

METAPETRINETS FOR CONTROLLING COMPLEX AND DYNAMIC PROCESSES¹

Vagan Terziyan

*Kharkov State Technical University of Radioelectronics
Department of Artificial Intelligence and Information Systems
14 Lenina av., 310726, Kharkov, Ukraine
phone: +380 572 409 890, fax: +380 572 409 113
e-mail: vagan@milab.kharkov.ua and vagan@jytko.jyu.fi*

Vesa Savolainen *Senior Member of IEEE*

*University of Jyväskylä
Department of Computer Science and Information Systems
P.O.Box 35, FIN-40351, Jyväskylä, Finland
phone: +358 14 603 026, fax: +358 14 603 011
e-mail: vesa.savolainen@jyu.fi*

ABSTRACT

Multilevel petrinets are proposed for facilitating flexible modeling and control of complicated dynamic processes. A *metapetrinet* is able not only to change the marking of a petrinet but also to reconfigure dynamically its structure. Each level of the metanet is an ordinary petrinet of some traditional type. A basic level petrinet simulates the process of some application. The second level, i.e. the metapetrinet, is used to simulate and help controlling the configuration change at the basic level. There is conformity between the places of the second level structure and places or transitions of the basic level structure. The main control rule is such that a certain place or transition is removed from the present configuration of the basic level if the corresponding place at the metalevel becomes empty. If at least one token appears to an empty metalevel place, then the originally defined corresponding basic level place or transition immediately is created back to the configuration. We also introduce multilevel metapetrinets where every upper level may change the configuration of the previous lower level. In such a case even a few levels with a simple structure are able to simulate quite complicated control processes at the basic level. We assert that metapetrinets offer more compact facilities to the models of complex dynamic processes than single level petrinets.

Keywords: petrinet, metanet, control network, multilevel net models, reconfiguration

¹ Terziyan V., Savolainen V., Metapetrinets for Controlling Complex and Dynamic Processes, *IEEE Transactions on Software Engineering* (submitted to International Journal, January 1998).

METAPETRINETS FOR CONTROLLING COMPLEX AND DYNAMIC PROCESSES

ABSTRACT

Multilevel petrinets are proposed for facilitating flexible modeling and control of complicated dynamic processes. A *metapetrinet* is able not only to change the marking of a petrinet but also to reconfigure dynamically its structure. Each level of the metanet is an ordinary petrinet of some traditional type. A basic level petrinet simulates the process of some application. The second level, i.e. the metapetrinet, is used to simulate and help controlling the configuration change at the basic level. There is conformity between the places of the second level structure and places or transitions of the basic level structure. The main control rule is such that a certain place or transition is removed from the present configuration of the basic level if the corresponding place at the metalevel becomes empty. If at least one token appears to an empty metalevel place, then the originally defined corresponding basic level place or transition immediately is created back to the configuration. We also introduce multilevel metapetrinets where every upper level may change the configuration of the previous lower level. In such a case even a few levels with a simple structure are able to simulate quite complicated control processes at the basic level. We assert that metapetrinets offer more compact facilities to the models of complex dynamic processes than single level petrinets.

Keywords: petrinet, metanet, control network, multilevel net models, reconfiguration.

1. INTRODUCTION

Ordinary petrinets that were introduced by Petri [16] have a long time been an object of intensive research aiming at the development of powerful and flexible tools for modeling, simulating and controlling complex flows and processes (see also Peterson [14, 15]). The most successful extensions and refinements of ordinary petrinets are *colored nets*, *time petrinets*, *timed petrinets*, and *reconfigured structures*.

Colored nets and stochastic nets were introduced to extend the modeling power of petrinets [3]. The main idea in colored nets is that the tokens may have values or colors. Colored nets facilitate a concise, flexible and manageable representation of systems. They can also be used to model data and resources that reside in the system. As long as the number of colors is finite, a colored net is equivalent to an ordinary but much larger petrinet.

Petrinet theory was one of the first concurrent formalisms to help solving real-time problems. Two basic timed versions of petrinets have been introduced: time petrinets [12] and timed petrinets [17]. They both have widely been used in later research, for example in [2, 6, 11]. There are two questions that arise when time is introduced to net theory: (a) the location of the time delays, i.e. either at places or at

transitions, and (b) the type of the delay, i.e. a fixed delay, an interval or a stochastic delay [20].

Kelling et al. [8, 9] applied petrinet based simulation of spatial systems in cooperative work. Their aim was the performance evaluation of systems that are likely to have failures in the form of fault transitions or incorrect reconfigurations. In order to cope with the complexity of these systems it was necessary to derive a petrinet representation from such other description tools as Fault Trees [5]. A parallel computer system with faults can often be logically reconfigured to simulate an identical fault-free parallel computer with some loss of speed. Kelling et al. developed an extensive theory of reconfiguring commonly used networks, such as meshes, hypercubes, butterflies, and trees, so as to minimize the loss in performance due to faults.

The importance of flexible modeling and performance evaluation for the design of manufacturing systems is obvious. Jensen [7] has presented computer-based tools for the construction and modification of petrinets. A new modeling method based on colored petrinets was introduced in [21], which is especially tailored for manufacturing systems. The separate modeling of the manufacturing system's structure and the production routes with dedicated colored petrinets was proposed. The work [20] is also connected with the problem of adequate modeling of flexible manufacturing systems with petrinets. With the method proposed, it is intended to combine the profitable properties of both colored and uncolored petrinets. This is done by the predefinition of color sets for the modeling of parts and resource states of the flexible manufacturing systems.

These extensions and refinements of petrinets have added the modeling power, flexibility and controllability of petrinets. Similar influences were achieved by such simple enlargements as switches and inhibitor arcs (see Peterson [14], Savolainen [18] and Leppanen and Savolainen [10]). However, very powerful tools for flexible restructuring of petrinets, that often are necessary and useful in the scheduling and control of dynamic changing-routed jobshops (see [4]) or traffic, have been missing so far. The present work is part of research dealing with theory development for multilevel dynamic mathematical models of knowledge, inference, simulation and control. We use a metasemantic approach (see Terziyan [19]) based on the following paradigms (c.f. Auramaki, Leppanen and Savolainen [1] and Mesarovic, Macko and Takahara [13]):

- the basic level of a mathematical model describes the domain in terms of objects, properties and relations,
- a higher level describes rules of the dynamics in configuration modification in the lower level,
- the active structure of each level defines an active part of the previous lower level at the current moment,
- similar tools are applied in each level of the model, and

- the behavior complexity of the modeled object defines the number of levels which are necessary for an exact description.

