INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2025, VOL. 71, NO. 3, PP. 1–9
Manuscript received March 6, 2025; revised June 2025. doi: 10.24425/ijet.2025.155452

Review of hierarchy in Petri Nets

Michał Markiewicz, and Lesław Gniewek

***Abstract*—Petri nets are increasingly being used to create IT and automation systems whose high complexity requires a new approach to design and implementation. As a result, new concepts for describing, analyzing, and presenting Petri nets have been developed, among which the introduction of a hierarchical network structure holds an important place. This structure allows for the presentation of a created system (network) at various levels of abstraction and facilitates the determination of the properties of its modules (subnetworks) and the all network. Although hierarchical networks are currently widely used in research and pracital applications, there is no uniform way of using them. Therefore, the aim of this article is to identify and present the basic concepts of applying hierarchy in Petri nets.**

***Keywords*—petri nets, hierarchy, modularization, formal model**

I. Introduction

IN modern information and automation systems, hierarchy, methods of its analysis, and modularization play an important role [1]–[6]. The application of a hierarchical structure can increase the innovation, functionality, and efficiency of a system if applied correctly. An essential aspect related to hierarchy is modularization, which is the process of dividing a system into subsystems (modules) to increase flexibility and scalability in the design and implementation of a complex system. This approach simplifies a system construction and facilitates its maintenance and updates. All modern software engineering is largely based on modularization.

An important issue related to hierarchy and modularization are the top-down and bottom-up approaches [7]–[9]. Both can be used for designing and creating new systems as well as analyzing the operation of existing ones. The first one involves refining the overall operation of the entire system and then determining how its individual modules work. Lower layers are defined, specifying more and more detailed operations of the system. The second approach refers to building and analyzing the system from its simplest components and gradually combining them into more complex modules. Often, both strategies are used together in practice to improve and optimize the operation of the system.

Petri nets (PNs), thanks to their undeniable advantages such as formal mathematical description, graphical presentation form, and the ability to model both sequential and concurrent systems, have become a tool widely used in both research and practical application [10]–[12]. The need to solve increasingly complex and intricate problems has led to the employment of new solutions and algorithms. One of these is the introduction of a hierarchical structure to PNs [13]–[15]. This approach allows for the modeling of complex systems by dividing them into interconnected subnetworks (modules). Each subnetwork can be independently analyzed, which facilitates understanding and verification of the behavior of the entire system.

From the latest studies [16]–[20], it can be seen that hierarchical Petri nets remain a current topic of research. It appears that as long as Petri nets are a topic of interest, research related to modularization and hierarchy will be inseparable, as these aspects are crucial for solving both practical and scientific problems.

The purpose of this manuscript is to present and identify various approaches to applying hierarchy in PNs. Based on the gathered literature (presented later) on PNs, several types of hierarchical structures are identified and described. It can be distinguished both concepts of the places and transitions refinement (section II), as well as techniques for their reduction (section III). Hierarchical PNs are formally described in the form of definitions (section IV), which, in the case of object networks, have characteristics typical of programming languages (section VII). Hierarchical networks have also found their place in industrial controller programming standards and in the methodology of dealing with complex IT systems (section VI). All these approaches will be covered in the following sections and summarized in section VIII and IX.

II. Refinement of places and transitions

One of the first works related to hierarchy in PNs is the article by R. Valette [21]. It presents the concept of a macrotransition with one input and one output. The author defined the concept of a well-formed block and examined the properties of the hierarchical network such as boundedness, safeness, and liveness. Figure 1 shows how to convert macrotransition (MT_1), when it is well-formed block. I. Suzuki and T. Murata [22] continued Valette's research, extending the concept of a well-formed block to a k-well-behaved block. Their research not only focused on macrotransition but also on determining the properties of a macroplace with one input and one output. Figure 2 shows how to convert macrotransition (MT_1) to examine subnet N_0 properties, when it is k-well-behaved block. W. Vogler [23] proposed an extension of the concepts described in [21], [22], involving the connection of

This article is supported by funds from Rzeszów University of Technology.

M. Markiewicz and L. Gniewek are with the Department of Electrical and Computer Engineering, Rzeszow University of Technology, Poland (e-mail: mmarkiewicz@prz.edu.pl, lgniewek@prz.edu.pl).

subnetworks (so-called daughter networks) with a transition with multiple inputs and outputs, considering certain structural constraints. W. Brauer et al. [24] described the extension of transition functionality for subnetworks with one initial and one final transition and the method of creating macroplaces.

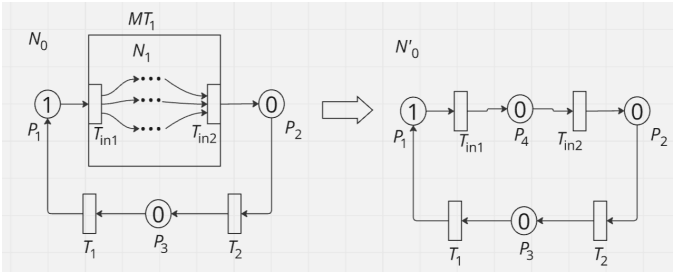

Fig. 1. Using well-formed block conception to examine a subnet N_0 properties (Three dots indicate any network structure that meets the well-formed block condition)

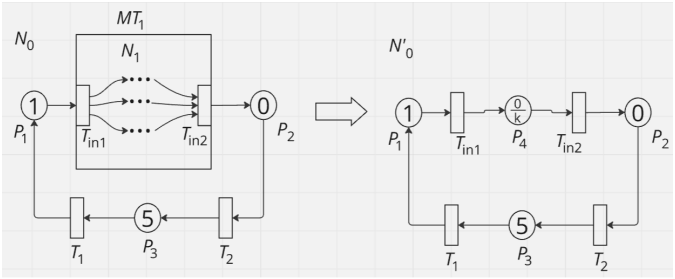

Fig. 2. Using k-well-behaved block conception to examine a subnet N_0 properties (three dots indicate any network structure that meets the k-well-behaved block condition)

L. Bernardinello and F. De Cindio [25] in their review study presented and compared various approaches to extending the functionality of places and transitions. Most of these concepts use the composition of mutually synchronized subnetworks with a state machine structure. W. M. Van Der Aalst [26] described a hierarchical workflow network that has features such as safeness, liveness, boundedness, free-choice, and well-structuredness. The network was used to model the control of various aspects of business processes. H. Huang et al. [27] proposed a method for extending functionality that ensures nineteen system properties. Their methodology is used to create and verify design specifications and includes five classes of operators: bottom-up extension, composition, functionality extension, reduction, and merging of places to solve resource-sharing problems.

The literature also considers extensions of the functionality of actions/operations [24], [28]–[32]. In this methodology, actions are assigned to transitions. Each transition associated with a subnetwork has an action label, which in this subnetwork is interpreted by more complex processes. This approach allows for hierarchical modeling of the system and its analysis at various levels of abstraction.

