Using the Evaluation Nets Modeling Tool Concept as an Enhancement of the Petri Net Tool

Michał Niedźwiecki, Krzysztof Cetnarowicz
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Krakow, Poland
Email: {nkg, cetnar}@agh.edu.pl

Krzysztof Rzecki
Cracow University of Technology
ul. Warszawska 24, 31-155 Krakow, Poland
Email: {krz}@iti.pk.edu.pl

***Abstract*—Petri net modeling has well-known algorithms, so it is easy to develop computer tools to build, edit, and analyse these networks. These tools are designed to be able to add extensions giving additional functionality, such as an extension for evaluation networks.**

Evaluation networks are not as popular as Petri Net modeling, but they turns out that, in modeling of some of the problems, evaluation networks makes analysis of them very clear and intuitive. Unfortunately there are no mathematical tools and computer programmes for evaluation networks use. Fortunately, under certain assumptions, an evaluation network can be converted into a Petri net. This article presents an idea of how to convert an existing Petri net computer programme to draw evaluation nets and convert them into Petri nets in order to use existing tools for Petri net analysis.

Evaluation nets are well suited for modeling negotiation protocols between two parties represented by servers or software agents. This article provides an example of such a protocol presented in three versions: a sequence diagram UML, Petri net and Evaluation nets.

I. Introduction

PETRI net is one of the few languages for modeling distributed systems. It was invented in the 1960s by the German mathematician Carl Adam Petri, and described in his doctoral dissertation [1]. Petri nets are used to model concurrent systems [2], the discrete [3] synchronisation process [4], etc. They are used in computer science, and in other fields such as biology, medicine and chemistry [5].

Modeling Petri nets may be performed by a number of computer programmes, such as WoPeD [6], YASPER [7], CPN Tools [8] or PIPE2 [9]. They allow for graphical editing, interactive visualisation, and analysis of automatic Petri net.

A new areas of Petri net application arose, so various extensions were created. As a result, there are coloured Petri nets [10], timed Petri nets [11], stochastic Petri nets [12], etc.

Among these extensions are evaluation nets (*E*-Nets), the first use of which was associated with an analysis of the operation of information systems. *E*-Nets were invented by Garry J. Nutt and Jerre D. Noe, and described in Nutt's dissertation [13]. *E*-Nets are very poorly distributed. Teaching materials for the *E*-Nets are not widely available, and nor are computer programms for it. Unfortunately, *E*-Nets were not developed particularly dynamically for many years.

E-Nets are well-suited for modeling communication between systems [14]. However, the lack of programmes that work with *E*-Nets makes use of them very difficult, and the implementation of such programmes from scratch increases the cost of the study.

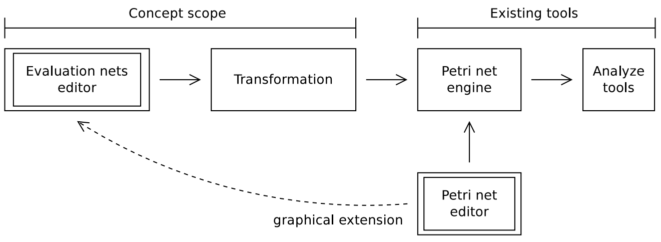

Fig. 1. Petri net editor is extended to support elements of evaluation nets. Then, when you call the analyser, evaluation nets will have been transformed to Petri net and as such referred to the process analyser.

The main motivations for writing an article on this theme are:

1) Evaluation networks are useful for designing algorithms for negotiations, such as between agents, servers, etc., yet their universal notation is very readable for humans.
2) The project of an algorithm may be verified by including the performance simulation of the evaluation nets. But there is no such network simulator.
3) Some elements of *E*-Nets can easily be presented in Petri net, so existing Petri net simulators can be used.

Our concept uses an opportunity to convert *E*-Nets (some restrictions were imposed) to Petri nets, and uses existing Petri net analysers for analysis by a visual extension of an existing Petri net editor, which draws evaluation nets on the screen and sends the ready Petri net to the analyser (Fig. 1).