In this paper, an interpretation of multilevel petrinets (MPNs) is proposed for flexible modeling of complicated dynamic processes and their control. An MPN is able not only to change its marking but also to reconfigure its structure dynamically, according to the triggering signals. Each level of the MPN is an ordinary petrinet of one of traditional types, according to the specific needs of the application. The basic level simulates the process of some application. The triggering signals must originate from the problems in the real world that the basic level petrinet models. The second level, i.e. the first metalevel, is used to simulate the configuration change at the basic level. There is a conformity between places of the second level structure and some attributes, either places or transitions, of the basic level structure, depending on the application. This is shown as an example in Figure 1.

The main control rule is such that a certain place or transition - depending on the current definition - is removed with all of its arc connections from the present configuration of the lower level if the corresponding place at the metalevel becomes empty. If at least one token comes to an empty metalevel place, then the corresponding basic level place or transition immediately is restored back to the configuration. In the multilevel metapetrinets every higher level may change the configuration of the previous lower level. In such a case even a few levels with a simple structure are able to simulate quite complicated processes at the basic level. We claim that an MPN structure facilitates a more compact modeling and control of a complex dynamic real world process than a traditional single level petrinet.

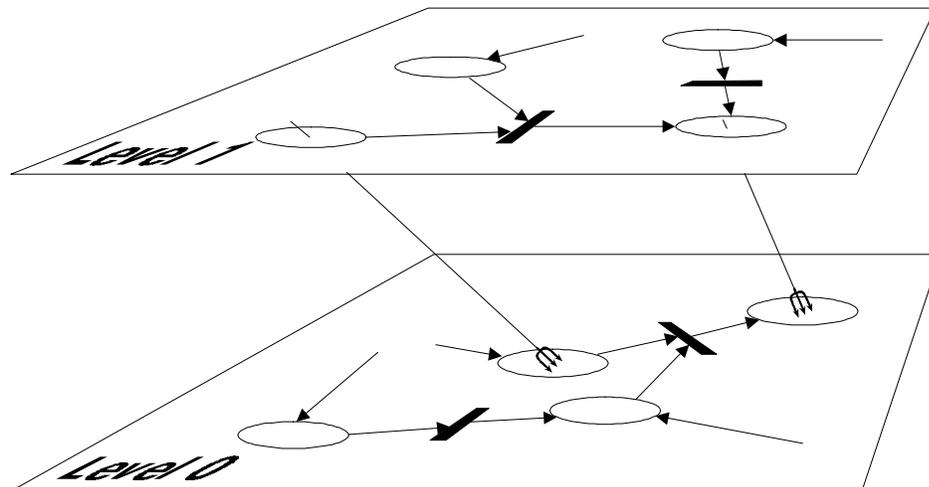


Figure 1. An example to illustrate the idea of metapetrinets

In this paper, Chapter 2 offers the notations and definitions of petrinets and metapetrinets. In the next chapter we define and analyze the main types of two-level MPNs. Chapter 4 gives illustrative examples of different types of metapetrinets.

Chapters 5 and 6 include the definition of generic multilevel metapetrinets. Finally, in the last chapter we present some conclusions.

2. PETRINET FOR SEPARATE LEVEL REPRESENTATION

Each level of the metapetrinet is represented by an ordinary petrinet. To study metanets, it is not so important which type of a petrinet is selected to describe a separate level. More important is to describe the interaction between the levels. Therefore we select the following petrinet formalism to develop our theory construction.

The classical *petrinet* is defined by a five-tuple:

$N = \langle P, V, F, H, M_0 \rangle$, where:

$P = \{P_1, P_2, \dots, P_n\}$ is a finite non-empty set of nodes (*places*),

$V = \{V_1, V_2, \dots, V_m\}$ is a finite non-empty set of nodes (*transitions*),

F is the input incidence function mapping from places to transitions,

H is the output incidence function mapping from transitions to places, and

M_0 is an initial marking of places, i.e. a nonnegative integer to describe the number of *tokens* in the places.

The simulation of a petrinet, that basically and traditionally is just a static model, is executed by *firing* the transitions that cause changes in the marking of their input places and output places. The transition can be fired if each of its input places has at least one token; then one token from each of its input places is removed and one token is added to each of its output places. For simplifying our metanet construction for the MPN, we shall apply a deterministic and non-conflicting rule to fire transitions: By suitable petrinet structure modifications, i.e. by applying petrinet subclasses (see Peterson [14]), the conflicts of common input places of transitions are avoided. Another simplifying assumption is that all transitions inside one level have the same fixed *time delay* between enabling and firing.

3. DEFINITION OF THE METAPETRINET

A *metapetrinet* can be constructed by adding two types of nodes, i.e. *metaplaces* and *metatransitions*, and the arcs connecting them. A metaplace may be assigned to each node of the initial petrinet. A metaplace is represented by a circle. A metaplace and the corresponding node are displayed inside a dotted-line oval and connected by a dotted-line arrow. In Figure 2a, a metaplace is presented to correspond to place P_i . In Figure 2b, a metaplace corresponds to transition V_j .