III. Reduction

Another approach to hierarchy in PNs is reduction. This method allows for determining the static and dynamic properties of a network based on its reduced equivalent, to which it has been transformed [33]. The transformation is performed based on specific rules applied in a certain order. Reduction is a method that facilitates proving the properties of the all network. The beginning of research related to it dates back to the 1970s [34]–[37]. This research was generalized by G. Berthelot [38], [39], who proposed transformations ensuring properties such as safety, boundedness, coverage by place invariants (S-invariant), liveness, marking reachable from any reachable marking, proper termination, home state, deadlock freeness, unavoidable states, and abstraction. Among the transformations, one can distinguish: place transformations, transition fusions, S-decomposition, and T-decomposition, which are mainly based on static properties and to a lesser extent on dynamic properties. S-decomposition and T-decomposition allow dividing a large complex network into smaller subnetworks and their separate analysis. Berthelot also described the merging of separate subnetworks through composition. K. H. Lee et al. [40], [41] proposed a hierarchical reduction method that uses decomposition. A large network is divided into smaller subnetworks, which are replaced by macroplaces and macrotransitions. The method is based on static properties and ensures the maintenance of properties such as liveness, boundedness, and proper termination.

J. Desel, E. Best, and J. Esparza [42]–[45] described the reduction of free-choice networks. In the works [42]–[44], extensions of free-choice networks using four types of reductions: P-reduction, T-reduction, F-reduction, and A-reduction were proposed. The authors showed that it is possible to reduce a free-choice network, after meeting certain conditions, to a marking graph and a state machine, and even a network consisting of two elements. Desel and Esparza [45] summarized the results related to free-choice networks and their properties, including proper formulation. L. Jiao et al. [46], [47] continued the research of Desel, Best, and Esparza. Their work focuses on asymmetric free-choice networks.

The reduction technique has also found application in time PN. R. H. Sloan and U. Buy [48] proposed extending Berthelot's reduction techniques [38] for time PN. J. Wang et al. [49] presented a component reduction method in time PN, which allows maintaining time properties and significantly simplifying the network structure by replacing its modules, e.g., with two places connected by a transition. E. Y. T. Juan et al. [50] presented the reduction of time PN used for modeling real-time systems. The authors described the concept of adding time intervals to the weights of arcs, delaying the movement of tokens between places and transitions. The reduction technique can also be significant in Flexible Manufacturing Systems (FMS), where managing shared resources and avoiding deadlocks are very important [51], [52].

IV. Formal definition

The introduction of hierarchy in PNs through formal description and network definition allows for the creation of

subnetworks in any way and with any properties, beyond imposing certain initial constraints (such as the number of network inputs and outputs).

One of the most complete concepts is the hierarchical Coloured PN (CPN) proposed by P. Huber, K. Jensen, and others [14], [53], [54]. In [53], Huber et al. showed various approaches to hierarchy in CPN, such as creating subnetworks (so-called pages linked with transitions and places), fusion sets, substitution of places and transitions, and the hierarchy graph of pages. Additionally, they introduced the concept of element instances and hierarchical structures modeled on object-oriented programming languages. Jensen, who continued research on CPN [14], [54], introduced a formal description and selected from the previously presented concepts those that would be used in practice, i.e., implemented in a computer simulator of coloured networks. He proposed a definition according to which the network can consist of separate subnetworks called pages, which are linked with transitions. The network includes instances of places, transitions, arcs, and subnetworks. Additionally, it is possible to define so-called place fusions: global place fusion, place fusion (shown in Figure 3) for pages of the same instance, and instance place fusion. Relationships between subnetworks (not related to place fusion) can be visualized on the hierarchy graph.

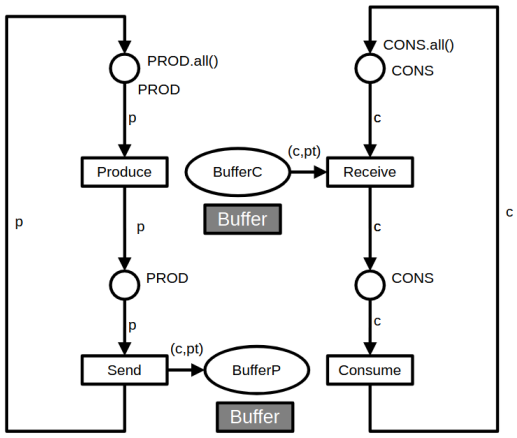

Fig. 3. Fusion places concept: BufferC and BufferP are in fact the same place

The concept of CPN is complemented by the CPN Tools simulator, which allows simulating the operation of a hierarchical network, analyzing the properties of the entire network and its subnetworks. This tool can be used to solve practical problems. Examples of industrial implementations using this tool can be found in the book by Jensen and L. M. Kristensen [55]. Many authors have eagerly referred to and continued research related to hierarchical CPN, e.g., by using this network to create multi-agent systems [56], or by using an extended version of the network to create decision systems [57]. Many works on object-oriented networks refer to colored Petri nets, and they will be presented in section VII.

A different approach to the use of hierarchy was presented by T. Holvoet and P. Verbaeten [58], namely an alternative definition of a hierarchical network. Recursion was used for the formal description. Each subnetwork can contain places and transitions, inside which there are subnetworks, which can contain further subnetworks, and so on. X. He [59] introduced a formal definition of hierarchical PN for modeling large, complex, and parallel distributed systems using the concepts of macrotransitions and macroplaces. In the proposed solution, the hierarchy in the network has a tree structure, which simplifies the definition. G. Andrzejewski [60] presented a formal model of a reactive system specification based on a hierarchical time interpreted PN, which has state memory at various levels of the network hierarchy. Remembering the marking of places inside a subnetwork asigned to a macroplace node, after the token leaves the node, was realized by assigning a history attribute to the macroplace node. H. Pan and J. Sun [61] proposed a hierarchical fuzzy PN, which can be used to model decision systems. The network defines both abstract places and transitions (macroplaces and macrotransitions).

M. Markiewicz and L. Gniewek [62], [63] proposed Hierarchical Fuzzy Interpreted PN (HFIPN), that can have macroplaces with several input, output and input-output places. Moreover, functionality of a macroplace instance was added to the network. The hierarchical network structure of HFIPN can be displayed on a hierarchy graph. Authors describe also formal algebraic representation of HFIPN, the rules of conversion of it to its flat version and the way of the combination of any two subnets in the hierarchical network. The whole concept is complemented by a software simulator called HFPIN-SML, that allows automatic code generation for PLC controllers based on the network graph [64].