The terms *E*-Nets and evaluation nets usually are used interchangeably, but for the record, in the rest of this article the names will be applied differently. The term "*E*-Nets" will be used for the original evaluation nets [15] and the variation created for this concept will be referred to as "evaluation nets."

The formalism of Petri Net were well described in [16] i [17]. That concept is based on those descriptions of Petri Net and it's extensions (Table I).

II. Evaluation nets

An *E*-net is a connected set of locations over the set of allowable transition schema. The formalism of original *E*-nets was described in [18] and [15].

In oryginal *E*-nets location may be connected with only one input and one output transition. Location in *E*-Nets can have no token (empty) or one token (full). For example in `F-TRANSITION` (Fig. 6) firing is not possible when the c location is full.

Since their creation, *E*-Nets have evolved. Some extensions were presented in [19]. These extensions were used in [14].

A lot of *E*-Nets structures can be transformed to Petri nets with coloured tokens [10] and an inhibitor arc [20] extension.

Token can have attributes. Attributes are in *Resolution procedure* and *transition procedure*. If attributes determine *E*-Nets, then conversion to Petri nets may be impossible. Therefore, for the simulation, we have to use external procedures to process the data in the token's attributes and to decide about the firing transition.

Unfortunately the usage of many external procedures in the cases of resolution locations and transition procedures seems to make the proposed model complex. The Petri net tools for analysis are not able to solve or make easiest the usage and the efficiency of proposed model. However, in some cases, it's enough to replace those procedures by simple conflict structures.

If we want to create an automatic converter from *E*-Nets to Petri nets, then we must impose additional restrictions on *E*-Nets. In this article we describe our variation of *E*-Nets. Some extensions are graphical only and these improve readability, but some extensions change the *E*-Net flow:

1) multi-connections between locations and transitions were Enabled, but this may produce conflict structures (similarly to with Petri nets),
2) if a conflict structure contains resolution locations then the order of firing is determined by the execution of these *resolution procedures*,
3) coloured token — a token with enumerated value (for example: colour) as an attribute,
4) transition schema — may use token colour instead of 0 and 1 values (something like switch-cache transition),
5) resolution location — can return a coloured token, return value cannot be determined from other tokens but can be randomised,
6) outer request location — a special kind of resolution location, represents request coming from outside the evaluation net (indicated by a rectangle with a double line),
7) transition with sender — transitions which are able to send tokens (represented by a horizontal arrow labeled with the corresponding message name), always after firing.

All these differences between evaluation nets and Petri nets, our extensions and transitions are described in Table I.

III. Transforming evaluation nets into Petri nets

In [18], there were described 5 primitives. Primitives are basic (trivial) constructions of evaluation nets (Fig. 6). All more complex evaluation nets are built by joining and extending of those primitives. *E*-Nets primitives may be replaced by their equivalents from Petri net (Fig. 7). Thus, we can acquire an evaluation net converted into a Petri net.

All original *E*-Nets primitives from Fig. 6 can be transformed with inhibitor arcs, colour tokens and conflict structures determined by e.g. controlled transitions (Fig. 7). `T-`, `F-` and `J-TRANSITION` may be transformed without any extensions but additional places, transitions and tokens are required.

Transformation of `X-` or `Y-TRANSITION` must be made by a lot of additional elements because we must emulate complex *transition schema* and all possible states of *resolution location*. With coloured Petri nets the number of additional elements may be reduced to 3 and 7 additional transitions for `X-` and `Y-TRANSITION`. Also added a lot of new connections. (Table II).

Transformation of a triangular location is trivial (use standard connection). The transformation of transition with sender is more difficult (Fig. 5).

Other transformation issues are described in Table I.

Following such transformation, Evaluation Nets can be transformed to Petri net analysis (Fig. 1).