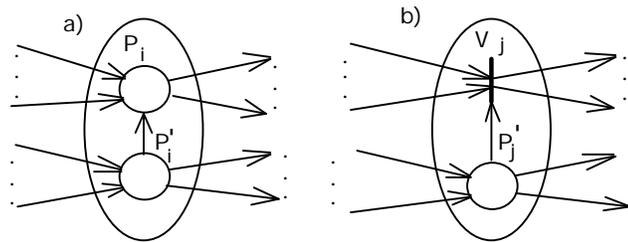


Figure 2. Two types of metaplaces

The construction controlling property of the MPN functions as follows: The nodes P_i and V_j in the Figure 2 are, accordingly, a place and a transition, and they belong to the basic level of petrinet. The nodes P'_i and P'_j are metaplaces that correspond to places P_i and V_j , and they belong to the metalevel. A node of a petrinet is defined to be *active* (available, live) if its corresponding metaplace contains at least one token. An empty metaplace means that its corresponding node on the basic level is *passive* (prohibited, dead) and the current structure of the net temporarily excludes this node with its connecting arrows. When at least one token comes to an empty metaplace then immediately its corresponding node on the basic level is restored together with its connecting arrows.

A metatransition is analogous to an ordinary transition. All its input and output places are metaplaces. In Figure 3, one can see an example of a metatransition, i.e. V'_i .

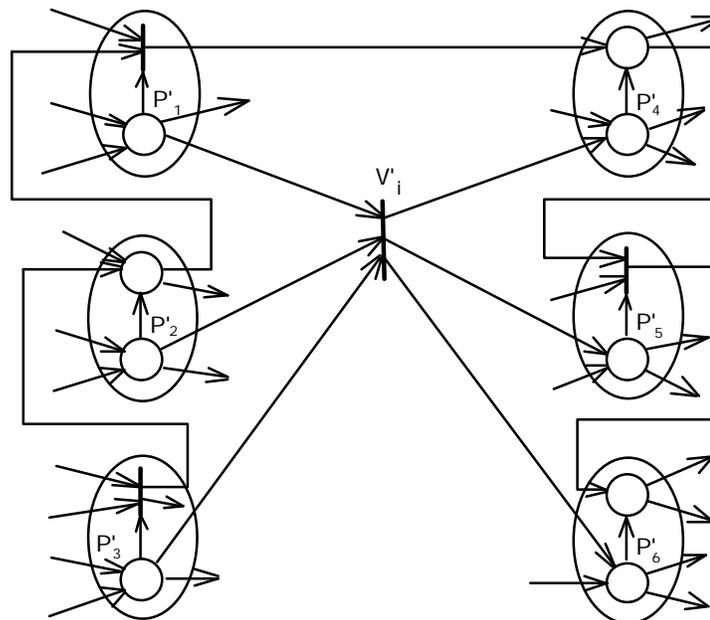


Figure 3. An example of a metatransition

In Figure 3, the nodes P'_1, P'_2, P'_3 are input metaplaces for the metatransition V'_i , and the nodes P'_4, P'_5, P'_6 are the output metaplaces for this metatransition. In this example, one can see that the nodes P'_2, P'_4, P'_6 are the metaplaces relative to the

places of the basic level petrinet, and P'_1, P'_3, P'_5 are the metaplaces relative to the transitions of the basic level petrinet.

The initial petrinet can be considered as the first level of a model. The structure of metaplaces and metatransitions represents the metapetrinet of the second level on the MPN. The execution of the metapetrinet defines the marking of metaplaces and so it defines the dynamic configuration of the basic level petrinet.

As it was noted above, each level of this stratified MPN construction has its own time delay between enabling and firing each transition. Let us suppose that the time measurement unit in this net model is the time delay of its basic level. We will call it a *beat*. We mark its value as symbol τ . It is necessary to note that the use of MPNs is reasonable if the time delay at the metalevel, i.e. the metatransitions' delay, is not shorter than that at the basic level. It means that the basic petrinet usually might operate during several beats with its current structure, only changing the marking, before some metatransitions fire and change this structure to another. If a transition at the same moment when the corresponding metaplace becomes empty is being to fire, then it first fires and disappears afterwards. For the controlling purposes it is essential to carefully define the time delays in each level.

We define our two-level MPN by the following vector:

$$N' = \langle P, V, F, H, M_0, P', V', F', H', M'_0, R', Q', k' \rangle, \text{ where:}$$

P, V, F, H and M_0 are components of an ordinary petrinet,

$P' = \{P'_1, P'_2, \dots, P'_n\}$ is a finite non-empty set of nodes, metaplaces,

$V' = \{V'_1, V'_2, \dots, V'_m\}$ is a finite set of metatransitions,

F' is the input incidence function mapping metaplaces to metatransitions,

H' is the output incidence function mapping metatransitions to metaplaces,

M'_0 is an initial marking of metaplaces,

R' is the incidence function mapping metaplaces to places,

Q' is the incidence function mapping metaplaces to transitions,

k' is the number of beats inside the fire time delay of a metatransition τ' so that $\tau' = k' \cdot \tau$; in other words, k' defines the number of one-bit steps of the petrinet execution between the nearest changes in its structure.

In the following we consider three varieties of metapetrinets. First one is a *transitional metanet*; metaplaces in such an MPN correspond only to transitions of the basic level. Second one is a *positional metanet*; metaplaces in it correspond only to places of the basic level. The last one is a *mixed metanet* where the metaplaces may be of any type.

4. EXAMPLES OF DIFFERENT TYPES OF MPN'S

Let us consider an example of a transitional MPN in Figure 4. This net is defined by the following sets: $P=\{P_1, P_2, P_3, P_4, P_5\}$ is a set of places, $V=\{V_1, V_2, V_3\}$ a set of transitions, the initial marking of places is: $P_1(1), P_2(5), P_3(2), P_4(0), P_5(0)$, $P'=\{P'_1, P'_2, P'_3\}$ is a set of metaplaces, $V'=\{V'_1, V'_2\}$ is a set of metatransitions, and the initial marking of metaplaces is: $P'_1(2), P'_2(0), P'_3(1)$.

Let us assume that $k'=5$. The execution table is presented in Table 1.

