Petri Nets and PNeS in Modeling and Analysis of Concurrent Systems

Zbigniew Suraj and Piotr Grochowalski

Department of Computer Science
University of Rzeszów, Rzeszów, Poland
e-mail:{zbigniew.suraj,piotrg}@ur.edu.pl

Abstract. Petri nets are widely used in both theoretical analysis and practical modeling of concurrent systems. The practical use of Petri nets is strongly dependent upon the existence of adequate computer tools. This paper describes PNeS, integrated graphical computer based tools for construction, modification, and analysis of Petri nets. It runs on any computer under any operating system. PNeS can be useful for the researchers and practitioners, both from academia and industry, who are actively involved in the work in the area of modeling and analysis of concurrent systems, as well as those who may potentially become involved in these areas.

Key words: Petri net tools, computer based tools, Petri nets

1 Introduction

Concurrent systems, such as automated manufacturing systems, process control, communication systems, computer operating systems, office information systems, etc., are of increasing importance in today's world because they are growing in number, size, and sophistication. The complex nature of modern concurrent systems creates numerous problems for their developers. The design and operation of these systems require modeling and analysis in order to achieve desirable performance and to avoid catastrophic errors. It is generally known that the failures in the modeling processes can vitally contribute to both the development time and cost, as well as the operational efficiency. Therefore special attention should be paid to the correctness of the models that are used at all planning phases.

Petri nets are widely used in both theoretical analysis and practical modeling of concurrent systems. Their graphical aspect allows representation of various interactions between discrete events more easily. However, the mathematical aspect allows formal modeling of these interactions and analysis of the modeled system properties. Petri nets were proposed by Carl A. Petri as a net-like mathematical tool for the study of communication with automata Petri (1962). Their further development was facilitated by the fact that they present two interesting characteristics. Firstly, they make it possible to model and visualize types of behavior having parallelism, concurrency, synchronization and resource sharing. These properties characterize concurrent systems. Secondly, the theoretic results are plentiful; the properties of these nets have been and still are extensively studied. There exists a large number of books, articles, and proceedings papers devoted to the theory and applications of Petri nets, see e.g. Peterson (1981), Reisig (1985), Murata (1989), Jensen, Rozenberg (eds.) (1991), David, Alla (1992), DiCesare et al. (1993), Zurawski, Zhou (1994), Desel et al. (eds.) (2004), and Zeugmann-Popova (2013).

The practical use of Petri nets is strongly dependent upon the existence of adequate computer tools - helping the user to handle all the details of a large and complex description. For Petri nets one needs at least graphical editor, analyzer, and simulator programs. The graphical editor gives an opportunity for entering, exiting, constructing, and editing Petri nets. The analyzer allows to perform a formal check of the properties related to the behavior of the underlying system e.g. concurrent operations, appropriate synchronization, freedom from deadlock, repetitive activities, and mutual exclusion of shared resources, etc. The simulator simulates execution of a Petri net, i.e., the flow of tokens in the places of the net through transitions. Simulation gives a vivid graphic description of a system's operation to aid in model design and debugging. Simulation becomes necessary when the performance cannot be predicted by the system performance evaluator described.

The article of Feldbrugge, Jensen (1991) provides a good overview of typical Petri net tools. Some of these tools and their applications are discussed in details in references of Murata (1989), Zurawski, Zhou (1994), and Desel et al. (eds.) (2004). A very good resource of information about Petri net theory and its applications is the website named *The Petri Nets World* PNW (2017).