V. Hierarchical Networks in Industrial Standards

The PN formalism, along with its hierarchical structure, has found application in industrial controller programming standards. The use of PN for writing control programs for PLC controllers was first standardized in France as the Grafcet standard, which was then described in the international standard IEC 848:1988 (currently IEC 60848:2013 [65]). Based on Grafcet, the Sequential Function Chart (SFC) was introduced into the IEC 1131-3:1992 standard (currently IEC 61131-3:2013 [66]) as one of the five programming languages for PLC controllers.

Leading hardware manufacturers have adapted to these standards. Programming PLC controllers using the SFC language is supported by commercial companies such as Siemens AG – Step7 Professional software [67], Omron Corp. – CX Programmer [68], and Rockwell Automation Inc. (Allen-Bradley) – Studio 5000 Logix Designer [69]. Schneider controllers (Modicon, Telemecanique) can be programmed using the SFC language – Unity Pro software [70] as well as the Grafcet language – PL7 software [71].

Formal analysis methods proposed by R. David and H. Alla [72] can be used to analyze the SFC and Grafcet languages. The theoretical foundations of Grafcet can also be found in another work by these authors [73]. Both Grafcet and SFC are based on safe networks, meaning that a maximum of one token can be stored in each place called a step.

In the Grafcet standard, it is possible to use several functionalities operating based on hierarchy. The first is the concept of

so-called macro steps, which assumes that a token transferred to the macro step node is automatically transferred to the input step of the subnetwork associated with the macro step. The token can leave the macro step node when the token in the subnetwork associated with it reaches the output step. A maximum of one token can be transferred to the macro step node at a time (subnetwork associated with it). Another concept is the so-called enclosure, in which the subnetwork is associated with a single enclosing step. When a token reaches the enclosing step, the initial steps (marked with an asterisk) are activated, and when the token leaves the enclosing step, all steps of the subnetwork associated with it are deactivated. A different functionality is the ability to force subnetwork states: freezing and resuming the operation of the subnetwork and activating and deactivating selected steps. Unlike the enclosure technique, the states of a given subnetwork can be controlled using different steps.

In the IEC 61131-3 standard, a hierarchical structure for the SFC language is not directly defined. However, the way actions assigned to steps are defined indirectly introduces the possibility of hierarchical nesting, as actions can be implemented using SFC and other languages from this standard. This concept is shown in Fig. 4.

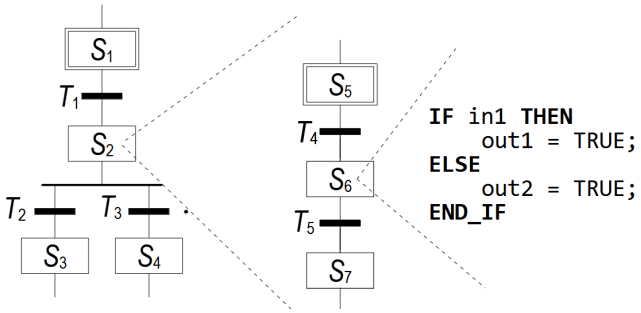

Fig. 4. Conception of nesting networks and other programs in SFC

VI. Methodology for Handling Complex Systems

As a separate direction for the application of hierarchy in PN, one can distinguish the preparation and use of methodologies for handling complex systems, such as production systems. This approach is usually presented in the form of a general verbal description supplemented with examples and formal descriptions.

M. Silva and R. Valette [74] proposed modeling FMS systems using hierarchical PN. The application of hierarchy in such systems is possible thanks to the stepwise extension of the functionality of places/transitions and modular composition. K. P. Valavanis [75] described a methodology consisting of three steps for modeling FMS systems using extended PN. This network includes six types of places, different tokens, and so-called inhibitor arcs and activator arcs. Additionally, the author applied decomposition (top-down) and composition (bottom-up) techniques. M. Zhou et al. [76] proposed a hybrid methodology (combining top-down and bottom-up analysis), which allows dividing the network into modules and ensuring them appropriate properties, such as boundedness, liveness, and reversibility. According to the concept of Zhou et al., decomposition (the creation of modules/subnetworks) is first carried out in the system. Then, non-shared resources used within these modules are determined. Finally, during the bottom-up analysis, shared resources between subnetworks are determined. M. Silva and R. Valette [74] proposed modeling FMS systems using hierarchical PN. The application of hierarchy in such systems is possible thanks to the stepwise extension of the functionality of places/transitions and modular composition.

M. D. Jeng and F. DiCesare [77] described the use of PN for modeling automated production systems with shared resources. The combination of modules is done through transitions shared by different subnetworks. The properties of the system that the methodology allows to achieve are liveness and boundedness. M. Zhou [78] showed a practical example of modular modeling of a semiconductor production system, adapted from [77]. The production system, operating based on a time PN, is divided into processes - subnetworks, in which various modules, e.g., representing resources, are distinguished. Zhou applied various methods, including reduction, which facilitate finding the properties of the entire system and described testing, simulating, scheduling, and controlling the production system.

Based on the cited works, it can be observed that they combine the concepts presented in the previous sections.

VII. Object-oriented Petri Nets

In this subsection, an overview of the work on Object-oriented PNs (OPNs), which are a popular research topic, is provided. One of the first works introducing object-orientation to PN is the article by C. Sibertin-Blanc [79], who proposed replacing the token known from classical PN with a set of objects (entities). This approach is based on a concept used in database theory. R. Valette et al. [80] extended the OPN described by Sibertin-Blanc to a fuzzy time network. R. Bastide and P. A. Palanque [81] described the application of Sibertin-Blanc's network to creating user interfaces.

G. Bruno and A. Balsamo [82] proposed a three-step methodology for modeling distributed systems (production systems) based on OPN. First, for each class of objects, e.g., machines, a control algorithm and synchronization using PN are created. Then, information defining the internal states of objects and the way of communication between objects is introduced. Finally, connections between objects are created using a class flow diagram. In [83], G. Bruno and M. Morisio continued their research and described a programming environment for creating specifications, modeling, and prototyping discrete event system based on OPN.

One of the first works presenting the addition of morphism to PN is the article by R. Fehling [84]. This author's research was later extended by the work of B. Farwer and K. Misra [85], who proposed combining hierarchical PN using morphism with features of object-oriented programming languages.

Y. K. Lee and S. J. Park [86] described OPN for modeling real-time systems. The work focuses on separating communication between objects and synchronization constraints from

the internal structure of objects. A two-step procedure is used to verify the correctness of a system consisting of hierarchically organized objects and relationships between them, allowing for the reduction of computational complexity.

C. Lakos and C. Keen [87] proposed the textual language LOOPN++ using OPN to model complex concurrent systems. This language allows for dynamic object creation. Thanks to the formal description of LOOPN++, it is possible to convert OPN to Coloured PN (CPN) and formally analyze the network. Lakos continued research related to combining object-oriented concepts and CPN in [88]–[90]. The proposed OPN allows for the use of inheritance mechanisms and related polymorphism and dynamic binding. Interaction between subnetworks can be synchronous and asynchronous. The network allows for the use of macroplaces, macrotransitions, place and transition fusions, and polymorphism. Possible application areas for Lakos's OPN include modeling information systems, controlling applets written in Java, and distributed applications.