A. Example

It turns out that evaluation networks are well-suited for modeling protocol negotiations between the two parties, such as that shown in Fig. 2. This protocol is based on the CNP [26] and is part of Complex Negotiations [27]. The specified protocol defines two participants: initiator and participant. If the initiator wants to start negotiations, it sends a message to the participant CFP (call for proposal). If the participant is interested in negotiations, the initiator receives the offer. Otherwise, the initiator sends the message *refuse* to end the negotiations as a failure. When Initiator is offered, decide whether a) accept the offer (accept), thus ending the negotiations successfully, b) reject the offer and end the negotiations a failure or c) rejects the offer but made a counteroffer and waits for a response from the participant.

Thanks to evaluation networks it is possible to design a computer system using the previously described negotiation protocol. Fig. 4 shows an evaluation net divided into two subnets, one belonging to the initiator, the second to the participant. Subnets do not have direct connections with each other (using arcs), but they exchange messages.

Initially, both participants are in a state of inactivity (`Idle`). Then the initiator is ordered to enter into negotiations, and as a result a token appears in l_1. This causes firing of a_1, which in addition to placing a token in l_4 sends a message called `CFP` (call for proposal). Initiator goes to l_4 (`Wait`) and waits for a response $\mathtt{S}_i$ from the participant. If the answer does not appear within the required time, the wait can be interrupted by a request l_3 (`Timer`). This is support for an emergency situation, which ends the Initiator net evaluation of the state of l_8 (`fail`).

In the meantime, the message `CFP` is received by l_{10} (located in the participant) which results in the appearance of the token and firing of a_5. Then the participant's evaluation

TABLE I
THE DIFFERENCES AND TRANSFORMATION METHOD BETWEEN EVALUATION NETS AND PETRI NET

Evaluation nets item	**Petri net equivalent**	**Original *E*-Nets limitations**	**Limitations of our evaluation nets variation**	**Original Petri net limitations**	**Useful Petri net variations**	**Transformation**
token	token	simple token or attributed token	first attribute is reserved for colour	simple token	colored token [10], object token [21]	When we use only coloured attribute value then we can use coloured token. Otherwise we must use object token.
location connections	place connections	connected to at least one transition	similarly to Petri net	no limitations	-	No conflicts.
token in location	token in place	no tokens (empty) or one single token (full) or one attributed token (full)	no tokens (empty) or one coloured token (full) with other attributes or not	no limitations of token count	inhibitor arcs [20], capacities of places [22]	Use inhibitor arcs of places.
transition schema	enabling rules, firing rules	schema values in tuple: 0 or 1 or "*e*" (only in right hand side of a schema), enabled transition cannot be disabled, it must fired (no conflicts)	may be use colour value instead to 0 or 1, conflict resolving determined by execution finish of resolution procedure or randomise function, transition can be disabled	input places must contains one or more token, output places not constrained enabling rules	inhibitor arcs [20], reset arcs [23], capacities of places [22]	Can be emulated by additional transitions, inhibitor arcs and loop connections (connect place with transition and transition with place). In output connections we can use capacities of places or inhibitor arcs. Reset arc may be used to reduce elements and improve readability of net.
transition procedure	-	-	not use	-	object token [21]	We can use outer procedures but existing Petri nets analysers may be unusable. However we can use tools for object tokens [8].
resolution location	conflict structure	returns token (single or attrubuted) or not	may return coloured token, return value cannot be dependent from previous events, returned value may be randomised or coming from outside of the evaluation net	firing is not deterministic	for decision of firing: Controlled Petri nets [24], Timed Petri nets [11], Stochastic Petri nets [12], Labeled Petri nets [25], etc.	Resolution location has resolution procedure. Resolution procedure running in enabling phase and result of it determine decision of firing. Automatic transformation to Petri nets without using external procedures is very difficult. For verifying the use cases we can use Timed Petri nets. For time benchmarking we can use Stochastic Petri nets.
triangular location [15]	connection	two interlinked triangular locations replaces long unreadable connection	-	-	-	It is only graphical notation. We can make extension for Petri net editor for supporting invisible connection and corresponding label.
transition with sender [14]	connection	-	it is sender triangular location and transition in one and when it fired then it sends coloured token from corresponding input location	-	-	Similarly to "triangular location." This item has been introduced for the improve readability of evaluation nets. Token was sent without depending of *transition schema*.