The objective of this paper is to present both the fundamental concepts of Petri nets and the analysis methods of the Petri net models, as well as the basic information about a set of Petri net tools, named as Petri Net System (PNeS in short). This system can be used for construction, modification and analysis of Petri net models. Currently, PNeS allows us to work with four basic classes of Petri nets, i.e., standard Petri nets, nets with inhibitor arcs, self-modifying nets, and priority nets. The resulting Petri nets are expressed in the PNML formalism, which is a standard format used by many analysis tools for Petri nets PNML (2017). Additionally, we introduce some of the most fundamental properties of Petri nets which are useful for analyzing modeled systems, and basic analysis methods which allow to check whether there exists one-to-one functional correspondence between the Petri net model and the original requirements specification; usually expressed in an informal way. PNeS can be useful for the researchers and practitioners, both from academia and industry, who are actively involved in the work in the area of modeling and analysis of concurrent systems, as well as those who may potentially be involved in these areas in the future. The paper mainly focuses on standard Petri nets, although other types of Petri nets are also discussed in the context of possible applications. PNeS runs on any computer under any operating system. It has been implemented in the *Java* programming language to guarantee portability and efficiency on any computers. PNeS is a follow-up of PN-tools which has been designed and implemented at the Pedagogical University in Rzeszów in 1986-1996. PN-tools were run on IBM PC computers under DOS operating system Suraj (1995).

This paper is organized as follows. Section 2 presents a review of basic definitions of Petri net concepts and notations. The examples of extensions to the standard Petri nets are discussed in Section 3. Some of the most fundamental properties of Petri nets are discussed in Section 4. In Section 5, the most fundamental methods of Petri net analysis are roughly described. A general description of PNeS is presented in Section 6. Section 7 includes some conclusions concerning among others future plans for PNeS.

2 Basic Definitions

The structure of a standard Petri net (a Petri net in short) is a directed graph with two kinds of nodes, *places* and *transitions*, interconnected by *arcs* - in such a way that each arc connects two different kinds of nodes (i.e., a place and a transition and vice versa).

Graphically, places are represented by circles and transitions as rectangles. A place is an input place to a transition if there exists a directed arc connecting this place to the transition. A place is an output place of a transition if there exists a directed arc connecting the transition to the place.

In a Petri net can exist directed (parallel) arcs connecting a place and a transition (a transition and a place). In such a case, in the graphical representation of a Petri net, parallel arcs connecting a place (transition) to a transition (place) are often represented by a single directed arc labeled with its multiplicity. The arc label '1' is omitted.

A Petri net is said to be *ordinary* if all of its arc multiplicities are 1's.

The dynamic behavior of the modeled system can be described in terms of its states and their changes, each place may potentially hold either none or a positive number of tokens. Pictorially, the tokens are represented by means of gray "dots" together with the suitable positive numbers placed inside the circles corresponding to appropriate places. We assume that if the number of tokens in a place equals 0 then the place is empty. A distribution of tokens on places of a Petri net is called *a marking*. It defines the current state of the system modeled by a Petri net. A marking of a Petri net with n places can be represented by an n-vector M, elements of which, denoted as $M(p)$, are nonnegative integers representing the number of tokens in the corresponding place p. The *initial marking*, denoted as M_0, is the marking determined by the initial state of a system. A Petri net containing tokens is called *a marked Petri net.*

In order to simulate the dynamic behavior of a system, a marking in a Petri net is changed according to the following *firing rule*. A transition t is said to be *enabled* if each input place p of t contains at least the number of tokens equal to the multiplicity of the directed arc connecting p to t. An enabled transition t may or may not fire (depending on whether or not the transition is selected). A firing of an enabled transition t removes from each input place p the number of tokens equal to the multiplicity of the directed arc connecting p to t, and adds to each output place p of t the number of tokens equal to the multiplicity of the directed arc connecting t to p.

A fragment of the graphical representation of a marked Petri net together with three enabled transitions (gray color) is shown in Fig. 1.

3 Extensions of Petri Nets

The simplest extension to standard Petri nets is *nets with inhibitor arcs* proposed by M. Hack (1975). An inhibitor arc leads from a place p to a transition t and inhibits the firing of t if the token load of p is not less than its multiplicity w. If $w > 1$, then, additionally, an ordinary arc from p to t with multiplicity less than w is allowed.