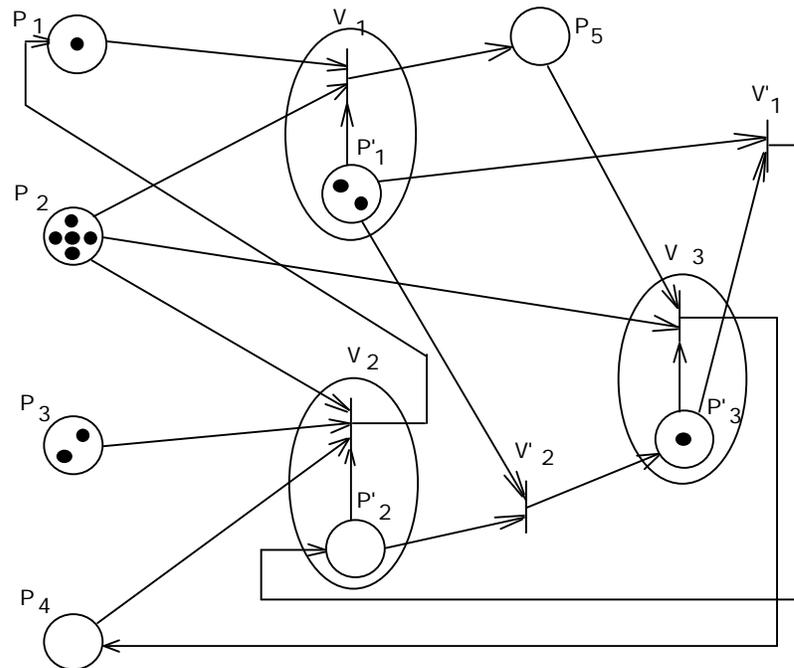


Figure 4. An example of a transitional metapetritnet

Step number	Firing metatransitions	Marking of metaplaces	Firing transitions	Marking of places
0	--	2;0;1	--	1;5;2;0;0
1	--	- ... -	V_1	0;4;2;0;1
2	--	- ... -	V_3	0;3;2;1;0
3	--	- ... -	--	-- ... --
4	--	- ... -	--	-- ... --
5	V'_1	1;1;0	--	-- ... --
6	--	- ... -	V_2	1;2;1;0;0
7	--	- ... -	V_1	0;1;1;0;1
8	--	- ... -	--	-- ... --
9	--	- ... -	--	-- ... --

10	V'_2	0;0;1	--	-- ... --
11	--	- ... -	V_3	0;0;1;1;0

Table 1. The execution table in the example with a transitional metanet

The next example presents a positional MPN in Figure 5 with the same basic level as in the previous example.

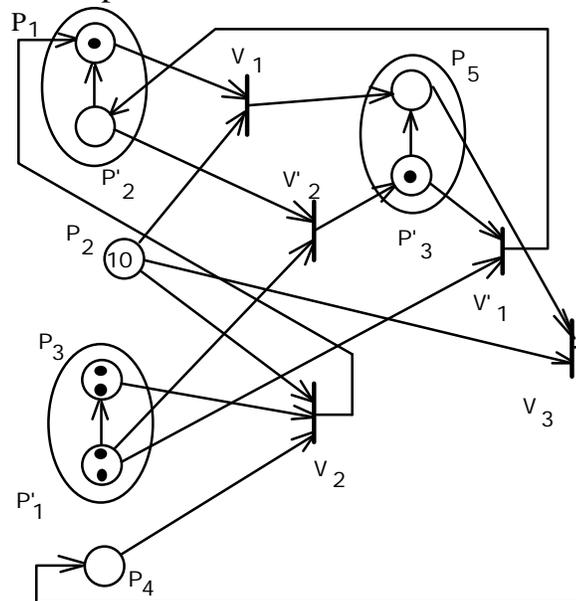


Figure 5. An example of a positional metapetrinet

As one can see in Figure 5, the same nets as in Figure 4 are used at both levels, although they are connected with each other through places instead of transitions. Using nearly the same marking (except P_2), we obtain execution results in Table 2.

Positional metapetrinets have one peculiarity: if a metaplace is empty, then the corresponding place "falls asleep" with all its input and output links. The marking of deleted place is preserved up to the metaplace receives token and "wakes up" the place.

Step number	Firing metatransitions	Marking of metaplaces	Firing transitions	Marking of places
0	--	2;0;1	--	1;10;2;0;0
1	--	- ... -	V_1	1;9;2;0;1
2	--	- ... -	V_1, V_3	1;8;2;1;1
3	--	- ... -	V_1, V_3	1;7;2;2;1
4	--	- ... -	V_1, V_3	1;6;2;3;1
5	V'_1	1;1;0	--	-- ... --
6	--	- ... -	V_2, V_3	2;5;1;3;1
7	--	- ... -	V_2, V_3	3;4;0;3;1

8	--	- ... -	V_3	3;3;0;4;1
9	--	- ... -	V_3	3;2;0;5;1
10	V_2'	0;0;1	--	-- ... --
11	--	- ... -	V_1, V_3	3;1;0;6;1
12	--	- ... -	V_1, V_3	3;0;0;7;1

Table 2. The execution table in the example with a positional metanet

Let us then consider an example of a mixed MPN in which metaplaces may correspond to transitions or places. In Figure 6, one can see a metanet obtained by combining the same two levels as in previous examples ($k'=5$).

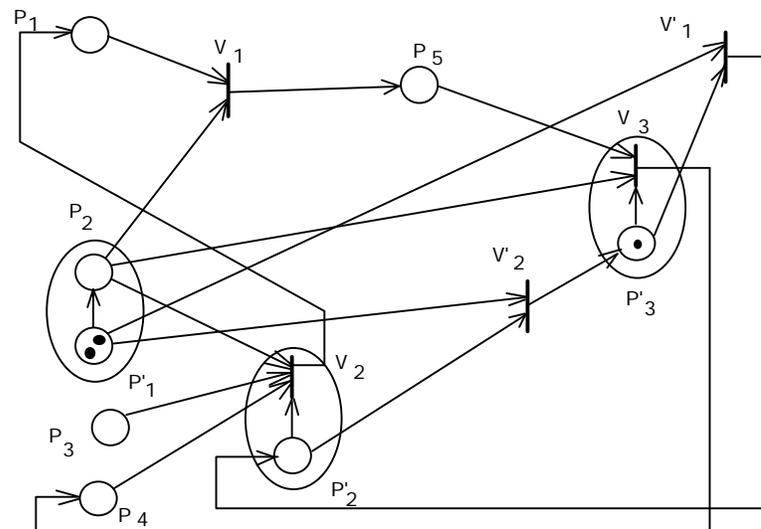


Figure 6. An example of a mixed metapetrinet

Let us show how the structure of the basic level petrinet in the example is changed in time under the control of the metalevel marking. First of all, the net presented in Figure 7 acts during five time beats.

