Annual Reviews in Control 37 (2013) 191–219

Contents lists available at ScienceDirect

Annual Reviews in Control

journal homepage: www.elsevier.com/locate/arcontrol

Review

Half a century after Carl Adam Petri's Ph.D. thesis: A perspective on the field ☆

Manuel Silva

Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain

ARTICLE INFO

Article history:
Received 25 March 2013
Accepted 28 July 2013
Available online 14 October 2013

ABSTRACT

It is certainly worth remarking on half a century of a work defining a landmark in *Discrete Event Dynamic Systems* (DEDS) theory. This invited contribution aims to combine some historical facts with elements of a conceptual view on concurrent DEDS, giving pointers about the development of the field. Simplifying the historical trajectory, it can be said that the seed sown by Carl Adam Petri in 1962 first grew in America (essentially until the mid 1970s), where an appropriate intellectual ambiance existed in computer science, business process management and switching systems design. Later, many other new lines of activity, including logic control and performance evaluation, flourished in Europe. Today *Petri nets* are widespread all over the world. The conceptual *paradigm* of Petri nets deals inter alia with modeling, logical analysis, performance evaluation, parametric optimization, dynamic control, diagnosis and implementation issues. In summary, *multidisciplinary* in themselves, formalisms belonging to the Petri nets paradigm may cover several phases of the life-cycle of complex DEDS.

Given the hundreds of research and text monographs on Petri nets, together with the many thousands of theoretical and applied contributions on the subject, not to mention the ISO (International Organization for Standardization) or IEC (International Electrotechnical Commission) standards for the use of Petri nets in engineering, this work cannot hope to be a complete survey or a tutorial in the more classical sense. It is more of an impressionistic overview of the field.

Contents

☆ This work has been partially supported by CICYT – FEDER project DPI2010-20413. The Group of Discrete Event Systems Engineering (GISED) is partially co-financed by the Aragonese Government (Ref. T27) and the European Social Fund.

E-mail address: silva@unizar.es

1367-5788/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.arcontrol.2013.09.001

1. Preliminary considerations

Science and Technology are social constructions. Nevertheless, certain individuals have contributed to their development in outstanding ways, as recognized by Isaac Newton in a letter to Robert Hooke (1676): "If I have seen further it is by standing on ye sholders of Giants".[1] Their achievements are sometimes partially recognized by giving their names to units of measurements, universal constants, algorithms, etc. For example, the *International System of Units* (SI) of measurement include the newton (named after Isaac Newton, 1642–1727), the coulomb (after Charles-Augustin de Coulomb, 1736–1806), the volt (after Alessandro Volta, 1745–1827), the ampere (after André Marie Ampère, 1775–1836) and the farad (after Michael Faraday, 1791–1867). Temperatures include the well-known scales of Celsius, Fahrenheit, Reaumur, Rankine and Kelvin (this last defines a universal constant: the 0 K). In other instances, researchers' names are given to specific items such as the force of Coriolis, the Planck constant, Avogadro's number, the Turing machine, the algorithm of Dijkstra (shortest paths), Wiener or the Kalman filters, or Forrester diagrams. In some cases, the equation or algorithm receives two or more names. For example, among the basic models of predator–prey problems can be found the so called Lotka–Volterra equations. Some "excesses" can be found in this well-intentioned practice. For example, we often hear about the "Watt governor" while making reference to the classical centrifugal or "flyball" governor. By no means can we doubt the outstanding contribution of James Watt to the steam engine and its clear consequences for the Industrial revolution, but this governor used by Watt was not invented by him. It had been patented previously (Mayr, 1970). In contrast, even if the paternity of a discovery is clear, the name of the discoverer is frequently not attached to it. For example, George Dantzig is properly acclaimed as the "father of linear programming", but his algorithm is just called the "simplex method" (1948).

In relatively very few cases, the name of a researcher is given to a theory for an entire subfield. For example, we speak of *Markov Chains* (MC) after Andrei Markov (1856–1922).[2] Of course, the theory of (semi-)Markov Processes is a collective work, not a personal one, but the Russian mathematician did play the role of pioneer. Analogous is the case of the so called *Petri nets*, a system theory initially inspired by Carl Adam Petri (1926–2010). The foundation stone of this construction, in which the so called Petri nets were not yet defined, is his Ph.D. dissertation (Petri, 1962).[3] Initially, Petri nets were considered as part of Computer Science (CS), but very quickly they also began to be employed in Automatic Control (AC) to design logic controllers. Last but not least, Petri nets were also incorporated into the background of Operations Research (OR). Therefore, Petri nets are perceived as part of the Discrete Event Dynamic Systems (DEDS) theory, at the intersection of CS, AC and OR. Moreover, fluid (or continuous) and different kinds of hybrid Petri nets are being extensively studied today. The important impact of Petri nets on information technology can be assessed considering the conferences, courses, books, tools or standard norms devoted to them, and the many application domains in which they have been/are used (see, in particular, Sections 2.3 and 7).

If the mathematics for continuous dynamic "views" of systems, particularly for control, go back more than three centuries (Sussmann & Willems, 1997), the formalization of discrete event "views" of dynamic systems is much more recent. Even if several precedents exist (Erlang, Shannon, Huffman, Moore, Mealy, etc.), roughly speaking it can be said that such "views" were really developed during the second half of the past century. Mainly in the framework of *switching systems*, in the past seventies, two IFAC symposia took place under the title *Discrete Systems*. The first was in Riga (USSR), in 1974, while the second one took place in Dresde (Deutsche Demokratische Republik), in 1977. Just to give a flavor of the problems considered at that times, the sections in this second case were: "General problems in the synthesis. Program systems for computer-aided design. Design using problem-oriented notations. Realization of switching circuits using complex modules. Reliability and dynamic behavior. Fault diagnosis. Analysis and simulation. General problems of switching and automata theory".[4] Moreover, in the framework of computer-based simulation, a subfield in computer engineering, there were important initiatives in the so called "discrete event simulation" (Fishman, 1973; Zeigler, 1976). In particular, in Zeigler (1976) the Discrete Event System Specification (DEVS) was introduced "to provide a formal basis for specifying the models expressible within discrete event simulation languages (SIMSCRIPT, SIMULA, GPSS, etc.)" (Zeigler, 1984).

The expression *Discrete Event Dynamic System* (DEDS) was first used in 1980, by Y. C. Ho and his team at Harvard. It became popular by the end of the same decade (see, for example, Ho (1989)). Like Molière's *Le Bourgeois Gentilhomme*, many of us, users or developers working on topics related to automata, Markov chains, queueing networks, Petri nets, process algebras, max/plus algebra, etc., more or less suddenly came to realize that we were not speaking prose, but "DEDS". In some sense, a new multidisciplinary community came to be recognized, mostly fed by specialities such as AC, CS and OR, but also by many others including Artificial Intelligence, Electronics and several application domains. Of course, as a formal system theory, its basis lies in mathematics and logic.

A quarter of a century ago, it was stated in the report about *Future directions in control theory: a mathematical perspective* (Fleming, 1988) that:

> there exist no formalisms for DEDS mathematically as compact or computationally as tractable as the differential equations are for continuous systems, particularly with the goal of control.

Certainly the field is much more mature today, as can be readily verified by looking at the many thousands of published works and their applications to real problems. Roughly speaking, despite significant changes, it can be said that the same basic operational formalisms remain today, Petri nets in particular, and considerable

[1] In modern English: "If I have seen further it is by standing on the shoulders of giants".

[2] As is well-known, this is a sequential view of stochastic systems that enjoy the memoryless property. In fact, "memorylessness" is the so called "Markov property".

[3] Translation into English, in Petri (1966).

[4] In the symposium of Dresde, the author of this overview published his first work concerning Petri nets (Silva & David, 1977).

diversity still prevails in the DEDS arena.[5] Without any doubt, *natural selection* (in the "Darwinian style") has and will continue to modify the conceptual and technical landscape.

Apart from undecidability issues, many analysis and synthesis techniques suffer from the well-known computational "disease" known as the *state explosion problem*. Fluidization of DEDS models is an efficient technique to partially overcome these problems, but the price paid is a loss of fidelity in the represented behaviors; in other words, this is another appearance of the classical tradeoff "fidelity vs computability". At the same time, the obtained models, continuous or hybrid, represent a bridge to the techniques addressed by the majority of the AC community.

Two broad perspectives on DEDS from the automatic control viewpoint are Cassandras and Lafortune (1999) and Seatzu, Silva, and Schuppen (2013), both dealing with automata and Petri nets. The first also addresses issues such as controlled Markov chains, DEDS simulation and sensitivity (perturbation) analysis. The second deals with distributed control and diagnosis, fluid approximations of DEDS, and systems in dioids (max-plus algebra).

This work is an extended version of the plenary talk presented at the WODES 2012 IFAC Workshop on Discrete Event Systems (Silva, 2012).[6] The purpose of this invited work is not to provide a broad view of Petri's life and contributions, nor to supply a technical overview of the development of the entire field. This would require much more than an article. In fact, the literature on proposals of derived formal Petri net models, analysis, synthesis and implementation (eventually fault-tolerant) techniques is enormous, and its consideration goes far beyond the scope of the panorama presented here which is purely "impressionistic" by nature. In Section 2 we provide just a few hints about Carl Adam Petri and the development of the field. Extended along many different directions, in Section 3 we provide a sketch of our view of Petri nets, first as autonomous formalisms with different levels of abstraction, even of different expressive power. The formal analysis of autonomous Petri net models and the enforcement of certain logical properties are considered in Section 4. In Section 5 Petri nets are considered as an integrated *modeling paradigm* rather than a collection of "unrelated" formalisms. Even if belonging to the Petri nets paradigm, the fluidization of net models is considered in Section 6, leading to continuous and hybrid formalisms. Reflections about the maturity of the field and pointers to applications are given in Section 7. Finally, in Section 8 a few general remarks close this panoramic view of a very extensive, multiformalist and multidisciplinary field dealing with a central systems theory for concurrency.

2. About Carl Adam Petri and concurrency theory

2.1. A mathematician educated at technical universities

Carl Adam Petri was born in Leipzig (1926). His father, a professor of mathematics, enjoyed relationships with distinguished mathematicians such as David Hilbert and Hermann Minkowski. Obliged to join the German army (1944), he was taken as a prisoner to England (until 1946), where he later taught Latin and chemistry. Petri returned to Germany in 1949, studying mathematics at the Technical University of Hannover (until 1956). From 1959 until 1962 he worked at the University of Bonn. In 1962 he obtained a doctorate degree from the Technical University of Darmstadt, *Kommunication mit Automaten*, winning an award for the best doctoral thesis of the year in the Faculty of Mathematics and Physics.[7]

Fig. 1. The investiture of Carl Adam Petri as Doctor *Honoris Causa* by the Universidad de Zaragoza (1999). The author of this article confers the cap, symbol of the doctorate.

This singular work does not try to solve a more or less specific problem. Petri deals with a number of fresh views, proposals to act, and insights. Later, Petri worked at the University of Bonn until 1968, becoming head of the computer installations. In that year the GMD (Gesellschaft für Mathematik und Datenverarbeitung), a national research center for applied mathematics and computer science, was founded. Carl Adam was called to manage the Institut für Informationssystemforschung (Institute for Information Systems Research), being later one of the directors of the Institut für Methodische Grundlagen (Institute for Methodological Foundations). Researchers such as H. Genrich, K. Lautenbach, C. Fernández, P.S. Thiagarajan and K. Voss worked in his prominent group at Schloss Birlinghoven. He retired from GMD in 1991. Carl Adam died in 2010 in a village close to Bonn.

During his career, Petri received several important recognitions. He was appointed Honorary Professor at the University of Hamburg in 1988 and received the Konrad–Zuse–Medal in 1993. The Society for Design and Process Science established the Carl A. Petri Distinguished Technical Achievement Award in 1997. In the same year he won the Werner-von-Siemens-Ring, one of the highest ranking awards for technical sciences in Germany, presented every three years. In the corresponding laudatory speech, prof. Gottzein said that:

> Petri nets brought engineers a breakthrough in their treatment of discretely controlled systems. Petri nets are a key to solve the design problem, as this is the first technique to allow for a unique description, as well as powerful analysis of discrete control systems. Based on Petri nets, it is now possible to formulate system invariants for discrete systems.

In 1999 he was awarded a Doctorate *Honoris Causa* by the Universidad de Zaragoza (Fig. 1). In 2003 Petri received the Orde van de Nederlandse Leeuw (Order of the Dutch Lion), with the grade of Commander. Finally, in 2008, together with Edward J. McCluskey, he received the IEEE Computer Pioneer award

> for establishing Petri net theory in 1962, which not only was cited by hundreds of thousands of scientific publications but also significantly advanced the fields of parallel and distributed computing

[5] The title of the report of a panel discussion at the International Workshop on Discrete Event Systems of a dozen of years ago, WODES2000, is very illustrative: *Unity in Diversity, Diversity in Unity* (Boel, 2002). Two panel members speak about formal models, "Petri nets" and "Max/+" algebra, and the other two about broad methodological issues, "supervisory control" and "perturbation analysis".

[6] The presentation and paper for WODES and our plenary talk at the Congresso Brasileiro de Automática (Campina Grande, 2 of September 2012), "On my relations with nets and Petri", have a more personal flavor.

[7] Both the original (in German) and English translations can be downloaded at: http://www.informatik.uni-hamburg.de/TGI/publikationen/public/biblio_petri.html.

Meaningfully, most recognitions of Carl Adam Petri's work were in the field of engineering.

2.2. Approaching concurrency in DEDS: comments about a System Theory view

We can say that Petri was a mathematician in a computer environment. Nevertheless, he was not a classical mathematician. Interested in the description of real-life situations, he essentially worked at a conceptual level, "opening windows" (i.e., providing foundations for new ways of thinking) to represent systems. This may be viewed as a profile less frequently exercised with success than that of a "theorem prover". In other words, he was much more conceptual than technical.

When in Computer Science the paradigm was local computations of mathematically intricate problems, Carl Adam Petri looked for a Systems Theory beyond the confines of problems in Informatics. He did not merely extend entities and constructions dealing with sequential computations. Certainly, he introduced a new and fresh approach to the conceptualization of concurrent systems, looking for a framework applicable to many types of discrete systems, being useful in a broad landscape of research fields (computer science, law, manufacturing, transportation, chemistry, epidemiology, demography, etc.). As an example of this preoccupation, in Petri and Smith (2007) it is claimed:

> Net theory has incorporated a touch of Pragmatics from its very beginning. It demands respect for, e.g.: limitation of all resources; inherent imprecision of measurement; partial independence of actions and decisions; and existence of illusions ("discrete" and "continuous" models), as the core of its "pragmatic" attitude.

As opposed to what is frequently quoted, in his PhD dissertation the graphical notation of Petri nets does not appear. The well-known bipartite graphs with conditions/places (local state variables, the values called markings) and events/transitions (locally bounded actions to neighboring places) came some three years later. When the Academy of Transdisciplinary Learning and Advanced Studies (ATLAS) awarded him its Gold Medal (2007), he confessed:

> In what follows I will describe some of the less well-known features of my work. The graphical representation of structural knowledge which is now in widespread use I invented it in a playful mood in August 1939, and practised it intensively for the purpose of memorizing chemical processes, using circles for substances and squares for reactions, interconnected by arrows to denote IN and OUT. In my dissertation on *Communication with Automata*, introducing the theory of such Nets in the context of Informatics, I did not mention my plaything. I did not want the theory to appear as a "graphical method" instead of a mathematical attack on the then prevailing Automata Theory, based on arguments taken from modern Physics. Only some years later, I was bold enough to propose Net Graphics as one of the standard features, and they were greatly welcomed.

Petri established Concurrency Theory as an axiomatic theory of binary relations of concurrency and causality, leading to expressive formalisms able to straightforwardly model *concurrency* and *synchronization*, thus also *cooperation* and *competition* relationships. Replacing temporal order by causal order, he regarded concurrency as mutually causal independent occurrences. Petri's *causal orders* and *preservation laws* were inspired by the laws of modern Physics. *Locality* (space) and *causality* are corner stones of an *untimed* (metric-free notions of time) theory of communication. This fact is specially interesting because computers are becoming more and more communication tools, not only computing engines, as they were in the sixties of the past century.

The two basic elements to build models of dynamic systems were *states* and *transitions* (e.g., substances and reactions, respectively), while the two relations among those (explicitly modeled by the arcs) represent the take or consume, and the create or give resources. With this point of view, using bipartite and animate graphs, Petri nets can reflect the structure of the system being modeled. Nevertheless, the ideas of Carl Adam Petri were in advance of the needs of the time. This is why we can understand that there was a certain "crossing of the desert" before Petri nets became well-accepted.

In the above context, we should not forget the role played by Anatole W. Holt and his group at Applied Data Research (later at Massachusetts Computer Associates) (Holt & Commoner, 1970; Holt, Saint, Shapiro, & Warshall, 1968). The *Information System Theory Project* (1968) was engaged in a program of basic research aimed at developing theory and techniques for the analysis and description of data structures in concurrent systems. Additionally, Holt was in close cooperation with the MIT, where Jack B. Dennis directed the project MAC (initially for "Mathematics And Computation", later backronymed, among other things, to "Multiple Access Computer"). After a first conference at MIT (Dennis, 1970), a second one devoted to *Petri Nets and Related Methods* took place five years later. Many researchers participated in this global scenario, including F. Commoner, M. Hack, S.S. Patil, C. Ramchandani, and R. Shapiro. A. Pnueli (Turing Award, 1996) also cooperated for a while. Among the earlier applications can be found speed independent design of switching circuits (S.S. Patil), certain resource-allocation problems (R. Shapiro), or production schemata (M. Hack).

Carl Adam Petri did explicitly recognize that Holt "contributed substantially to the deepening and dissemination of net things".[8] Among many other contributions to this dissemination, Holt directed the translation into English of Carl Adam's Ph.D. dissertation (Petri, 1966), played a key role in bringing Petri nets to the attention of relevant computer scientists, among others to Dennis at MIT, and "baptized" the nets as *Petri nets*. Thanks to Holt, the name of Petri is well-known worldwide, and is constantly repeated.

As clearly noted in Peterson (1977), it is worth pointing out the existence of some alternative views concerning the development of the field from those early times:

> In contrast to the work of Petri, Holt, and many European researchers, which emphasizes the fundamental concepts of systems, the work at MIT and many other American research centers concentrates on those mathematical aspects of Petri nets that are more closely related to automata theory [...] This mechanistic approach is quite different in orientation from the more philosophical approaches of Holt and Petri.

In this sense, it is very significant that Holt and Commoner (1970) declare:

> Perhaps we are closest in spirit to operations research techniques, but with an insistence on conceptual economy and rigor more common in purer branches of mathematics. Also, it is necessary that our descriptions be built up part by part in analogy to the way in which the systems being described are built up part by part.

[8] From the private correspondence of Petri to Holt; communication by Anastasia Pagnoni, widow of Anatole Holt, at the *Carl Adam Petri Memorial Symposium*, 4 of February, 2011, Berlin. At the same presentation, she summarized their relationship: "Anatol has been one of Petri's closest scientific counterparts – both soul buddy and rival – since they first met in 1964", or "Carl Adam and Tolly: crossed destinies. Half a century of fierce scientific fights, of respect, and friendship".

Carl Adam Petri persistently claimed that formal languages (in the automata theory sense), in which the well-known hierarchy of Chomsky defines levels of generality, was not appropriate to deal with the expressiveness of net systems models. In fact, their sequentialized views (sequences of events/occurrences of transitions) does not provide explicit information about concurrency and distribution of the modeled system. Informally speaking, some kind of "isomorphism" between the described system and the model contribute to the "faithfulness and understandability" of those formal constructions.

To this impressionistic picture should be added the existence in America of works like those of R.M. Karp and R.E. Miller. While dealing with control aspects of their "Parallel Program Schemata", around 1967 they introduced *Vector Addition Systems* (VAS). This was made as a "simple geometric structure" to solve certain decision procedures for properties such as determinacy, boundedness or termination (see, for example, Karp & Miller, 1969). The similarity of VAS and Petri nets with weights on the arcs (initially called "generalized Petri nets") was perceived very soon, although not exploited in a systematic way till some years later. For example, in Hack (1974b) it is proved that "Vector Addition Systems, Petri nets, Vector Replacement Systems and Generalized Petri nets are equivalent to each other, in the sense that any problem expressed in one formalism can be translated by a standard procedure into another formalism". Interestingly, it is explicitly declared that "the graphical appeal of Petri net methods permits a better grasp for intuitive arguments, which can help enormously to find rigorous proofs of various facts". This usefulness of Petri nets is demonstrated in Hack (1974a, 1976).

Moreover, placing this incomplete view of the first decade of concurrency theory in a broader context, it is interesting to see, for example, the survey published by Baer (1973). Reviewed in the context of modeling and analysis of concurrent/parallel systems in Computer Science, the works deal with several formalisms that admit graphical representation: graph models, Petri nets, parallel flowcharts, and flow graph schemata. As a last comment, let us point out the existence of important activities on the American west coast during this period, particularly at UCLA under the leadership of G. Estrin. Among numerous works can be found the Ph.D. of Cerf (1972), one of the fathers of the Internet (Turing Award, 2004).

The significant number of results in the theory of Petri nets produced by the end of the 1970s (Brauer, 1980) can be viewed as an integration of the two lines deriving from Petri's first proposal (the system theory – more properly inspired by Petri – and the automata approaches) and of the above-mentioned VAS and other graphical models for parallel computations, independently introduced in America in the late 1960s. Observe that the dual views of research in America also have a parallel in Europe. For example, results of a collaboration between Hamburg and Paris can be seen in Valk and Vidal-Naquet (1977), while in Paris the group led by Claude Girault (Univ. Paris VI) or Joseph Sifakis's group at Grenoble were already working on what was identified as system perspective, with fundamental contributions to the "structure theory" of Petri nets. In parallel, in America other researchers were following on from the developments of Holt and MIT. For example, Tadao Murata, with a circuit theory and automatic control background (Murata, 1977), meaningfully became interested in Petri nets in 1975 after reading Hack (1972); or Mike Molloy, one of the originators of basic stochastic timing in Petri nets (Molloy, 1982).

In the 1970s, Robin Milner focused his approach of concurrent systems on the "interactions of smaller components" (Milner, 1980), one of the more celebrated steps in the definition of process algebras. In other words, Milner paid central attention to the process of construction of the model. Moreover, he also rejected the possibility of limiting the approach to computer systems, searching equally for minimality in the number of basic concepts. In his Turing Award lecture (1991), Milner (1993) recognized that several of these questions were already understood by Petri in the sixties, who

> [...] pioneered the scientific modeling of discrete concurrent systems. Petri's work has a secure place at the root of concurrency theory. He declared the aim that his theory of nets should – at its lowest levels – serve impartially as a model of the physical world and in terms of its primitive constructions. What I always wanted to advance, to complement Petri net theory, is the synthetic or compositional view of systems which is familiar from programming.

The view proposed by Milner was algebraic, but this perspective can also be integrated in the Petri net theory, keeping what is sometimes called the "architecture" of the model; in other words, the trace of how the model was constructed. An attempt to combine process algebra and Petri nets is considered in Best, Devillers, and Koutny (2001); an algebraic compositional approach to the construction of Petri net models appears in Huang, Jiao, Cheung, and Mak (2012).

While bipartition at graph level was a salient feature in Petri's approach, it is not (explicitly) present in Milner's. Nevertheless, the use of bipartite graphs was not an exclusive feature of Petri nets. Other modeling approaches at the beginning of the sixties used it. For example, in queueing networks (QNs), queues – as places – are static containers of clients, while servers in stations provide the services. Additionally, the *Forrester Diagrams* (Forrester, 1961), also meaningfully referred to as *stock and flow diagrams*, use deposits or stocks as "warehouses", the state-values being levels, while flows pass through "valves". Introduced for continuous "views" of systems, Forrester Diagrams do not have discrete counterparts. Both QNs and Petri nets lead to relaxed "views" by means of fluidization, allowing certain analyzes to be computationally feasible (see Section 6).

The last time I met Carl Adam Petri was in 2005 at the 26th Int. Conf. on Applications and Theory of Petri nets, in Miami. On that occasion we had the honor of giving a keynote speech on "Continuization of Timed Petri nets: From Performance Evaluation to Observation and Control". Therefore, we (re)discussed fluid "views" of discrete models. Once again, he told me about the importance of the *duality* of reactants and reactions, which in time suggested the idea of the separation of places from transitions to him. In an enjoyable conversation I claimed that my game was "in moles, not in molecules, so I was able to consider fractions of moles". Affectionately, he stated that my interest in fluid net systems was "not surprising", because of my degree in Industrial–Chemical Engineering from the Universidad de Sevilla. Carl Adam Petri always was a warm-hearted person.

