*, vol. 8, no. 3, pp. 350–361, 1992.
[110] M. C. Zhou, K. McDermott, and P. A. Patel, "Petri net synthesis and analysis of an FMS cell," *IEEE Trans. Syst., Man, and Cybern.*, vol. 23, no. 2, pp. 523–531, 1993.
[111] M. C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Boston, MA: Kluwer, 1993.
[112] R. Zurawski and T. Dillon, "Systematic construction of functional abstractions of Petri net models of typical components of flexible manufacturing systems," in *Proc. Int. Workshop on Petri Nets and Performance Models*, Melbourne, Australia, 1991, pp. 248–257.
[113] ——, "Modelling and verification of flexible manufacturing systems using Petri nets," *Modern Tools for Manufacturing Systems*, R. Zurawski and T. Dillon, Eds. Amsterdam: Elsevier, 1993, pp. 237–261.

Richard Zurawski (M'85), for a photograph and biography please see page 566 of this issue.

MengChu Zhou (S'88-M'90-M'90-SM'93), for a photograph and biography please see page 566 of this issue.