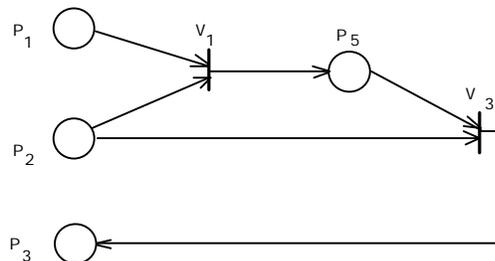


Figure 7. The first five-beat structure of the basic level of a mixed metanet

The following net in Figure 8 is executed during the next five beats.

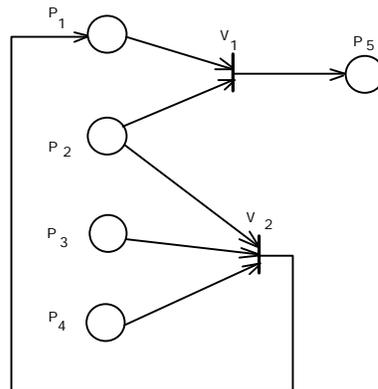


Figure 8. The second five-beat structure of the basic level of a mixed metanet
The structure presented in Figure 9 is executed during the next five beats.

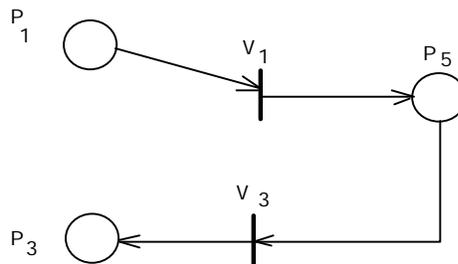


Figure 9. The third five-beat structure of the basic level of a mixed metanet

If we try to simulate the processes presented in these examples by using an ordinary petrinet only, we can convince of the fact that the obtained structure is somewhat more complex and intricate than shown here by means of metanets. For example, the process described in Table 2 can be simulated by a petrinet shown in Figure 10.

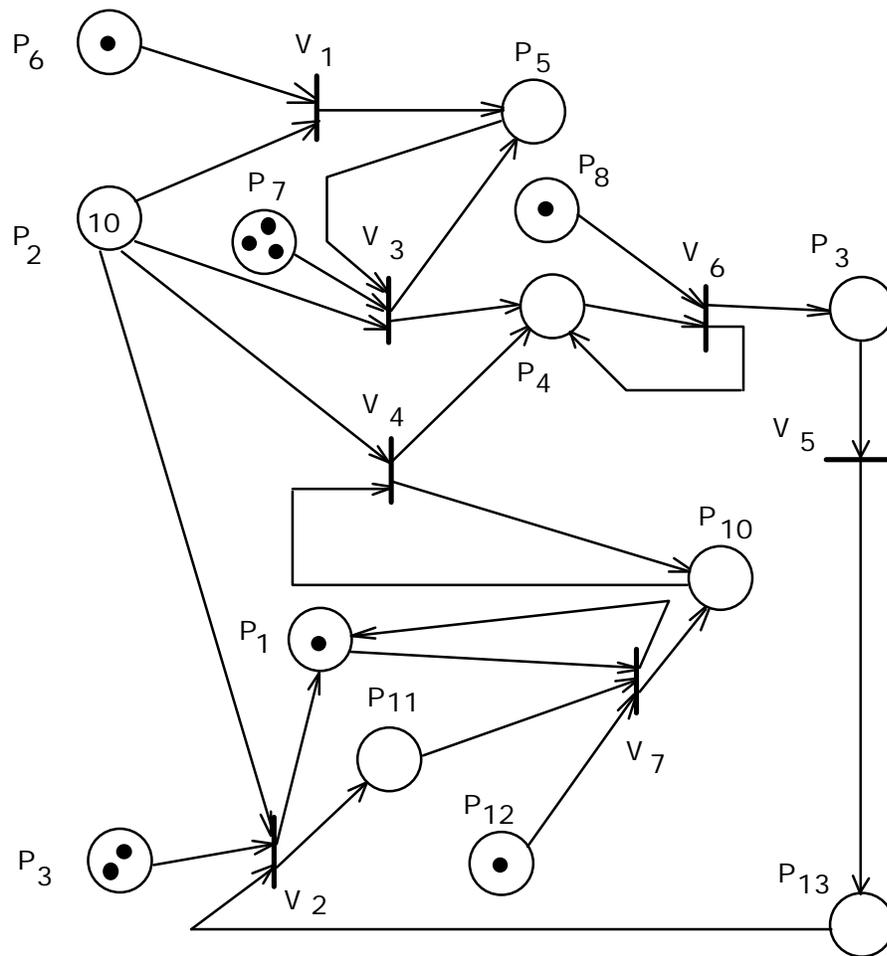


Figure 10. An ordinary petrinet simulation of the example with a positional metapetrinet

As one can see it was necessary to add 8 places and 4 transitions to the initial petrinet of Figure 5. At the same time, the metanet construction adds only 3 metaplaces and 2 metatransitions to the controlling metalevel.

It is not so simple but it is possible to simulate positional metanets by colored timed petrinets. In that case there should be one special type of tokens, i.e. a switch [10, 14, 15, 18]. The transitional fire delay for switches should be several times more than for other tokens. If a switch comes to some place, then this place changes its status from passive to active or vice versa. A passive state means that the place becomes temporarily inaccessible for all other tokens and it can be used only by switches. However, it seems to us, that the processes in transitional and mixed metapetrinets cannot be represented so compactly by only a basic level petrinet as using our metanet representation.