2.3. An aerial perspective on the field

Surprisingly perhaps, Petri was not prolific in terms of published works: only some 35,[9] several belonging to the so called "gray literature". But the influence of his concurrency theory is impressive. This can be seen in various ways. For example, "The Petri Nets Bibliography",[10] that contains entries until 2006 only, has more than seven thousand items. Nevertheless, it is uncertain how representative this lower bound is. It may in fact represent less than 10% of the total.[11] Another perspective comes from conferences. We have

[9] http://www.informatik.uni-hamburg.de/TGI/mitarbeiter/profs/petri_eng.html. All web references were last accessed on March 24, 2013.

[10] http://www.informatik.uni-hamburg.de/TGI/pnbib/.

[11] Searching in *Google Scholar* "Petri net" and "Petri nets", more than a quarter of million of entries are found, of course with redundancies and absences.

already mentioned the two at MIT in 1970 and 1975. In some sense a symbol of the taking over of the task on the European side, one of the first monographic meetings in the field in France was the Journées AFCET sur les Réseaux de Petri (Toulouse, April, 1976), organized by Michel Diaz and Claude Girault. But the most significant inflexion point was, without any doubt, the Advanced Course on General Net Theory of Processes and Systems organized in Hamburg in 1979.

Since 1980 there has been an annual meeting devoted to Petri nets. Under the name European Workshop on Application and Theory of Petri Nets (EWATPN), the first one was organized by Claude Girault (Paris) and Rüdiger Valk (Hamburg) in Strasbourg (France). Selected papers of the first two meetings (also Bad Honnef, West Germany, 1981) were compiled in (Girault & Reisig, 1982). From 1989 this series of meetings was renamed as International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN). Essentially from a computer science perspective and specially in the first decades, a significant part of the core of contributions to Petri nets theory has been presented in this series of meetings. Moreover, under the leadership of Grezgorz Rozenberg, a subseries of Lecture Notes in Computer Science (LNCS) named *Advances in Petri Nets* was started in 1985. Initially, the volumes were compiled by complementing selected papers from one or two consecutive meetings (i.e., EWATPN or ICATPN) with additional invited articles (Rozenberg, 1985–1991, 1993). Other volumes of this subseries were devoted to some advanced courses (to be commented on later), or to diffusing the results of relevant Petri nets-centered European research projects, such as (Rozenberg, 1992).

Starting in 1993, the subseries of Lecture Notes in Computer Science devoted to the proceedings of the ICATPN, *Application and Theory of Petri Nets*, has appeared on a yearly basis. After 1999, the issues of *Advances* were devoted to special theoretical or applied topics (see Subsection 7.1). The *Transactions on Petri Nets and Other Models of Concurrency* (ToPNoC), a journal mainly comprising selected papers from satellite workshops around ICATPN, has been published since 2008; it is a subseries of Lecture Notes on Computer Science of which kurt Jensen (2008–2012) has been the editor-in-chief till this year.

From 1985 till 2003, ten meetings of the IEEE/ACM Int. Symp. on Petri Nets and Performance Models (PNPM) took place (celebrated in Torino, the first one was named IEEE/ACM Int. Symp. on Timed Petri Nets). Selected papers from these meetings were frequently published as special sections in the *IEEE Transactions on Software Engineering*. This series of meetings merged in 2004 with others, such as that on Process Algebra and Performance Models (PAPM), inspired by PNPM, and giving light to the annual Int. Conf. on Quantitative Evaluation of Systems (QEST). In addition, even more impressive is the number of meetings in which Petri nets is an explicitly mentioned topic, with special sessions frequently being devoted to them. For example, "The Petri Nets: Meetings and Events"[12] lists more than 150 during 2005–2011, more than twenty per year. In fact this would require some correction, because in the above list of "Meetings and Events" some prestigious ones such as the Workshops On Discrete Event Systems (WODES) and the American Control Conferences (ACC) appear only once, while the Conferences on Decision and Control (CDC), the European Control Conferences (ECC) or the IFAC World Congresses do not even appear at all! This may be partly understood because of the relative importance given to computer science/engineering conferences on that site with respect to automatic control and operations research.

As a complementary consideration, let us point out the existence of many journals that have devoted at least one special issue or special section to Petri nets. Among many others: *Discrete Event Dynamic Systems, Nonlinear Analysis: Hybrid Systems* (under preparation), *In Silico Biology, Performance Evaluation, Theoretical Computer Science, The International Journal of Advanced Manufacturing Systems, TSI-Technique et Science Informatiques, Asian Journal of Control, Transactions of the Institute of Measurement and Control* (Journal of Biological Systems Modeling and Simulation), *Natural Computing*, or several *IEEE Transactions (on Industrial Electronics; on Software Engineering; on Systems, Man, and Cybernetics (Part C), etc.)*. Some international advanced courses have also played an important role in the diffusion of the theory and its applications. The first, organized in Hamburg in 1979, led to (Brauer, 1980). In the preface, Wilfried Brauer points the new needs:

> Complex organizations and their behaviors cannot be adequately described by classical sequential system models; the problems related to concurrency of actions of different subunits, to conflicts between local and global goals, to limitations of resources, to different levels of exactness of descriptions [...] necessitate new approaches.

Inserted in the subseries of *Advances in Petri Nets* are the proceedings of three other two week courses (Brauer et al., 1987a, 1987b; Desel et al., 2004; Reisig & Rozenberg, 1998a, 1998b). Advanced courses, fruits of a Human Capital and Mobility (HCM) European research project are Girault and Valk (2003) and Balbo and Silva (1998), while Seatzu et al. (2013) is the result of a Specific Targeted Research Project (STREP) of the European Union. Moreover, there are software tools and international standards for the use of Petri nets in industrial environments. All the above is evidence that from the seed sown by Carl Adam Petri, a big tree has flourished. As recognized three decades ago and true today, "the theory of Petri nets has been developed by a number of people working at different times in different places with different backgrounds and motivations" (Peterson, 1981).

3. Autonomous Petri Nets: from relevant features to abstraction levels

Petri net is a generic term used to designate a family of related DEDS formalisms, all sharing some basic relevant features as *minimality* in the number of primitives, *non-determinism*, *locality* of the states and actions (with consequences over model building and structuration) or *temporal realism*. The global state of a system is obtained by the juxtaposition of the different local states. The occurrence or firing of a transition captures the basic notion of behavior of the model and its effect on the local states. Other notions of system behavior can be captured by a Petri net model when occurrences of sets of transitions are considered, expressing concurrency, synchronization, choice or total ordering. Local state variables (*places*), state transformers (*transitions*) and the declared relations between them define the *structure* of the Petri net model.The value of a local state variable is called its *marking*. The global state of the system, the marking of the system, is the concatenation of the marking of the different places.

At a first level we should distinguish between *autonomous* and *interpreted* formalisms. The first group deals with fully non-deterministic objects, its evolution depends only on the marking; models of the second group are obtained by restricting the behaviors by means of constraints that can be related to different kinds of external events, to time in particular. The presentation in this section starts focusing on the so called *Place/Transition* nets (PT-nets), an intermediate level, by default usually called Petri net. Later, other models – simpler or more complex – are introduced. Interpreted formalisms are considered in Section 5. Because of the extremely large number of concepts and techniques (for modeling, analysis, synthesis and implementation) it is outside the scope of this over-

[12] http://www.informatik.uni-hamburg.de/TGI/PetriNets/meetings/.

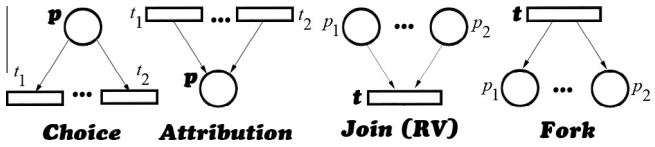

Fig. 2. The logical OR is present around places, in *choices* (or branches) and *attributions* (or meets); the logical AND is formed around transitions, in *joins* (or waits) and *forks* (or splits). *Branches, meets, waits* and *splits* are the names given by Konrad Zuse (1980), a computer pioneer who in 1941 constructed the world's first functional program-controlled computer, Z3.

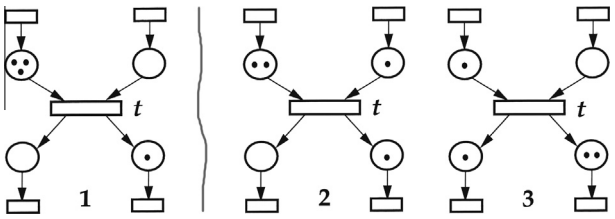

Fig. 3. Transition t is not enabled in case 1; it is enabled in case 2, and its firing leads to the marking in case 3.

view to provide a "reasonably complete" bibliography on Petri Nets (PNs).

The modeling of DEDS by means of Petri nets has been a recurrent topic in the literature. Among the early surveys, we find Peterson (1977), written with a certain computer science flavor; and Agerwala (1979) presented in more of a basic tutorial style. To the best of our knowledge, following an initiative by Joseph Sifakis (Turing Award, 2007), the first overview of PNs dealing with modeling, analysis and the implementation of logic controllers (in the context of safe design) is Moalla, Sifakis, and Silva (1980). What can be described as a second generation of broad views is the well-known extensive survey by Murata (1989), also Silva (1993) and David and Alla (1994). Half tutorial-half survey, with examples of applications, the second work presents a simple but technical view of different PN-based formalisms at the PT-net level and their analysis techniques. The third work, also essentially devoted to PT-nets and some extensions by interpretation, places relative emphasis on the Grafcet (see Section 7.2) and continuous and hybrid net systems (see Section 6). Concerning books, the first broad view of the knowledge of the time is the proceedings of the Advanced Course on General Net Theory of Processes and Systems of Hamburg (Brauer, 1980). Immediately after this came Starke (1980), Peterson (1981), Brams (1983), Reisig (1985), and Silva (1985); as proof of the rapid dissemination of this knowledge, it can be observed that they are written in English, French, German and Spanish. Approaching the present day and written with complementary perspectives, relevant works include, inter alia, DiCesare, Harhalakis, Proth, Silva, and Vernadat, 1993, Jensen (1997), Reisig and Rozenberg (1998a, 1998b), Girault and Valk (2003), Diaz (2009), Jensen and Kristensen (2009), David and Alla (2010), Haddad et al. (2011), and Seatzu et al. (2013).

3.1. Structure (net) and distributed state (marking): some basic comments

It is not the purpose of this subsection to present a more or less detailed explanation of what Petri nets are. There are many textbooks providing such explanations. The purpose of this subsection is very modest: merely to try to describe some elements in order that those readers relatively unfamiliar with the topic can have a general basic idea of some of the properties of this kind of formalism. Because of its central role, we approach here the level of so-called Place/Transitions nets (PT-nets).[13]

Petri nets can be represented as *bipartite graphs*, where "state variables" are called *places* (p),[14] and "state transformers", *transitions* (t). Places are usually represented by circles, while transitions are represented by bars or rectangles. The set of places (P) and transitions (T) are connected through *pre-incidence* (or *input*), and *post-incidence* (or *output*) functions defined in the naturals, ***Pre*** and ***Post***. Function ***Pre***(***Post***) defines the connections from places to transitions (transitions to places). The weight of the arc from a place to a transition represents the required number of resources (to be consumed!), a *precondition*, for the occurrence of the transition. Analogously, an arc from a transition to a place represents what is produced by the occurrence of the transition, the *postcondition*. These central notions make this family of models specially suitable for the modeling of concurrent and distributed DEDS. Functions ***Pre*** and ***Post*** can alternatively be defined as *weighted flow* relations (nets as graphs) or as two *incidence matrices* (which allow the use of algebraic techniques). The net structure represents fixed relations, the *static* part of the model.

Over the set of places is defined the "distributed state", the *marking*. It is numerically quantified (not in an arbitrary alphabet, as in automata), associating natural values to the local state variables, the places. If a place p has a value v ($\boldsymbol{m}(p) = v$), it is said to have v *tokens* (frequently depicted in graphic terms putting v black dots inside the place, or just the number). Clarifying, the places are "state variables" while the markings are their "values"; the *global state* is defined through the concatenation of local states. The net structure provided with an initial marking is a *Petri net system*, or *marked Petri net*, a model of a DEDS.

Summarizing, a net (structure) can be viewed as $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post}\rangle$, where:

- P and T are disjoint and finite sets of places and transitions, respectively.
- ***Pre*** and ***Post*** are $|P| \times |T|$ sized, natural valued (zero included), incidence matrices. The net is said to be *ordinary* if ***Pre*** and ***Post*** are valued on $\{0,1\}$. Weighted arcs permit the abstract modeling of *bulk* services and arrivals.

A PN system will be denoted as $\langle \mathcal{N}, \boldsymbol{m}_0\rangle$. Most basic PN constructions are depicted in Fig. 2. The last two (join and fork) do not appear in sequential automata; moreover, the arcs may be valued with natural numbers. Observe that the negation is missing in PT-nets. Its inclusion leads to the so called *inhibitor arcs*, an extension to be considered later.

The dynamic behavior of the net system (trajectories with changes in the marking) is produced by the firing of transitions. The firing of a transition is a "local operation", which follows very simple rules. Because the evolution of the marking means the evolution of tokens in places, these rules define what is usually called the *token game* (in some sense, "a game" similar to checkers, the chessboard being the net). A marking in a net system evolves according to the following *firing* (or *occurrence*) rules:

- A transition is said to be *enabled* at a given marking if each input place has at least as many tokens as the weight of the arc joining them.
- The *firing* or *occurrence* of an enabled transition is an instantaneous operation that removes from (adds to) each input (output) place a number of tokens equal to the weight of the arc joining the place (transition) to the transition (place).

[13] We mostly follow the lines set out in Silva (1993).

[14] From a notational point of view, Petri usually used S to represent the set of places, because they are "state elements" (it can also be interpreted as making reference to the German word "Stellen").

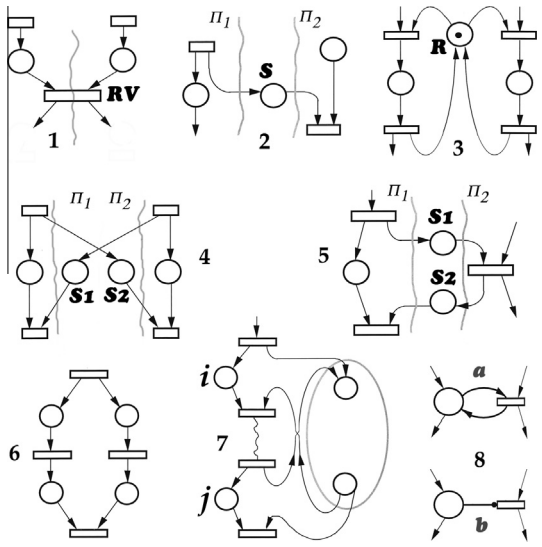

Fig. 4. Basic synchronization schemes: (1) Join or *Rendezvous*, *RV*; (2) *Semaphore*, *S*; (3) Mutual exclusion semaphore (*mutex*), *R*, representing a shared resource; (4) Symmetric RV built with two semaphores; (5) Asymmetric RV built with two semaphores (*master/slave*); (6) Fork-join (or *par-begin/par-end*); (7) Non-recursive *subprogram* (places *i* and *j* must be in mutex, to remember the returning point; for simplicity, it is assumed that it is single-input/single-output); (8) Guard (a) self-loop from a place through a transition, "similar" – but not exactly equivalent – to a *reading arc*, as denoted in (8.b); its role is like a traffic light, "somewhat" like a catalyst in chemistry: if present, it allows the evolution, but it is not consumed.

Fig. 3 shows a transition, *t*, its place environment, and the consequences, if it is enabled and fired.

According to what has been said, three important observations to be taken into account are:

- The underlying logic in the firing of a transition is non-monotonic! It is a *consumption/production* logic.
- Enabled transitions are *never forced* to fire: this is a form of non-determinism, a topic on which we will comment later.
- An *occurrence sequence* is a sequence of fired transitions $\sigma = t_1 \cdot \ldots t_k$. In the evolution from $\boldsymbol{m}_0$, the reached marking $\boldsymbol{m}$ can be easily computed as:

$$\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \quad \boldsymbol{m}, \boldsymbol{\sigma} \in \mathbb{N} \tag{1}$$

where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the *token flow* matrix (*incidence* matrix if $\mathcal{N}$ is self-loop free) and $\boldsymbol{\sigma}$ the firing count vector corresponding to σ.

The previous equation is the *state-transition* equation (frequently named as *fundamental* or, simply, *state* equation). Nevertheless, two important remarks should be made:

- It represents a necessary but not sufficient condition for reachability; the problem is that the existence of a $\boldsymbol{\sigma}$ does not guarantee that a corresponding sequence σ is firable from $\boldsymbol{m}_0$, thus certain solutions – called *spurious* (Silva, Teruel, & Colom, 1998) – are not reachable. This imply that most frequently – except in certain net system subclasses – only semi-decision algorithms can be derived by using the Eq. (1).
- All variables are natural numbers, which implies computational complexity.

With this conceptually simple formalism, concurrent or simultaneous firing of transitions can be modeled in a straightforward manner. Moreover, it is not difficult to express basic synchronization schemas (Fig. 4). In the illustrated examples

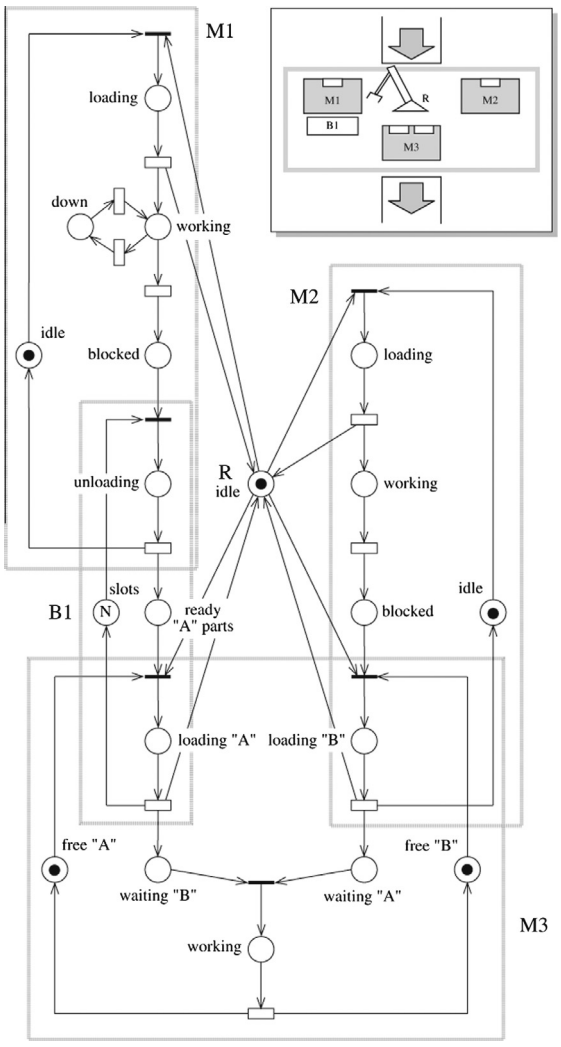

Fig. 5. A manufacturing cell: Basic concurrent DEDS model showing the interleaving of competition (for the robot) and cooperation relationships.

of synchronization, all schemes use *joins*; when *weights* are allowed in the arcs, another kind of synchronization appears: several copies of the same resource are needed (or produced) in a single operation. Being able to express concurrency and synchronization, when viewing the system at a higher level, it is possible to build *cooperation* and *competition* relationships. It should be pointed out that in automata theory, the state of a finite state machine is a "single variable, taking values in a symbolic unstructured set", while in PT-systems (i.e., PT-nets provided with an initial marking) it is structured as a vector of non-negative integers. Therefore, analysis techniques that do not require the enumeration of the state space are now possible.

Fig. 5 depicts a representation of a manufacturing cell composed of two processing machines (simple sequential automata, M1 and M2), an assembly machine (M3, like a cyclic PERT, with a rendezvous), a loading/unloading robot (*R*, that can be viewed as a mutual exclusion semaphore, or as defining another simple sequential machine), and a storage element (which can be viewed as two semaphores, *Slots* and *Ready "A" parts*, or as a *k*-marked cycle, thus it is not a sequential automata). The two processing machines and the assembly machine *compete* for the robot, while they *cooperate* in the production plan.

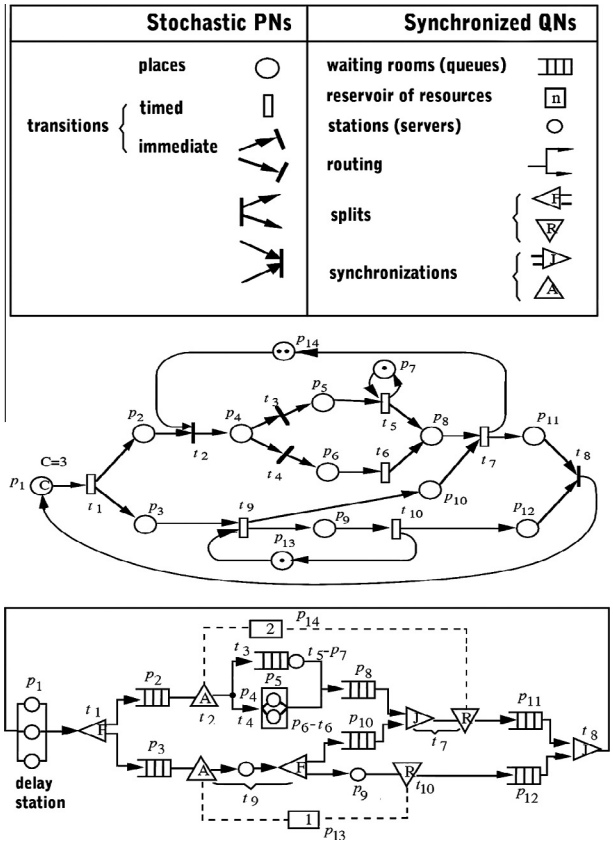

Fig. 6. A transition-timed stochastic PN and the corresponding extended queueing network. It is assumed that there exist no customers classes, routing is Bernouilli, and all services are exponential. The models represent a customer/server system with passive resources and synchronizations. The minimality of concepts in PNs is clear: places and transitions vs queues, delays, stations, routings, acquisitions, releases, forks, joins and reservoirs.

3.2. Some relevant features of Petri net models

Among the qualities that Petri nets enjoy are:

Minimality in the number of primitives. This is usually a must in the construction of any conceptual framework, raising a classical trade-off between the engineering and scientific perspectives: while engineers appreciate a rich ontology with different concepts suited for different purposes, scientists mainly look for basic underlying notions. Obviously, a diversity and specificity of primitives may be convenient in order to develop concise and elegant models, but generally this tends to make formal reasoning and theory construction more difficult. An ideal solution to conciliate reasoning capabilities and practical expressivity consists of having a minimal number of basic primitives in terms of which richer ones can be constructed. In this way, the basic PN formalism is quite spare, having only two simple and somehow orthogonal primitives (see Fig. 6). Observe that *queues* (for clients) and *reservoirs* (for resources) correspond to places; *stations* (where the services are provided), *routings*, *splits* (forks and releases, used resources being liberated in the latter case), and *synchronizations* (joins or rendezvous, and acquisitions of resources) correspond to transitions.

Locality and structuration. With PNs, the description is in *local* terms: local states (places) and local state changes (transitions). The locality of places and transitions is a central issue in PNs. It appears as a cornerstone for the construction of net models. For instance, it is possible to *refine* a place or a transition (i.e., to detail the model), or to *compose* two modules by the identification of shared transitions (*merging* two or more into one) or places (*fusion* operation). Refinements and modularity can be based either on states or actions, thanks to their treatment on an equal footing. In brief: both *top-down* and *bottom-up* modeling methodologies can be freely interleaved, opening the window to "team-based" model constructions.

Non-determinism: A humble position. In classical finite automata theory, non-determinism can be viewed as a means to obtain "smaller" descriptions of systems because of its "power to be in several states at once [...] an ability to 'guess' something about its input" (Hopcroft, Motwani, & Ullman, 2001). Another way of explaining the interest of non-deterministic finite automata is to accept that an event at a state may cause transitions to more than one new state because of "our own ignorance [Due to the fact that] sometimes, we cannot say with certainty what the effect of one event might be" (Cassandras & Lafortune, 1999). In both cases, the evolution of the non-deterministic automata is due to the existence of a sequence of input symbols.

At a conceptual level, concerned by implementation issues of switching diagrams, or by the coordination of automata, according to Carl Adam Petri non-determinism is based on the

> unjustified reliance upon precise time delays for switching and transmission. The only way out of this fundamental dilemma – it seemed to me – was the total renunciation of real-valued metric scales, and their replacement by the sole reliance upon combinatorial principles founded on natural laws.[15]

In other words, assuming the possibility of important temporal uncertainty-unpredictability, it can be understood as the starting point for a way to replace time-quantified reasoning about the dynamics of a system. This led him to search for special metrics such as the *Synchronic Distance* (SD; see, for example, Petri (1976) and Reisig (1985)); as a distance, it verifies the axioms of symmetry, positive definiteness and triangle inequality. According to Petri, the SD is "the" net *metrics*, it looks for degrees of dependence among the firing of (subsets of) transitions.[16]

Temporal realism and occurrence nets. The representation of concurrency by means of causal independence of transition occurrences is more faithful than in sequentialized views (the so called *interleaving* semantics). This has several consequences. On the one hand, it allows "temporal realism" in timed interpretations, which is crucial for (non-markovian) performance evaluation or control (scheduling, etc). A PN system underlies an event structure with causality and compatibility relations. Thus, it allows the use of *unfolding* techniques and *occurrence nets* (where the behavior is expressed by means of partial ordered sets) (Best & Fernández, 1988; Esparza, Römer, & Vogler, 2002; Reisig, 1985). Among other things, these techniques have the potential for reducing the computational burden of purely sequentialized approaches (e.g., reachability or coverability graphs), something better understood, for example, for 1-bounded systems.