Petri nets with inhibitor arcs are intuitively the most direct approach to increasing the modeling power of Petri nets. All other extensions to standard Petri nets which are defined in this section are in fact equivalent to Petri nets with inhibitor arcs Peterson (1981).

Another extension of standard Petri nets are the so-called *self-modifying* Petri net introduced by R. Valk (1978). Basically self-modifying nets are standard Petri nets in which integers as well as places are associated with the arcs. Considering places as 'variables' and the marking of a place as its 'value', the multiplicity of an arc is then defined as the sum of the integers and the 'value' of the places associated with this particular arc.

Petri nets with *priorities* have been suggested by M. Hack (1975). Priorities can be associated with the transition such that if t and t' are both enabled, then the transition with the highest priority will fire first.

4 Properties of Petri Nets

Petri nets as a mathematical tool have many properties. These properties, interpreted in the context of the modeled system, enable the system designer to identify the presence or absence of specific features of the designed system. It is possible to distinct two types of properties: behavioral and structural. The first kind of properties depends on the initial marking of a Petri net, whereas the second one depends on the net structure of a Petri net.

4.1 Behavioral properties

Reachability. An important issue in designing concurrent systems is whether a system can reach a specific state, or exhibit a particular functional behavior. In a Petri net model, this should be reflected in the existence of specific sequences of transitions firings, which transform an initial marking M_0 to the required marking M'. The existence in the Petri net model of additional sequences of transitions firings transforming M_0 to M' indicates that the Petri net model may not be reflecting exactly the structure and behavior of the underlying system. This may also indicate the presence of unanticipated aspects of the functional behavior of the real system.

A marking M is said to be *reachable* from a marking M_0 if there exists a sequence of transitions firings which transforms a marking M_0 to M. The set of all possible markings reachable from M_0 is called the *reachability set*, and is denoted by $R(M_0)$.

Boundedness and safeness. Places in a Petri net are often used to represent buffers for storing intermediate data in a modeled system. By verifying that the net is bounded or safe, it is guaranteed that there will be no overflows in the buffers, no matter what firing sequence is taken.

A Petri net is said to be *k-bounded* or simply *bounded* if the number of tokens in each place does not exceed a finite number k for any marking reachable from the initial marking M_0. A Petri net is said to be *safe* if it is 1-bounded.

Stability. In real systems, the number of resources in use is typically restricted by different constraints, e.g. financial. If tokens are used to represent resources, the number of which in a system is typically fixed, then the number of tokens in a Petri net model of this system should remain unchanged irrespective of the marking the net takes on. This follows from the fact that resources are neither created nor destroyed. This requirement is closely related to the stability property of a Petri net.

A Petri net is said to be *stable* if total number of tokens in the net remains constant.

Liveness. Another problem which may arise in the design of a real system is deadlock. A deadlock in a system is a state in which no progress can be made.

A *deadlock* in a Petri net is a marking such that no transition is enabled. A transition t in a Petri net is *live* if for every marking M in $R(M_0)$, t is firable (enabled) at a certain marking M' reachable from M. A Petri net is *live* if every transition is live.

Reversibility. An important issue in the operation of real systems is the ability of these systems for an error recovery. These systems are required to return from the failure states to the preceding correct states. This requirement is closely related to the reversibility property of a Petri net.

A Petri net, for the initial marking M_0, is said to be *reversible* if the initial marking M_0 can be reached from each marking M in $R(M_0)$. Thus, in a reversible Petri net one can always get back to the initial marking.

4.2 Structural properties

For ordinary Petri nets several kinds of analysis methods are known. They can be formulated without considering the behavior of Petri net to deduce behavioral properties.

Let a Petri net be ordinary.

A Petri net is called: (1) a *marked graph* if each place has exactly one input transition and exactly one output transition, (2) a *state machine* if each transition has exactly one input place and exactly one output place.