Thus, the first advantage of our PMN representation seems to be possibility to use more simple and compact petrinets at each level for simulating quite complicated processes.

The second advantage is that the effectiveness of executing PMNs is higher in computer-based applications because the permanent test of transitions can be made only for currently active substructures that is extracted from the basic level petrinet under the control of metalevels.

Further on, it seems to us that metapetrinets can be useful for discrete modeling due to its compactness and possibility to confine an exhaustion within the active part of the PMN structure.

5. MULTILEVEL METAPETRINETS

If we use the metapetrinet for a too complex process simulation there may occur some cases where the second metanet level by itself is as compound as a net being under its control. In such cases it is reasonable to add another controlling metalevel to the MPN. Thus we obtain metaplaces of the third level that control to metaplaces and metatransitions of the second level of the MPN. Metatransitions of the third level connect these metaplaces. As presented in Section 4, the second level of the MPN changes the configuration of the basic petrinet level.

In a general case, metapetrinets allow any number of metalevels. The more complex process it is necessary to simulate, the more metalevels are needed to describe it in an adequate and compact way. It is usually required that the firing delay time for the i -th level transition is several times higher than time for the $(i-1)$ -th level transition.

Figure 11 presents an example of a three-level MPN.

The nodes P_1, P_2, P_3, P_4 are places of the basic level of MPN, V_1, V_2 are transitions of the basic level, P'_1, P'_2, P'_3, P'_4 are places of the second level (metaplaces), V'_1, V'_2 are transitions of the second level (metatransitions), P''_1, P''_2, P''_3 are places of third level (metametaplaces), V''_1 is a transition of the third level (metametatransition).

To define a metapetrinet of the $(r+1)$ -th level we use the following vector:

$$N^{(r)} = \langle P, V, F, H, M_0; P', V', F', H', M'_0, R', Q', k' ;$$

$$P'', V'', F'', H'', M''_0, R'', Q'', k'' ; \dots ; P^{(r)}, V^{(r)}, F^{(r)}, H^{(r)}, M_0^{(r)}, R^{(r)}, Q^{(r)}, k^{(r)} \rangle,$$

where:

$P^{(r)}$ is the finite non-empty set of nodes (metaplaces of the $(r+1)$ -th level of MPN

$$\{P_1^{(r)}, P_2^{(r)}, \dots, P_{n^{(r)}}^{(r)}\},$$

$V^{(r)}$ is the finite set of metatransitions of the $(r+1)$ -th level of MPN $\{V_1^{(r)}, V_2^{(r)}, \dots, V_{m^{(r)}}^{(r)}\},$

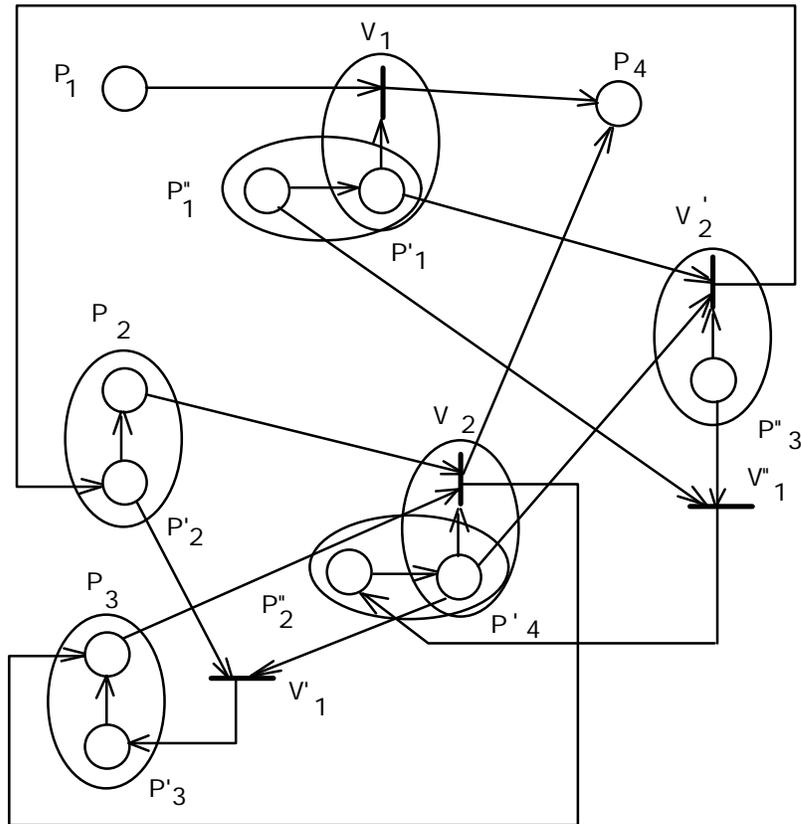


Figure 11. An example of a three-level metapetri net

$F^{(r)}$ is the input incidence function mapping metaplaces of the $(r+1)$ -th level to metatransitions of the same level,

$H^{(r)}$ is the output incidence function mapping metatransitions of the $(r+1)$ -th level to metaplaces of the same level,

$M_0^{(r)}$ is an initial marking of metaplaces of the $(r+1)$ -th level,

$R^{(r)}$ is the incidence function mapping metaplaces of the $(r+1)$ -th level to places of the r -th level,

Q' is the incidence function mapping metaplaces of the $(r+1)$ -th level to transitions of the r -th level, and

$k^{(r)}$ is the number of metatransitions' fire time delays of the r -th level inside the metatransitions' fire time delay of the $(r+1)$ -th level.

The more levels the MPN has the more configurations of the basic level it can produce. One can control changes at the basic level by changing marking of metaplaces at any level. The higher level is chosen for changing the marking, the more essential changes may be expected in the execution of the basic level.

6. CLASSIFICATION OF GENERIC METAPETRINETTS

In order to advance to a more general classification of possible metapetrinet types in this section, we consider, without any loss of generalization, two-level metapetrinets where the controlling metalevel is a classical petrinet and the basic level is one or a combination of the following petrinet types: classical, timed or colored. In principle, the controlling level could effect all of the four attributes of a petrinet, i.e. places, transitions, links and tokens. Let us consider an example presented in Figure 12. The possible ways of a basic level control are presented in Table 3.