3.3. On ways of approach to PT-systems and basic but powerful extensions

The classical proverb "all roads lead to Rome" is a clear recognition that Rome has been a very important – the most important – city in Europe for several centuries. In analogous sense, given that using different roads we may arrive at "the city" of PNs (with several nuances not to be considered here), this may be "interpreted"

[15] Carl Adam Petri, Discourse of the *Honoris Causa* Doctorate, Universidad de Zaragoza, April, 15th, 1999.

[16] Associating weights to the firing of transitions, the measure was later generalized. Nevertheless, firing dependencies exist that are not measured by weighted synchronic distances, but captured by *Bounded Fairness*, BF (Silva, 1987; Silva & Murata, 1992); see later Fig. 8.

as an indication of their relative importance. The axiomatic approach of Petri led to *Condition/Event* systems (CE-systems), where places (conditions) may have one token at most. Places represent boolean variables, thus the systems have a finite state space, i.e., are bounded. Revisiting CE-sytems are defined the *Elementary Nets* (EN): pure (i.e., self-loop free) CE-system with no repeated node and with no isolated element (Thiagarajan, 1987). The initial marking should be such that every event may be fired at some reachable marking.

The case of PT-nets is different. They are an extension, because the places are natural variables (Holt & Commoner, 1970) and thus the models may have an unbounded number of states. A second generalization in this direction is the association of weights to arcs, leading to what were in time called *generalized PNs*. Now, nets with weights on arcs are just PT-nets, while they are called "ordinary" if not weighted. PT-systems can be "(re)discovered" in different ways. The possibilities include:

- *analogy* with the state equation of continuous variable systems, which leads to *Vector Addition Systems* (as already said, VAS were introduced as particular mathematical structures by Karp and Miller); PT-nets are "found" in a didactic way in (Silva & Teruel, 1996).
- particular codings of state graphs, something which is computable by means of the *Theory of Regions* of a graph (Ehrenfeucht & Rozenberg, 1990).[17] This can be viewed as the "reverse" problem of the reachability graph construction for bounded net systems (from PN systems to state graphs, now from state graphs to PN systems). Embedded in the so called *synthesis problem* for nets, it consists in deciding whether a given finite deterministic automaton is isomorphic to the reachability graph of a net system, and then constructing it. In other terms, it can be viewed as dealing with a problem of *coding the state* of the automaton by means of places (the state variables). For bounded and self-loop free (weighted) PT-systems it is a polynomial time problem (Badouel & Darondeau, 1998). Apparently paradoxical, the synthesis problem for EN-systems is NP-complete (Badouel, Bernardinello, & Darondeau, 1997).
- using the "sequent calculus" of *Linear Logic*. This logic is a non-monotonic one that emphasizes the role of formulas as resources (Girard, 1987; Girard, 1995). Individual PT-nets form models of Girard's linear logic (see, for example, Martin-Oliet & Meseguer (1991)). In fact, the equivalence between the provability of certain sequents of linear logic (without the additive connectives, i.e., using the multiplicative fragment) and reachability in Petri nets has been shown in various ways. One quite "natural" way is to denote markings as some monomials – logical atoms are tokens – and transition firings by some implicative formulas; a reachability relation among markings is expressed by a sequent which has to be proven (Fanchon, Rivière, Pradin-Chézalviel, & Valette, 2003).

Although it is possible to simulate (implement) generalized PT-systems by ordinary ones preserving the (projected) language (i.e., identical expressivity under interleaving semantics (Hack, 1974b)), several reasons justify dealing with weighted PT-systems directly rather than with their "ordinary" simulations: the models are more concise, the transformations do not appropriately preserve concurrent semantics in general, and the ordinary implementations fall typically out of the subclasses which enjoy strong analytical results, even in the simplest cases.

PT-systems are able to model infinite state (i.e., unbounded) systems, and constitute a "frontier formalism". By this we mean that they are very expressive, while most classical properties are decidable (for example, reachability (Kosaraju, 1982; Mayr, 1984; Reutenauer, 1990)), but "small" extensions transform them into formalisms able to simulate *Turing machines*. Thus, excellent expressivity, but many decision problems can be undecidable. Among the most well-known extensions in autonomous net models are the addition of *inhibitor arcs* (or *zero tests*, where arcs are used that substitute a small circle for the classical arrow, a notation borrowed from logic diagrams), or of *priority levels* to the firing of transitions. The inhibitor arcs "simply add" the negation missing in Fig. 2. In their more basic form, they disable the transition when the input place is marked. PT-nets provided with inhibitor arcs (or transition priorities) have the same modeling power as Turing machines (Agerwala, 1975; Hack, 1975).[18] PT-nets with *reset* arcs (from a place to a transition, the firing of the transition empties the place) are not able to simulate Turing machines, but reachability is undecidable (Dufourd, Finkel, & Schoebelen, 1998). All three additions here mentioned (inhibitor arcs, priorities on transitions, and reset arcs) increase the (theoretical) expressiveness of PT-systems, introducing undecidabilities or increasing complexities for the analysis.

3.4. On abstraction: diversity of High-Level Petri Net (HLPN) models

The addition to PN models of the notion of *individuals* (for example, from anonymous to personalized, labeled or colored tokens) allows more abstract formalisms to be derived. Information in tokens allows the objects to be named (they are no more indistinguishable) and dynamic associations to be created. Abstraction from PT-nets to so called *High-Level PNs* (HLPNs) is something like "moving from assembler to high-level programming languages", or, at the computational level, like "moving from pure numerical to a symbolic level". Sometimes, this type of abstraction exhibits the same theoretical expressiveness as PT-nets; in other words, high-level views may lead to more compact and structured models, but keeping the same expressive power of PT-nets systems (i.e., we can talk of "abbreviations", not of "extensions"). A broad compendium on the topic published some two decades ago is (Jensen & Rozenberg, 1991); it "contains reprints of some of the most important papers on the application and theory of high-level Petri nets [... representing] the current *state of the art*".

Conceptually and historically speaking the two basic HLPNs formalisms are:

- *Predicate/Transition Nets* (PrT-nets) (Genrich, 1987; Genrich & Lautenbach, 1979, 1981). PrT-nets are PT-nets with annotations in a first-order language affecting the whole net, the places, the transitions and the arcs. The marking of the places is a symbolic sum of tuples of constants. Predicates allow us to express complex conditions on the selection of transitions to be enabled.
- *Colored Petri nets* (CP-nets) (Jensen, 1981, 1994; Jensen & Kristensen, 2009). CP-nets are based on a PT-net-like structure and a set of finite and non-empty set of data types, called *color* sets. The color function maps each place into a set of possible token colors. The evaluation of the arc expressions yields a multi-set over the color set that is attached to the corresponding place.

[17] A region is a subset of states such that equally labeled transitions either all enter it, or all stay within it, or all leave it, or all stay outside it. Regions in state graphs and places in PNs are in strong correspondence.

[18] Nevertheless, if the place at the origin of an inhibitor arc is bounded, this kind of arc can be removed by adding a "complementary" place and a guard or reading arc; i.e., if the place is bounded, it is a modeling convenience, an *abbreviation*, but it does not extend the modeling power.

The PrT-net systems represented a real breakthrough in modeling convenience (i.e., easiness of modeling). Somehow very close to PrT-nets, the CP-nets systems allow some more intuitive considerations, particularly in the present context. It is not easy to introduce that kind of formalisms in a so broad perspective as this one is. Just to informally refine a little the basic idea, let us consider that in a PT-net we split places and transitions into several disjoint groups (i.e., we consider partitions on P and T). Thus, both ***Pre*** and ***Post*** matrices can be "viewed" as formed by sub-matrices (domain restricted functions), not by natural numbers as in PT-nets. The transitions in a group (folded into a single transition in the CP-net) still play the role of a state transformer; the change in the place values produced by the occurrence of the transition is described by the function that is defined by the corresponding sub-matrix, where each column represents an *occurrence mode* of the folded transition. So a description on *two levels* is obtained: a colored place deals with a subset of PT-places, which makes up a *data type* (the same can be argued for transitions). The new places still play the role of state variables, but their values are now somehow *structured* (e.g., represented by a vector with as many components as there were original places in the element of the partition). Otherwise stated, there now exists an "explicit" or high-level net structure (i.e., the relation of colored places and colored transitions) and the "implicit" structure due to the functions attached to the arcs.[19] Of course, this kind of idea can be iterated at more than two levels. If the number of colors is finite, PT-systems and CP-net systems are equally expressive (i.e., they can model the same systems), but certain models can be expressed much more easily in the high-level formalism. From a different perspective, we can see colored Petri net systems as a formal net-based representation that instead of having several "identical" copies of a PT-subsystem (with "black" or indistinguishable tokens) a single colored-subsystem with colored tokens is used. Of course, token colors can be structured as cartesian products (lists, records, etc.) of more basic colors, what may allows very compact models. Colored net systems may be particularly helpful in systems with strong symmetries or in those systems in which the value of data strongly influence the global behavior.[20]

The above basic HLPN models have experienced several variants and extensions. Moreover, motivated either by specific application domains or by theoretical works, many other proposals have been made exhibiting specific modeling features and analysis methods. For example, in order to deal with sequential processes communicating by FIFO-channels, a model of parallel computation was introduced (Memmi & Finkel, 1985; Roucairol, 1986). As in CP-nets, in *FIFO-nets*, tokens are distinguished (in order to model different kinds of messages). Places behave as FIFO-queues, instead of counters: their markings are "sequences" of tokens rather than unordered collections. Then, the occurrence rule is modified to model the FIFO mechanism: tokens are removed at the head of the sequence and are added at the tail. FIFO-nets can simulate CP-nets. A subclass (alphabetical FIFO-nets) has the same algorithmic power as Turing machine, thus this model is an "extension" of PT-systems. Summarizing, in the previous HLPN models, tokens may receive *identity* and may be *ordered*.

Among other HLPN formalisms are "object oriented" net systems, a broad framework with different proposals (for a state of the art review, see Miyamoto & Kumagai (2005)). For example:

- *OBJSA-nets* (Battiston, De Cindio, & Mauri, 1988), where it can be said that the main interest is to provide object-oriented languages with a formal basis.
- *Cooperative Nets* (Sibertin-Blanc, 1994), where objects have distinguishing identity or name, and may store the names of other objects. This class deals with some object oriented concepts from programming languages into the PN formalism.
- Lakos (1995) is more interested in showing how the definitions of *Object-Oriented Petri Nets* (OOPNs) and *Object Petri Nets* (OPN) – supporting encapsulation, inheritance, polymorphism and dynamic binding – can be derived from CP-nets.
- *Nets within Nets* (Valk, 1991, 1998, 2003), in which tokens are nets (called *object nets*), and are within a net, called a *system net*. Object nets and the system nets are PT-nets. Object nets can move through a system net and interact with both the system net and with other object nets. These interactions allow changes of the marking of the object nets, but not of their net structure.

It is true that there exist many proposals for HLPNs. Despite their differences, they share basic notions, such as those considered in Subsection 3.2. These include the existence of token conservation laws or of structural objects that define repetitive sequences, etc. In consequence, concepts and results are frequently transferred – adapted or generalized – from some formalisms to others.

4. On analysis and enforcement of properties on autonomous net systems

The present section deals with the main lines in the study of basic model properties, even on the possibility of enforcing certain properties (when considering the control of a model). While fundamental, it is somewhat more technical than the previous sections and can be skipped in an initial reading.

4.1. Basic model properties: from analysis to subclasses of net systems

In general, the understanding of the behaviors of intricate concurrent systems, particularly with interleaved cooperation and competition relationships, is not easy. Thus the designer should appreciate answers to questions such as *reachability*: Given marking $\boldsymbol{m}$, is it reachable from $\boldsymbol{m}_0$? Other classical and basic properties are: (1) *boundedness* (is the state space finite?); (2) *mutex* (are the markings of some places in mutual exclusion?); (3) *deadlock-freeness* (does every reachable markings enables at least one transition? In mechanical terms, this is something like: is the engine not seizing?); (4) *liveness* (can all transitions – activities – always be fired sometime in the future?); (5) *reversibility* (does the net system always have the possibility to return to the initial marking?); etc. For decidability and complexity issues with respect to these or other problems see, for example, Esparza (1998) and Rosa-Velardo and de Frutos-Escrig (2010). The above are a few of the most frequently used generic behavioral properties. Additionally, the use of different *temporal logics* allows us to express special properties for the system, a task complementary to its modeling/specification. The literature on analysis techniques is extremely broad, and its consideration goes beyond the scope of the present impressionistic panorama. Among significant books dealing more or less in detail with these topics and others at PT-net level, see Brams, 1983, DiCesare et al. (1993), Reisig and Rozenberg (1998a), Girault and Valk (2003), Diaz (2009), and David and Alla (2010).

The main analysis techniques can be classified into three groups: (1) Enumeration, (2) Transformation (mainly reduction), and (3) Structural analysis. In the first case the work is performed

[19] In fact, in CP-nets two different representations can be used. The *function* representation uses linear functions between multi-sets; alternatively, more inspired by Pr/T nets, the *expression* representation employs arc expressions and guards. They are "equivalent", and formal translations can be done in both directions.

[20] Colored nets have an important impact in modeling industrial case studies (http://cs.au.dk/about/cpnets/industrial-use/). In June 2013, examples are structured in protocols and networks, software, workflows and business processes, hardware, control of systems or military systems, among other groups.

at a behavioral level; in the last case the work is directly done analyzing the model description.

Enumeration. Assuming the system is bounded (finite state space), this kind of approach is based on the idea of exploring the full state space, leading to the *reachability graph*, R-graph. Nevertheless, while providing the most detailed information about the behavior of the systems, the approach suffers from the *state explosion problem*, and non-trivial models may easily be too big for computer processing. Among the methods to "alleviate" the problem are the so called *stubborn set* and the *sleep set* methods (Karatkevich, 2007; Kristensen, Schmidt, & Valmari, 2006; Valmari, 1998). Complementary techniques deals with the identification of symmetries, where the idea is not to construct a full reachability space, but a condensed one (Starke, 1991). Usually used in the framework of High-Level Petri Nets (Section 3.4), the states that differ in permutations of symmetric components are condensed into one symbolic state (Chiola, Dutheillet, Franceschinis, & Haddad, 1997; Jensen & Kristensen, 2009). If the system is not bounded, a finite *coverability graph* can be computed (Finkel, 1993; Geeraerts, Raskin, & Begin, 2007; Karp & Miller, 1969). As already advanced in Section 3.3, reachability is decidable (see, for example, Reutenauer (1990)), but it cannot be decided with the coverability graph; nevertheless, the *marking covering* problem (given marking $\boldsymbol{m}$, exist $\boldsymbol{m}' \geqslant \boldsymbol{m}$ reachable from $\boldsymbol{m}_0$?) can be decided.

Firing sequences of an unbounded PN system can be matched by a "walk" inside the corresponding coverability graph, but the reverse is not true. This leads to important weaknesses such as the following one: two different net systems may have the same coverability graph, one system being deadlock-free, the other not (Peterson, 1981). Another problem with the R-graph is that it basically provides a "sequentialized" view of the behavior of the system; moreover, any change in the initial marking obliges the full R-graph to be recalculated. As a last point, let us comment that once the R-graph is computed, strategies to prove the properties (sometimes expressed in some *Temporal Logic*) are needed. *Model Checking* techniques provide important improvements in this respect.[21]

Transformation (in most cases based on a set of *reduction rules*). The idea is to convert the PN system into another one with the same properties being studied in the original case, but simpler to analyze (see, for example, Berthelot (1986)). Most practical transformation techniques are defined on the basis of the net structure, the marking being a parameter. If the transformations are computationally "cheap" (e.g., feasible in polynomial time), and the transformed system is computationally much easier to analyze (a much smaller state space, or belonging to a certain subclass of net systems, for example), then the full process may be very efficient. This kind of approach is related to rewriting techniques in computer science. The limitation of transformations approaches resides in the fact that, given a set of transformation rules, the transformation power is limited. Increasing the number of rules increases the transformation power, but then the application becomes more expensive. All in all, in practice, some systems cannot be appropriately reduced, and no final decision can be obtained. Observe that if the application of reduction techniques is inverted, the result is a *stepwise refinement*.

Structural analysis. Proper of PNs, this kind of approaches put the focus on the graph structure of the net, or on its state equation [eq. 1]. Net-driven approaches, the initial marking is now a parameter. In the first case, graph theory objects and techniques are used (circuits, strongly connected components, siphons, traps, handles, bridges, etc.); in the second one, linear algebra/mathematical programming approaches are important. This last kind of approach was introduced in Lautenbach and Schmid (1974), and among other early important works are Sifakis (1978) and Memmi and Roucairol (1980). Most frequently, semi-decision algorithms are available (for example, a sufficient condition for deadlock-freeness, a necessary condition for reversibility, etc). Among others, three kinds of strongly related notions that must be differentiated are:

- *P-semiflows* and *T-semiflows*, that are natural vectors;
- *Token conservation* and *repetitive behaviors*, that are invariant laws; and
- *Conservative* and *consistent* components, that are subnets generated by the subset of nodes in the support of P- and T-semiflows, respectively.

The *support* of a vector $\boldsymbol{v} \geqslant \boldsymbol{0}$ is $\|\boldsymbol{v}\| = \{v_i | v_i > 0\}$, the set of positive elements of $\boldsymbol{v}$. Left ($\boldsymbol{y} \cdot \boldsymbol{C} = 0$) and right ($\boldsymbol{C} \cdot \boldsymbol{x} = 0$) natural annullers of the token flow matrix are called P-*semiflows* and T-*semiflows*, respectively. A semiflow is *minimal* when its support is not a proper superset of the support of any other semiflow, and the greatest common divisor of its elements is one.

P-semiflows establish some *token conservation* laws, i.e., if $\exists \boldsymbol{y} \gneq \boldsymbol{0}$ then, given an arbitrary $\boldsymbol{m}_0$, $\boldsymbol{y}^T \cdot \boldsymbol{m}_0 = \boldsymbol{y}^T \cdot \boldsymbol{m}$ for every reachable marking $\boldsymbol{m}$. The P-subnet generated by the support of a P-semiflow is called a *conservative component*, meaning that it is a part of the net that conserves its weighted token content. On the other hand, T-semiflows identify potentially cyclic behaviors in the system, i.e., if $\exists \boldsymbol{x} \gneq \boldsymbol{0}$ s.t. $\boldsymbol{C} \cdot \boldsymbol{x} = \boldsymbol{0}$, and σ is a firing sequence whose firing count vector is equal to $\boldsymbol{x}$ (i.e., $\boldsymbol{\sigma} = \boldsymbol{x}$), the initial marking is recovered.

For example, the net in Fig. 5 has six minimal P-semiflows. For one of those the support is:

$$\|\boldsymbol{y}\| = \{unloading, ready\ \text{"}A\text{"}\ parts, loading\ \text{"}A\text{"}, slots\}.$$

Considering the initial marking in the figure, observe that $m[unloading] + m[ready\ \text{"}A\text{"}\ parts] + m[loading\ \text{"}A\text{"}] + m[slots] = N$ for any marking $\boldsymbol{m}$ reachable from $\boldsymbol{m}_0$. The set of places in that support define the buffer B1 (associated to machine M1). The same net has two minimal T-semiflows, one concerning the input and output transitions of place *down*, modeling the failure and repair of machine M1. The second one deal with the failure-free behavior (in it all components are equal to one, i.e., $\boldsymbol{x} = \boldsymbol{1}$). Thus, the net being considered has two T-components; moreover, if every transition in any of the two T-components is fired once, the system returns to the initial marking.

Invariant laws (conservation principles) for information processing and communication appear in "analogy" with physics or chemistry. Conservative and consistent components provide interesting decomposed views of net systems, useful for analysis, controller synthesis or implementation issues.

A major limitation of the state equation method for analyzing net systems is that those descriptions of the behavior are, in general, relaxations, and there exist non-reachable solutions (the so called *spurious solutions*). Nevertheless, in some net subclasses spurious solutions do not exist, or if they exist are not problematic with respect to certain properties (for example, in some net system subclasses spurious solutions cannot be deadlocks); moreover, techniques are available to remove some spurious solutions, improving the quality of the possible analysis.

Two interesting graph-based structural concepts that generate additional kinds of invariant laws are siphons and traps. Given a node $v \in P \cup T$, its *preset*, ${}^\bullet v$, is defined as the set of its input nodes, and its *postset*, $v^\bullet$, as the set of its output nodes. For example, in Fig. 2, ${}^\bullet t_1 = \{p\}$ in the "choice", while $t^\bullet = \{p_1, p_2\}$ in the "fork". These

[21] E.M. Clarke, E.A. Emerson, and J. Sifakis won the 2007 Turing Award for transforming model checking "from a theoretical technique to a highly effective verification technology that enables computer hardware and software engineers to find errors efficiently in complex system designs". Among the first topics on the field, using Interpreted Petri nets, as an intermediate language, and model checking, see, for example, Queille and Sifakis (1982).

definitions can be naturally extended to sets of nodes. A set of places Σ is a *siphon* if ${}^\bullet\Sigma \subseteq \Sigma^\bullet$. To simplify things, assume the net is ordinary (i.e., all arc weights are one). Inversely, a set of places Θ is a *trap* if $\Theta^\bullet \subseteq {}^\bullet\Theta$. According to the previous concepts, two new kind of invariant laws are: (1) if a siphon is empty or is emptied (i.e., there exist no token in the places), then it would always remain empty (thus all the transitions in its postset will be non-live); and (2) if a trap is marked or become marked, it would always remain marked (i.e., it would never be emptied). Analysis techniques based on graph-defined objects such as siphons and traps can also be computed by means of linear algebra. For a broad perspective on this field of structural analysis and linear algebra/mathematical programming, see Silva et al. (1998). Last but not least, it is important to observe that the token flow matrix enjoys several interesting rank properties allowing liveness to be analyzed, as detailed in Recalde, Teruel, and Silva (1998b).

The existence of the three main analysis approaches (enumeration, transformation and structural) means that neither one provides fully satisfactory answers in isolation. Nevertheless, the combined use of enumeration, transformation and structural analysis techniques frequently leads to satisfactory results in engineering practice. The transformation rules are computationally very cheap and frequently very efficient. Thus a kind of "meta-rule" in any analysis strategy may be: before anything else, try to apply as much as possible the reductions rules. Additionally, *simulation* of autonomous models refers to techniques that straightforwardly increase confidence about correctness, but its help is mainly in "token game animation" (a basic help to understanding) or in finding some counterexamples or bugs, not in proving properties. The nondeterminism of concurrent/distributed systems makes simulation a non-easy task.

Dealing with the tradeoff between modeling generality vs complexity of analysis (the more general the model, the less amenable to analysis), with differential equations it is customary to deal with special subclasses (for example, ordinary differential equations, among which the most popular are linear invariant equations; in continuous control theory, for example, the Ricatti equation is well known). The same occurs in PN theory, and certain *subclasses* of PNs are able to model interesting types of systems in practice, being easier to analyze. Usually done by imposing structural constraints to the way in which places and transitions can be connected, these restrictions give rise to constraints in the possible behavior of the different net systems (after defining an initial marking). This is an important difference with automata based models, whose structure coincides with that of one and only one exhibited behavior.

The understanding of the relationships between the behavior and the structure of special subclasses of net systems is of great interest. In the case of PT-nets, among well-known subclasses are *ordinary* nets (unweighed models). The most basic and dual subclasses of ordinary nets are: *State Machines* (SM, forks and joins are not allowed) and *Marked Graphs* (MG, choices and attributions are not allowed). *Free-Choice Nets* (FC, the choices are not externally conditioned) generalize the previous ones (Desel & Esparza, 1995; Hack, 1972). Allowing arc weights, the generalization of FC leads to *Equal Conflict Nets* (EQ, see, for example, Teruel & Silva, 1993 and Teruel & Silva (1996)). In other cases, structural restrictions are complemented with constraints on the possible initial markings, as in

- *Deterministic Systems of Sequential Processes* (DSSP), a generalization of EQ systems (Recalde, Teruel, & Silva, 1998a). In DSSPs the structure of the net is *not flat*; there exist two levels: 1-safe SMs and destination-private *buffers* interconnecting the SMs. Otherwise stated, DSSPs model distributed systems in which SMs *cooperate* asynchronously through the buffers. With the same kind of ideas in mind, a recursively defined superclass in Recalde, Teruel, and Silva (2001).
- *System of Simple Sequential Processes with Resources* (S3PR) and some generalizations (Colom, 2003; Ezpeleta, Colom, & Martínez, 1995; Park & Reveliotis, 2001). Here the net modules of the system are interconnected by means of places representing *shared resources*. The present subclasses of net models, also defined at two levels, are particularly appropriate to model *resource allocation systems* (RAS), where modules mainly *compete*.