A Petri net is said to be: (1) *free-choice* if each shared place is the only input place of its output transitions, (2) *extended-free-choice* if the output transitions of each shared place have the same input places, (3) *extended-simple* if it holds that if two places have at least one common output transition then the set of all output transitions of one of the places is a subset of the set of all output transitions of the other one, (4) *state machine coverable* if its set of places is a union of components which have, as subnets, the state machine property, where by a component of a Petri net we understand a set of places where each input transition is also an output transition and reversely, (5) *state machine decomposable* if its set of places is a union of components which, on their part, are strongly connected state machines.

A nonempty subset of places S in a Petri net is called a *siphon* if every transition having an output place in S has an input place in S. A nonempty subset of places Q in a Petri net is called a *trap* if every transition having an input place in Q has an output place in Q. A siphon (trap) is said to be *minimal* if it does not contain any other siphon (trap). A trap is said to be *maximal* if it is not included in any other trap.

A net has the *siphon-trap-property* if the maximal trap of each minimal siphon is marked.

5 Analysis Methods

Analysis by means of simulation. Simulation can be supported by a computer too or it can be totally manually. Simulation can reveal errors, but in practice never be sufficient to prove the correctness of a system. Simulation is often used in the design phases and the early investigation of a system design.

Analysis by means of reachability graphs. The basic idea behind reachability graphs is to construct a graph which contains a node for each reachable state and an arc for each possible change of state. A reachability graph can be used to prove properties of the modeled system. For bounded systems a large number of questions can be answered. Deadlocks, reachability and existence of marking bounds can be decided by a simple search through the nodes of the reachability graph, while liveness and home markings can be decided by constructing and inspecting the strongly connected components. The reachability graph method can be totally automated - and this means that the modeler can use the method, and interpret the results, without having much knowledge about the underlying mathematics. For more information see Karp, Miller (1969).

Analysis by means of invariants. Invariant analysis allows logical properties of Petri nets to be investigated in a formal way. There are two dual classes of invariants. A place invariant (P-invariant) characterizes the conservation of a weighted set of tokens, while a transition invariant (T-invariant) characterizes a set of transition sequences having no effect, i.e., with identical start and end markings. The main advantages of invariant analysis are the low computational complexity (in particular, compared to the method of reachability graphs). For more information see Memmi, Vautherin (1987).

Analysis by means of reductions. The basic idea behind this method is to modify a Petri net - without changing a selected set of properties, e.g. safeness, boundedness, and liveness. The modification of the net is performed by means of a set of transformations rules and may be carried out manually, automatically or interactively. In the latter case the strategy is decided by a person, while the detailed computations and checks are made by a computer. The purpose of the transformation is to obtain a small and simple net for which it is easy to investigate the given properties. For more information see Berthelot (1987).

Other analysis methods. For ordinary Petri nets several kinds of analysis methods are known. One of the methods uses structural properties, it means properties, which can be formulated without considering the behavior (i.e., transition sequences) of Petri net to deduce behavioral properties. For more information see Best (1987).