R. Bastide [91] described two main trends within object networks: objects inside networks and networks inside objects. The solution proposed by the author combines both approaches. R. Esser [92] proposed a methodology for designing embedded systems. The presented time OPN allows for the automatic design of complex systems subject to real-time constraints.

M. Češka et al. [93] described an OPN that uses the RPC (Remote Procedure Call) mechanism for communication between objects and the net invocation mechanism known from the work of P. Huber et al. [53]. Objects in the network have the functionality of active servers that communicate with other objects through services (methods). Methods and independent activities of objects are described using PN. The OPN consists of networks organized into classes. J. E. Hong and D. H. Bae [94], [95] presented the definition of hierarchical OPN, which supports many object-oriented features such as abstraction, encapsulation, modularization of objects, communication between objects, polymorphism, and inheritance.

Z. Jiang et al. [96] proposed OPN with a variable structure, which is realized by modifying the message-passing relationships between separate objects and by adding and removing objects in the network. In [97], Jiang et al. continued their research and modified the network to a time OPN with a variable structure to analyze the performance of the production system.

M. Dong and F. F. Chen [98] presented the use of OPN for modular modeling and analysis of the flow of information, raw materials, products, and services within the Manufacturing Supply Chain (MSC). MSC includes all business activities from acquiring raw materials to final delivery to the customer. The authors use the P-invariant method for analysis.

R. Valk in his works [99]–[101] described OPN, in which a subnetwork being an object is inside the token (Fig. 5). Such a subnetwork combines the features of a token and a regular subnetwork. It can change its position in the network between places and transitions and its states (marking) during movement. This approach enables two-level modeling and can be used in systems requiring agent-oriented programming, workflow modeling, and task organization, as well as in network solutions. The properties of the OPN proposed by Valk were studied by M. Köhler and H. Rölke [102], who examined the reachability of marking and the boundedness of the network. The conclusion from their work is that the reachability of marking cannot be predicted, while it is possible to determine whether an elementary network is bounded.

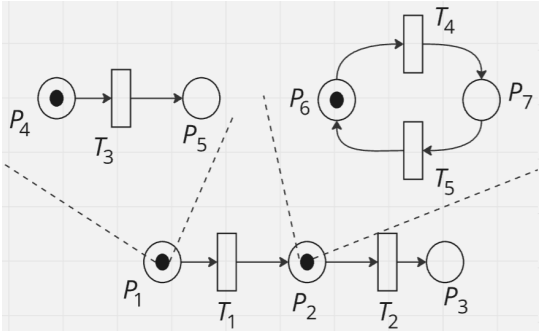

Fig. 5. Object-oriented network with token subnetworks

D. Moldt and F. Wienberg described [56] an agent-oriented CPN (with object-oriented features) that can be used to model multi-agent systems. This network is consistent with the agent programming concept proposed by Y. Shoham [103]. D. Moldt and R. Valk [104] proposed a Coloured OPN (COPN) used for modeling business processes. This network, thanks to the combination of the OPN concept proposed by Moldt [105] and Valk [99], takes into account the changing workflow requirements in business systems. Modeling these systems requires dynamic adaptation by adding, replacing, and removing tokens being subnetworks.

C. Maier and D. Moldt [106] presented COPN that can be used for dynamic system modeling. In this solution, traditional UML-based modeling, including state charts, communication diagrams, and sequence diagrams, is extended. The disadvantage of UML diagrams is their lack of unambiguous conversion to the implemented application. This problem can be solved by the formal description of the proposed COPN. J. Saldhana and S. M. Shatz [107] presented a methodology for analyzing and validating UML state diagrams using OPN. First, state diagrams are converted to a flat state machine, based on which OPN models are generated. These models are then combined into a UML communication diagram, from which a regular CPN is obtained.

X. F. Zha [108] proposed a fuzzy OPN intended for knowledge-based expert systems implementation. The author presented a methodology combining design and assembly planning processes, which uses, among other things, artificial intelligence concepts based on agents. O. Biberstein et al. [109] described the CO-OPN/2 network formalism. This OPN can be used to create specifications for complex concurrent systems. Object-orientation elements introduced to CO-OPN/2 include concepts such as classes and objects, object references, and inheritance mechanisms. Classes are treated as networks, places as attributes, and methods as external parameterized transitions. S. Chachkov and D. Buchs [110] presented a method for generating code based on the CO-OPN network. The proposed methodology uses transactional mechanisms.

X. Meng [111] showed the application of time OPN for modeling reconfigurable production systems. This network allows for assessing the change in system performance in response to, e.g., adding machines to the system or changing the software of machines and storage tools. Decomposition and modular composition techniques are employed to achieve this purpose. The resulting network can be subjected to formal analysis and validation.

T. Miyamoto and K. Horiguchi [112] proposed a reachability graph for analyzing the operation of multi-agent PN, a variant of OPN based on CPN. To study the properties of OPN, its graph must be transformed into a modular PN and analyzed using the method proposed by S. Christensen and L. Petrucci [113]. A. Schumann and K. Pancerz [114] described the use of OPN for modeling the behavior of Physarum Polycephalum, a single-celled organism that can be used to solve various computational problems and serve, for example, to implement logic gates.

S. Hammami and H. Mathkour [115] proposed using agent-based OPN to implement an adaptive e-learning system. The research addresses two important areas: first, the construction of a multi-agent architecture that adapts to the learner's preferences, and second, the analysis and control of the communication between interacting agents in the system.

Ö. Başak and Y. E. Albayrak [116] described the use of OPN for designing and implementing a control algorithm for FMS. This OPN allows for system performance analysis and increases the effectiveness of production process control. X. Y. Wu and X. Y. Wu [117] presented the use of extended OPN to simulate mission reliability in repairable systems with phased missions. The authors implemented a software tool in C# that allows for the analysis and simulation of the proposed network.

Many other works address topics related to OPNs, but based on the study shown in this section, it can be seen that OPNs have found applications in many different scientific and practical fields. This allows to conclude that the use of OPNs is very versatile and can provide many benefits by combining the advantages of PN (e.g., formal description of operation and graphical presentation) with the features of object-oriented programming languages.

VIII. Discussion

The boundary between the research areas discussed in this manuscript is not clear, as they often overlap and can complement each other. For example, an important part of the methodology for handling large systems can be the reduction technique [78]. Furthermore, when applying hierarchy, the decomposition (top-down), composition (bottom-up), and their combination techniques can be used [33], [73], [118]. In some cases, high-level hierarchical networks may also have features of object-oriented programming languages [84], [85], [94], [99], [101], [119], [120].