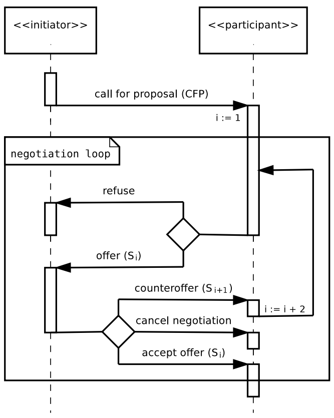

Fig. 2. Negotiation protocol in sequence diagram.

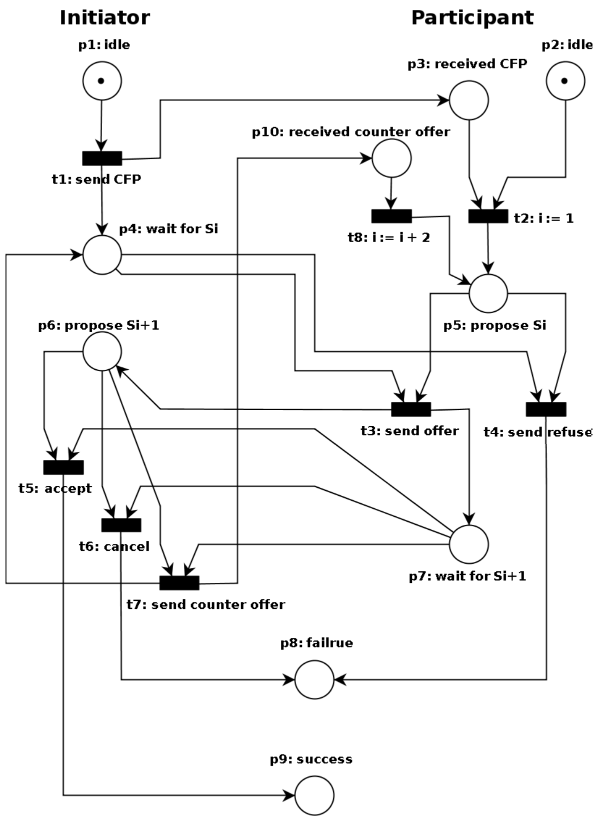

Fig. 3. Negotiation protocol in Petri net.

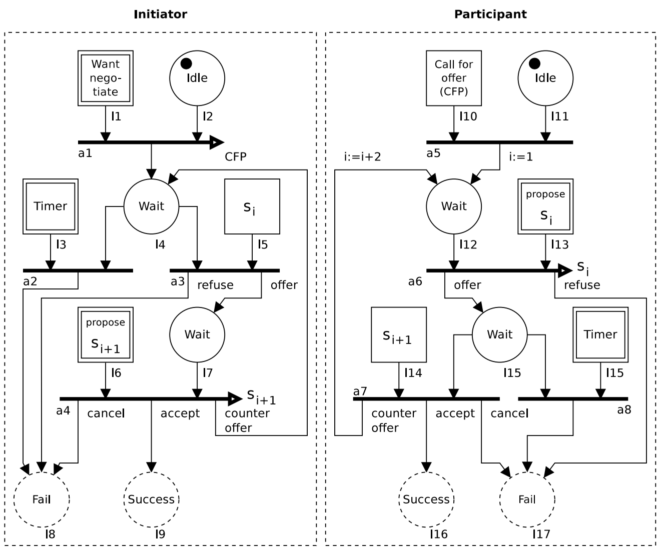

Fig. 4. Negotiation protocol in evaluation nets (our variation). S_i token can have *offer* or *refuse* enumeration value. S_{i+1} token can have *cancel*, *accept* and *counter offer* enumeration value. $i := 1$ and $i := i + 2$ have no effect on the evaluation net flow.