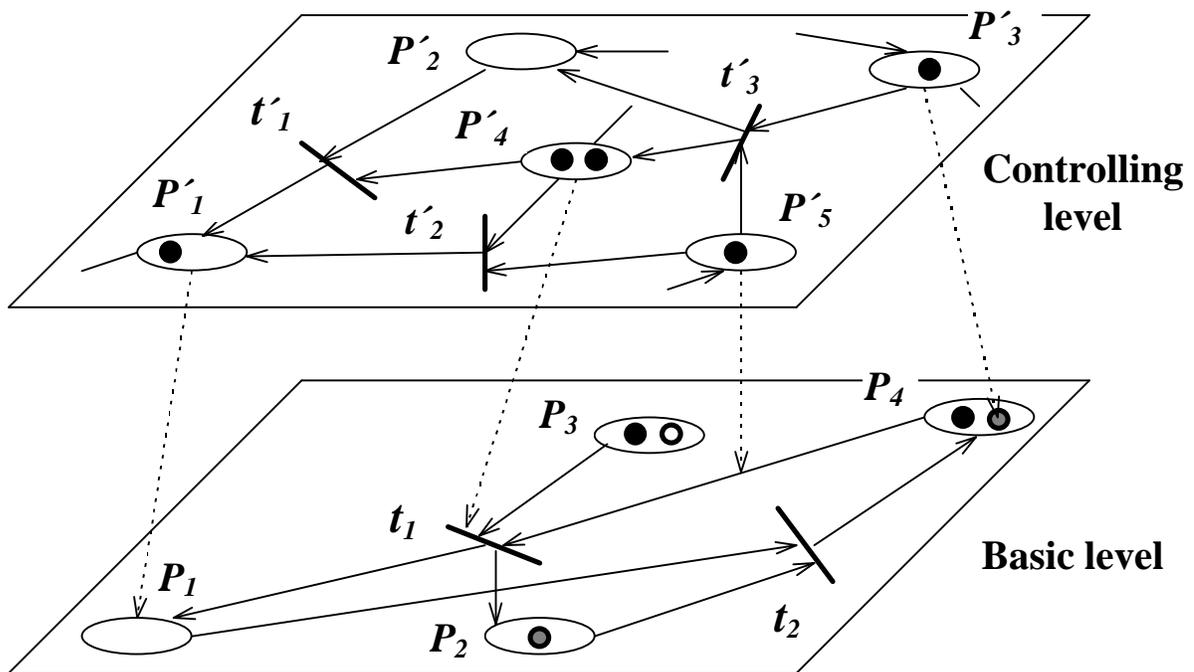


Fig. 12. An example to illustrate the metapetrinet control types

basic level petrinet attributes				
	<place>	<transition>	<link>	<token>

controlling effect	1) removing a place; 2) restoring a place; 3) changing a place's capacity; 4) changing a place's marking	1) removing a transition; 2) restoring a transition; 3) changing time settings; 4) changing the fire rule	1) removing a link; 2) restoring a link; 3) changing a link's direction; 4) changing a link's capacity	1) removing a token; 2) restoring a token; 3) changing a token's color; 4) changing a token's place
--------------------	---	--	---	--

Table 3. Classes of controlling interactions between the metapetrinet levels

In Figure 12, a two-level metapetrinet is shown. It is supposed that the controlling level of the metanet is a classical petrinet. The basic level of the metanet is a coloured petrinet. There are four controlling interlevel links in this figure.

The metaplace $P'1$ controls the place $P1$. Depending on how we wish to control the basic level, this might mean, for example, one of the following alternatives:

- if the metaplace becomes empty, then the place is removed out of the net together with all its links,
- if the metaplace becomes nonempty, then the place is being restored into the net together with all its links;
- if the metaplace changes its marking, then the capacity of the place is assigned to the number of tokens in the metaplace;
- if the metaplace is increases its marking, then the current amount of black tokens of the place is increased,
- if the metaplace is decreases its marking, then the current amount of white tokens of the place is decreased.

The metaplace $P'3$ controls the gray token of the place $P4$. Depending on how we wish to control the basic level, this might mean, for example, one of the following alternatives:

- if the metaplace becomes empty, then the controlled token is being removed out of the net,

- if the metaplace becomes nonempty, then the token is restored into the net with the same color into the same place;
- if the metaplace changes its marking to b , then the color of the token is changed from the current color to the color number b ;
- if the metaplace changes its marking to b , then the token is removed from the current place to the place number b .

The metaplace $P'4$ controls the transition $t1$. This might mean, for example, one of the following alternatives:

- if the metaplace becomes empty, then the transition is removed out of the net together with all its links,
- if the metaplace becomes nonempty, then the transition is restored into the net together with all its links;
- if the metaplace changes its marking, then a certain time setting of the transition is assigned to the number of tokens in the metaplace;
- if the metaplace changes its marking to b , then the fire rule which is currently used with the transition is changed to a fire rule with number b .

The metaplace $P'5$ controls the link between the place $P4$ and the transition $t1$. Depending on how we wish to control the basic level, this might mean, for example, one of the following alternatives:

- if the metaplace becomes empty, then the link is being removed out of the net,
- if the metaplace becomes nonempty, then the link is restored into the net at the same place with the same direction;
- if the metaplace changes its marking to b , then the capacity of the link is changed from the current value to b ;
- if the metaplace changes its marking, then the link changes its direction.

We summarize these control classes in Table 3. We claim that the possibility of utilizing any or a combination of these control types facilitates the modeling of any required dynamics in the MPN structures.

7. CONCLUSIONS

Our constructive study of metapetrinets was made as an attempt to formulate a flexible and simple model for facilitating deterministic simulation and control of complicated and very dynamic processes. The MPN is able not only to change its marking but also to reconfigure dynamically its structure. It was shown by examples that our metanet representation uses only simple and compact petrinets at each level. Our examples

justify us to assume that the effectiveness of the execution of the MPN will be higher in computer-based modeling and simulation applications because the permanent test of transitions can be made only for currently active substructure that is extracted from the basic level petrinet under the control of metalevels.