Hierarchy in PN can be implemented in various ways. The mentioned literature distinguishes two main concepts of hierarchical structure. The first is the use of a macrotransition (also called an abstract transition), inside which there is a separate subnetwork. The second is the use of a macroplace (also called an abstract place), where the subnetwork is placed inside it. The choice of whether a given network fragment should be classified as a macroplace or a macrotransition is not clear for different classes of PN. In both approaches, the decisive condition is the type of input and output elements. According to one concept, a given network fragment is classified as a macroplace if there are places at its input and output, or as a macrotransition if there are transitions at its input and output [21], [22], [40], [58]–[60], [65], [84]. In other approaches, it is exactly the opposite [14], [53], [54], [61].

The possibility of using different types of macros and the number of their input and output elements also varies for different PN. In some networks, only macroplaces are used [60], [65], in others only macrotransitions [14], [53], [54], while in others both macroplaces and macrotransitions [40], [58], [59], [61], [84]. Moreover, hierarchical networks use macros with one input and output element, any number of input/output elements, and macros that have a strictly defined number of inputs and outputs [40].

Another aspect of hierarchy is the number of tokens and the conditions that determine their movement through macros, i.e., through subnetworks associated with them. The most common approaches are two. Any number of tokens can be inserted into the macro, resulting from the structure of the subnetwork [14], [53], [54], or only one token [65]. However, in [58], this is conditioned by the sum of the weights of the input and output arcs of the macro. Moreover, in the case of steps enclosure [65], one token is transferred to a step, while the tokens appear in all initial places of the subnet associated with this parent node. Forcing subnetwork steps states by command assigned to the parent node can also be used [65].

Based on the literature cited in the previous subsections, it can be concluded that the mere application of hierarchy in PN will not speed up or increase the efficiency of the system. However, using hierarchy by dividing the system into components, separate analysis, modification, and simulation can improve its performance. The hierarchical approach can also be used at the system design stage, thus speeding up its creation process by allowing the network to be considered at different levels of detail.

IX. Conclusion

This paper discusses different approaches to introducing hierarchy in PNs. Based on the analyzed literature, the following group have been identified:

- refinement of places and transitions,
- reduction,
- formal definition,
- hierarchical networks in industrial standards,
- methodology for handling complex systems,
- object-oriented PNs.

The boundary between the these approaches is not clear and they often share common elements. The application of hierarchy undoubtedly increases the practical value of PNs and facilitates the process of implementation and design of the system based on PNs.

The authors claim the following contributions to this manuscript:

- collecting and analyzing the literature related to hierarchy in PNs,
- identifying various ways of applying hierarchical structure and analyzing them,
- drawing conclusions and summarizing the application of hierarchy in PNs.

ACKNOWLEDGMENT

The authors would like to thank experts for their appropriate and constructive suggestions to improve this template.

REFERENCES

[1] M. D. Mesarovic, D. Macko, and Y. Takahara, *Theory of hierarchical, multilevel, systems*. Elsevier, 2000.

[2] C. Sunder, A. Zoitl, M. Rainbauer, and B. Favre-Bulle, "Hierarchical control modelling architecture for modular distributed automation systems," in *2006 4th IEEE International Conference on Industrial Informatics*. IEEE, 2006, pp. 12–17.

[3] A. Bidram and A. Davoudi, "Hierarchical structure of microgrids control system," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1963–1976, 2012.

[4] B. P. Zeigler, *Object-oriented simulation with hierarchical, modular models: intelligent agents and endomorphic systems*. Academic press, 2014.

[5] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2010.

[6] S. Makris, *Cooperating robots for flexible manufacturing*. Springer, 2021.

[7] X. Zhang, R. Mangal, M. Naik, and H. Yang, "Hybrid top-down and bottom-up interprocedural analysis," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2014, pp. 249–258.

[8] E. M. Nystrom, H.-S. Kim, and W.-M. W. Hwu, "Bottom-up and top-down context-sensitive summary-based pointer analysis," in *Static Analysis: 11th International Symposium, SAS 2004, Verona, Italy, August 26-28, 2004. Proceedings 11*. Springer, 2004, pp. 165–180.

[9] J. M. Rodriguez, M. Crasso, C. Mateos, A. Zunino, and M. Campo, "Bottom-up and top-down cobol system migration to web services," *IEEE Internet Computing*, vol. 17, no. 2, pp. 44–51, 2011.

[10] K.-Q. Zhou and A. M. Zain, "Fuzzy petri nets and industrial applications: a review," *Artificial Intelligence Review*, vol. 45, pp. 405–446, 2016.

[11] A. Giua and M. Silva, "Petri nets and automatic control: A historical perspective," *Annual Reviews in Control*, vol. 45, pp. 223–239, 2018.

[12] Z. Li, N. Wu, and M. Zhou, "Deadlock control of automated manufacturing systems based on petri nets—a literature review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 437–462, 2011.

[13] J. L. Peterson, "Petri nets," *ACM Computing Surveys (CSUR)*, vol. 9, no. 3, pp. 223–252, 1977.

[14] K. Jensen, *Coloured Petri nets: basic concepts, analysis methods and practical use*. Springer Science & Business Media, 1997, vol. 1.

[15] R. David and H. Alla, *Discrete, continuous, and hybrid Petri nets*. Springer, 2005, vol. 1.

[16] C. Yuan, Y. Liao, L. Kong, and H. Xiao, "Fault diagnosis method of distribution network based on time sequence hierarchical fuzzy petri nets," *Electric Power Systems Research*, vol. 191, p. 106870, 2021.

[17] M. Figat and C. Zieliński, "Parameterised robotic system meta-model expressed by hierarchical petri nets," *Robotics and Autonomous Systems*, vol. 150, p. 103987, 2022.

[18] S. Souravlas, S. Anastasiadou, and I. Kostoglou, "A novel method for general hierarchical system modeling via colored petri nets based on transition extractions from real datasets," *Applied Sciences*, vol. 13, no. 1, p. 339, 2022.

[19] N. Ali, S. Punnekkat, and A. Rauf, "Modeling and safety analysis for collaborative safety-critical systems using hierarchical colored petri nets," *Journal of Systems and Software*, vol. 210, p. 111958, 2024.

[20] W. Yue, L. Hou, X. Wan, X. Chen, and W. Gui, "Superheat degree recognition of aluminum electrolysis cell using unbalance double hierarchy hesitant linguistic petri nets," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–15, 2023.

[21] R. Valette, "Analysis of petri nets by stepwise refinements," *Journal of computer and system sciences*, vol. 18, no. 1, pp. 35–46, 1979.

[22] I. Suzuki and T. Murata, "A method for stepwise refinement and abstraction of petri nets," *Journal of computer and system sciences*, vol. 27, no. 1, pp. 51–76, 1983.