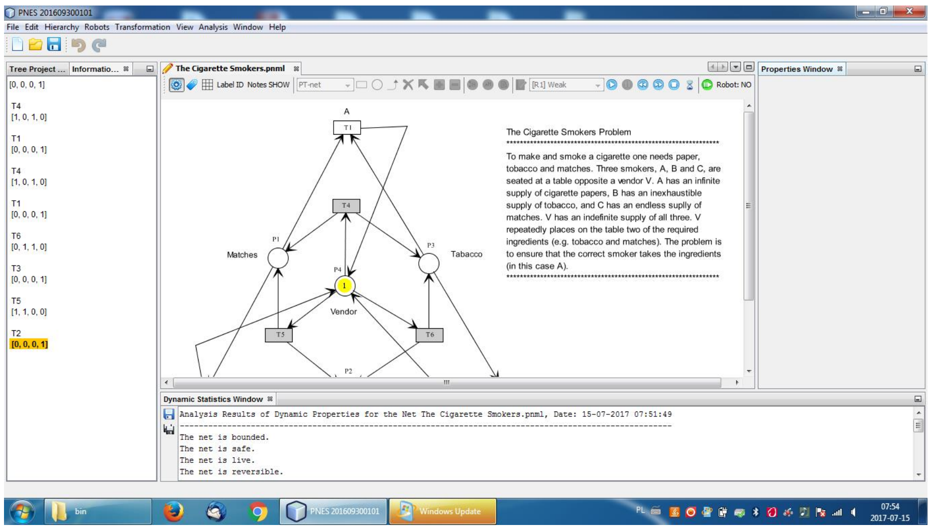
6 PNeS

6.1 General

PNeS is a fully integrated environment designed to aid engineers in modeling and solving concurrency related problems in parallel and distributed computing systems by using Petri net technology. This system supports the user in constructing of Petri net models as well as in modifying and analyzing. In particular, PNeS may be used to help the system designer in proving the correctness of his or her design.

PNeS consists of three main logical parts as follows: editor, simulator, and analyzer. Editor is a window-based graphical editor for entering, exiting, constructing, and editing Petri nets. Simulator is a program for graphical and textual simulation of a Petri net model. The results of simulation allow to detect design errors. Analyzer is a set of programs by means of which for standard Petri nets: (1) Basic structural and behavioral properties of Petri nets can be checked. The results of such analysis allow to detect syntactic (sometimes even semantic) design errors. For certain subclasses of standard Petri nets the structural properties can be used to deduce behavioral properties. (2) Simple reductions of the size of a net (and in a consequence of its reachability graph) preserving safeness, boundedness, and liveness can be executed. The use of such reduction methods is necessary if the storage capacity of the given computer system is not sufficient to analyze a given net. (3) Calculation of the invariants and other structural information (e.g. state machine components) from a given net which reflect certain structural properties of the modeled system are possible. Invariant analysis can be done by computing generator sets of all P-/T-invariants and of all nonnegative invariants. Additionally, vectors can be tested for invariant properties.

PNeS offers also an user-friendly Windows GUI environment, see Fig. 1.



```
[0, 0, 0, 1]

T4
[1, 0, 1, 0]

T1
[0, 0, 0, 1]

T4
[1, 0, 1, 0]

T1
[0, 0, 0, 1]

T6
[0, 1, 1, 0]

T3
[0, 0, 0, 1]

T5
[1, 1, 0, 0]

T2
[0, 0, 0, 1]
```

The Cigarette Smokers Problem
****
To make and smoke a cigarette one needs paper, tobacco and matches. Three smokers, A, B and C, are seated at a table opposite a vendor V. A has an infinite supply of cigarette papers, B has an inexhaustible supply of tobacco, and C has an endless suplly of matches. V has an indefinite supply of all three. V repeatedly places on the table two of the required ingredients (e.g. tobacco and matches). The problem is to ensure that the correct smoker takes the ingredients (in this case A).
****

```
Analysis Results of Dynamic Properties for the Net The Cigarette Smokers.pnml, Date: 15-07-2017 07:51:49
------------------------------------------------------------------------------------------------------
The net is bounded.
The net is safe.
The net is live.
The net is reversible.
```

Fig. 1. A graphical user interface of PNeS.

6.2 Behavior analyzers

The proof of the correctness of a system specification in a Petri net based language usually is done in two substeps. First, the underlying net is analyzed with respect to basic dynamic and structural properties (see section 4) are tested. Next, using the results of such an analysis the correctness of the complete model is verified. It appears, that it is rather difficult, if not impossible, to do this in a systematic way. But, the more information has been collected and the more design errors have been detected during the analysis and simulation of the underlying Petri net, the easier the second substep is to carry out.