As an example of *synergy* between the theories of formalisms belonging to different levels of abstraction, let us just mention that basic concepts and techniques for the analysis were immediately transferred from PT-nets to CP-nets, even if at the computational level, nice and profound developments were needed. Enumeration, transformation and structural methods in PT-systems have been transposed and adapted to HLPNs, for example to CP-nets (Diaz, 2009; Haddad & Pradat-Peyre, 2006; Jensen & Kristensen, 2009; Jensen & Rozenberg, 1991).

4.2. About control, observation and diagnosis, identification and synthesis

In verification techniques the goal is to ensure that a given system is correct with respect to its specification. More than analysis, control leads to synthesis problems; the idea is to enforce the given system in order to fulfill a specification (for example, to enforce certain mutual exclusion properties or deadlock-freeness). Technically speaking, some elements are "added" to constrain the behavior of the original model in such a way that a correct execution is obtained. To the already mentioned basic properties to analyze (reachability, boundedness, etc.) many others are now added. In particular, there are properties inspired by Control Theory, combining analysis and synthesis problems. The main questions relate to control, observation, diagnosis or identification, all areas of ongoing research. Of course, we do not claim that these problems concern control theorists and engineers only!

With respect to classical control theory, now there exist two main differences: models are discrete event and untimed (autonomous, fully non-deterministic, eventually labeling the transitions in order to be able to consider the PNs languages). Let us say that for control purposes the transitions should be partitioned into *controllable* (when enabled, you can either force or block the firing) and *uncontrollable* (if enabled, the firing is non-deterministic). Similarly, transitions (or places) can be partitioned into *observable* and *unobservable*. Related to observability, diagnosis is the process of detecting a failure (any deviation of a system from its intended behavior) and identifying the cause of the abnormality.

The most classical approach to control is to directly design (directly conceived as an algorithm) and implement the controller for the plant. An alternative point of view is to model the plant (by means of a PN), and subsequently synthesize a controller, using some theoretical control techniques (see Holloway, Krogh, & Giua (1997), for an early overview). Obviously, the point of inspiration is the theoretical controller synthesis paradigm for continuous systems: Given a model of the plant dynamics (**P**) and a specification for the desired closed-loop system (**S**), the goal is to *synthesize a controller* (**L**) such that **S** equals the parallel-composition of **P** and **L**; in other words, controllers (called "supervisors") are designed to ensure that only behaviors consistent with the specification may occur. The previous equality is not always possible, and the goal is usually relaxed to minimally-limit the behavior within the specified legality (i.e., to compute *maximally-permissive* controllers). Based on the *duality* of places and transitions in net models,

the focus may be on two complementary approaches: *state feedback* and *event feedback* control. In the first case, the idea is to force the system to remain in a specified set of allowed states, or legal markings; in other words, to prevent it from reaching certain *forbidden* markings. In the second case, using PNs with labeled transitions, the control problem is considered in a *formal language* setting, as in the basic framework of *Supervisory Control* (Ramage & Wonham, 1989): the specification is given as an acceptable language, thus a "sequentialized" specification of the legal behavior is provided (for a recent perspective on those language-based approaches, see, Giua (2013)). The issue of distributed control based on Petri nets is considered in Darondeau and Ricker (2012), where "a survey of DES control for PN researchers and a survey of distributed PN synthesis for DES researchers" are provided.

Control approaches rooted in the structure theory of PNs (e.g., on the fundamental equation) deal with problems such as imposing constraints in the net based model to verify certain complex mutex properties (moreover, keeping others such as liveness). In this context, *Generalized Mutual Exclusion Constraints* (GMECs) are straightforwardly implemented by means of distinctive controllers (*monitor* places (Giua, DiCesare, & Silva, 1992)). If the original model is non-live (e.g., has a deadlock) or non-liveness is introduced by the added monitors, for example, then the so called *liveness-enforcing* problem is raised (in fact, the question is to additionally restrict the behaviors to exclude non-liveness). Books dealing with these kind of structural techniques (later considered as *supervision based on place invariants*) for the control of Petri nets, eventually with uncontrollable and unobservable transitions, are Moody and Antsaklis (1998), Stremersch (2001), Iordache and Antsaklis (2006), and Li and Zhou (2009). In many cases, the main goal is to ensure that deadlocks can never occur. This problem is very important in *Resource Allocation Systems* (RAS), where the strong competition for resources is crucial (Colom, 2003; López-Grao & Colom, 2013; Nazeem & Reveliotis, 2012). For this kind of problems, liveness may be enforced by adding certain monitor places computed by means of "siphon-based" analysis (a siphon is a subset of places that, if they become unmarked, will remain unmarked, i.e., they cannot be refilled with tokens), a distinguishing feature in PN theory.

A classical complementary problem in control is *observation*, i.e., to reconstruct the state of a PN based on the appreciation of the occurrence of certain events, or on partial marking observation. Related to these problems are the proper control problems on the one hand and the diagnosis problems on the other. Based on event observation, assuming that the net structure is known and the initial marking is (partially) unknown, in Giua and Seatzu (2002) the marking of a PN is estimated. An analogous kind of problem for a labeled Petri net system is considered in Ramírez-Treviño, Rivera-Rangel, and López-Mellado (2003); moreover, models in which some transitions can be labeled with the empty string (*silent* transition, i.e., unobservable) are considered in Giua, Seatzu, and Corona (2007), where the notion of *basis marking* is introduced (a subset of markings consistent with an observation of sequences of minimal length; all other markings consistent with the observation can be obtained from the knowledge of those markings). At this level we cannot enter into the many technicalities involved, in particular because the assumptions made in the many different works are not the same and may even be rather different. Unfortunately, to the best of our knowledge, no wide-ranging survey on observation has been published (Giua, 2011 is an extended abstract providing a brief and non technical overview).

A recent state of the art survey of research focussed on *fault diagnosis* is Zaytoon et al. (2013). As we only deal here with "model-driven engineering", we only mention *model-based* approaches in fault diagnosis. Roughly speaking, in order to detect, isolate and identify the faults, the idea is to compare the behavior of the system and the model. This work can be done using *fault-free* models (i.e., those only concerned with the normal behavior) or more complex models in which "some" faults are explicitely taken into account. The use of fault-free models can be based on ideas such as, for example, Hamming distance in a code (the marking code) or comparing systems with a "reduced" model (for historical reasons, because introduced in the context of *fault-tolerant* implementations, these kinds of ideas are considered in Section 5.1, when dealing with *logic controllers*). The advantage of using models that consider certain faults increases (but does not guarantee) the possibility of locating those faults. Among the range of proposals dealing with PN models, we can cite works such as Dotoli, Fanti, Mangini, and Ukovich (2009), where *integer linear programming* is used, or Basile, Chiacchio, and Tommasi (2009) where emphasis is placed on mathematical programming in *on-line* computations. PN *structural techniques* are also employed in Ramírez-Treviño, Ruiz-Beltrán, Rivera-Rangel, and López-Mellado (2007), Lefebvre and Delherm (2007), Ramírez-Treviño, Ruiz-Beltrán, Arámburo-Lizárraga, and López-Mellado (2012) or Basile, Chiacchio, and Tommasi (2012); in a complementary manner, *unfolding* is used in Benveniste, Fabre, Haar, and Jard (2003) and Haar and Fabre (2013), while *distribution* is a key issue in Genc and Lafortune (2007). Let us summarize by saying that using PN models, structural concepts and techniques are most frequently taken into consideration.

Diagnosability, like observability or controllability, is a logical criterion. If a model is diagnosable (i.e., it is possible to detect in finite time the occurrence of faults), a diagnoser can be constructed. Complementary to this, it can be said that diagnosability is the dual notion of *opacity*, a criterion employed in computer science to characterize whether some hidden information may be inserted into messages. If diagnosability can be improved, opacity can also be enforced; for example, by using the conceptual framework of supervisory control (Dubreil, Darondeau, & Marchand, 2010).

Identification of DEDS is also a question requiring attention. In general, the starting point is a behavioral observation, the goal being to construct a PN model that generates the observed behavior, either from examples/counterexamples of its language or from the structure of a reachability graph. So the results are *derived*-models, not human-made models (i.e., not made by designers). If a priori we limit ourselves to certain subclasses of models (for example, to free-choice and self-loop free nets), there is first a decision procedure to solve (does a solution exist?). In general terms, identification is strongly related to the *synthesis* problem, to which we make a first approach in Section 3.3, as a way of approaching PNs. In synthesis problems it is assumed that the given behavior is a "complete" description of the system; in some cases, forbidden behaviors (i.e., counterexamples) are also specified. Fortunately, in this area there are two recent complementary overviews of the field (Cabasino, Darondeau, Fanti, & Seatzu, 2013; van Dongen, Alves de Medeiros, & Wen, 2009); it can be said that both are "surveys of surveys", which gives an idea of the broadness of the landscape of approaches to identification and synthesis of PNs models. One remark to justify the spread of approaches, notations, etc. is significative of the vitality of the field:

> the same problem formulation has been studied by people from two different areas, namely computer science and automatic control. The use of different terminologies and notations, as well as the separate reference journals and conferences for these two areas, lead to the fact that very similar solutions to the same problem – either identification or synthesis – have been developed independently by different research groups. (Cabasino et al., 2013)

Once again, the *duality* of places and transitions appears in the approaches to identify/synthesize PNs. Here it "transpires" the dual views of sequential(ized) discrete event systems as languages or automata. If the starting point is a labeled graph (a sequential automata, as already mentioned), the PT-system may appear as a "coding schema" of the sequentialized model. Alternatively, in many practical cases the starting point for identification or synthesis may be a set of sequences of events (event logs, where *logs* stand for captured "real" executions). They define a language that may have its origin in a model with different labels associated to each transition or not (i.e., several transitions may be identically labeled). The *theory of regions* and *integer programming* approaches are used in several contributions. Observe that there exist complementary state-based and language-based approaches in region theory, where regular, step and partial languages may be used.

In the context of *process mining*, identification or synthesis is called *process discovery* (van Dongen et al., 2009). In this application domain, specific approaches are used usually employing some (heuristic) rules exploiting locality and concurrency properties (abstraction, direct succession of events or parallel dependencies). The goal is that "mined" (i.e., constructed) models provide insights into the behavior captured in the log (eventually, analyzing the model), or to generate the corresponding run-time support. One important open point is to deal with noised data (missing, redundant or wrongly sequentialized events, etc.). In any case, among the lessons learned when dealing with real life cases are the fact that automatically derived models frequently (1) do not reflect the "reality"; (2) tend to present naive views; (3) are difficult to understand; and (4) lack expressiveness. Usually, "spaghetti" representations (probably rooted in the complexity and variability of the underlying processes) are constructed, with potentially significant differences between the reference process and the processes in the discovered procedural PN model (van der Aalst, 2009).

5. Petri nets: a modeling paradigm

A *formalism* is a conceptual framework in which a kind of formal model of a system can be expressed. For instance, the ordinary differential equations constitute a formalism to model the dynamic behavior of continuous systems with lumped parameters. Conceived for diverse purposes, some examples of formalisms for DEDS are *state diagrams* (for sequential operational description of the intended behavior); *Markov chains* and the different types of QNs (for performance evaluation); *PERT* and *conjunctive/disjunctive graphs* (for scheduling); etc.

In view of the long *life cycle* of many systems (conception and modeling; analysis and synthesis from different perspectives; implementation and operation) and the diversity of application domains, it seems desirable to have a *family* of formalisms rather than a collection of "unrelated" or weakly related formalisms. The expected advantages would include *coherence* among models usable on different phases, *economy* in the transformations and *synergy* in the development of models and theories (thus derived objects as reachability graphs or structural components, and proof logics). As an example of synergy at the theory level, let us just mention that while studying the computation of the *visit ratio* of transitions in a timed stochastic (Markovian) PN model, the first *rank theorem* for the study of deadlock-freeness was found (Campos, Chiola, & Silva, 1991).[22]

A *paradigm* is "the total pattern of perceiving, conceptualizing, acting, validating, and valuing associated with a particular image

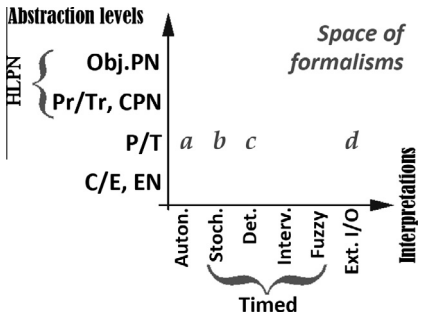

Fig. 7. A possible global view of the PN-based modeling paradigm. (**Remark:** letters "a", "b", "c" and "d" refers here to particular formalisms used for the design and analysis of the manufacturing cell considered in this work, but each letter may refer to many formalisms; for example, "b" contain interpretations in which time may be associated to transitions, to places, to arcs or to tokens, and many combinations of those; also the pdfs may have an infinity of possible forms, etc.)

of reality that prevails in a science or a branch of science" (Kuhn, 1962). In particular, a *modeling paradigm* is a conceptual framework that allows particular formalisms to be obtained from some common concepts and principles. Nevertheless, even if we believe that PNs constitute an adequate conceptual framework for the operational description of concurrent DEDS,

> we do not believe that it is always possible to select a single formalism, or family of them, to deal in a reasonable way with every aspect of every DEDS. The complexity and variety of systems suggest instead the interest of having *multi-paradigm* environments, where the existence of sound and efficient bridges between different paradigms becomes a major issue. (Silva & Teruel, 1996)

In fact, even if it is natural to try to reduce the diversity of formalisms to a common framework (modeling paradigm), different languages, calculi and theories may be needed for the different possible perspectives.

The conceptual seeds proposed by Carl Adam Petri do not flourish on a single formalism only. His work, and the improvements and extensions developed by many other researchers and engineers, inspired a set of formalisms that constitute the foundations of a modeling paradigm. The idea is already explicit, because of the several *autonomous* formalisms considered in Section 3, some a few "abbreviations" of PT-nets (such as CP-nets), others being important "extensions" (such as PT-nets with inhibitor arcs).

Strictly speaking, to *interpret* a net system is merely to assign a meaning or interpretation to the various entities: places, transitions and tokens (in some cases, also to the arcs). In this sense it can be said that autonomous PN systems are *partially-interpreted* bipartite and animate graphs, where places have the meaning of state variables, the marking represents their value, and transitions are state transformers. But this does not alter at all the behavior of the net model; it remains "autonomous". The marking informs about what *may* happen, but does not condition *what* (among alternatives) actually happens and *when* it will happen. Nevertheless, in control problems the model of a controller should be in *close loop* (otherwise stated, *concurrently composed*) with the plant being controlled. Thus inputs and outputs are needed to *synchronize* the PN model with the external world; moreover, means are needed to express *time dependent* evolutions. In Subsection 5.1 we deal with some *interpreted extensions*, and the net system is said to be *non-autonomous* because its evolution depends not only on the net marking, but also on the state of the *environment* being considered (i.e., a non-autonomous PN system is "reactive" with respect to its environment).

In our view, the Petri nets based modeling paradigm derives from the "cross-product" of the different levels of *abstraction* of

[22] As already advanced, the results were generalized in a pure untimed framework (Teruel & Silva, 1996; Recalde et al., 1998a; Recalde et al., 1998b), even taking into account the *process of construction* of the net (i.e., not viewing it as *a flat structure*).

autonomous PNs and the different *interpreted extensions* (i.e., formalisms extended by interpretations)(Fig. 7). Letters in the figure refers to some models of the manufacturing example that is considered for illustration: (a) concerns the autonomous PT-model of Fig. 5; (b) refers to the stochastic PT-system of Fig. 9, used to evaluate the performance; (c) deals with the deterministic model in Fig. 10, used to compute a minimum-time optimal steady-state; finally, (d) is the *marking diagram*, MD, describing the control of the manufacturing cell, assuming that the conflicts for the robot are not yet solved (Fig. 11). Obviously, proceeding within the PN modeling paradigm, the work may be really *multidisciplinary*! Elements for the first axis and column (CE; PT; PrT, Colored PN... systems) have been discussed in Section 3.

5.1. On some extensions by interpretation: non-autonomous net systems

With a simple illustrative purpose, here we just consider two types of "interpreted extensions". Associating time to the autonomous net models results in an very significative set of proposals. There are also some interpretations generalizing the well-known *State Diagram* formalism (in this last case by adding boolean conditions and events to "constrain and force" the firing of enabled transitions of the underlying autonomous models). In summary, the added constraints mean that: (1) safety properties (boundedness, deadlock-freeness, etc.) are preserved, but the reverse is not true (e.g., an unbounded autonomous model may become bounded for appropriate timings); and (2) liveness properties of the autonomous model are neither necessary nor sufficient for the same properties of the interpreted one. So analysis techniques (sharing some basic principles with that of autonomous models, as presented in Section 4) need to be (re)considered.

5.1.1. Putting time in Petri Nets

Although initially fully *non-deterministic*, Petri nets were provided with notions of time during the 1970s. One of the many possible ways to incorporate time in a PN system is to associate timings to transitions (alternatively or complementarily; it is also possible to assign timings to places, to arcs, to tokens, etc.; see, for example, Boyer & Roux (2008)). The association can be done as a *delay* (the firing remains atomic) or as a *duration* (the occurrence is in three phases: start/activity/end). "True concurrency" leads to temporal realism of these models. The applications may range from dealing with performance and performability analysis, with optimal control (as in scheduling), or with real-time applications which for logical correctness include timeliness constraints (from liveness to explicit time-bounds in response, otherwise there may be a risk of severe consequences, even of full system failure).

Different ways of constraining time lapses (i.e., restricting the non-determinism) are:

- Giving a *time interval* or window as in *Time* PNs (TPNs, introduced in Merlin (1974)). The interval may be just a point so that timing is deterministic, as in *Timed* PNs (TdPNs, introduced in Ramchandani (1973)).
- In a *stochastic* way, defining the time-pdfs: Symons (1980), Natkin (1980), Molloy (1982), Ajmone Marsan, Balbo, and Conte (1984).
- In a *possibilistic* form, using fuzzy sets: Murata (1996) and Cardoso and Camargo (1999).

The precedent work by Murata deals with temporal uncertainty. He introduces fuzzy timing in a HLPN model (by means of Fuzzy set theoretic functions – F – as: F-timestamps, F-enabling time, F-occurrence and F-delay) in a way that respects Petri's axioms on nets. In Cardoso and Camargo (1999) is presented a general overview of several – and some very different – fuzzy PN proposals. It must be observed that not all of them are related to timing, but fuzziness is sometimes introduced to deal with reasoning (using fuzzy production rules, in this kind of approach there exists no timed interpretation (Pedrycz, 1995); see, also Cao & Sanderson (1996), where task planning for automated "intelligent" systems is considered, discussing task representation, planning and error recovery, by means of AND/OR nets, and fuzzy Petri nets).[23]

Returning to time interpreted models, different ways of constraining non-determinism in conflict resolution include:

- Providing some fairness constraints. The constraints may be rigid, and then the resolution of the conflict is deterministic.
- Probabilistically, as in stochastic PNs.
- Possibilistically, by the way of fuzzy sets.

Defining a sound interpretation so that the model faithfully reflects the intended behavior is not always an easy task. For example, in the case of stochastic interpretations, Ajmone Marsan et al. (1989) explores different possibilities and shows that the net structure should be carefully taken into account. In order to deal appropriately with confusion and conflicts among the many questions arising, Teruel, Franceschinis, and Pierro (2003) considers a net-level method to guide the modeler in defining the priorities and weights of immediate transitions.

Performance indices of interest in *transient analysis* include, for example, the probability of reaching a given marking or of the satisfaction of given deadlines. In the study of *steady states*, it is common to look for the probability of being in a specific marking, or of the utilization of a given resource; also the expected values of flows (throughput), of utilization rates of resources, or of waiting times of clients at certain places. Pioneering works on the performance evaluation of PT-systems with *deterministic* timings include (Ramamoorthy & Ho, 1980; Sifakis, 1977). In the same line, basic *stochastic* proposals are the historically called *Stochastic PNs* (SPNs, where only exponential pdfs are associated to the transitions, thus the underlying Markov chain and the reachability graph of the autonomous net model are isomorphous), *Generalized SPNs* (GSPNs, where immediate transitions, inhibitor arcs or priorities among transitions are added to the SPN formalism) or *Queueing PNs* (QPNs, where places are either ordinary or queueing, the latter introduced to eliminate the representation of scheduling strategies by integrating the concept of queues into a colored version of GSPNs). Books concerning performance and performability evaluation include Ajmone Marsan, Balbo, and Conte (1986), Ajmone Marsan, Balbo, Conte, Donatelli, and Franceschinis (1995), Bause and Kritzinger (1996), Wang (1998), Lindemann (1998), Balbo and Silva (1998), Haas (2002), and Zimmermann (2007). Analytical techniques for performance evaluation range from "exact" computations (typically Markov chains generation, net-driven tensor algebra methods, net-driven aggregation using symmetries, net-driven product forms, etc.), through "approximations" (using, for example, "divide and conquer" approaches), to the computation of "bounds" (for arbitrary pdfs of the timings associated to the transitions, or improvements for particular cases as exponential pdfs). The use of stochastic timing with HLPNs leads to joint consideration of the underlying "data typed" untimed net model and stochastic concepts and techniques (among many examples, Chiola, Dutheillet, Franceschinis, & Haddad (1993)).

[23] When dealing with models for reasoning, by abuse of language some authors speak about fuzzy PNs, while they are really considering certain boolean-like and monotonous logics (i.e., not a compsumption/production logics, in which the P- and T-semiflows and derived invariants hold), closer to AND/OR graphs. Typically, in those reasoning processes "nets" are acyclic and assumed to be 1-bounded.

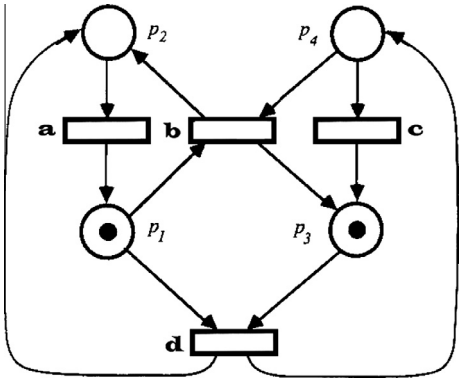

Fig. 8. Live as autonomous, deterministic timings associated to transitions can kill transition *c*. Moreover, transitions *a* and *d* are in B-fairness relation (no infinite number of firings of one of those transitions without firing infinitely often the other), but there does not exist any finite synchronic distance for them.

Simulation of timed models is frequently very helpful in practice. In performance evaluation, simulation is a technique for getting estimates of the characteristics of a random process, thus it is based on the use of statistical output analysis techniques (this means that the repetition of the experiment is needed to build a "sample" of independent observations of the same random process from which interval estimates can be computed). Simulation techniques are particularly useful to study difficult and complex problems that do not fit well into the classes of situations (models and properties) for which nice theoretical or technical results exist (see, for example, Chiola & Ferscha (1993), Ferscha (1999), and Haas (2002)). Very often simulation is used to study the transient behavior of a system; in other cases the goal is to study the steady-state (its existence is related to ergodicity).

It is not possible to provide an overview here of the works on the analysis of classical properties such as boundedness and deadlock-freeness in (HL)PN models with the many different time interpretations. At this point it is crucial to recall that any timing is a set of constraints on the possible behavior of the autonomous model, thus only a subset of reachable markings in the autonomous net system are now reachable; in consequence, properties of the autonomous and timed models may differ. For example, it is obvious that an autonomous PN model may be unbounded, while the timed one is bounded. In general, safety properties (boundedness, deadlock-freeness, etc.) are preserved when time is added; nevertheless, it is not the case with properties such as liveness. For example, the PN in Fig. 8 is live as autonomous; nevertheless, if a deterministic 1 time unit is associated to all transitions, except to transition *c*, to which 3 t.u. are associated, under classical *race policy* this transition will never fire, and the TPN is not live! Moreover, the association of time interpretations may induce many undecidabilities. Provided with time (deterministic or in intervals), PN models are also used to model *real-time* systems. In this class of applications, certain deadlines may be critical. Among other works on the domain concerned with analysis, expressivity or real-time considerations, see Berthomieu and Diaz (1991), Ghezzi, Mandrioli, Morasca, and Pezzè (1991), Bucci and Vicario (1995), Serugendo, Mandrioli, Buchs, and Guelfi (2002), Bucci, Fedeli, Sassoli, and Vicario (2004), Bérard, Cassez, Haddad, Lime, and Roux (2005), Lime and Roux (2009), Bérard, Cassez, Haddad, Lime, and Roux (2013). For books dealing with topics related to time in PNs, see, for example, Wang, 1998, Penczek and Polrola (2006), and Diaz (2009). The second is focussed on its verification by means of *model checking* techniques (see, also, Hadjidj & Boucheneb (2011)). Let us point out that beyond classical properties, others, such as diagnosis, have also been considered (Boel & Jiroveanu, 2013).