[23] W. Vogler, "Behaviour preserving refinements of petri nets," in *Graph-Theoretic Concepts in Computer Science*, ser. Lecture Notes in Computer Science, G. Tinhofer and G. Schmidt, Eds. Springer Berlin Heidelberg, Jun. 1986, no. 246, pp. 82–93. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-17218-1_51

[24] W. Brauer, R. Gold, and W. Vogler, "A survey of behaviour and equivalence preserving refinements of petri nets," in *Advances in Petri Nets 1990*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Berlin Heidelberg, Jun. 1989, pp. 1–46. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-53863-1_19

[25] L. Bernardinello and F. De Cindio, "A survey of basic net models and modular net classes," in *Advances in Petri Nets 1992*. London, UK: Springer-Verlag, 1992, pp. 304–351. [Online]. Available: http://dl.acm.org/citation.cfm?id=647749.734386

[26] W. M. Van Der Aalst, "Workflow verification: finding control-flow errors using petri-net-based techniques," in *Business Process Management*. Springer, 2000, pp. 161–183.

[27] H. Huang, L. Jiao, and T.-Y. Cheung, *Property-Preserving Petri Net Process Algebra in Software Engineering*. World Scientific, 2012.

[28] R. Van Glabbeek and U. Goltz, "Refinement of actions in causality based models," in *Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness*. Springer, 1989, pp. 267–300. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-52559-9_68

[29] E. Best, R. Devillers, A. Kiehn, and L. Pomello, "Concurrent bisimulations in petri nets," *Acta Informatica*, vol. 28, no. 3, pp. 231–264, Mar. 1991. [Online]. Available: http://link.springer.com/article/10.1007/BF01178506

[30] W. Vogler, "Bisimulation and action refinement," *Theoretical Computer Science*, vol. 114, no. 1, pp. 173–200, Jun. 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/030439759390157O

[31] R. Van Glabbeek and U. Goltz, "Refinement of actions and equivalence notions for concurrent systems," *Acta Informatica*, vol. 37, no. 4-5, pp. 229–327, 2001. [Online]. Available: http://link.springer.com/article/10.1007/s002360000041

[32] L. Jiao, "Refining and verifying regular petri nets," *International Journal of Systems Science*, vol. 39, no. 1, pp. 17–27, 2008. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/00207720701621959

[33] C. Girault and R. Valk, *Petri nets for systems engineering: a guide to modeling, verification, and applications*. Springer Science & Business Media, 2013.

[34] M. Hack, "Analysis of production schemata by petri nets," Massachusetts Institute of Technology. Cambridge, Mass, Master's thesis, 1972.

[35] R. J. Lipton, "Reduction: A method of proving properties of parallel programs," *Communications of the ACM*, vol. 18, no. 12, pp. 717–721, 1975. [Online]. Available: http://dl.acm.org/citation.cfm?id=361234

[36] Y. S. Kwong, "On reduction of asynchronous systems," *Theoretical Computer Science*, vol. 5, no. 1, pp. 25–50, aug 1977. [Online]. Available: http://www.sciencedirect.com/science/article/pii/030439757790041X

[37] W. Kowalk and R. Valk, "On reductions of parallel programs," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, H. A. Maurer, Ed. Springer Berlin Heidelberg, jul 1979, no. 71, pp. 356–369. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-09510-1_29

[38] G. Berthelot, "Checking properties of nets using transformations," in *European Workshop on Applications and Theory in Petri Nets*. Springer, 1985, pp. 19–40.

[39] ——, "Transformations and decompositions of nets," in *Petri Nets: Central Models and Their Properties*, ser. Lecture Notes in Computer Science, W. Brauer, W. Reisig, and G. Rozenberg, Eds. Springer Berlin Heidelberg, 1987, no. 254, pp. 359–376. [Online]. Available: http://link.springer.com/chapter/10.1007/BFb0046845

[40] A. Name, "Hierarchical reduction method for analysis and decomposition of petri nets," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, pp. 272–280, March-April 1985.

[41] K. H. Lee, J. Favrel, and P. Baptiste, "Generalized petri net reduction method," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 17, no. 2, pp. 297–303, 1987. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4309041

[42] J. Desel, "Reduction and design of well-behaved concurrent systems," in *CONCUR'90 Theories of Concurrency: Unification and Extension*. Springer, 1990, pp. 166–181. [Online]. Available: http://link.springer.com/chapter/10.1007/BFb0039059

[43] E. Best and J. Desel, "Partial order behaviour and structure of petri nets," *Formal aspects of computing*, vol. 2, no. 1, pp. 123–138, 1990. [Online]. Available: http://link.springer.com/article/10.1007/BF01888220

[44] J. Esparza, "Reduction and synthesis of live and bounded free choice petri nets," *Information and Computation*, vol. 114, no. 1, pp. 50–87, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0890540184710807

[45] J. Desel and J. Esparza, *Free choice Petri nets*. Cambridge University Press, 2005, vol. 40.

[46] L. Jiao, T.-Y. Cheung, and W. Lu, "Characterizing liveness of petri nets in terms of siphons," in *International Conference on Application and Theory of Petri Nets*. Springer, 2002, pp. 203–216.

[47] ——, "On liveness and boundedness of asymmetric choice nets," *Theoretical Computer Science*, vol. 311, no. 1, pp. 165–197, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397503003591

[48] R. H. Sloan and U. Buy, "Reduction rules for time petri nets," *Acta Informatica*, vol. 33, no. 7, pp. 687–706, 1996. [Online]. Available: http://link.springer.com/article/10.1007/s002360050066

[49] J. Wang, Y. Deng, and M. Zhou, "Compositional time petri nets and reduction rules," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 30, no. 4, pp. 562–572, aug 2000. [Online]. Available: https://doi.org/10.1109/3477.865173

[50] E. Y. T. Juan, J. J. P. Tsai, T. Murata, and Y. Zhou, "Reduction methods for real-time systems using delay time petri nets," *IEEE Transactions on Software Engineering*, vol. 27, no. 5, pp. 422–448, may 2001. [Online]. Available: https://doi.org/10.1109/32.922714

[51] J.-M. Proth and X. Xie, *Petri nets: a tool for design and management of manufacturing systems*. John Wiley & Sons, 1996, vol. 6.

[52] M. Uzam, "The use of the petri net reduction approach for an optimal deadlock prevention policy for flexible manufacturing systems," *The International Journal of Advanced Manufacturing Technology*, vol. 23, no. 3, pp. 204–219, feb 2004. [Online]. Available: https://doi.org/10.1007/s00170-002-1526-5

[53] P. Huber, K. Jensen, and R. M. Shapiro, "Hierarchies in coloured petri nets," in *Advances in Petri Nets 1990*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Berlin Heidelberg, Jun. 1989, no. 483, pp. 313–341. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-53863-1_30

[54] K. Jensen, "Coloured petri nets: A high level language for system design and analysis," in *Advances in Petri Nets 1990*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Berlin Heidelberg, Jun. 1989, no. 483, pp. 342–416. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-53863-1_31

[55] K. Jensen and L. M. Kristensen, *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media, 2009.