Structure checking and liveness. For testing structural properties from which, in case of ordinary Petri nets, liveness properties follow, the module "Structural Properties..." (available in the option "Analysis" of the Menubar), is used. If one is given an unknown net, using the module "Basic Properties" one can obtain information on elementary net properties such as: the number of places (transitions), the minimal (maximal) number of the net nodes, as well as the maximal entrance (exit) degree of the net nodes. One can check basic structural properties. Additionally, it is tested, whether the read-in net considered as an undirected graph, is connected. If a net is ordinary one can check whether it is a state machine, a free-choice net, an extended free-choice, or an extended simple net, using the "Structural Properties..." module invoked in the "Analysis" option. These properties are related with the liveness via the siphon-trap-property. Then the minimal siphons are computed. If there exists a clean (not marked) siphon, the net is not live. Next, the possible conclusion connected with the liveness of a net follows: (1) If a given net is an extended simple net and the siphon-trap-property holds, then a net is live. (2) If a given net is an extended free-choice and the siphon-trap-property does not hold, then a net is not live. (3) If for a given net the siphon-trap-property holds, then no dead marking can be reached.

This module allows also to check whether the net is a state machine decomposable (which implies that it is bounded under any initial marking). In addition, this module decides whether a given net is a state machine coverable, i.e., coverable by components which are state machines.

Reachability graph analysis. In the module described above possible conclusions on the behavior properties are inferred from the structural properties on the basis of an initial marking. If there are no information for such conclusions, we have to investigate the reachability graph. The module "Coverability/Reachability Graphs..." (available in the option "Analysis" of the Menubar) computes the coverability tree/graph Karp, Miller (1969), which is identical with the reachability tree/graph in the case of a bounded net. If a given net is bounded, first, this module builds the reachability tree/graph, then it checks whether the net is bounded, safe, live, reversible, stable, and deadlock-free. This module also lists the dead/live transitions at an initial marking as well as dead/live markings. Then, the module checks at each reachable marking at which several transitions have concession whether one of these transitions takes the concession of another upon firing. The obtained dynamic conflicts are also written out. Moreover, the module creates the graphical representation of reachability tree/graph and indicates nodes in the tree/graph which are dead, duplicated. If a given net is unbounded, first, this module builds the coverability tree/graph, then it writes out that the net is unsafe, unbounded, and nonstable. Moreover, unbounded places are output.

Reductions. The module "Reductions..." can be used to reduce the size of the given net, so that it becomes analyzable by other modules from the option "Analysis" of the Menubar on the one hand, and it can be used to find an equivalent small net with known properties, i.e., boundedness, safeness, liveness. In this module has been implemented the most essential local reductions steps known from the literature, e.g. merging of nodes which share all predecessors and successors, fusion of equivalent places, reduction of different kinds of place/transition chains.

Computation of invariants. A basis for the set of all place (transition) invariants is computed by the module "Invariants" from the option "Analysis" of the Menubar. First, this module computes an (transposed) incidence matrix for a given Petri net as a basis for computation of all (nonnegative) place (transition) invariants Reisig (1985). Next, from this it derives information on boundedness and reversability properties. Moreover, in general, invariants have an interpretation in terms of the modeled system which can be useful for its verification. It is worth to notice that several conclusions derived from invariants are related only to nonnegative invariants (e.g. net coverability by P-invariants). This module computes a basis for the set of nonnegative place (transition) invariants. Correspondingly, the set of all nonnegative invariants is the set of vectors that can be generated from the computed set by means of linear combinations with nonnegative coefficients.

Tests. The reachability/coverability problems can be solved in dialogue using the module "Tests". For this, first, one should use the option "Reachability/ Coverability". Then, the respective marking should be input in the way shown on the screen. The module checks whether this marking is reachable/coverable and announces the result. This module permits also to check the properties of place and transition vectors, respectively. It shows whether the vector investigated is an invariant.