From a more Operations Research perspective, works include Carlier, Chrétienne, and Girault (1984), where it is shown "how to model with a timed Petri net, tasks, resources and constraints of a scheduling problem", and Carlier and Chrétienne (1988). There is significant literature on scheduling of TPNs models using different kinds of heuristics, due to the computational complexity of "realistic" optimization problems. In this context, it should be considered that the classical *control problem* of minimizing the time to evolve from a given marking to another is a minimum *makespan problem* (i.e., of minimizing the completion time) in operations research.

5.1.2. Logic controllers and their implementation

In sequential switching systems, it is customary to consider *State Diagrams*. If the (logical) state of a PN model is the marking, it seems reasonable to speak about *Marking Diagrams* (MDs) (Silva, 1993). Enabled transitions in autonomous PN models may be fired, but they are not forced to do so. In MDs transitions have associated *external boolean conditions* (if missing, the condition is assumed to have the "true" constant) and *events* (if missing, it is interpreted as being always "true"). If a transition is enabled, the associated external condition is true, and the event happens, then the transition is "instantaneously" forced to fire. If some transitions in conflict are simultaneously firable, the conflict is "instantaneously and non-deterministically" solved (with some technical nuances not to be considered here, the *Synchronized* and *Interpreted* Petri nets of chapter 3 in David and Alla (2010) are closely related to the kind of formalisms considered here). MDs allow more concise and natural representation of concurrency and sequencing compared to state diagrams and relay ladder logic diagrams, for example. MDs use some interpretations similar to that of the Grafcet (Subsection 7.2), but the underlying model is a PT-system (in particular, the token conservation laws are fully respected).

Beyond analysis techniques, *implementation* issues (i.e., simulation of the PN model by playing the "token game") are very important in building *logic controllers*. Implementations may be hardwired, microprogrammed, based on programmable logic controllers, and based on general purpose computers (see, for example, Silva (1985)).

In industrial applications, *programmable logic controller* (PLC) based implementations are very important. PLCs work cyclically: they (1) read the inputs; (2) compute the evolution of the marking; and (3) generate the outputs. Among the arguments that justify the success of this special kind of computers are their special programming languages and their physical capabilities of working in very aggressive environments (in conditions of high humidity and high dust levels, where electrical signals are very noisy, where there is a large number of I/O signals, etc.). The most basic languages of PLCs are based on *ladder diagrams* (LDs), *instruction lists* (ILs; with AND, OR, XOR, NOT, etc. operations, without indirect addressing), or *function block diagrams* (FBDs). Their main quality is to be close to the mentality of technicians. The frequent presence of LDs, ILs or other similar languages in PLCs bring to the development of implementation techniques for PT-net based formalisms as MDs. This topic was already considered in Silva and David (1977). If at that time the main kind of problems in hardwired asynchronous implementations were *essential hazards* (Unger, 1969), the sequentialization of the implementation in PLCs leads to new "programming hazards". The key issue was how to code the memory cells and how to sequentialize the instructions in order to avoid a poor "simulation". In any case, observe that the approach was already a "model-driven code generation" (from formal PN-based models to automatic code generation). The automatic generation of code in some basic PLC languages became the subject of an important body of literature. Among many "small" variants on the PN formalisms (all very close to MDs) and target programming languages, gener-

ation techniques for *Signal Interpreted Petri Nets*, *Timed Interpreted Petri Nets* and *Control Petri Nets* are presented in Frey (2000), Jiménez, López, and Ramírez (2000), and Lee, Zandong, and Lee (2004), respectively. For code generation according to the *Sequential Flow Chart* (SFC) standard (Subsection 7.2), see, for example, Park, Tilbury, and Khargonekar (2001) (in fact the SFCs are "derived" from 1-safe PT-systems, so this coding is straightforward).

PN-based PLCs have been implemented in general purpose computers. Among other possibilities, software implementations may be *compiled* (the "game" derives from the direct execution of a program) or *interpreted* (a program "plays" reading some data structures concerning the net and the marking); they may also be sequential or concurrent (parallel or distributed), etc. A critical point is to efficiently determine whether a transition is enabled. To accomplish this, some techniques are *place-driven* (e.g., input places to net joins are partitioned into "representative" and "synchronization" places, the latter class being implemented like "semaphores" (Silva & David, 1979)), while other techniques are *transition-based* (e.g., certain linear functions are used to compute the enableness of a given transition (Briz & Colom, 1994)). There are many other possibilities. PT-net and CP-net systems can be implemented as in Colom, Silva, and Villarroel (1986), where decentralized implementations are also considered. In the implementation of higher control levels, some convergence exists between the fields of PNs and Artificial Intelligence (see, for instance, Martínez, Muro, Silva, Smith, & Villarroel (1989) and Valette & Courvoisier (218-238)). In this sense, transitions play the role of *expert rules* while the working memory can be split into several nodes corresponding to the respective input places. With respect to classical PN implementations, the search for enabled transitions is carried out by the matching phase in the rule system, which can take advantage of the partition into local working memories. For the selection phase, transitions can be grouped into conflict sets by inspecting the net structure, and each one can be provided with a particular resolution strategy. Implementation of PN-based controllers and simulation of PN models are closely related technical problems.

In the above context, development has been *model driven*: a "good enough" formal model is designed, validated, verified, and implemented (in the context of software engineering, see Selic (2003)). The reverse procedure (i.e., going from the code in a programming language to a formal model) is done in Shatz, Tu, Murata, and Duri (1996), obtaining interesting conclusions about an ADA program with the analysis of the PN model extracted from it (the program in ADA is contemplated as a "given artifact"). Analogously, the generation of a PN model from programs in LD-language allows to allows verifications to be performed (see, for example, Bender et al. (2008)).

Let us end the present comments by pointing out that fault detection/correction capabilities may first be improved by increasing the Hamming distance in the space of the markings. This task can be achieved by embedding the PN model in a larger net representation obtained by adding some redundant information; for example, *implicit* places in Sifakis (1979) or *test* places in Silva and Velilla (1985). The idea is that the augmented net system keeps the functionality, i.e., maintains the previous behavior. As is well known in coding theory (McWilliams & Sloan, 1981), if the Hamming distance of a code (here the marking code) is δ, then all the errors of multiplicity:

- $\delta - 1$ can be detected;
- η, where $2\eta + 1 \leqslant \delta \leqslant 2\eta$, can be corrected!

Prock (1991) deals with monitoring the number of tokens on the unique global P-invariant of a strictly conservative PN (i.e., such that the firing of transitions does not change the total amount of tokens): if the total number of tokens changes, an error is detected. The previously mentioned idea of increasing the Hamming distance is also addressed in Hadjicostis and Verghese (1999) and Wu and Hadjicostis (2005). As a complementary technique, the design of *self-checking* systems (i.e., systems able to detect some erroneous behaviors as soon as they become observable) has also been considered by means of the *spy* (Velilla & Silva,14(1),14(1) and *observer* (Diaz, Juanole, & Courtiat, 1994) concepts. Now, transitions are partitioned into *observable* (i.e., *check-points*) and *non-observable*. In the former work, a PN reduction method builds the projection of the PN system on the subset of observable transitions (usually the rendezvous, at least). Any mismatch between the observed system and the spy indicates the presence of an error. Of course, the increasing of the Hamming distance and the use of observers or spies are topics related to fault-diagnosis, as presented in Subsection 4.2. They are considered here because they arose within the context of the implementation of PN models.

5.2. A simple exercise: manufacturing cell example and life-cycle

When dealing with complex systems, multiformalism approaches are frequently used. "Unrelated formalisms" (eventually one per phase of the life-cycle), based on different view points and using particular underlying theories, are frequently used; for example, automata (for functional specification), QNs (for performance evaluation), PERTs (for basic scheduling/control), different coding schemas (for software implementation), etc. Eventually, provided with appropriate interpretations, PNs can do an analogous kind of job: autonomous PNs, stochastic PNs (many possibilities), marking diagrams, etc, till the (semi)automatic generation of code, eventually fault-tolerant to a certain level.

As a simple exercise, we present a set of models of the paradigm of Petri nets useful for dealing from different perspectives with the manufacturing cell considered in Fig. 5. The net system in the figure is an autonomous functional description of the intended behavior (this kind of model is marked "a" in Fig. 7). Let us consider now that a study of the performance is needed in order to optimize the size of the buffer. Assuming some exponential timings associated to the firing of transitions, and that transitions **a, b, c** and **d** are immediate, the model in Fig. 9 describes the system (point "b" in Fig. 7). It is clear that the performance of this system is monotonously increasing with the buffer capacity, N. Moreover, the kind of descriptive function with respect to $\lambda(fail)$ shows the role of "high-pass" filter implemented by the buffer (like a capacitor in electrical circuits).

To simplify arguments, let us now assume that all firing delays are deterministic, and the failure and repair cycle is very improbable, so the cycle is removed. Now a question may arise: how to allocate the robot to the four activities modeled in Fig. 10.1 by the firing transitions **a**, **b**, **c** and **d**, and the subsequent ones, in sequence? The PN is structurally bounded (i.e., bounded for any initial marking), and the only minimal T-semiflow (natural right annuler of the token flow matrix) in the net is $X = \mathbf{1}$, so all transitions should fire in the same proportion in steady-state. Assume that the behavior is *1-periodic* (i.e., a repetitive sequence of one occurrence of each transition defines the optimal behavior). Then the problem of optimizing the throughput is reduced to finding one of the best cyclic sequences of firings of the four immediate transitions. There exist four transitions; thus, because of the cyclic behavior, the number of possibilities is $(4-1)! = 6$. Now, if at least one token is shifted to the buffer (see Fig. 10), then **a**–**c** can be fired without having to wait for the shared resource, the robot. After **d** we can be fire **b**, and at least one optimal solution will remain. In this way, only one circular possibility remains: **d**–**b** followed by **a**–**c** (Fig. 10(2) shows the circular use of the resource). Adding the places that sort the sequence **b–a–c–d**, then the place **R** is im-

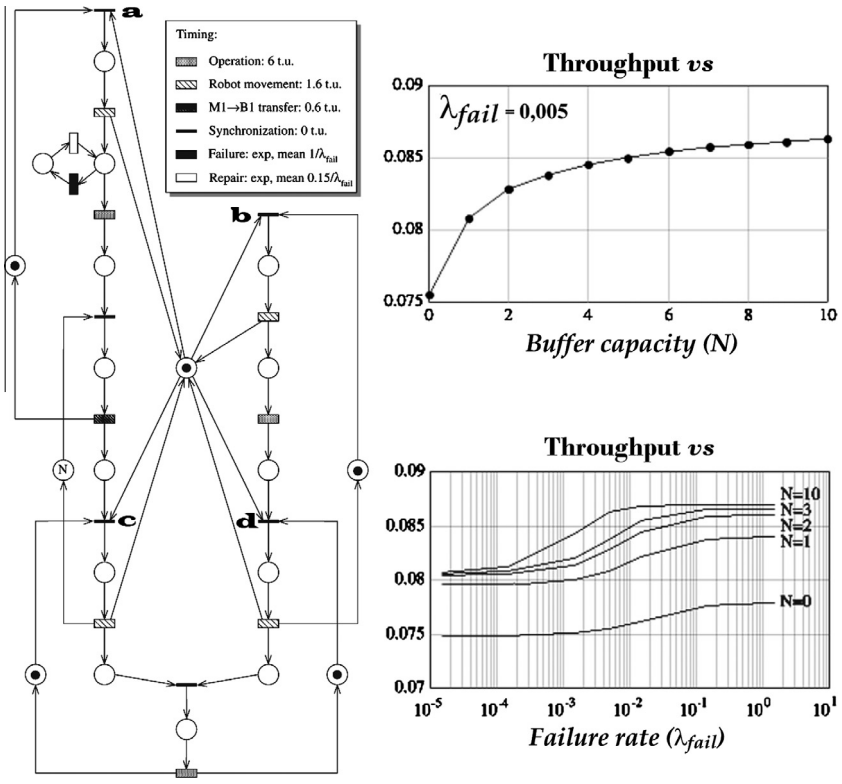

Fig. 9. GSPN model, and evolution of the throughput depending on the size of the buffer and on the rate of failure and repair (adjusted in such a way that unavailability is constant).

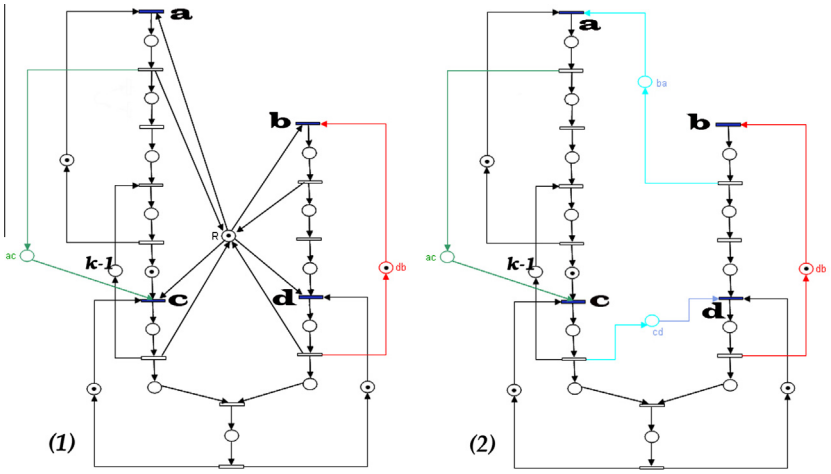

Fig. 10. Progressive evolution of the Petri net model of a manufacturing cell: A maximal throughput is obtained for deterministic timings.

plicit (i.e., it can be removed, because it does not constrain the behavior at all: the use of the resource is already scheduled in a pure sequential way!). With the pure deterministic timing being considered, the cycle time of the original system (using a greedy approach in the firings) is 10.8 t.u.; the obtained cycle time after control is 9.2 t.u. In other words, the controlled system is 14.8% faster! Moreover, it is optimal (minimum time), because 9.2 t.u. is the lower bound for the cycle time of the original system (that can be computed according to a simple LPP (Campos & Silva, 1992)).

Finally, the model in Fig. 11 (this kind of model is marked "d" in Fig. 7) describes a logic controller with *inputs* (lower case) and *outputs* (upper case) associated to the transitions. Using PN structural objects, this system can be implemented by four sequential tasks (M1, M2, M3.1 and M3.2, the last two to implement the assembly machine), one mutual exclusion semaphore (**R**) and two semaphores, one for the empty-slots in the buffer and the other for the full-slots. Let us observe that the events "sched" correspond to the decisions of a scheduler on top of the controller that solve the conflict in the use of the shared resource, the robot. If the marked graph obtained in Fig. 10(2) is used, the (fixed) schedule policy is embedded in the controller.

In summary, a simple family of formalisms was helpful throughout the considered process.

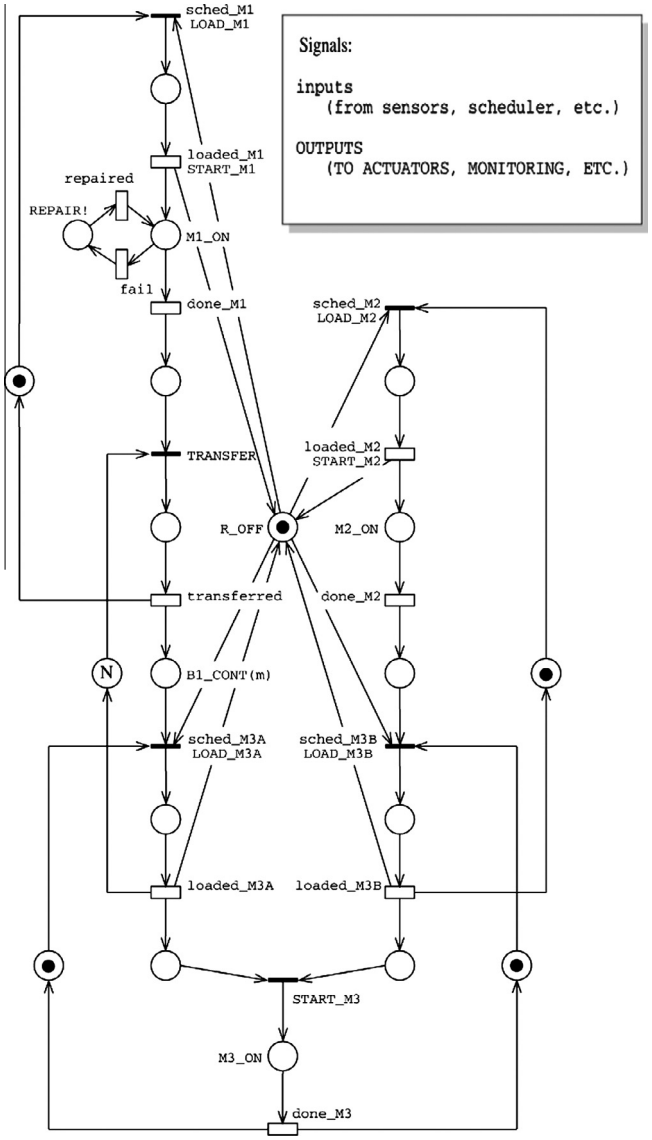

Fig. 11. A marking diagram (MD) that specifies the behavior of the logic controller for the manufacturing cell. Here, inputs and outputs are associated to transitions. Upper case labels refer to "OUTPUTs" (to actuators, monitoring, etc.), while lower case refers to "inputs" (from sensors, scheduler, etc.). Because the use of the robot exhibits a conflict, labels "sched-Mi" deal with an upper level of control to solve it. In other words, the MD plays an intermediate controller position among the plant and a scheduler to solve the conflicts.

6. Fluid relaxation of DEDS models

Because of their special character, the models derived by fluidization are now addressed, even though they form part of the Petri nets modeling paradigm. Our purpose here is twofold. First, to deal with the fact that, if decidable, many analysis and synthesis problems in DEDS, in Petri nets in particular, suffer from the so called the *state explosion problem*, which may lead to computational infeasibility. Of course, this problem tends to be much more acute when there exists a "high" number of clients or resources in the system; in other words, when we deal with a "large" initial marking (i.e., the system is heavily populated). One way to try to overcome this problem is to relax the model by means of a simple and "classical" operation: to (partially) fluidize it, in this case by fluidizing the firing of transitions. If only a few transitions are fluidized, the models obtained form a class of *hybrid* Petri nets (some transitions remain discrete, some become "continuous"); if all are fluidized, the model is a *fluid* (or *Continuous*) Petri Net (CPN). A good understanding of continuous PNs is a fundamental issue for improving our knowledge of many hybrid PN formalisms. In any case, the relaxed models are technically hybrid. Our second main goal here is to present the idea that the classes of formalisms obtained may be not so far from the preoccupations of a large majority of the readers of this journal. To put it another way, the intention of this section is to provide a "bridge" between the preoccupations among the DEDS community and the rest of the community interested in automatic control. For a broad perspective on fluid or continuous PNs, recent comprehensive works are (David & Alla, 2010; Silva, Júlvez, Mahulea, & Vázquez, 2011). The following three works provide additional technical information: Vázquez, Mahulea, Júlvez, and Silva (2013), Mahulea, Júlvez, Vázquez, and Silva (2013), Júlvez, Vázquez, Mahulea, and Silva (2013). In this journal, Silva and Recalde (2004) appeared almost a decade ago.

Fluid-flow "views" of systems that conceptually may be "more properly" contemplated as DEDS have been employed directly in very different fields. These include:

- Population Dynamics (Ecology, Sociology, etc.), where the classical model of Lotka–Volterra is of paradigmatic simplicity (Renshaw, 1986).
- Manufacturing Systems, in which fluidization fits particularly well when heavily loaded, long production lines are considered (see, for example, Gershwin (1994)).
- Communication Systems (Low, Paganini, & Doyle, 2002; Srikant, 2003).
- Road traffic Systems, an application domain in which, due to compressibility of traffic flows, frequently macroscopic models deal with partial differential equations (see, for instance, Kachroo & Özbay (1999)).
- Biochemical systems theory, where it is important to capture stoichiometric relationships and kinetic laws; based on ordinary differential equations (ODE), biochemical processes are modeled by means of power-law expansions in the state variables (concentrations, levels of gene expression, etc.) (Savageau, 1976).

Fluidization is a relaxation technique used in Queueing Networks from the beginning of the 1970s (Newell, 1971). Following along the lines of fluidization of time interpreted PNs, Stochastic Process Algebra models also deal with this over-approximation (Galpin, 2010; Hillston, 2005). In particular, through an intermediate "numerical representation [...] a Place/Transition structure [appears] underlying each PEPA model. Based on this structure and the theories developed for Petri nets, some important techniques for the structural analysis of PEPA have been given" (Ding, 2010).

Due to the intricate interleaving of cooperation and competition relationships that can be modeled with Petri nets, among the first considerations to be made is the fact that not all PT-systems can be satisfactorily fluidized, in the same way that not all differential equations can be linearized. This is true even if only one transition is fluidized (Silva & Recalde, 2004). For example, fluidization does not preserve deadlock-freeness in any sense (i.e., the discrete model may be DF, but the continuous one may not; and the discrete model may have a deadlock, but the continuous one may not). Some general characterizations and sufficient conditions to check "fluidizability" are given in (Fraca, Júlvez, & Silva, 2012). Among the peculiarities of fluid PNs should be mentioned the need to introduce a reachability extension for infinitely long firing sequences, *lim-reachability* (i.e., reachability at the limit; this concept is necessary to deal with questions like the classical Zeno's problem).

The addition of time to continuous PNs is made in ways analogous to the discrete counterparts. Nevertheless, different timing rules may try to approximate the behavior of the underlying discrete model. The two most commonly used in engineering are the so called: (1) *finite servers semantics* or *constant speed* (specially treated in David & Alla (2010)) and (2) *infinite server semantics* or *variable speed* (specially treated in Silva et al. (2011)). Under those server semantics the fluid systems are continuous piecewise linear with polyhedral regions. Most frequently, the approximation by the second semantics provides better results; additionally, for certain net subclasses, it has been proved that it provides a better approximation of the throughput (Mahulea, Recalde, & Silva, 2009). Moreover, from infinite server semantics, products of markings of places in rendezvous (so called *population semantics*) can be determined through decoloration of colored PNs (Silva & Recalde, 2002). Generalizations of this or of alternative server semantics are needed in order to deal, for example, with kinetic laws in biochemistry. Under this last kind of server semantics the fluid systems are continuous non-linear in a single polyhedral region. In this case, it is straightforward to represent *chaotic* behaviors (in particular, the classical Lorenz model) with continuous PNs. In *Timed Differentiable Petri Nets* (TDPNs) (Recalde, Haddad, & Silva, 2010) the idea is analogous to having two separate channels (like in Forrester Diagrams (Jiménez, Júlvez, Recalde, & Silva, 2004)): one devoted to defining how tokens flow, i.e., the "material" flow, and the other fixing the value of the flow, i.e., the "information" flow. Moreover, it has been proved that TDPNs can be simulated by timed continuous models under infinite server semantics. From a different perspective, in Hiraishi (2008) lower and upper bounds are considered for the firing rates (in fact, some "interval firing speeds" are used for approximating probabilistic deviation on the underlying discrete net model).

Let us concentrate on continuous PNs under infinite server semantics. They are marking and firing rate-homothetic, but due to the presence of *minimum* operators, the superposition does not hold. Their intrinsic non-linearity allows certain "paradoxes" to emerge. For example, in the steady-state: (1) there are non-monotonicities (e.g., increasing a firing speed of certain transitions, the system evolves slowly); or (2) fluidization does not necessarily lead to faster systems (i.e., to fluidize does not necessarily improve the throughput!). Both "peculiarities" are present in the net system of Fig. 12, for $\lambda(t1) > 2$. Finally, it must be pointed out that, even if continuous PNs under infinite servers semantics may "seem" a simple class of models, they can:

- exhibit bifurcations at steady state (Júlvez, Recalde, & Silva, 2005); and
- simulate Turing machines (Recalde et al., 2010).

Bifurcations merited their first monographic work in Meyer (2012), where it is shown that they may appear even in firing rate-monotonic systems. The small but tricky continuous net system in Fig. 12.a shows a bifurcation at $\lambda(t2) = 2$. Moreover, for $\lambda(t2) > 2$ the basic fluid model has a null-throughput (because it is at "deadlock" $\boldsymbol{m} = [0\ 1\ 10]$). Nevertheless, the discrete model does not have such a deadlock: it is a *spurious* one (a non-reachable solution of the state equation!). The qualitative and performance analysis of the system through the fluid approximation can be greatly improved if spurious deadlocks are removed. This can be done in polynomial time by adding *implicit* places (i.e., places that do not constrain the behavior of the discrete model). In this case, it is enough to add a place parallel to p_2 with only 9 tokens (thus, in the original discrete model, p_2 has two "frozen tokens"). Then p_2 can be removed and the state-equation has no deadlock solution. Observe that this improvement from the "original" net system (CPNo) does not depend on the time-interpretation. "Removing" the deadlock leads to the net system CPNr. Clearly, the quantitative approximation of the performance of the Markovian PN (MPN, where the stochastic PN interpretation uses only exponential pdfs) is also improved (Fig. 12b).