[56] D. Moldt and F. Wienberg, "Multi-agent-systems based on coloured petri nets," in *Application and Theory of Petri Nets 1997*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jun. 1997, pp. 82–101. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-63139-9_31

[57] J. F. Peters, A. Skowron, Z. Suraj, W. Pedrycz, and S. Ramanna, "Approximate real-time decision making: concepts and rough fuzzy petri net models," *International Journal of Intelligent Systems*, vol. 14, no. 8, pp. 805–839, 1999. [Online]. Available: https://www.researchgate.net/profile/James_Peters/publication/2804691_Approximate_Real-Time_Decision_Making_Concepts_and_Rough_Fuzzy_Petri_Net_Models/links/0fcfd510a880d3eff0000000.pdf

[58] T. Holvoet and P. Verbaeten, "Petri charts: an alternative technique for hierarchical net construction," in *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, 1995, vol. 3, pp. 2688–2693.

[59] X. He, "A formal definition of hierarchical predicate transition nets," in *Application and Theory of Petri Nets 1996*, ser. Lecture Notes in Computer Science, J. Billington and W. Reisig, Eds. Springer Berlin Heidelberg, Jun. 1996, no. 1091, pp. 212–229. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-61363-3_12

[60] G. Andrzejewski, "Hierarchical petri nets for digital controller design," in *Design of embedded control systems*. Springer, 2005, pp. 27–36.

[61] H. Pan and J. Sun, "Complex knowledge system modeling based on hierarchical fuzzy petri net," in *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*. IEEE, 2007, pp. 31–34.

[62] M. Markiewicz and L. Gniewek, "Conception of hierarchical fuzzy interpreted petri net," *Stud. Inf. Control*, vol. 26, pp. 151–160, 2017.

[63] M. Markiewicz, L. Gniewek, and D. Warchoł, "Extended hierarchical fuzzy interpreted petri net," *Sensors*, vol. 21, no. 24, p. 8433, 2021.

[64] M. Markiewicz and L. Gniewek, "A program model of fuzzy interpreted petri net to control discrete event systems," *Applied Sciences*, vol. 7, no. 4, p. 422, 2017.

[65] International Electrotechnical Commission, "International standard iec 60848:2013: Grafcet specification language for sequential function charts approach," International Electrotechnical Commission, Geneva, CH, International Standard, 2013.

[66] "International standard iec 61131-3: Programmable controllers - part 3: Programming languages," International Electrotechnical Commission, Geneva, CH, International Standard, 2013.

[67] *System Manual, STEP 7 Professional V12.0*, Siemens AG, 2013.

[68] *Operation Manual, SFC Programming*, Omron Corporation, 2010.

[69] *Programming Manual, Logix5000 Controllers Sequential Function Charts*, Rockwell Automation Publication, 1756-PM006F-EN-P, 2014.

[70] *Reference manual, Unity Pro. Program Languages and Structure*, Schneider Electric, 2014.

[71] *Reference manual, PL7 Micro/Junior/Pro. Description of the PL7 software*, Schneider Electric, 2008.

[72] R. David and H. Alla, *Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems*. Prentice Hall, 1992.

[73] ——, *Discrete, Continuous, and Hybrid Petri Nets*. Springer, 2010.

[74] M. Silva and R. Valette, "Petri nets and flexible manufacturing," in *European Workshop on Applications and Theory in Petri Nets*. Springer, 1988, pp. 374–417.

[75] K. P. Valavanis, "On the hierarchical modeling analysis and simulation of flexible manufacturing systems with extended petri nets," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 1, pp. 94–110, jan 1990. [Online]. Available: https://doi.org/10.1109/21.47812

[76] M. Zhou, F. DiCesare, and A. A. Desrochers, "A hybrid methodology for synthesis of petri net models for manufacturing systems," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 350–361, 1992.

[77] M. D. Jeng and F. DiCesare, "Synthesis using resource control nets for modeling shared-resource systems," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 317–327, jun 1995. [Online]. Available: https://doi.org/10.1109/70.388774

[78] M. Zhou, "Modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems: A petri net approach," *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, no. 3, pp. 333–357, aug 1998. [Online]. Available: https://doi.org/10.1109/66.705370

[79] G. Sibertin-Blanc, "High-level petri nets," *Journal of Computer Science*, vol. 12, no. 4, pp. 345–356, 1985.

[80] R. Valette, J. Cardoso, and D. Dubois, "Monitoring manufacturing systems by means of petri nets with imprecise markings," in *IEEE International Symposium on Intelligent Control*, vol. 2526. Albany NY USA., 1989.

[81] R. Bastide and P. A. Palanque, "Petri net objects for the design, validation and prototyping of user-driven interfaces." in *INTERACT'90 Conference Proceedings*, vol. 90, 1990, pp. 625–631.

[82] G. Bruno and A. Balsamo, "Petri net-based object-oriented modelling of distributed systems," *ACM SIGPLAN Notices*, vol. 21, no. 11, pp. 284–293, Jun. 1986. [Online]. Available: https://doi.org/10.1145/960112.28725

[83] G. Bruno and M. Morisio, "Petri-net based simulation of manufacturing cells," in *1987 IEEE International Conference on Robotics and Automation Proceedings*, vol. 4, Mar. 1987, pp. 1174–1179. [Online]. Available: https://doi.org/10.1109/ROBOT.1987.1087859

[84] R. Fehling, "A concept of hierarchical petri nets with building blocks," in *Advances in Petri Nets 1993*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Berlin Heidelberg, Jun. 1991, no. 674, pp. 148–168. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-56689-9_43

[85] B. Farwer and K. Misra, "Modelling with hierarchical object petri nets," *Fundamenta Informaticae*, vol. 55, no. 2, pp. 129–147, Jan. 2003. [Online]. Available: http://content.iospress.com/articles/fundamenta-informaticae/fi55-2-04

[86] Y. K. Lee and S. J. Park, "OPNets: an object-oriented high-level Petri net model for real-time system modeling," *Journal of Systems and Software*, vol. 20, no. 1, pp. 69–86, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0164121293900494

[87] C. A. Lakos and C. D. Keen, *LOOPN++: a new language for object-oriented Petri nets*. Department of Computer Science, University of Tasmania, 1994.