7 Conclusion and Further Work

We have recalled the backgrounds of Petri nets and have presented PNeS. By using PNeS, researchers and practitioners interested in the theory and application of Petri nets obtain a method and a tool with which main design aspects of modeled systems can be analyzed quickly and correctly, as well described in a presentable way. Analysts, system designers and everyone who, in the framework of project developments, has to describe coherently and vividly complex procedures of system engineering on the basis of a theoretical method, are able to carry out their task in a more economical and time-saving way.

Although in the last three decades, a large number of tools have been reported in the Petri net literature, a majority of these tools are used mostly for research and educational purposes. From our observations, in most cases, existing tools concern a concrete class of Petri nets or a particular method for designing and/or analyzing of Petri net model. In a case presented here, the situation is different. We describe here an integrated and modular PNeS which possess features universality in this sense that it can be used for several classes of Petri nets and makes accessible various methods of the designing and analysis of Petri net models. We plan to extend the functional possibilities of the existing PNeS about additional Petri net classes and new analysis techniques. Taking into account the need for representing approximate and uncertain information, as well as the need for the qualitative specification of the industrial control, we plan the development and implementation of various types of fuzzy Petri nets. Fuzzy Petri nets are useful, among others, for knowledge representation and approximate reasoning in decision support systems Looney (1988), Chen et al. (1990), Pedrycz, Gomide (1994), Suraj (2013), Suraj, Bandyopadhyay (2016).

Standard Petri nets are not always sufficient to represent and analyze complex real systems. Tokens in standard Petri nets have no identity. This causes some problems for modeling real systems such as manufacturing and communication systems, which require the physical resources or messages, if represented by tokens, to have identity. Without this identity, it is impossible to trace the flow of different resources or messages in the system. In order to address this issue, we are going to add to our system the module serving colored Petri nets Jensen, Rozenberg (1991). In these nets, a token can be a compound object carrying data. This data can be of arbitrary complexity involving integers, reals, text strings, records, lists and tuples.

Another reason why the use of Petri nets is largely confined to academic and research institution is the difficulty involved in constructing Petri nets models. Constructing net models of systems, especially large scale systems, is not a trivial task. It requires a great deal of experience. No methodology is available yet, which would allow for a fully automatic construction of Petri nets models. In our opinion, in most cases, Petri net models are constructed in an ad hoc manner. In the past four decades, many approaches to the systematic construction of Petri net models have been proposed, and the work in this area still continues. These approaches, using the terms of software engineering, can be classified into bottom-up, top-down, and hybrid approaches. A comprehensive discussion of these approaches can be found in Jeng, DiCesare (1993). It is worth to add that we plan to extend our PNeS about the bottom-up and top-down methods for constructing standard Petri net models.

In the past decade, a number of approaches have been reported which allow for the automatic construction of restricted classes of Petri net models from requirements specifications expressed using production rules, flow diagrams, state machines, temporal logic, etc. ROSECON system Pancerz, Suraj (2004), which can cooperate with PNeS, allows to construct colored Petri net models from the specification presented in the form of Pawlak's information systems using rough set methods Pawlak (1982). We plan also the further research in this direction.

Acknowledgment. This work was partially supported by the Center for Innovation and Transfer of Natural Sciences and Engineering Knowledge at the University of Rzeszów. The authors are grateful to the anonymous referees for critical remarks and useful suggestions to improve the presentation of the paper.

References

1. Berthelot G. (1987) Transformations and Decompositions of Nets, *Lecture Notes in Comput. Sci.*, vol. 254, Springer, Berlin, 359-376.
2. Best E. (1987) Structure theory of Petri nets: the free choice hiatus, *Lecture Notes in Comput. Sci.*, vol. 254, 168-205, Springer, Berlin.
3. Chen S.M., Ke J.S., Chang J.F. (1990): Knowledge representation using fuzzy Petri nets, *IEEE Trans. on Knowledge and Data Engineering* 2(3), Sept. 1990, 311-319.
4. David R., Alla H. (1992) *Petri nets and Grafcet: Tools for modelling discrete event systems.* Prentice Hall, New York.
5. Desel J., Reisig W., Rozenberg G. (2004) *Lectures on Concurrency and Petri Nets.* Springer, Berlin.