A complementary way to try to get a better approximation of the stochastic MPN is to add (truncated) gaussian-noise to the firing of transitions of the fluid approximation (Vázquez & Silva, 2012). Proceeding in this manner, the system is no longer deterministic, but *stochastic* (SCPN). The addition of noise is especially interesting when the system frequently commutes among adjacent linear systems (i.e., among adjacent regions of the polyhedra). If both improvements are simultaneously employed, the SCPNr approximation in Fig. 12b is obtained. A transient behavior for the MPN and the SCPNr approximation is shown in Fig. 12c.

Comparisons of CPNs under infinite servers semantics with *positive compartmental systems* or with *Forrester* (or *stock* and *flow*) diagrams, observability and controllability concepts and criteria, the design of observers and controllers, parametric optimization and diagnosis issues are all surveyed, for continuous PNs, in the already mentioned works (Júlvez et al., 2013; Mahulea et al., 2013; Silva et al., 2011). Structural concepts and techniques are also taken into consideration. For example, a notion of *structural observability* gives expression to the idea that a net system is observable for all possible values of the firing rates of transitions (i.e., the observability only depends on the structure of the net). Moreover, *weak structural* (or *generic*) *observability* abstracts the value of the firing rates of transitions, except in a variety of lower dimensions (e.g., if $\lambda\ 1 = \lambda 2$). Using results from *linear structured* systems (Dion, Commault, & van der Woude, 2003), the last concept has been considered. In addition to the evoked structural approaches, more behavioral ones, with certain peculiarities such as model checking, have also been developed (for example, Kloetzer, Mahulea, Belta, & Silva (2010)).

If only some transitions are fluidized, a kind of *hybrid* PN is obtained. See (Cassandras, Giua, Seatzu, & Zaytoon, 2008; David

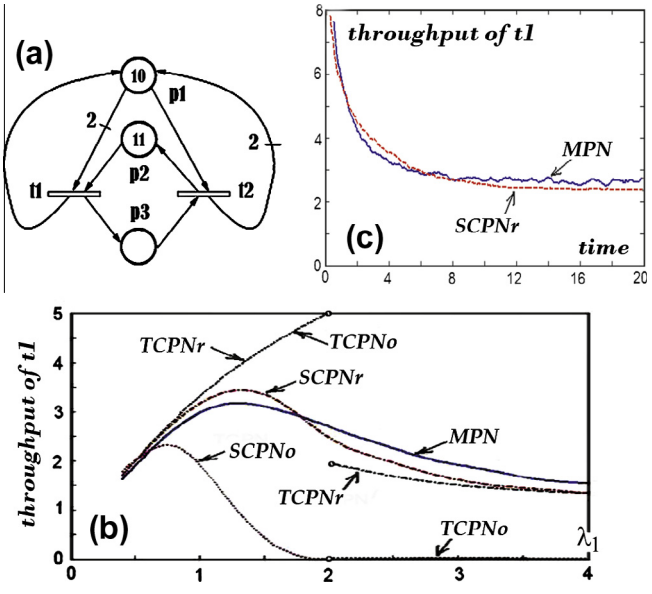

Fig. 12. (a) A "tricky" Markovian PN system (MPN) that is fluidized (TCPNo). (b) The steady-state throughput of $t1$ is plotted for $\lambda(t1) \geqslant 0{,}4$; TCPNo exhibits a discontinuity at $\lambda(t1) = 2$. Removing the spurious deadlock, the improvement is clear in TCPNr for $\lambda(t1) > 2$ (nevertheless, the system is non-monotonous); the addition of noise to TCPNo leads to the stochastic SCPNo; if both improvements are mixed, the result is SCPNr, the best approximation. (c) The transient comparison of the discrete-MPN vs the stochastic improved approximation (SCPNr).

& Alla, 2010; Di Febbraro, Giua, & Menga, 2001) in which a number of alternative hybrid PN approaches and applications are presented. Let us also mention that under the heading "fluid", there exists another line of activity with stochastic PN models in which only a few places are fluidized (Ciardo, Nicol, & Trivedi, 1999; Horton, Kulkarni, Nicol, & Trivedi, 1998; Trivedi & Kulkarni, 1993).

Certain PN based hybrid formalisms use a completely different approach to incorporate some continuous part. This is the case of *Differential Predicate Transitions PNs* (Champagnat, Valette, Hochon, & Pingaud, 2001; Villani, Miyagi, & Valette, 2007): to each place is associated a set of differential equations that describes some evolution, while the place is marked. In other words, ideas are used analogous to those employed in *hybrid automata* (see, for example, Alur et al. (1995)). The essential difference is that the discrete dynamics is here described with a (discrete) Petri net.

As stated in Pnueli and Sifakis (1995), "we strongly believe that understanding, analysis, and successful systematic development of hybrid systems can only result from the proper combination of computer science techniques with methods taken from control theory and dynamic systems". Inter-cultural experiences are almost always enriching, frequently fun and practical.

7. On applications and maturity

7.1. A restricted view on the broad spectrum of application domains

As a system theory modeling paradigm, Petri nets allow an enormous variety of application fields to be addressed concerning concurrent DEDS views of specific systems. Such applications may be presented under some functional/conceptual perspectives; for example, PNs can be considered in the framework of logical control or for performance evaluation in many different classes of concrete specific application domains (e.g., to control traffic lights, or to evaluate the performance of a concrete manufacturing facility). Therefore, "applications" may be viewed through specific technical domains (e.g., flexible manufacturing systems), or through transversal activities usable in different technical domains (e.g., logic controllers can be used to control a manufacturing or a traffic light system). Applications from the latter more conceptual perspective are partly implicit in the previous section. Moreover, concerning specific technical domains, PNs can be used in a given application at different levels of abstraction, eventually provided with different kinds of interpretation. For example, in computer hardware, PNs can represent subsystems at the level of sequential finite state machines, or at the level of competition–cooperation among complex functional units; in manufacturing systems, certain PT-systems may model local controllers, while some HLPN-systems model de coordination at cell or line level.

Focussing here on the idea of application domains, in what follows we limit our references mainly to some surveys or books providing global views of a few specific technical domains. From the very first overviews of the field (Agerwala, 1979; Peterson, 1977, 1981), special emphasis was placed on computer hardware (for example, on speed-independent circuit design) and software (for example, on operating systems and compilers; distributed data bases and communication protocols); additionally, hard/soft integrated views were put forward. Peterson (1981) also mentioned other domains such as production and legal systems. The former subsequently had an enormous impact. As regards the latter, Meldman and Holt (1971) and Meldman (1978) are representative of the attempt to formalize legal procedures. Almost a decade later, Murata (1989) explicitly added office-information systems (which would develop into Work Flow Management (WFM) systems), also flexible manufacturing and industrial control systems, both subsequently very successful. Complementary views of office-information systems are offered in De Michelis and Ellis (1996) and van der Aalst (1998) while a comprehensive updated introduction can be found in van der Aalst and Stahl (2011).

Industrial control and PNs is a transversal application domain started in France in the 1970s (for an early overview, in a broader panorama, see André, Diaz, Girault, & Sifakis (1980)). Overviews on the application of PNs to flexible manufacturing systems can be found in Silva and Valette (1989), Zurawski and Zhou (1994) and Silva and Teruel (1997). A significant number of books are specifically devoted to broad perspectives on this topic. These include DiCesare et al. (1993), Zhou and DiCesare (1993), Desrochers and Al-Jaar (1995), Proth and Xie (1996), Zhou and Venkatesh (1999), Guasch, Piera, Casanovas, and Figueras (2002), and Villani et al. (2007). The latter, more of a monograph, concerns hybrid supervision. Special volumes dealing with other monographic topics are, for example: *Communication Networks* (Billington et al., 1999), *Concurrent Object-Oriented Programming* (Agha, de Cindio, & Rozenberg, 2001), *Concurrency and Hardware Design* (Cortadella et al., 2002; Yakovlev et al., 2000) or *Communication-Based Systems* (Ehrig et al., 2003; German, 2000). A new application domain for Petri nets emerging over the last two decades has been *Systems Biology*, a model based approach devoted to the analysis of biological systems. Devoted to the study of the interactions between biological components, it focuses on emerging properties. There have been several surveys of this fast-developing multidisciplinary field and PNs started with Reddy, Mavrovouniotis, and Liebman (1993); recent broad perspectives include Wingender, 2011, Koch et al. (2011), and Heiner (2011). The intrinsic relationships between Petri nets and membrane and reaction systems, two formal models inspired by features of the behavior of living cells, are surveyed in Kleijn, Koutny, and Rozenberg (2011), focusing on the mutual benefits deriving from the strong semantic links that can be established. In the fields of biology and medicine, examples of applications can be found for population dynamics, epidemiology or the management of health care systems. Furthermore, it should be pointed out that Petri nets have also been employed in many other application domains.

As a pointer to some additional applications, the management of traffic systems can be mentioned as an example. In this domain, colored discrete (Dotoli & Fanti, 2006) and hybrid PN formalisms have been used (DiFebbraro, Giglio, & Sacco, 2004; Tolba, Lefebvre, Thomas, & Moudni, 2005). Hybrid abstractions may be employed, for example, to represent "platoons" of vehicles in road traffic problems, these being formed due to the synchronization imposed by traffic lights (Vázquez, Sutarto, Boel, & Silva, 2010). Continuous PNs have been used in (Júlvez & Boel, 2010). In order to deal with problems in some ways related, *Generalized Batches PNs* (GBPNs) were defined for the modeling and analysis of bottling lines (Demongodin, 2001; Demongodin & Giua, 2012). *First-Order Hybrid Petri Nets* (FOHPNs) represent an alternative definition of a timed PN based hybrid formalism. Using LPP techniques, in Balduzzi, Giua, and Menga (2000) online control and structural optimization problems are studied; moreover, a multiclass production network described with a queueing network is considered in the FOHPNs framework. In the area of logistics, Jiménez Macías and Pérez de la Parte (2004) is of some basic tutorial value. Gudiño-Mendoza, López-Mellado, and Alla (2012) deals with modeling and simulation of water distribution networks. Associating music objects to places and music operators to transitions, a specific interpretation (*Music Petri Nets*) is developed in Barata (2008). Of course, these represent just a few areas and proposals from the wide spectrum of applications.

7.2. On maturity in engineering and standardization

Engineering is an art and a science. In practice, engineering disciplines combine scientific-based knowledge, frequently created, adapted or completed in their own field, with less formal, more creative activities, partially based on experience and intuition and supported by means of trial and error approaches.

Maturity in an engineering discipline requires *formal methods* in order to appropriately model the kinds of systems required. These methods include the use of *formalisms* which support co-operative work, and the existence of *paradigmatic* or *standard* models (also the idea of reuse). In this context, the availability of powerful *analysis* and *synthesis* techniques is very important, even if they are sometimes partially based on certain heuristics,[24] because of decidability or computability difficulties, for example. In this sense, it can be said that Petri net theory and applications is a mature field, as it can be also said of continuous control theory and applications.

In order to efficiently benefit in practice from Petri net theory, *software tools* are necessary. They should support the construction of models, perform related analysis and synthesis, and address implementation issues (even to guarantee certain levels of fault detection/correction). Proceeding in this way, engineering productivity is increased and costs are reduced.[25] Fortunately, Petri nets offer a host of well-founded possibilities in the DEDS arena. Among the several initiatives at the beginning of the 1980s, Robert Shapiro, that did cooperate with Anatole Holt from the second half of the sixties, was making a net based software tool for the GMD during 1982-83; in the framework of the software project Galileo of for Standard Eléctrica-ITT S.A., our group of Zaragoza did make a tool to analyze PT-net systems (1984–1986), including reachability, reduction and convex geometry based techniques.

Two reports providing a certain historical perspective about PN tools are Feldbrugge (1986, 1993). It is outside the scope of this work to review the diversity of software tools dealing with PN formalisms and the reader is referred to the "Petri Nets Tools Database Quick Overview".[26] An interesting recent initiative to analyze the behavior of different tools on a same set of case studies is the *Model Checking Contest @ Petri Nets* (Kordon et al., 2012), where "the objective is to compare the efficiency of verification techniques according to the characteristics of the models". This contest is still (edition 2013) limited to logical properties.[27] To complement the set of tools presented to the contest, let us provide here some additional pointers, partially historical (i.e., not all of the following tools are equally actualized and maintained): (1) GreatSPN (developed at the University of Torino), was basically designed to evaluate performance (Chiola, Franceschinis, Gaeta, & Ribaudo, 1995); (2) CPN-AMI (http://move.lip6.fr/software/CPNAMI/), developed by the University of Paris VI, is a CASE environment that offers functions for modeling, simulation, model checking and computation of structural properties; (3) CPN-TOOLS, developed by the University of Aarhus (now maintained by the Eindhoven University of Technology), is specialized in editing, simulating, and analyzing Colored Petri nets. In several respects, this is the heiress of Design/CPN, a tool built under the sponsorship of a company called Metasoftware (in cooperation with the group of the University of Aarhus, also of Hartman Genrich from GMD); (4) TINA (TIme petri Net Analyzer (Berthomieu & Vernadat, 2006)), developed at the LAAS-Toulouse, is a toolbox for PNs with extensions as inhibitor and read arcs, priorities or Time Petri Nets; (5) Roméo (Lime, Roux, Seidner, & Traonouez, 2009) is a software that deals with Time Petri Net (IRCCyN, Nantes), performs analysis on T-Time Petri nets and on one of their extensions to scheduling; (6) Snoopy (Heiner, Herajy, Liu, Rohr, & Schwarick, 2012) was conceived (University of Technology in Cottbus) to design and animate hierarchical graphs and has been used specially for the validation of natural systems, i.e. biochemical networks such as metabolic or gene regulatory networks; and (7) embedded in the MATLAB environment, SimHPN (Júlvez, Mahulea, & Vzquez, 2012) (University of Zaragoza) integrates a collection of tools devoted to simulation, analysis and synthesis of systems modeled with Hybrid PNs.

Let us now consider a very important issue in most industrial contexts: *standardization*. The existence of standard norms adopted by international organizations such as the ISO (International Organization for Standardization) or the IEC (International Electrotechnical Commission) is important for many reasons, not least because such norms derive from technical consensus.

Due to its industrial impact on automation and because of its pioneering character, an important standard formalism is the *Grafcet*, in some sense a "child" of PNs. It was defined by the "Commission de Normalisation de la Représentation du Cahier des charges d'un Automatisme Logique" (1975–1977), under the leadership of Michel Blanchard. We will not attempt a definition here (see, for example, David & Alla (1992)). Roughly speaking, it can be said that the Grafcet is "like" an ordinary (i.e., no weights on arcs) PT-system implemented with a flip–flop per place; thus, if a place is not 1-bounded, this "implementation" breaks the token conservation laws, because some tokens "disappear". In contrast, if a conflict is not properly solved by means of external conditions or events associated to the transitions, the transitions can be "simultaneously" fired, and tokens are anomalously "created", also breaking the token conservation laws. Thus, care must be taken in order to avoid such "strange" behaviors! The "Commission pour la Normalisation" was formed by the working group "Systèmes Logiques" of the AFCET.[28] Even though at that time the author was a young Ph.D. student at the INPG, he had the chance of being one of its 24 "Chevaliers de la Table Ronde" representing research centers and powerful industries. The final report was extensively disseminated, for example in Blanchard (1977) (also in other journals such as *Automatisme* in Mars-April 1978). It is worth remembering that the name originally came from "GRaphe de l'AFCET", the learned society. The idea of disseminating it at an international level quickly made it "advisable" to change the explanation of the acronym to "GRAphe Fonctionnel de Commande des Etapes-Transitions". The interest in propagating it as an engineering norm led, with the help of the ADEPA ("Agence nationale pour le Développement de la Production Automatisée"), to a French proposal in 1982: UTE NF C 03-190, *Diagramme fonctionnel "GRAFCET" pour la description des systèmes logiques de commande*. Years later, in 1988, appeared the IEC 848 standard, drafted by the *International Electrotechnical Commission* (IEC): *Preparation of function charts for control systems*. This defines the *Sequential Function Chart* (SFC), a graphical programming language for PLCs based on Grafcet (see, later, the IEC 61131-3, of 1993).

If norms for logical controllers were based on PNs through the Grafcet, the existence of other normalization initiatives in complementary fields must be pointed out. For example, in the framework of software engineering, an important international effort over more than a decade culminated with an *International Standard Organization* (ISO) norm. This is the ISO/IEC 15909, published in

[24] This may be the case, for example, of "suboptimal" designs with respect to the really desired objective function, or when the goal or objective is not necessarily the required one but it is adopted because synthesis techniques are available.

[25] Nevertheless, "it is very important to recognize that using mature tools in themselves solve very little. It is the analytical process maturity that ensures success in the field of model based development. Or to put it another way: A fool with a tool, is still but a fool" (Torngren & Larses, 2004).

[26] http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html.

[27] The "Raw report on the model checking contest at Petri nets 2012" is available at http://arxiv.org/abs/1209.2382.

[28] A learned society which represented technical profiles such as automatic control, computer science and operations research (Hoffsaesb, 1990).

two parts: (1) *Systems and software engineering – High-level Petri nets – Part 1: Concepts, definitions and graphical notation*, 2004; and (2) *Systems and software engineering – High-level Petri nets – Part 2: Transfer format*, 2011 (see, for details: http://www.petri-nets.info/standard.php). This norm provides a unified descriptive language allowing exchanges between different software tools; therefore to take advantage of the specific analysis/synthesis/implementation techniques of each one.

8. Concluding remarks

The appealing characteristics of Petri nets include: (1) the ability to represent concurrency, causality, synchronizations, resource sharing, conflicts, bulk or lot-based services and arrivals, etc., in a natural way; (2) the graphic representation, which facilitates their use as a means of communication; (3) the locality of states and actions, that leads to distributed and structured state representations, allowing model compactness and top-down and bottom-up "team-based" model construction (i.e., refinements, modularity, reusability, etc.); (4) their adequacy for representing essential features of a given system by means of the selection of the appropriate abstraction level; (5) their interpretability, as providing the possibility to associate a wide range of meanings and connections to the external world; (6) their formal/precise semantics, which allows the undertaking of rigorous analysis or automation of the implementation, eventually leading to control; and (7) the ease of translating formal models to executable code, allowing simulation, rapid prototyping and detailed code generation (eventually becoming fault-tolerant). Many modeling, analysis and synthesis techniques have been developed in the Petri nets paradigm to overcome the so called *state explosion problem* inherent to DEDS, a real "sword of Damocles".

On the development of the theories of the Petri net paradigm, it is easy to observe a strong interleaving with many other well-established theories: graph, algebra (linear, max/plus, process, etc.), mathematical programming, formal languages, automata, logics (propositional, predicate, linear, fuzzy, etc.), possibility, interval arithmetics, stochastic processes, etc.

Models of the Petri nets paradigm have been used in many application domains, selecting the appropriate formalisms (i.e., the abstraction level and the interpreted extension). The diverse general purposes include: (1) design and its documentation; (2) validation and verification of logical and time-based specifications; (3) performance and performability evaluation and optimization; (4) dynamic control, including supervisory approaches (forbidding undesirable states to be reached), and performance control (to reach or maintain particular states under some performance index, scheduling problems in particular); and (5) monitoring and diagnosis, aiming at finding as soon as possible when a process diverges from its intended behavior in a non random way.

With humor, it can be said that as Petri nets are a concurrency theory, the field has been/is being "concurrently" developed. The success and usefulness of its basic conceptual framework has led research groups with quite different backgrounds and goals, motivated either by applications or by theoretical developments, to suggest proposals of extensions or adaptations for analysis or synthesis techniques. Petri nets have been built up by a large and diverse community! This has led to

> a very heterogeneous landscape of diverse models and this in turn has stimulated research on concepts and approaches that provide some (often partial) unification/structuring of this landscape (Ehrig et al., 2001).

Nevertheless, beyond the variety of models and techniques, Petri nets share a common basis, combining a well defined mathematical theory with an appealing graphical representation. Like Latin for romance languages (Italian, French, Spanish, Catalan or Portuguese), the different variants of Petri net formalisms share a common root, are founded on a "common language", making them a privileged means of dialog (even of self-dialog). Of course, once again, Darwinian "natural selection rules" are continuously filtering and improving the conceptual and technical legacy.

In Murata (1989) it was said that "What has been presented in this tutorial paper is a brief review of a rich body of knowledge in the field of Petri nets. It is not possible to discuss all aspects of the field in a single paper..." Nearly a quarter of century later, the problem is at least one order of magnitude bigger. So, this work cannot be considered either as a technical survey or as a tutorial on the field, at least in the classical sense. More modestly, it tries to present a panorama of the broad landscape, providing some historical points of reference. We are aware that all the topics mentioned here have been considered all too briefly, and many others have not been considered at all. Our intention was limited to draw an impressionistic – sometimes pointillist – view of a domain with about a hundred thousand pieces of work, as the order of magnitude. This is why, whenever possible, we have referred to surveys or books, therefore "implicitly" to the references cited therein. Together with those main "brush-strokes", some fine touches have been added in an attempt to bring some liveliness to the painting, as Francisco de Goya knew how to do in such a masterly fashion. If we succeed in arousing interest and stimulating additional research or applications, our goal will have been achieved.

Acknowledgements

The author wishes to thank several colleagues and friends, pioneers in the field, for their valuable suggestions. In particular to: Claude Girault (Paris, FR), Tadao Murata (Chicago, USA) and Robert Valette (Toulouse, FR), now emeritus or retired professors, that were initially inspired by works at MIT; Rüdiger Valk, today emeritus professor at the University of Hamburg (DE), that was partially inspired by works at GMD; Giorgio De Michelis (Milano, IT), that join the field because of the suggestion of Gianni Degli Antoni, that studied some of the first papers by Petri and Holt. All of them started to work on Petri nets around 1973–1975. Also to Gianfranco Balbo (Torino, IT), who started to work on stochastic Petri nets at the very beginning of the 1980s, and was initially inspired by contributions from UCLA. We are also very grateful for their helpful advice to Sebastián Dormido (Madrid, SP), Janan Zaytoon (Reims, FR) and several (ex)members of our Grupo de Ingeniería de Sistemas de Eventos Discretos (GISED), at the University of Zaragoza. Any shortcoming is, of course, our responsibility.

References

Agerwala, T. (1975). *Towards a theory for the analysis and synthesis of systems exhibiting concurrency*. Ph.D. thesis. The Johns Hopkins University.

Agerwala, T. (1979). Putting Petri nets to work. *Computer, 12*(12), 85–94.

Agha, G., de Cindio, F., & Rozenberg, G. (Eds.). (2001). *Concurrent object-oriented programming and Petri nets. Advances in Petri nets, LNCS* (Vol. 2001). Springer.

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

Ajmone Marsan, M., Balbo, G., & Conte, G. (1984). A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems. *ACM Transactions on Computer Systems, 2*(2), 93–122.

Ajmone Marsan, M., Balbo, G., & Conte, G. (1986). *Performance models of multiprocessor systems*. MIT Press.

Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., & Franceschinis, G. (1995). *Modelling with generalized stochastic Petri nets*. Wiley.

Alur, R., Coucoubetis, C., Henzinger, T., Ho, P. H., Nicollin, X., Olivero, A., et al. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science, 138*(1), 3–34.

André, C., Diaz, M., Girault, C., & Sifakis, J. (1980). Survey of french research and applications based on Petri nets. In *Brauer* (pp. 321–345).

Badouel, E. & Darondeau, P. (1998). Theory of regions. In *Reisig and Rozenberg* (pp. 529–586).

Badouel, E., Bernardinello, L., & Darondeau, P. (1997). The synthesis problem for elementary net systems is NP-complete. *Theoretical Computer Science, 186*, 107–134.

Baer, J. L. (1973). A survey of some theoretical aspects of multiprocessing. *ACM Computing Surveys, 5*(1), 31–80.

Balbo, G. & Silva, M. (Eds.) (1998). Performance models for discrete event systems with synchronizations: Formalisms and analysis techniques. In *Proceedings of human capital and mobility MATCH performance advanced school*. Jaca, Spain <http://webdiis.unizar.es/GISED/?q=news/matchbook>.

Balduzzi, F., Giua, A., & Menga, G. (2000). First-order hybrid Petri nets: A model for optimization and control. *IEEE Transactions on Robotics and Automation, 16*(4), 382–399.

Barata, A. (2008). Music description and processing: An approach based on Petri nets and XML. In V. Kordic (Ed.), *Petri net theory and applications* (pp. 525–534). Vienna: I-TECH Education and Publishing.

Basile, F., Chiacchio, P., & Tommasi, G. D. (2009). An efficient approach for online diagnosis of discrete event systems. *IEEE Transactions on Automatic Control, 54*(4), 748–759.