The MPN is useful for discrete modeling due to its compactness and possibility to confine an exhaustion within the active part of the MPN structure.

The semantics for the MPNs was only informally sketched in this paper. Our extensions mean that the analysis methods constructed for ordinary petrinets cannot always be applied as such. The transitions must be fired in maximal steps. This makes the modeling power equivalent to that of Turing machines since the simulation of counter machines is simple.

It is an interesting feature of this model that as such it does not allow information transfer from the lower levels to the higher levels. But this kind of additional model features can be easily constructed by using petrinet links.

It seems to us that a completely deterministic behaviour of an MPN can possibly find interesting applications in artificial intelligence in such areas as cellular metaautomata or genetic metaalgorithms.

Our research today aims at analyzing and constructing the application environments for the MPN. The metapetrinet technique was experimentally tested within a software shell called Meta Net Editor for the simulation of discrete dynamic processes and it was applied to simulate the flow of precious metals within technological processes at the Kharkov's Jewelry Plant. Further investigation is planned in the area of flight connection control. Rigorous mathematical justification for metapetrinets might also be of interest.

REFERENCES

- [1] E. Auramaki, M. Leppanen, and V. Savolainen, Universal Framework for Information Activities. *Data Base*, Vol. 19, No. 1, 1988, pp. 11-20.
- [2] B. Berthomieu and M. Diaz, "Modeling and Verification of Time Dependent Systems Using Time Petri Nets", *IEEE Trans. Software Engineering*, Vol. 17, No. 3, 1991, pp. 259-273.
- [3] J. Billington, G.R. Wheeler, and M.C. Wilbur-Ham, "PROTEAN: A High-Level Petri Net Tool for the Specification and Verification of Communication Protocols", *IEEE Trans. Software Engineering*, Vol. 14, No. 3, 1988, pp. 301-316.
- [4] R.W. Conway, W.L. Maxwell, and L.W. Miller, Theory of Scheduling, Addison-Wesley, London, 1967.
- [5] J. F. Ereau, H. Demmou, M. Saleman, "Dynamic Fault Trees Based on Synchronized Petri Nets", *Proc. 7th European Simulation Symposium*, Erlangen, October 26-28 1995.
- [6] F. S. Etesami and G.S. Hura, "Rule-Based Design Methodology for Solving Control Problems", *IEEE Trans. Software Engineering*, Vol. 17, No. 3, 1991, pp. 274-282.
- [7] K. Jensen, "Computer Tools for Construction, Modification and Analysis of Petri Nets," in *Petri Nets: Applications and Relationships to Other Models of Concurrency*, W. Bauer, W.

- Reisig, and G. Rozenberg, Eds., LNCS Vol. 255, Springer-Verlag, Berlin, Feb. 1987, pp. 4-19
- [8] Ch. Kelling and G. Hommel, "Modeling Priorities in Token Protocols", *Int. Journal on Mini and Microcomputers*, Special Issue, Vol. 17, No. 1, 1995, pp. 35-41.
- [9] Ch. Kelling, J. Henz, and G. Hommel, "Design of a Communication Scheme for a Distributed Controller Architecture Using Stochastic Petri Nets", *Proc. of the 3rd. Workshop on Parallel and Distributed Real-Time Systems*, Santa Barbara, IEEE Press, 1995, pp. 147-154.
- [10] L. Leppanen and V. Savolainen, "Petri Net Modeling of the Control Structures of Programming Languages", in *Algebra, Combinatorics and Logic in Computer Science*, J. Demetrovics, G. Katona, and A. Salomaa, Eds., Colloguia Mathematica Soc. J. Bolyai, Vol. 42, North-Holland, Amsterdam, 1985, pp. 583-595.
- [11] N. G. Leveson and J. L. Stolzy, "Safety Analysis Using Petri Nets", *IEEE Trans. Software Engineering*, Vol. 13, No. 3, 1987, pp. 386-397.
- [12] P. M. Merlin and A. Segall, "Recoverability of Communication Protocols - Implications of a Theoretical Study", *IEEE Trans. Communications*, Sept. 1976, pp. 1036-1043.
- [13] M. Mesarovic, D. Macko, and Y. Takahara, Y., *Theory of Hierarchical Multilevel Systems*, Academic Press, London, 1970.
- [14] J.L. Peterson, "Petri Nets", *Computing Surveys*, Vol. 9, No. 3, 1977, pp. 223-252.
- [15] J.L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1981.
- [16] Petri C.A., *Kommunikation mit Automaten*, Schriften des IIM, No. 2, Bonn, 1962.
- [17] C. Ramchandani, *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*, Technical Report MAC TR 120, MIT, February 1974.
- [18] V. Savolainen, "Concurrency Maximizing Database Scheduler and Time-indicating Petri Net Description", in *Proceedings of the 4th Hungarian Computer Science Conference*, M. Arató, I. Katai, L. Varga L., Eds., Budapest, 1985, pp. 159-169.
- [19] V. Terziyan, *Multilevel Models for Knowledge Bases Control and Their Applications to Automated Information Systems*, Doctor of Technical Sciences Degree Thesis, Kharkov State Technical University of Radioelectronics, 1993 (in Russian).
- [20] W. M. Van der Aalst, *Timed Coloured Petri Nets and their Application to Logistics*, Ph.D. thesis, Eindhoven University of Technology, 1992.
- [21] A. Zimmermann, "A Modeling Method for Flexible Manufacturing Systems based on Colored Petri Nets", *Proc. Int. Workshop on New Directions in Control and Manufacturing*, Hong Kong, 1994, pp. 147-154.
- [22] A. Zimmermann, S. Bode, and G. Hommel, "Performance and Dependability Evaluation of Manufacturing Systems Using Petri Nets", *Proc. 1st Workshop Manufacturing Systems and Petri Nets, 17th Int. Conf. on Application and Theory of Petri Nets*, Osaka, Japan, 1996, pp. 235-250.