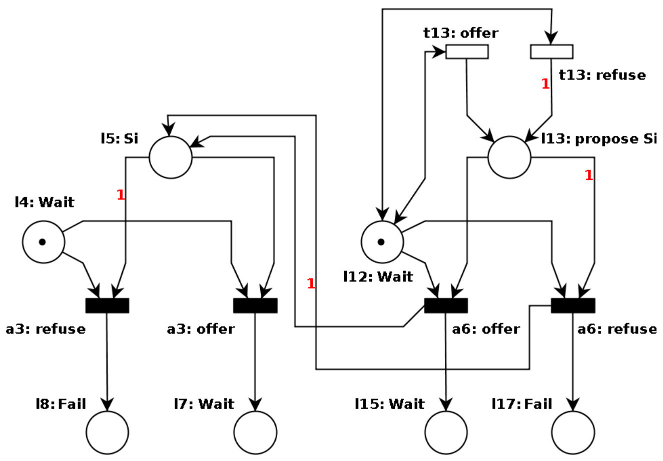

Fig. 5. Equivalent of l_4, l_5, l_7, l_8, l_{12}, l_{13}, l_{16}, l_{17}, a_3 and a_6 from Fig. 4 in coloured Petri nets. Inhibitor arcs are skipped.

net goes to l_{12} (`Wait`) and expects him to make a decision (request l_{13}). As a result of the decision, a coloured token appears in l_{13} and the colour represents the `offer` (submission of proposals initiator) or `refuse` (breaking off of negotiations). The coloured token is processed by a_6, which decides whether to proceed according to evaluation net state l_{15} (`Wait`) or l_{17} (`fail`). Then the token is placed in the message `S`$_i$. The participant is in state l_{15} and is expecting a message from the initiator to be received by l_{14}. If the message does not appear

TABLE II
NUMBER OF OLD ELEMENTS/ADDITIONAL ELEMENTS BY CONVERTER

Primitive	Places	Transitions		Connections		Colo-urs
		Orig.	Lab.	Orig.	Inh.	
T-TR. Fig. 7 (a)	2/0	1/0	0	1/0	1	1/0
T-TR. Fig. 7 (b)	2/1	1/0	-	1/0	-	-
F-TRANSITION	3/0	1/0	0	3/0	2	1/0
J-TRANSITION	3/0	1/0	0	3/0	1	1/0
X-TRANSITION	4/0	1/1	2	4/6	4	1/1
T-TRANSITION	4/0	1/3	4	4/16	8	1/1
Switch: Fig. 5	8/0	2/2	2	8/12	-	2/0

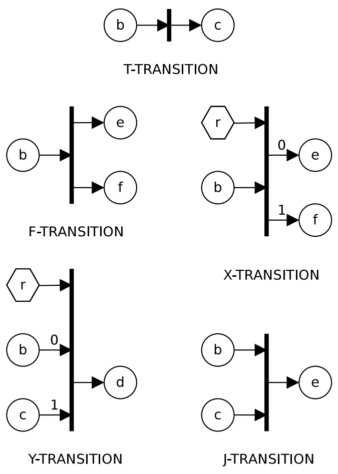

Fig. 6. Primitive E-Nets from [18].