Basile, F., Chiacchio, P., & Tommasi, G. D. (2012). On k-diagnosability of Petri nets via integer linear programming. *Automatica, 48*(9), 2047–2058.

Battiston, E., De Cindio, F., & Mauri, G. (1988). OBJSA nets: A class of high-level Petri nets having objects as domains. In G. Rozenberg (Ed.), *Advances in Petri nets 1988. LNCS* (Vol. 340, pp. 20–43). Springer.

Bause, F., & Kritzinger, P. (1996). *Stochastic Petri nets. An introduction to the theory*. Braunschweig: Vieweg.

Bender, D. F., Combemale, B., Crégut, X., Farines, J. M., Berthomieu, B., & Vernadat, F. (2008). Ladder metamodeling and PLC program validation through time Petri nets. In *Model driven architecture – Foundations and applications (4th European conference on ECMDA-FA). LNCS* (Vol. 5095, pp. 121–136). Berlin: Springer.

Benveniste, A., Fabre, E., Haar, S., & Jard, C. (2003). Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Transactions on Automatic Control, 48*(5), 714–727.

Bérard, B., Cassez, F., Haddad, S., Lime, D., & Roux, O. H. (2005). Comparison of different semantics for time Petri nets. In D. Peled & Y. K. Tsay (Eds.), *Third international symposium on automated technology for verification and analysis, ATVA 2005. LNCS* (Vol. 3707, pp. 293–307). Springer.

Bérard, B., Cassez, F., Haddad, S., Lime, D., & Roux, O. H. (2013). The expressive power of time Petri nets. *Theoretical Computer Science, 474*, 1–20.

Berthelot, G. (1986). Checking properties of nets using transformations. In G. Rozenberg (Ed.), *Advances in Petri Nets 1985. LNCS* (Vol. 222, pp. 19–40). Springer.

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

Berthomieu, B., & Vernadat, F. (2006). Time Petri nets analysis with tina. In *Third international conference on the quantitative evaluation of systems (QEST 2006)* (pp. 123–124). IEEE Computer Society.

Best, E., Devillers, R. R., & Koutny, M. (2001). *Petri net algebra*. Springer.

Best, E., & Fernández, C. (1988). *Nonsequential processes – A Petri net view. EATCS monographs on theoretical computer science* (Vol. 13). Springer.

Billington, J., Diaz, M., & Rozenberg, G. (Eds.). (1999). *Application of Petri nets to communication networks. Advances in Petri nets, LNCS* (Vol. 16). Springer.

Blanchard, M. et al. (1977). Pour une représentation normalisée du cahier des charges d'un automatisme logique. *Automatique et Informatique Industrielle* (61-62), 27–32. 36–40.

Boel, R. K. (2002). Unity in diversity, diversity in unity: Retrospective and prospective views on control of discrete event systems. *Discrete Event Dynamic Systems, 12*(3), 253–264.

Boel, R. K., & Jiroveanu, G. (2013). The on-line diagnosis of time Petri nets. In *Seatzu et al. (2013)* (pp. 343–364).

Boyer, M., & Roux, O. H. (2008). On the compared expressiveness of arc, place and transition time Petri nets. *Fundamenta Informaticae, 88*(3), 225–249.

Brams, G. W. (1983). *Réseaux de Petri: Théorie et Pratique*. Paris: Masson.

Brauer, W. (Ed.). (1980). *Net theory and applications. LNCS* (Vol. 84). Springer.

Brauer, W., Reisig, W., & Rozenberg, G. (Eds.). (1987a). *Petri nets: Applications and relationships to other models of concurrency. Advances in Petri Nets 1986, Part II, LNCS* (Vol. 255). Springer.

Brauer, W., Reisig, W., & Rozenberg, G. (Eds.). (1987b). *Petri nets: Central models and their properties. Advances in Petri Nets 1986, Part I, LNCS* (Vol. 254). Springer.

Briz, J. L., & Colom, J. M. (1994). Implementation of weighted P/T nets based on linear enabling functions. In *Valette (1994)* (pp. 99–118).

Bucci, G., Fedeli, A., Sassoli, L., & Vicario, E. (2004). Timed state space analysis of real-time preemptive systems. *IEEE Transactions on Software Engineering, 30*(2), 97–111.

Bucci, G., & Vicario, E. (1995). Compositional validation of time-critical system using communicating time Petri nets. *IEEE Transactions on Software Engineering, 21*(12), 969–992.

Cabasino, M. P., Darondeau, P., Fanti, M. P., & Seatzu, C. (2013). Model identification and synthesis of discrete-event systems. In M. C. Zhou, H. X. Li, & M. Weijnen (Eds.), *Contemporary issues in systems science and engineering. IEEE Press series on systems science and engineering* (pp. 62–84). Wiley.

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

Campos, J., & Silva, M. (1992). Structural techniques and performance bounds of stochastic Petri net models. In G. Rozenberg (Ed.), *Advances in Petri nets 1992. LNCS* (Vol. 609, pp. 352–391). Springer.

Cao, T., & Sanderson, A. (1996). *Intelligent task planning using fuzzy Petri nets. Series in intelligent control and intelligent automation*. World Scientific.

Cardoso, J., & Camargo, H. (Eds.). (1999). *Fuzziness in Petri nets. Studies in fuzziness and soft computing* (Vol. 22). Springer.

Carlier, J., & Chrétienne, P. (1988). Timed Petri net schedules. In G. Rozenberg (Ed.), *Advances in Petri nets 1988. LNCS* (Vol. 340, pp. 62–84). Springer.

Carlier, J., Chrétienne, P., & Girault, C. (1984). Modelling scheduling problems with timed Petri nets. In G. Rozenberg (Ed.), *Advances in Petri nets. LNCS* (Vol. 188, pp. 62–82). Springer.

Cassandras, C. G., Giua, A., Seatzu, C., & Zaytoon, J. (Eds.) (2008). Special issue on discrete event methodologies for hybrid systems. *Discrete Event Dynamic Systems, 18*(2).

Cassandras, C. G., & Lafortune, S. (1999). *Introduction to discrete event systems*. Boston: Kluwer Academic Publishers.

Cerf, V. G. (1972). *Multiprocessors, semaphores and a graph model of computations*. Ph.D. thesis. Computer Science Dept., UCLA, Los Angeles.

Champagnat, R., Valette, R., Hochon, J. C., & Pingaud, H. (2001). Modeling, simulation and analysis of batch production systems. *Discrete Event Dynamic Systems: Theory and Applications, 11*, 119–136.

Chiola, G., Dutheillet, C., Franceschinis, G., & Haddad, S. (1993). Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Transactions on Computers, 42*(11), 1343–1360.

Chiola, G., Dutheillet, C., Franceschinis, G., & Haddad, S. (1997). A symbolic reachability graph for coloured Petri nets. *Theoretical Computer Science, 176*(1–2), 39–65.

Chiola, G., & Ferscha, A. (1993). Distributed simulation of timed Petri nets: Exploiting the net structure to obtain efficiency. In *Application and Theory of Petri Nets. LNCS* (Vol. 691, pp. 146–165). Springer.

Chiola, G., Franceschinis, G., Gaeta, R., & Ribaudo, M. (1995). GreatSPN 1.7: Graphical editor and analyzer for timed and stochastic Petri nets. *Performance Evaluation, 24*(1–2), 47–68.

Ciardo, G., Nicol, D., & Trivedi, K. S. (1999). Discrete-event simulation of fluid stochastic Petri nets. *IEEE Transactions on Software Engineering, 25*(2), 207–217.

Colom, J. M. (2003). The resource allocation problem in flexible manufacturing systems. In W.-M.-P. van der Aalst & E. Best (Eds.), *Proceedings of the 24th international conference on applications and theory of Petri nets. LNCS* (Vol. 2679, pp. 23–35). Eindhoven, Netherlands: Springer.

Colom, J. M., Silva, M., & Villarroel, J. L. (1986). On software implementation of Petri nets and colored Petri nets using high-level concurrent languages. In *Proceedings of the 7th European workshop on application and theory of Petri nets* (pp. 207–241). Oxford.

Cortadella, J., Yakovlev, A., & Rozenberg, G. (Eds.). (2002). *Concurrency and hardware design. Advances in Petri nets, LNCS* (Vol. 2549). Springer.

Darondeau, P., & Ricker, L. (2012). Distributed control of discrete-event systems: A first step. *Transactions on Petri Nets and Other Models of Concurrency VI, 7400*, 24–45.

David, R., & Alla, H. (1992). *Petri nets and Grafcet*. Prentice-Hall.

David, R., & Alla, H. (1994). Petri nets for modeling of dynamic systems – A survey. *Automatica, 30*(2), 175–202.

David, R., & Alla, H. (2010). *Discrete, continuous and hybrid Petri nets* (1st ed.). Berlin: Springer. 2004.

De Michelis, G., & Ellis, C. A. (1996). Computer supported cooperative work and Petri nets. In *Reisig and Rozenberg (1998b)* (pp. 125–153).

Demongodin, I. (2001). Generalised batches Petri net: Hybrid model for high speed systems with variable delays. *Discrete Event Dynamic Systems, 11*(1–2), 137–162.

Demongodin, I., & Giua, A. (2012). Linear programming techniques for analysis and control of Batches Petri nets. In *International workshop on discrete event systems, WODES'12* (pp. 54–60). IFAC-PapersOnLine, Guadalajara, Mx.

Dennis, J. B. (Ed.). (1970). *Record of the Project MAC conference on concurrent systems and parallel computation*. New York: ACM.

Desel, J., & Esparza, J. (1995). *Free choice Petri nets. Cambridge tracts in theoretical computer science* (Vol. 40). Cambridge University Press.

Desel, J., Reisig, W., & Rozenberg, G. (Eds.). (2004). *Lectures on concurrency and Petri nets. LNCS, Advances in Petri Nets* (Vol. 3098). Springer.

Desrochers, A., & Al-Jaar, R. Y. (1995). *Applications of Petri nets in manufacturing systems*. IEEE Press.

Diaz, M. (Ed.). (2009). *Petri nets: fundamental models, verification and applications. Control systems, robotics and manufacturing series (CAM)*. London: Wiley.

Diaz, M., Juanole, G., & Courtiat, J. P. (1994). Observer – a concept for formal on-line validation of distributes systems. *IEEE Transactions on Software Engineering, 20*(12), 900–913.

DiCesare, F., Harhalakis, G., Proth, J. M., Silva, M., & Vernadat, F. B. (1993). *Practice of Petri nets in manufacturing*. Chapman & Hall.

Di Febbraro, A., Giua, A., & Menga, G. (Eds.) (2001). Special issue on hybrid Petri nets. *Discrete Event Dynamic Systems, 11*(1–2).

DiFebbraro, A., Giglio, D., & Sacco, N. (2004). Urban traffic control structure based on hybrid Petri nets. *IEEE Transactions on Intelligent Transportation Systems, 5*(4), 224–237.

Ding, J. (2010). *Structural and fluid analysis for large scale PEPA models with applications to content adaptation systems*. Ph.D. thesis, The University of Edinburgh.

Dion, J., Commault, C., & van der Woude, J. (2003). Generic properties and control of linear structured systems: a survey. *Automatica, 39*(7), 1125–1144.

Dotoli, M., & Fanti, M. P. (2006). An urban traffic network model via coloured timed Petri nets. *Control Engineering Practice, 14*, 1213–1229.

Dotoli, M., Fanti, M., Mangini, A., & Ukovich, W. (2009). On-line fault detection in discrete event systems by Petri nets and integer linear programming. *Automatica, 45*(11), 2665–2672.

Dubreil, J., Darondeau, P., & Marchand, H. (2010). Supervisory control for opacity. *IEEE Transactions on Automatic Control, 55*(5), 1089–1100.

Dufourd, C., Finkel, A., & Schoebelen, P. (1998). Reset nets between decidability and undecidability. In *Proceedings of ICALP 98. LNCS* (Vol. 1443, pp. 103–115). Springer.

Ehrenfeucht, A., & Rozenberg, G. (1990). Partial 2-Structures: Part I. Basic notions and the representation problem; Part II. State spaces of concurrent systems. *Acta Informatica, 27*, 315–368.

Ehrig, H., Juhás, G., Padberg, J., & Rozenberg, G. (Eds.). (2001). *Unifying Petri nets. Advances in Petri Nets, LNCS* (Vol. 2128). Springer.

Ehrig, H., Reisig, W., Rozenberg, G., & Weber, H. (Eds.). (2003). *Petri net technology for communication-based systems. Advances in Petri Nets, LNCS* (Vol. 2472). Springer.

Esparza, J. (1998). Decidability and complexity of Petri net problems – An introduction. In *Reisig and Rozenberg (1998a)* (pp. 374–428).

Esparza, J., Römer, S., & Vogler, W. (2002). An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design, 20*(3), 285–310.

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

Fanchon, J., Rivière, N., Pradin-Chézalviel, B., & Valette, R. (2003). Preuves de logique linéaire et processus de réseaux de Petri. In E. Lavoisier (Ed.), *Modélisation des systémes réactifs, MSR'03* (pp. 261–276). Metz.

Feldbrugge, F. (1986). Petri net tools. In G. Rozenberg (Ed.), *Applications and theory in Petri nets 1986. LNCS* (Vol. 222, pp. 203–223). Springer.

Feldbrugge, F. (1993). Petri net tool overview. In G. Rozenberg (Ed.), *Advances in Petri nets 1993. LNCS* (Vol. 674, pp. 169–209). Springer.

Ferscha, A. (1999). Adaptive time warp simulation of timed Petri nets. *IEEE Transactions on Software Engineering, 25*(2), 237–257.

Finkel, A. (1993). The minimal coverability graph for Petri nets. In G. Rozenberg (Ed.), *Advances in Petri nets 1993. LNCS* (Vol.674, pp. 210–243). Springer.

Fishman, G. S. (1973). *Concepts and methods in discrete event digital simulation*. Wiley.

Fleming, W. H. (1988). *Future directions in control theory: A mathematical perspective. Report of the panel on future directions in control theory*. SIAM.

Forrester, J. W. (1961). *Industrial dynamics*. Cambridge, Mass: MIT Press.

Fraca, E., Júlvez, J., & Silva, M. (2012). Marking homothetic monotonicity and fluidization of untimed Petri nets. In *International workshop on discrete event systems, WODES'12* (pp. 21–27). IFAC-PapersOnLine, Guadalajara, Mx.

Frey, G. (2000). Automatic implementation of Petri Net based control algorithms on PLC. In *Proceedings of the American control conference on ACC 2000* (pp. 2829–2823). Chicago.

Galpin, V. (2010). Continuous approximation of PEPA models and Petri nets. *International Journal of Computer Aided Engineering and Technology, 2*(4), 324–339.

Geeraerts, G., Raskin, J. F., & Begin, L. V. (2007). On the efficient computation of the minimal coverability set for Petri nets. In K. S. Namjoshi, T. Yoneda, T. Higashino, & Y. Okamura (Eds.), *Automated technology for verification and analysis. LNCS* (Vol. 4762, pp. 98–113). Springer.

Genc, S., & Lafortune, S. (2007). Distributed diagnosis of place-bordered Petri nets. *IEEE Transactions on Automation Science and Engineering, 4*(2), 206–219.

Genrich, H. (1987). Predicate/transition nets. In W. Brauer, W. Reisig, & G. Rozenberg (Eds.), *Petri Nets: Central models and their properties – Advances in Petri nets 1986. Part I – Proceedings of an advanced course, Bad Honnef, September 1986. LNCS* (Vol. 254, pp. 207–247). Berlin: Springer.

Genrich, H., & Lautenbach, K. (1979). The analysis of distributed systems by means of predicate/transition nets. In G. Khan (Ed.), *Semantics of concurrent computation. LNCS* (Vol. 70, pp. 123–146). Springer.

Genrich, H. J., & Lautenbach, K. (1981). System modeling with high level Petri nets. *Theoretical Computer Science, 13*, 109–136.

German, R. (2000). *Performance analysis of communication systems: Modeling with non-markovian stochastic Petri nets*. John Wiley and Sons.

Gershwin, S. B. (1994). *Manufacturing systems engineering*. Englewood Cliffs, NJ: Prentice-Hall.

Ghezzi, C., Mandrioli, D., Morasca, S., & Pezzè, M. (1991). A unified high-level Petri net formalism for time-critical systems. *IEEE Transactions on Software Engineering, 17*(2), 160–172.

Girard, J. Y. (1987). Linear logic. *Theoretical Computer Science, 50*, 1–102.

Girard, J. Y. (1995). Linear logic: Its syntax and semantics. In *Advances in linear logic. Lecture Notes Series. London Mathematical Society* (Vol. 222). Cambridge University Press.

Girault, C., & Reisig, W. (Eds.). (1982). *Application and theory of Petri nets. Informatik Fachberichte* (Vol. 52). Springer.

Girault, C., & Valk, R. (2003). *Petri nets for systems engineering. A guide to modeling, verification, and applications*. Berlin: Springer.

Giua, A. (2011). State estimation and fault detection using Petri nets. In L. M. Kristensen & L. Petrucci (Eds.), *Applications and theory of Petri nets 2011. LNCS* (Vol. 6709, pp. 38–48). Springer.

Giua, A. (2013). Supervisory control of Petri nets with language specifications. In *Seatzu et al. (2013)* (pp. 235–255).

Giua, A., DiCesare, F., & Silva, M. (1992). Generalized mutual exclusions constraints on nets with uncontrollable transitions. In *IEEE international conference on systems, man and cybernetics* (pp. 974–979). Chicago.

Giua, A., & Seatzu, A. (2002). Observability of place/transition nets. *IEEE Transactions on Automatic Control, 47*(9), 1424–1437.

Giua, A., Seatzu, C., & Corona, D. (2007). Marking estimation of Petri nets with silent transitions. *IEEE Transactions on Automatic Control, 52*(9), 1695–1699.

Guasch, T., Piera, M.A., Casanovas, J., & Figueras, J. (2002). *Modelado y Simulación. Aplicación a procesos logísticos de fabricación y servicios*. Edicions UPC, Barcelona.

Gudiño-Mendoza, B., López-Mellado, E., & Alla, H. (2012). Modeling and simulation of water distribution systems using timed hybrid Petri nets. *Simulation: Transactions of the Society for Modeling and Simulation, 88*(3), 329–347.

Haar, S., & Fabre, E. (2013). Diagnosis with Petri nets unfoldings. In *Seatzu et al. (2013)* (pp. 301–317).

Haas, P. J. (2002). Stochastic Petri nets. *Modelling, stability, simulation. Springer series in operations research*. New York: Springer.

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

Hack, M. H. (1974a). The recursive equivalence of the reachability problem and the liveness problem for Petri nets and vector addition systems. In *Proceedings of the 15th annual IEEE symposium on switching and automata theory*. New Orleans.

Hack, M. H. T. (1974b). *Decision problems for Petri nets and vector addition systems*. Technical Report Computation Structures Group Memo 95. Project MAC, Laboratory for Computer Science, M.I.T., Cambridge, MA.

Hack, M. H. T. (1975). *Decidability questions for Petri nets*. Ph.D. thesis, M.I.T., Cambridge, MA. Also Tech. Report 161, Lab. for Computer Science, June 1976.

Hack, M. (1976). The equality problem for vector addition systems is undecidable. *Theoretical Computer Science, 2*(1), 77–95.

Haddad, S., Kordon, F., Pautet, L., & Petrucci, L. (Eds.). (2011). *Models and analysis in distributed systems*. New York: Wiley.

Haddad, S., & Pradat-Peyre, J. (2006). New efficient Petri nets reductions for parallel programs verification. *Parallel Processing Letters, 16*(1), 101–116.

Hadjicostis, C. N., & Verghese, G. C. (1999). Monitoring discrete event systems using Petri net embeddings. In S. Donatelli & H. C. M. Kleijn (Eds.), *Application and Theory of Petri Nets 1999. LNCS* (Vol. 1639, pp. 188–207). Springer.

Hadjidj, R., & Boucheneb, H. (2011). Efficient reachability analysis for time Petri nets. *IEEE Transactions on Computers, 60*(8), 1085–1099.

Heiner, M. (Ed.) (2011). Special issue Petri nets for systems and synthetic biology. *Natural Computing, 10*(2–3).

Heiner, M., Herajy, M., Liu, F., Rohr, C., & Schwarick, M. (2012). Snoopy – A unifying Petri net tool. In S. Haddad & L. Pomello (Eds.), *Application and theory of Petri nets – Hamburg, June. LNCS* (Vol. 7347, pp. 398–407). Springer.

Hillston, J. (2005). Fluid flow approximation of PEPA models. In *Proceedings of the second international conference on the quantitative evaluation of systems (QEST)* (pp. 33–43). IEEE Computer Society.

Hiraishi, K. (2008). Performance evaluation of workflows using continuous Petri nets with interval firing speeds. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 91*(11), 3219–3228.

Ho, Y.C. (Ed.) (1989). Special issue on discrete event systems. *Proceedings of the IEEE, 77*(1).

Hoffsaesb, C. (1990). The French society of computer scientists: AFCET. *Annals of the History of Computing, 12*(3), 167–176.

Holloway, L. E., Krogh, B. H., & Giua, A. (1997). A survey of Petri net methods for controlled discrete event systems. *Journal of Discrete Event Dynamic Systems, 7*, 151–190.

Holt, A. W., & Commoner, F. (1970). Events and conditions. *Record Project MAC Conference on Concurrent Systems Parallel Computation*, 3–52.

Holt, A., Saint, H., Shapiro, R., & Warshall, S. (1968). Final report of the information system theory project. In *Tech. Rep. RADC-TR-68-305*. Rome Air Development Center, Griffiss Air Force Base, New York.

Hopcroft, J., Motwani, R., & Ullman, J. (2001). *Introduction to automata theory, languages, and computation*. Reading, Massachusetts: Addison-Wesley.

Horton, G., Kulkarni, V., Nicol, D., & Trivedi, K. (1998). Fluid stochastic Petri nets: Theory, applications, and solution techniques. *European Journal of Operational Research, 105*, 184–201.

Huang, H., Jiao, L., Cheung, T., & Mak, W. M. (2012). *Property-preserving Petri net process algebra in software engineering*. Singapore: World Scientific.

Iordache, M. V., & Antsaklis, P. J. (2006). *Supervisory control of concurrent systems: A Petri net structural approach*. Boston: Birkhauser.

Jensen, K. (1981). Coloured Petri nets and the invariant-method. *Theoretical Computer Science, 14*, 317–336.

Jensen, K. (1994). *Coloured Petri nets: Basic concepts, analysis methods, and practical use. EATCS monographs on theoretical computer Science*. Springer.

Jensen, K. (1997). *Coloured Petri nets. Basic concepts, analysis methods and practical use* (2nd corr. printing ed., Vol. 1). Berlin: Springer.

Jensen, K. (Ed.) (2008–2012). *Transactions on Petri nets and other models of concurrency, LNCS* (Vols. 5100, 5460, 5800, 6550, 6900, 7400). Springer.

Jensen, K., & Kristensen, L. M. (2009). *Coloured Petri nets. Modelling and validation of concurrent systems*. Berlin: Springer.

Jensen, K., & Rozenberg, G. (Eds.). (1991). *High-level Petri nets*. Berlin: Springer.

Jiménez, I., López, E., & Ramírez, A. (2000). Synthesis of ladder diagrams from Petri nets controller models. In *IEEE international symposium on intelligent control (ISIC'01)* (pp. 225–230). Mexico-DF.

Jiménez, E., Júlvez, J., Recalde, L., & Silva, M. (2004). Relaxed continuous views of discrete event systems: Considerations on Forrester diagrams and Petri nets. In *IEEE international conference on systems, man, and cybernetics (SMC)* (Vol. 5, pp. 4897–4904). The Hague.

Jiménez Macías, E., & Pérez de la Parte, M. (2004). Simulation and optimization of logistic and production systems using discrete and continuous Petri nets. *Simulation, 80*(3), 143–152.

Júlvez, J., & Boel, R. (2010). A continuous Petri net approach for model predictive control of traffic systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*.

Júlvez, J., Vázquez, C. R., Mahulea, C., & Silva, M. (2013). Continuous Petri nets: Controllability and control. In *Seatzu et al. (2013)* (pp. 407–429).

Júlvez, J., Mahulea, C., & Vzquez, C. (2012). SimHPN: A MATLAB toolbox for simulation, analysis and design with hybrid Petri nets. *Nonlinear Analysis: Hybrid Systems, 6*(2), 806–817.

Júlvez, J., Recalde, L., & Silva, M. (2005). Steady state performance evaluation of continuous mono-T-semiflow Petri nets. *Automatica, 41*(4), 605–616.

Kachroo, P., & Özbay, K. (1999). *Feedback control theory for dynamic traffic assignment. Series on advances in industrial control*. London: Springer.

Karatkevich, A. (2007). *Dynamic analysis of Petri net-based discrete systems. Lecture notes in control and information sciences* (Vol. 356). Berlin: Springer.

Karp, R. M., & Miller, R. E. (1969). Parallel program schemata. *Journal on Computer Systems Science, 3*, 147–195.

Kleijn, J., Koutny, M., & Rozenberg, G. (2011). Petri nets for biologically motivated computing. *Scientific Annals of Computer Science, 21*(2), 199–225.