6. DiCesare F., Harhalakis G., Proth J.M., Silva M., Vernadat F.B. (1993) *Practice of Petri nets in Manufacturing.* Chapman and Hall, New York.
7. Feldbrugge F., Jensen K. (1991) Computer tools for High-level Petri nets. In: *High-level Petri Nets. Theory and Application*, Springer, Berlin, 691-717.
8. Hack M. (1975) *Decidability questions for Petri nets.* PhD dissertation, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge.
9. Jeng M.D., DiCesare F. (1993) A review of synthesis techniques for Petri nets with applications to automated manufacturing systems, *IEEE Trans. Syst., Man, and Cybern.* 23(1), 301-312.
10. Jensen K., Rozenberg G. (eds.) (1991) *High-level Petri Nets. Theory and Application.* Springer, Berlin.
11. Karp R.M., Miller R.E. (1969) Parallel program schemata, *J. Comput. & Syst. Sci.* 3, 147-195.
12. Looney C.G. (1988) Fuzzy Petri Nets for Rule-Based Decision-making, *IEEE Trans. Syst., Man, Cybern.* 18-1, 178-183.
13. Memmi G., Vautherin J. (1987) Analysing nets by the invariant method, *Lecture Notes in Comput. Sci.*, vol. 254, Springer, Berlin, 300-336.
14. Murata T. (1989) Petri nets: Properties, Analysis and Applications, *Proc. of the IEEE*, vol. 77, no. 4, 541-580.
15. Pancerz K., Suraj Z. (2004) Discovering Concurrent Models from Data Tables with the ROSECON System, *Fundam. Informat.* 60(1-4), 251-268.
16. Pawlak Z. (1982) Rough sets, *Int. J. Comput. & Informat. Sci.* 11, 341-356.
17. Pedrycz W., Gomide F. (1994)A generalized fuzzy Petri net model, *IEEE Trans. on Fuzzy Systems* 2-4, 295-301.
18. Peterson J.L. (1981) *Petri Net Theory and the Modeling of Systems,* Prentice-Hall, Inc., Englewood Cliffs, N.J.
19. Petri C.A. (1966) *Kommunikation mit Automaten.* Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr.3, 1962.
20. Reisig W. (1985) *Petri Nets,* Springer Publ. Company.
21. Suraj Z. (1995) PN-tools: environment for the design and analysis of Petri Nets, *Control and Cybernetics* 24(2), 199-222.
22. Suraj, Z. (2013) A new class of fuzzy Petri nets for knowledge representation and reasoning, *Fundam. Informat.* 128(1-2), 193-207.
23. Suraj Z, Bandyopadhyay S. (2016) Generalized Weighted Fuzzy Petri Net in Intuitionistic Fuzzy Environment, *Proc. of the IEEE World Congress on Computational Intelligence*, 25-29 July, 2016, Vancouver, Canada, 2385-2392.
24. The Petri Net Markup Language (2017). Available: http://www.pnml.org
25. The Petri Nets World (2017). Available: http://www.informatik.uni-hamburg.de/TGI/PetriNets
26. Valk R. (1978) Self-modifying Nets: a Natural Extension of Petri Nets, *Lecture Notes in Comput. Sci.* vol. 62, Springer, Berlin, 464-476.
27. Zeugmann-Popova L. (2013) *Time and Petri Nets,* Springer, Berlin.
28. Zurawski R., Zhou MengChu (1994) Petri Nets and Industrial Applications: A Tutorial, *IEEE: Transactions on Industrial Electronics* 41(6), 567-583.