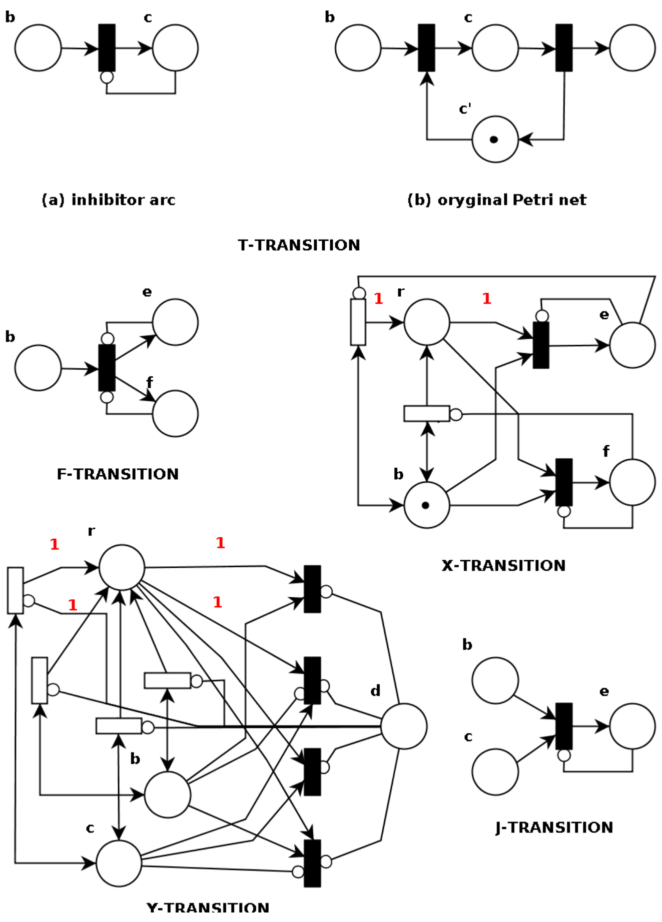

Fig. 7. E-Nets primitives from Fig. 6 transformed to Petri nets with extension: coloured, controlled and inhibitor arc. Additionally `T-TRANSITION` was transformed in two versions: (a) Petri net with extensions and (b) original Petri nets (b).

within a given time, the emergency procedure similar to that associated with l_3 will start in the initiator.

After receiving the message, initiator $\mathtt{S}_i$ is processed by a_3, which acts in a similar way to a_6. If the message contains the `refuse` attribute, the network is referred to the state of l_8 (`fail`), but when the message contains the attribute `offer`, the evaluation net goes into standby l_7 to make a decision by the initiator (decision will appear in the l_6).

Next, the attributed token is processed by a_4 and by a_7, which receives it through error $\mathtt{S}_{i+1}$ passed through a_4 and l_{14}.

We have tried to show the same example in Figure 3 using a controlled Petri net. This chart was prepared by hand. We tried to maintain maximum readability and so have omitted the emergency 'time out' in waiting for a message from the other side. We also abandoned the isolated, separate subnet Petri net for initiator and participant (such as in the evaluation net) to increase the number of transitions and arcs.

However, we provided standby. Cost reduction readability in this case was small because the whole Petri net is simple. Attributed tokens and resolution locations could be replaced with controlled transitions. In the former case, the initiator and participant received from their evaluation nets a request for attributed tokens. In this case, the initiator and participant will request a specific, enabled fired controlled transition.

Thus, this example shows that the use of an evaluation net instead of a Petri net makes sense visually. However, the lack of tools for analysis means that the evaluation drawings of the usefulness of the net are slightly larger than usual. This condition can be adjusted by using the described concept, which, however, entails the generation of a significant amount of additional elements. Only a portion of the changes resulting from the transformation in Fig. 5 is shown in Fig. 4.

IV. Conclusion

Evaluation nets are useful for some communication protocols e.g. the negotiation protocol from Fig. 2. However, they are not so popular, and tools for making and analysing them do not exist. Fortunately, evaluation nets with some restrictions can be transformed into Petri net.

The Petri net has many tools including mathematical tools, analysers and graphic editors. It is possible to extend the existing editor to support evaluation nets, transform them into Petri net and then use Petri net analysis tools.

Moreover, it is impossible to transform all things. The *transition procedure* and *resolution procedure* are programming instructions that operate on the attributes of tokens.

For this reason, we must either abandon them and possibly replace the *resolution procedure* function that returns a random result, or take the code of these procedures outside the Petri net and give them control over the firing transitions. In this article, we focused on the first solution. In the future, we want to focus on the other. There is a third solution, which is based on Petri net generating source code [28]. Maybe then it would be able to make a converter that is 100% compatible with the evaluation nets.