Kloetzer, M., Mahulea, C., Belta, C., & Silva, M. (2010). An automated framework for formal verification of timed continuous Petri nets. *IEEE Transactions on Industrial Informatics, 6*(3), 460–471.

Koch, I., Reisig, W., & Schreiber, F. (Eds.). (2011). *Modeling in systems biology. The Petri net approach, computational biology* (Vol. 16). Berlin: Springer.

Kordon, F., Linard, A., Buchs, D., Colange, M., Evangelista, S., Lampka, K., et al. (2012). Report on the model checking contest at Petri nets 2011. *Transaction of the Petri Nets and Other Models of Concurrency VI, 7400*, 169–196.

Kosaraju, S. R. (1982). Decidability of reachability in vector addition systems. In *14th Annual symposium on theory of computing* (pp. 267–281). San Francisco.

Kristensen, L. M., Schmidt, K., & Valmari, A. (2006). Question-guided stubborn set methods for state properties. *Formal Methods in System Design, 29*(3), 215–251.

Kuhn, T. S. (1962). *The structure of scientific revolutions*. University of Chicago Press.

Lakos, C. (1995). From coloured Petri nets to object Petri nets. In *16th international conference on the application and theory of Petri nets. LNCS* (Vol. 935, pp. 278–297). Berlin: Springer.

Lautenbach, K., & Schmid, H. A. (1974). Use of Petri nets for proving correctness of concurrent process systems. In *IFIP congress* (pp. 187–191).

Lee, G. B., Zandong, H., & Lee, J. S. (2004). Automatic generation of ladder diagram with control Petri net. *Journal of Intelligent Manufacturing, 15*, 245–252.

Lefebvre, D., & Delherm, C. (2007). Diagnosis of DES with Petri net models. *IEEE Transactions on Automation Science and Engineering, 4*(1), 114–118.

Lime, D., & Roux, O. H. (2009). Formal verification of real-time systems with preemptive scheduling. *Real-Time Systems, 41*(2), 118–151.

Lime, D., Roux, O. H., Seidner, C., & Traonouez, L. M. (2009). Romeo: A parametric model-checker for Petri nets with stopwatches. In S. Kowalewski & A. Philippou (Eds.), *Tools and algorithms for the construction and analysis of systems, TACAS 2009. LNCS* (Vol. 5505, pp. 54–57). Springer.

Lindemann, C. (1998). *Performance modelling with deterministic and stochastic Petri nets*. John Wiley and Sons.

Li, Z. W., & Zhou, M. C. (2009). Deadlock resolution in automated manufacturing systems. *A novel Petri net approach. Advances in industrial control*. London: Springer.

López-Grao, J. P., & Colom, J. M. (2013). Structural methods for the control of discrete event dynamic systems. the case of the resource allocation problem. In *Seatzu et al. (2013)* (pp. 257–278).

Low, S. H., Paganini, F., & Doyle, J. C. (2002). Internet congestion control. *IEEE Control Systems Magazine, 22*(February), 28–43.

Mahulea, C., Júlvez, J., Vázquez, C.R., & Silva, M. (2013). Continuous Petri nets: Observability and diagnosis. In *Seatzu et al. (2013)* (pp. 387–406).

Mahulea, C., Recalde, L., & Silva, M. (2009). Basic server semantics and performance monotonicity of continuous Petri nets. *Discrete Event Dynamic Systems, 19*(2), 189–212.

Martínez, J., Muro, P., Silva, M., Smith, S. F., & Villarroel, J. L. (1989). Merging artificial intelligence techniques and Petri nets for real time scheduling and control of production systems. In R. Huber et al. (Eds.), *Artificial intelligence in scientific computation* (pp. 307–313). Scientific Publishing Co..

Martin-Oliet, N., & Meseguer, J. (1991). From Petri nets to linear logic: A survey. *International Journal of Foundations of Computer Science, 2*(4), 297–399.

Mayr, O. (1970). *The origins of feedback control*. Cambridge, MA: The MIT Press.

Mayr, E. (1984). An algorithm for the general Petri net reachability problem. *SIAM Journal of Computing, 13*(3), 441–460.

McWilliams, F., & Sloan, N. (1981). *The theory of error correcting codes*. New York: North-Holland.

Meldman, J. A. (1978). A Petri-net representation of civil procedure. *IDEA: The Journal of Law and Technology, 19*(2), 123–148.

Meldman, J. A., & Holt, A. W. (1971). Petri nets and legal systems. *Jurimetrics Journal, 12*(2), 65–75.

Memmi, G., & Roucairol, G. (1980). Linear algebra in net theory. In *Brauer (1980)* (pp. 213–223).

Memmi, G., & Finkel, A. (1985). An introduction to FIFO Nets-monogeneous nets: A subclass of FIFO nets. *Theoretical Computer Science, 35*, 191–214.

Merlin, P. (1974). *A study of the recoverability of computer systems*. Ph.D. thesis, Univ. California, Irvine.

Meyer, A. L. (2012). Discontinuity induced bifurcations in timed continuous Petri nets. In *International workshop on discrete event systems, WODES'12* (pp. 28–33). IFAC-PapersOnLine, Guadalajara, Mx.

Milner, R. (1980). *A calculus of communicating systems. LNCS* (Vol. 92). Springer.

Milner, R. (1993). Elements of interaction. *Communications of the ACM, 36*(1), 78–89.

Miyamoto, T., & Kumagai, S. (2005). A survey of object-oriented petri nets and analysis methods. *IEICE Transactions, 88-A*(11), 2964–2971.

Moalla, M., Sifakis, J., & Silva, M. (1980). A la recherche d'une méthodologie de conception sûre des automatismes logiques basée sur l'utilisation des réseaux de Petri. In *Monographie AFCET: Sureté de Fonctionnement des Systèmes Informatiques* (pp. 133–167). Hommes et Techniques, Suresnes.

Molloy, M. K. (1982). Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers, 31*(9), 913–917.

Moody, J., & Antsaklis, J. (1998). *Supervisory control of discrete event systems using Petri nets*. Boston: Kluwer Academic Publishers.

Murata, T. (1977). State equation, controllability, and maximal matchings of Petri nets. *IEEE Transactions on Automatic Control, 22*(3), 412–416.

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

Murata, T. (1996). Temporal uncertainty and fuzzy-timing high-level Petri nets. In J. Billington and W. Reisig (Eds.), *Application and theory of Petri nets 1996* (pp. 11–28). Osaka.

Natkin, S. (1980). *Réseaux de Petri Stochastiques*. Ph.D. thesis. CNAM, Paris.

Nazeem, A., & Reveliotis, S. (2012). Designing compact and maximally permissive deadlock avoidance policies for complex resource allocation systems through classification theory: The nonlinear case. *IEEE Transactions on Automatic Control, 57*(7), 1670–1684.

Newell, G. F. (1971). *Applications of queuing theory* (2nd ed.). Chapman and Hall. 1982.

Park, J., & Reveliotis, S. A. (2001). Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings. *IEEE Transactions on Automatic Control, 46*(10), 1572–1583.

Park, E., Tilbury, D. M., & Khargonekar, P. P. (2001). A modeling and analysis methodology for modular logic controllers of machining systems using petri net formalism. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, 31*(2), 168–188.

Pedrycz, W. (Ed.). (1995). *Fuzzy sets engineering*. Boca Raton: CRC Press.

Penczek, W., & Polrola, A. (2006). *A dvances in verification of time Petri nets and timed automata. An Emporal Logic Approach, Studies in Computational Intelligence* (Vol. 20). Berlin, Heidelberg: Springer.

Peterson, J. L. (1977). Petri nets. *ACM Computing Surveys, 9*(3), 223–252.

Peterson, J. L. (1981). *Petri Net theory and the modeling of systems*. Upper Saddle River, NJ: Prentice-Hall.

Petri, C. A. (1962). *Kommunication mit Automaten*. Ph.D. thesis. Technischen Hoschule Darmstadt.

Petri, C. A. (1966). *Communication with automata*. Rome Air Development Center-TR-65-377, New York.

Petri, C. A. (1976). *Interpretations of net theory*. Technical Report 75-07, Gessellschaft fur Mathematik und Datenverarbeitung, Bonn.

Petri, C.A., & Smith, E. (2007). The pragmatic dimension of net theory. In *Proceedings of the 8th European workshop on applications and theory of Petri nets*. Zaragoza.

Pnueli, A., & Sifakis, J. (Eds.) (1995). Special issue on hybrid systems. *Theoretical Computer Science, 138*(1).

Prock, J. (1991). A new technique for fault detection using Petri nets. *Automatica, 27*, 239–245.

Proth, J. M., & Xie, X. (1996). *Petri nets. A tool for design and management of manufacturing systems*. Wiley.

Queille, J. P., & Sifakis, J. (1982). Specification and verification of concurrent systems in Cesar. In *5th International symposium on programming. LNCS* (Vol. 137, pp. 337–351). Springer.

Ramage, P. J., & Wonham, W. M. (1989). The control of discrete event systems. *IEEE Proceedings, 77*(1), 81–98.

Ramamoorthy, C. V., & Ho, G. S. (1980). Performance evaluation of asynchronous concurrent systems using Petri nets. *IEEE Transactions on Software Engineering, 6*(5), 440–449.

Ramchandani, C. (1973). *Analysis of asynchronous concurrent systems by timed Petri nets*. Ph.D. thesis. MIT, Boston.

Ramírez-Treviño, A., Rivera-Rangel, I., & López-Mellado, E. (2003). Observability of discrete event systems modeled by interpreted Petri nets. *IEEE Transactions on Robotics and Automation, 19*(4), 557–565.

Ramírez-Treviño, A., Ruiz-Beltrán, E., Arámburo-Lizárraga, J., & López-Mellado, E. (2012). Structural diagnosability of DES and design of reduced Petri net diagnosers. *IEEE Transactions on Systems, Man, and Cybernetics, Part A, 42*(2), 416–429.

Ramírez-Treviño, A., Ruiz-Beltrán, E., Rivera-Rangel, I., & López-Mellado, E. (2007). Online fault diagnosis of discrete event systems. A Petri net-based approach. *IEEE Transactions on Automation Science and Engineering, 4*(1), 31–39.

Recalde, L., Haddad, S., & Silva, M. (2010). Continuous Petri nets: Expressive power and decidability issues. *International Journal of Foundations of Computer Science, 21*(2), 235–256.

Recalde, L., Teruel, E., & Silva, M. (1998a). Modeling and analysis of sequential processes that cooperate through buffers. *IEEE Transactions on Robotics and Automation, 14*(2), 267–277.

Recalde, L., Teruel, E., & Silva, M. (1998b). On linear algebraic techniques for liveness analysis of P/T systems. *Journal of Circuits, Systems, and Computers, 8*(1), 223–265.

Recalde, L., Teruel, E., & Silva, M. (2001). Structure theory of multi-level deterministically synchronized sequential processes. *Theoretical Computer Science, 254*(1–2), 1–33.

Reddy, V. N., Mavrovouniotis, M. L., & Liebman, M. N. (1993). Petri net representations in metabolic pathways. In *Proceedings of the 1st international conference on intelligent systems for molecular biology* (pp. 328–336). Bethesda, MD: Association for Advancement of Artifical Intelligence, AAAI.

Reisig, W. (1985). *Petri nets. An introduction, EATCS monographs on theoretical computer science* (Vol. 4). Springer.

Reisig, W., & Rozenberg, G. (Eds.). (1998a). *Lectures on Petri nets I: Basic models. LNCS, Advances in Petri Nets* (Vol. 1491). Springer.

Reisig, W., & Rozenberg, G. (Eds.). (1998b). *Lectures on Petri nets II: Applications. LNCS, Advances in Petri nets* (Vol. 1492). Springer.

Renshaw, E. (1986). A survey of stepping-stone models in population dynamics. *Advances in Applied Probability, 18*, 581–627.

Reutenauer, C. (1990). *The mathematics of Petri nets*. London: Prentice-Hall.

Rosa-Velardo, F., & de Frutos-Escrig, D. (2010). Decidability problems in Petri nets with names and replication. *Fundamenta Informatica, 105*(3), 291–317.

Roucairol, G. (1986). FIFO-nets. In *Brauer et al. (1987b)* (pp. 436–459).

Rozenberg, G. (Ed.), (1985–1991, 1993). *Advances in Petri nets. Lecture notes on computer science* (Vols. 188, 222, 266, 340, 424, 483, 524, 674). Springer.

Rozenberg, G. (Ed.). (1992). *The DEMON project–Advances in Petri nets. LNCS* (Vol. 609). Springer.

Savageau, M. (1976). *Biochemical systems analysis: A study of function and design in molecular biology*. Reading, MA: Addison-Wesley.

Seatzu, C., Silva, M., & Schuppen, J. (Eds.) (2013). *Control of discrete-event systems. Automata and Petri net perspectives. Lecture notes in control and information sciences* (Vol. 433). London: Springer.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE Software, 20*(5), 1925.

Serugendo, G. D. M., Mandrioli, D., Buchs, D., & Guelfi, N. (2002). Real-time synchronised Petri nets. In *Proceedings of the international conference on applications and theory of Petri nets 2002. LNCS* (Vol. 2360, pp. 142–162). Springer.

Shatz, S. M., Tu, S., Murata, T., & Duri, S. (1996). An application of Petri net reduction for Ada tasking deadlock analysis. *IEEE Transactions on Parallel Distributed Systems, 7*(12), 1307–1322.

Sibertin-Blanc, C. (1994). Cooperative nets. (In *Valette (1994)* (pp. 377–396)).

Sifakis, J. (1977). Use of Petri nets for performance evaluation. In H. Beilner & E. Gelenbe (Eds.), *Measuring, Modelling and Evaluating Computer Systems* (pp. 75–93). North-Holland.

Sifakis, J. (1978). Structural properties of Petri nets. In *Mathematical Foundations of Computer Science. LNCS* (Vol. 64, pp. 474–483). Springer.

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

Silva, M. (1985). Las Redes de Petri: en la Automática y la Informática. In: Madrid, A. C., (Ed.), *Thomson-AC, 2002* (2nd ed.).

Silva, M. (1987). Towards a synchrony theory for P/T nets. In K. Voss, H. Genrich, & G. Rozenberg (Eds.), *Concurrency and nets* (pp. 435–460). Berlin: Springer.

Silva, M. (1993). Introducing Petri nets. In *Practice of Petri nets in manufacturing* (pp. 1–62). Chapman and Hall.

Silva, M. (2012). 50 Years after the Ph.D. thesis of Carl Adam Petri: a perspective. In *International workshop on discrete event systems, WODES'12* (pp. 13–20). IFAC-PapersOnLine, Guadalajara, Mx.

Silva, M., & David, R. (1977). On the programming of asynchronous sequential systems by logic equations. In IFAC international symposium on discrete systems (pp. 52–62). Dresde (DDR).

Silva, M. & Teruel, E. (1996). A systems theory perspective of discrete event dynamic systems: The Petri net paradigm. In *Symposium on discrete events and manufacturing systems* (pp. 1–12). CESA '96 IMACS Multiconference. Lille.

Silva, M., & Velilla, S. (1985). Error detection and correction on Petri net models of discrete event control systems. In *Proceedings of the ISCAS 85* (pp. 921–924). Kyoto.

Silva, M., & David, R. (1979). Synthèse programmée des automatismes logiques décrits par réseaux de Petri: Une méthode de mise en oeuvre sur microcalculateurs. *Rairo-Automatique, 13*(4), 369–393.

Silva, M., Teruel, E., & Colom, J.M. (1998). Linear algebraic and linear programming techniques for the analysis of net systems. In *Reisig and Rozenberg (1998a)* (pp. 309–373).

Silva, M., Júlvez, J., Mahulea, C., & Vázquez, C. (2011). On fluidization of discrete event models: Observation and control of continuous Petri nets. *Discrete Event Dynamic Systems, 21*, 427–497.

Silva, M., & Murata, T. (1992). B-Fairness and structural B-fairness in Petri net models of concurrent systems. *Journal of Computer and System Sciences, 44*, 447–477.

Silva, M., & Recalde, L. (2002). Petri nets and integrality relaxations: A view of continuous Petri net models. *IEEE Transactions on Systems, Man, and Cybernetics, 32*(4), 314–327.

Silva, M., & Recalde, L. (2004). On fluidification of Petri net models: From discrete to hybrid and continuous models. *Annual Reviews in Control, 28*, 253–266.

Silva, M., & Teruel, E. (1997). Petri nets for the design and operation of manufacturing systems. *European Journal of Control, 3*(3), 182–199.

Silva, M., & Valette, R. (1989). Petri nets and flexible manufacturing. In G. Rozenberg (Ed.), *Advances in Petri nets 1989. LNCS* (Vol. 424, pp. 374–417). Springer.

Srikant, R. (2003). *The mathematics of internet congestion control*. Boston: Birkäuse.

Starke, P. (1980). *Petri-Netze*. Berlin: Deutcher Verlag der Wissenschaften.

Starke, P. (1991). Reachability analysis of Petri net using symmetries. *Journal of Systems Analysis, Modelling and Simulation, 8*(5/6), 294–303.

Stremersch, G. (2001). *Supervision of Petri nets. The International Series on Discrete Event Dynamic Systems* (Vol. 13). Kluwer Academic Publishers.

Sussmann, H., & Willems, J. C. (1997). 300 Years of optimal control: From the brachystochrone to the maximum principle. *IEEE Control Systems Magazine, 17*(3), 32–44.

Symons, F. (1980). Introduction to numerical Petri nets, a general graphical model of concurrent processing systems. *Australian Telecommunication Research, 14*(1), 28–33.

Teruel, E., Franceschinis, G., & Pierro, M. D. (2003). Well-defined generalized stochastic Petri nets: A net-level method to specify priorities. *IEEE Transactions on Software Engineering, 29*(11), 962–973.

Teruel, E., & Silva, M. (1993). Liveness and home states in equal conflict systems. In M. A. Marsan (Ed.), *Application and theory of Petri nets. LNCS* (Vol. 691, pp. 415–432). Springer.

Teruel, E., & Silva, M. (1996). Structure theory of equal conflict systems. *Theoretical Computer Science, 153*(1–2), 271–300.

Thiagarajan, P. S. (1987). Elementary net systems. In *Brauer et al. (1987b)* (pp. 26–59).

Tolba, C., Lefebvre, D., Thomas, P., & Moudni, A. E. (2005). Continuous and timed Petri nets for the macroscopic and microscopic traffic flow modelling. *Simulation Modelling Practice and Theory, 13*(5), 407–436.

Torngren, M., & Larses, O. (2004). Characterization of model based development of embedded control systems from a mechatronic perspective; drivers, processes, technology and their maturity. In *Summer school on model driven engineering for embedded systems*. Brest, France.

Trivedi, K., & Kulkarni, V. G. (1993). FSPNs: Fluid stochastic Petri nets. In M. Ajmone Marsan (Ed.), *Application and theory of Petri nets 1993. LNCS* (Vol. 691, pp. 24–31). Berlin: Springer.

Unger, S. H. (1969). *Asynchronous sequential systems*. New York: Wiley-Intescience.

Valette, R. (Ed.). (1994). *Application and theory of Petri nets 1994. LNCS* (Vol. 815). Springer.

Valette, R., & Courvoisier, M. (1992). Petri nets and artificial intelligence. In *Proceedings of the IEEE/SICE international workshop on emerging technologies for factory automation* (pp. 218–238). North Queensland, Australia.

Valk, R. (1991). Modelling concurrency by task/flow EN systems. In *3rd Workshop on concurrency and compositionality, GMD-Studien* (Vol. 191). Gesellschaft fnr Mathematik und Datenverarbeitung, St. Augustin, Bonn.

Valk, R. (1998). Petri nets as object tokens: An introduction to elementary object nets. In J. Desel & M. Silva (Eds.), *Proceedings of the international conference on applications and theory of Petri nets. LNCS* (Vol. 1420, pp. 1–25). Berlin: Springer.

Valk, R. (2003). Object Petri nets: Using the nets-within-nets paradigm. In J. Desel, W. Reisig, & G. Rozenberg (Eds.), *Advanced course on Petri nets 2003. LNCS* (Vol. 3098, pp. 819–848). Berlin, Germany: Springer.

Valk, R., & Vidal-Naquet, G. (1977). On the rationality of Petri net languages. In *Theoretical computer science, LNCS* (Vol. 48, pp. 319–328).

Valmari, A. (1998). The state explosion problem. In *Reisig and Rozenberg (1998a)* (pp. 429–528).

van der Aalst, W. (1998). The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers, 8*(1), 21–66.

van der Aalst, W. (2009). Process-aware information systems: Lessons to learned from process mining. *LNCS Transactions on Petri Nets and Other Models of Concurrency II, LNCS, 5460*, 1–26.

van der Aalst, W., & Stahl, C. (2011). *Modeling business processes: A Petri net oriented approach*. Cambridge, MA: MIT Press.

van Dongen, B. F., Alves de Medeiros, A. K., & Wen, L. (2009). Process mining: Overview and outlook of Petri net discovery algorithms. *LNCS Transactions on Petri Nets and Other Models of Concurrency II, LNCS, 5460*, 225–241.

Vázquez, C., Sutarto, H., Boel, R., & Silva, M. (2010). Hybrid Petri net model of a traffic intersection in an urban network. In *IEEE multiconference on systems and control*. Yokohama.

Vázquez, C. R., Mahulea, C., Júlvez, J., & Silva, M. (2013). Introduction to fluid Petri nets. In *Seatzu et al. (2013)* (pp. 365–386).

Vázquez, C., & Silva, M. (2012). Stochastic continuous Petri nets: An approximation of markovian net models. *IEEE Transactions on System, Man and Cybernetics, Part A: Systems and Humans, 42*(4), 641–653.

Velilla, S., & Silva, M. (1988). The spy: A mechanism for safe implementation of highly concurrent systems. In *Annual review in automatic programming (Sp. Issue: 15th IFAC/IFIP workshop on real time programming)* (Vol. 14(1), pp. 75-81).

Villani, E., Miyagi, P. E., & Valette, R. (2007). *Modelling and analysis of hybrid supervisory systems. A Petri net approach*. Berlin: Springer.
Wang, J. (1998). *Timed Petri nets: Theory and application. International series on discrete event dynamic system* (9). Norwell, MA: Kluwer.
Wingender, E. (Ed.). (2011). *Biological Petri nets. Studies in health technology and informatics* (Vol. 162). Lansdale, PA: IOS Press.
Wu, Y., & Hadjicostis, C. (2005). Algebraic approaches for fault identification in discrete-event systems. *IEEE Transactions on Automatic Control, 50*(12), 2048–2053.
Yakovlev, A., Gomes, L., & Lavagno, L. (Eds.). (2000). *Hardware design and Petri nets*. Kluwer Academic Press.
Zaytoon, J., & Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control, 37*, 308–320.
Zeigler, B. P. (1976). *Theory of modeling and simulation*. John Wiley.
Zeigler, B. P. (1984). *Multifacetted modelling and discrete event simulation*. Academic Press.
Zhou, M. C., & DiCesare, F. (1993). *Petri net synthesis for discrete event control of manufacturing systems*. Kluwer Academic Publishers.
Zhou, M., & Venkatesh, K. (1999). *Modeling, simulation, and control of flexible manufacturing systems. A Petri net approach*. New Jersey: World Scientific.
Zimmermann, A. (2007). *Stochastic discrete event systems. Modeling, evaluation, applications*. Berlin: Springer.
Zurawski, R., & Zhou, M. C. (Eds.) (1994). Special issue on petri nets in manufacturing. *IEEE Transactions on Industrial Electronics, 41*(6).
Zuse, K. (1980). Petri-nets from the engineering viewpoint. In *Brauer (1980)* (pp. 441–479).

Manuel Silva received the Industrial-Chemical Engineering degree from the University of Sevilla (1974) and the postgraduate (1975) and Ph.D. (1978) degrees in Control Engineering from the INP de Grenoble. From 1975 to 1978, he worked for the CNRS at the Laboratoire d'Automatique de Grenoble. In 1978 he started the group of Systems Engineering and Computer Science at the University of Zaragoza. His main research interests include modeling, validation, performance evaluation, and implementation of distributed concurrent systems using Petri Nets. He is author of the book *Las Redes de Petri en la Automática y la Informática* (AC, 1985; reedited by Thomson-AC, 2002), and coauthor of the book *Practice of Petri Nets in Manufacturing* (Chapman & Hall, 1993). Prof. Silva was dean of the Centro Politécnico Superior, University of Zaragoza, from 1986 to 1992. He has been associate editor of several journals and co-organizer of some two hundred international conferences. Interested in the History of Technology, he is the editor of *Técnica e Ingeniería en España* (now 8 volumes, some 6.000 pages). Prof. Silva has been distinguished with a medal from the city of Lille (France) and by the Association of Telecommunication Engineers of Aragón. He is Honoris Causa Doctorate by the University of Reims-Champagne-Ardennes, member of the Royal Academy of Engineering of Spain, and elected member of the Royal Academy of Sciences of Zaragoza.