[88] C. Lakos, "From coloured petri nets to object petri nets," in *Application and Theory of Petri Nets 1995*, ser. Lecture Notes in Computer Science, G. D. Michelis and M. Diaz, Eds., no. 935. Springer Berlin Heidelberg, jun 1995, pp. 278–297. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-60029-9_45

[89] ——, "The consistent use of names and polymorphism in the definition of object petri nets," in *Application and Theory of Petri Nets 1996*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, jun 1996, pp. 380–399. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-61363-3_21

[90] ——, "Object oriented modelling with object petri nets," in *Concurrent object-oriented programming and petri nets*. Springer, 2001, pp. 1–37.

[91] R. Bastide, "Approaches in unifying petri nets and the object-oriented approach," in *1st Workshop on Object-Oriented Programming and Models of Concurrency, within the 16th International Conference on Application and Theory of Petri nets*, 1995.

[92] R. Esser, *An object oriented Petri net approach to embedded system design*. Eidgenössische Technische Hochschule [ETH] Zürich, 1996.

[93] M. Češka, V. Janoušek, and T. Vojnar, "Pntalk — a computerized tool for object oriented petri nets modelling," in *Computer Aided Systems Theory — EUROCAST'97*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, feb 1997, pp. 591–610. [Online]. Available: https://link.springer.com/chapter/10.1007/BFb0025078

[94] J. E. Hong and D.-H. Bae, "Hoonets: Hierarchical object-oriented petri nets for system modeling and analysis," *KAIST Technical Report CS/TR*, pp. 98–132, 1998.

[95] J.-E. Hong and D.-H. Bae, "Software modeling and analysis using a hierarchical object-oriented petri net," *Information Sciences*, vol. 130, no. 1–4, pp. 133–164, Dec. 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025500000906

[96] Z. Jiang, M. J. Zuo, P. Y. Tu, and R. Y. K. Fung, "Object-oriented petri nets with changeable structure (opns-cs) for production system modelling," *The International Journal of Advanced Manufacturing Technology*, vol. 15, no. 6, pp. 445–459, jun 1999. [Online]. Available: https://link.springer.com/article/10.1007/s001700050089

[97] Z. Jiang, M. J. Zuo, R. Y. Fung, and P. Y. Tu, "Performance modelling of complex dynamic production systems using temporised object-oriented petri nets with changeable structure (topns-cs)," *The International Journal of Advanced Manufacturing Technology*, vol. 16, no. 7, pp. 521–536, 2000.

[98] M. Dong and F. F. Chen, "Process modeling and analysis of manufacturing supply chain networks using object-oriented petri nets," *Robotics and Computer-Integrated Manufacturing*, vol. 17, no. 1–2, pp. 121–129, feb 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584500000454

[99] R. Valk, "Petri nets as token objects," in *International Conference on Application and Theory of Petri Nets*. Springer, 1998, pp. 1–24.

[100] ——, "Concurrency in communicating object petri nets," in *Concurrent object-oriented programming and Petri nets*. Springer, 2001, pp. 164–195.

[101] ——, "Object petri nets," in *Advanced Course on Petri Nets*. Springer, 2003, pp. 819–848.

[102] M. K"ohler and H. R"olke, "Properties of object petri nets," pp. 278–297, 2004.

[103] Y. Shoham, "Agent-oriented programming," *Artificial intelligence*, vol. 60, no. 1, pp. 51–92, 1993.

[104] D. Moldt and R. Valk, "Object oriented petri nets in business process modeling," in *Business Process Management*. Springer, 2000, pp. 254–273.

[105] D. Moldt, *H"ohere Petrinetze als Grundlage f"ur Systemspezifikationen*. Department of Computer Science, University of Hamburg, 1996.

[106] C. Maier and D. Moldt, "Object coloured petri nets-a formal technique for object oriented modelling," in *Concurrent object-oriented programming and petri nets*. Springer, 2001, pp. 406–427.

[107] J. Saldhana and S. M. Shatz, "Uml diagrams to object petri net models: an approach for modeling and analysis," in *International Conference on Software Engineering and Knowledge Engineering*, 2000, pp. 103–110.

[108] X. F. Zha, "An object-oriented knowledge based petri net approach to intelligent integration of design and assembly planning," *Artificial Intelligence in Engineering*, vol. 14, no. 1, pp. 83–112, 2000.

[109] O. Biberstein, D. Buchs, and N. Guelfi, "Object-oriented nets with algebraic specifications: the co-opn/2 formalism," in *Concurrent Object-Oriented Programming and Petri Nets*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, pp. 73–130. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-45397-0_3

[110] S. Chachkov and D. Buchs, "From formal specifications to ready-to-use software components: the concurrent object oriented Petri net approach," in *Proceedings Second International Conference on Application of Concurrency to System Design*, 2001, pp. 99–110. [Online]. Available: https://doi.org/10.1109/CSD.2001.981768

[111] X. Meng, "Modeling of reconfigurable manufacturing systems based on colored timed object-oriented Petri nets," *Journal of Manufacturing Systems*, vol. 29, no. 2, pp. 81–90, Jul. 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612510000518

[112] T. Miyamoto and K. Horiguchi, "Modular reachability analysis in fundamental class of multi-agent nets," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, Nov. 2011, pp. 3782–3787. [Online]. Available: https://doi.org/10.1109/IECON.2011.6119925

[113] S. Christensen and L. Petrucci, "Modular analysis of petri nets," *The Computer Journal*, vol. 43, no. 3, pp. 224–242, 2000.

[114] A. Schumann and K. Pancerz, "Towards an object-oriented programming language for physarum polycephalum computing: A petri net model approach," *Fundamenta Informaticae*, vol. 133, no. 2-3, pp. 271–285, 2014.

[115] S. Hammami and H. Mathkour, "Adaptive e-learning system based on agents and object petri nets (AELS-A/OPN)," *Computer Applications in Engineering Education*, vol. 23, no. 2, pp. 170–190, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.21587

[116] Ö. Başak and Y. E. Albayrak, "Petri net based decision system modeling in real-time scheduling and control of flexible automotive manufacturing systems," *Computers & Industrial Engineering*, vol. 86, pp. 116–126, 2015.

[117] X.-Y. Wu and X.-Y. Wu, "Extended object-oriented petri net model for mission reliability simulation of repairable pms with common cause failures," *Reliability Engineering & System Safety*, vol. 136, pp. 109–119, 2015.

[118] M. C. Zhou and K. Venkatesh, *Modeling, simulation, and control of flexible manufacturing systems, a Petri net approach*. World Scientific, 1999.

[119] U. Becker and D. Moldt, "Object-oriented concepts for coloured petri nets," in *Proceedings of IEEE Systems Man and Cybernetics Conference - SMC*, vol. 3, Oct. 1993, pp. 279–285. [Online]. Available: https://doi.org/10.1109/ICSMC.1993.385024

[120] C. Lakos, "From coloured petri nets to object petri nets," in *International Conference on Application and Theory of Petri Nets*. Springer, 1995, pp. 278–297.