Another problem is that transformation adds a lot of new elements. Additional elements may require some modifications to analysers, but this exists only for the analyser and not for the user. For example, without a special filter on the analyser result, we saw our elements and new elements added by the converter. Without a filter our elements may disappear among the others.

Therefore, future work should focus on adjusting existing analysers. There are plans to write an interpreter, that will be able to perform net evaluation in a way similar to the BPEL interpreter.

Acknowledgements

The research leading to these results has received funding from the dean grant no. 15.11.230.082, AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications. Author Michał Niedźwiecki would like to thank EFS of POKL 4.1.1 (POKL.04.01.01-00-367/08-00), European Union programme for support.

References

[1] C. A. Petri, "Kommunikation mit automaten," Ph.D. dissertation, Universität Hamburg, 1962.

[2] F. Ahmad, H. Huang, and X. Wang, "Analysis of the petri net model of parallel manufacturing processes with shared resources," *Information Sciences*, vol. 181, no. 23, pp. 5249 – 5266, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025511003665

[3] I. Rivera-Rangel, A. Ramírez-Treviño, and E. López-Mellado, "Building reduced petri net models of discrete manufacturing systems," *Mathematical and Computer Modelling*, vol. 41, no. 8–9, pp. 923 – 937, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895717705001548

[4] A. Merabet, "Synchronization of operations in a flexible manufacturing cell: The petri net approach," *Journal of Manufacturing Systems*, vol. 5, no. 3, pp. 161 – 169, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0278612586900439

[5] J. Will and M. Heiner, *Petri Nets in Biology, Chemistry, and Medicine: Bibliography*, ser. Computer science reports. Inst. of Computer Science, 2002. [Online]. Available: http://books.google.pl/books?id=6DB8GwAACAAJ

[6] A. Eckleder and T. Freytag, "Woped – a tool for teaching, analyzing and visualizing workflow nets," *Petri Net Newsletter*, vol. 75, Dec. 2008. [Online]. Available: http://www.informatik.uni-augsburg.de/pnnl/PDFs/PNNL75_WWW.pdf

[7] K. van Hee, O. Oanea, R. Post, L. Somers, and J. M. v. an der Werf, "Yasper: a tool for workflow modeling and analysis," in *Proceedings of the Sixth International Conference on Application of Concurrency to System Design*, ser. ACSD '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 279–282. [Online]. Available: http://dx.doi.org/10.1109/ACSD.2006.37

[8] M. Westergaard and L. Kristensen, "The access/cpn framework: A tool for interacting with the cpn tools simulator," in *Applications and Theory of Petri Nets*, ser. Lecture Notes in Computer Science, G. Franceschinis and K. Wolf, Eds. Springer Berlin Heidelberg, 2009, vol. 5606, pp. 313–322. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02424-5_19

[9] N. J. Dingle, W. J. Knottenbelt, and T. Suto, "Pipe2: a tool for the performance evaluation of generalised stochastic petri nets," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 34–39, Mar. 2009. [Online]. Available: http://doi.acm.org/10.1145/1530873.1530881

[10] K. Jensen, "A brief introduction to coloured petri nets," in *Proceedings of the Third International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, ser. TACAS '97. London, UK, UK: Springer-Verlag, 1997, pp. 203–208. [Online]. Available: http://dl.acm.org/citation.cfm?id=646481.691443

[11] C. Ramchandani, "Analysis of asynchronous concurrent systems by timed petri nets," Cambridge, MA, USA, Tech. Rep., 1974.

[12] G. Balbo, "Introduction to stochastic petri nets," in *Lectures on Formal Methods and PerformanceAnalysis*, ser. Lecture Notes in Computer Science, E. Brinksma, H. Hermanns, and J.-P. Katoen, Eds. Springer Berlin Heidelberg, 2001, vol. 2090, pp. 84–155. [Online]. Available: http://dx.doi.org/10.1007/3-540-44667-2_3

[13] G. J. Nutt, "The formulation and application of evaluation nets," Ph.D. dissertation, University of Washington, 1972.

[14] M. Żabińska and K. Cetnarowicz, "Application of m-agent multi-profile architecture and evaluation nets to negotiation algorithm design," in *ISSPIT' 2002 : the 2nd IEEE International Symposium on Signal Processing and Information Technology : December 18–21*, K. Y. M. Amin, Ed., 2002, pp. 214–219.

[15] J. D. Noe and G. Nutt, "Macro e-nets for representation of parallel systems," *Computers, IEEE Transactions on*, vol. C-22, no. 8, pp. 718–727, 1973.

[16] G. Rozenberg and J. Engelfriet, "Elementary net systems," in *Lectures on Petri Nets I: Basic Models*, ser. Lecture Notes in Computer Science, W. Reisig and G. Rozenberg, Eds. Springer Berlin Heidelberg, 1998, vol. 1491, pp. 12–121. [Online]. Available: http://dx.doi.org/10.1007/3-540-65306-6_14

[17] R. Zurawski and M. Zhou, "Petri nets and industrial applications: A tutorial," *Industrial Electronics, IEEE Transactions on*, vol. 41, no. 6, pp. 567–583, 1994.

[18] G. J. Nutt, "Evaluation nets for computer system performance analysis," in *Proceedings of the December 5-7, 1972, fall joint computer conference, part I*, ser. AFIPS '72 (Fall, part I). New York, NY, USA: ACM, 1972, pp. 279–286. [Online]. Available: http://doi.acm.org/10.1145/1479992.1480030

[19] F. André and C. (Grup), *Distributed Computing Systems: Communication, Cooperation, Consistency*. Elsevier Science Pub., 1985. [Online]. Available: http://books.google.pl/books?id=dH8EAQAAIAAJ

[20] K. Reinhardt, "Reachability in petri nets with inhibitor arcs," *Electron. Notes Theor. Comput. Sci.*, vol. 223, pp. 239–264, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2008.12.042

[21] M. Köhler, *Object Petri Nets: Definitions, Properties, and Related Models*. Univ., Bibliothek des Fachbereichs Informatik, 2003.

[22] S. Christensen and N. D. Hansen, "Coloured petri nets extended with place capacities, test arcs and inhibitor arcs," in *Applications and Theory of Petri Nets, volume 691 of LNCS*. Springer Verlag, 1992, pp. 186–205.

[23] C. Dufourd, A. Finkel, and P. Schnoebelen, "Reset nets between decidability and undecidability," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, K. Larsen, S. Skyum, and G. Winskel, Eds. Springer Berlin Heidelberg, 1998, vol. 1443, pp. 103–115. [Online]. Available: http://dx.doi.org/10.1007/BFb0055044

[24] L. Holloway and B. Krogh, "Controlled petri nets: A tutorial survey," in *11th International Conference on Analysis and Optimization of Systems Discrete Event Systems*, ser. Lecture Notes in Control and Information Sciences, G. Cohen and J.-P. Quadrat, Eds. Springer Berlin Heidelberg, 1994, vol. 199, pp. 158–168. [Online]. Available: http://dx.doi.org/10.1007/BFb0033544

[25] M. Cabasino, A. Giua, M. Pocci, and C. Seatzu, "Discrete event diagnosis using labeled petri nets. an application to manufacturing systems," *Control Engineering Practice*, vol. 19, no. 9, pp. 989 – 1001, 2011, <ce:title>Special Section: DCDS'09 – The 2nd {IFAC} Workshop on Dependable Control of Discrete Systems</ce:title>. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066111000025

[26] R. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on computers*, vol. 29, no. 12, pp. 1104–1113, 1980.

[27] M. Niedźwiecki, K. Rzecki, and K. Cetnarowicz, "Complex negotiations in the conclusion and realisation of the contract," July 2013, international Conference on Computational Science.

[28] J.-B. Voron and F. Kordon, "Transforming sources to petri nets: a way to analyze execution of parallel programs," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, ser. Simutools '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 13:1–13:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1416222.1416240