The Method of Translation of the Diagram with One Type Directed Links into the Inhibitor Petri Net

Nikolay Voit, Dmitry Kanev, Sergey Kirillov(✉), and Maria Ukhanova

Ulyanovsk State Technical University, Ulyanovsk, Russia
n.voit@ulstu.ru, dima.kanev@gmail.com, kirillovsyu@gmail.com, mari-u@inbox.ru

Abstract. The fundamental scientific problem of the business process management theory is to increase the efficiency of automated systems workflow synthesis and processing in order to reduce the time spent on their development, increasing the success of processing diagram models, namely the implementation of the requirements for resource constraints, functionality, financial component and deadlines, as well as improving the diagram models quality in terms of error control, narrowing the semantic gap between business process analysis and execution. The article proposes a method of translation of the diagram with one type of directed communication into the Petri inhibitor network. The results of the research represent that the approach has significant advantages over similar methods of analysis. The effectiveness of this analytical approach is proved by concrete real and relevant examples.

Keywords: Business processes · Graphical language analysis · EPC · Petri net component

1 Introduction

In modern practice of organizational management, graphical models of business processes have become widespread. However, graphical models research, even carried out in accordance with the rules of the structural approach (limited context, limiting the number of elements at each level of decomposition, etc.), presents considerable complexity. Therefore, a mandatory step in the modeling of business processes of enterprises is the automatic verification of the models obtained. The issues of analysis of defect-free completion are relevant, since the complexity of models is constantly increasing, and the test tools built into the simulation environment are far from perfect.

The main goal of this work is to reduce the time to develop graphical models of business processes and improve their quality by expanding the class of diagnosed errors.

Petri net is one of the approaches for it. Since its inception in 1962, Petri nets have been used in a wide variety of applications. Although Petri nets are graphical and easy to understand, they have formal semantics and allow analysis methods from model checking and structural analysis to process and performance analysis. Over time, Petri nets have become a solid foundation for research in the field of business process management. Petri nets allow semantic and syntactic models analysis. In particular, to obtain all possible scenarios for the business process execution, to determine never-executing functions and system deadlocks, which is especially important when analyzing cyclic processes.

At the same time, the existing methods for translating graphical diagrams of business process management into the Petri net are specialized and aimed at working with one or two graphical languages.

2 Analysis of Existing Approaches

In article [1], the rules for the transformation of UML activity diagrams in the Petri net are considered and an algorithm for their implementation is proposed. The waiting state is transformed into a position, the action state is transformed into a transition. The separation and merging of control, branch and condition flows is converted into the corresponding Petri net; constraint conditions are represented using transition constraint conditions.

Practically in each developed system one can meet the sequence and condition, branching and merging. Consequently, one of the advantages of the described rules seto is that it contains recommendations for the mutual obtaining of the more frequently encountered elements in the UML activity diagrams and Petri nets algorithms. Algorithmic rules presentation contributes to their formalization.

In the article [2] the technique of software design using UML-diagram and Petri net consisting of 7 stages is considered:

1. Compilation of a case diagram with the identification of all entities in the form of actors and use cases.
2. Drawing a class diagram and highlighting the class methods.
3. Selection based on the class diagram of objects that are involved in the system and will help determine the initial values for marking all the vertices of the Petri net.
4. Construction of activity diagrams to describe the dynamic properties of the system.
5. Translation of the constructed activity diagram into the Petri net using formal transformation rules.
6. Analysis of properties using automated software packages for working with Petri nets, which allow identifying logical errors in modeling activity diagrams: finding unused work scenarios, identifying dead-end branches of algorithms, etc. For classes or objects that need detailed elaboration of their logic, a separate activity diagram should be drawn up and analyzed using Petri nets. If necessary, the dynamics of work should be tracked on the state of the system. This is possible using Petri nets apparatus, whose analysis results will be the basis of the state space construction and exploration.
7. In case of successful analysis and set of diagrams that do not need to be improved, the simulated diagrams are ready for automatic code generation.

In [3], the converting UML diagrams into Petri nets rules are given. These are the following.

1. The waiting state is transformed into a position, and the action state is transformed into a transition that starts the action. It is possible that two states of the action are arranged in series, then the end of the first and the beginning of the second action are combined and shown by one transition.
2. Separation and merging of parallel control flows of UML diagrams is converted to the corresponding Petri net equivalent.
3. The branch in the activity diagram, denoted by the decision symbol, is converted to the corresponding Petri net equivalent.
4. Restriction conditions are shown with transition restriction conditions.
5. The critical section (access to the critical resource realized by the critical section) is converted to a position with one mark in the initial marking. When a critical section is executed, the label is removed from the position and returns upon completion of the critical Sect.
6. The semaphore used to simulate the employment of a specific resource from the resource pool is converted to a position with the same number of labels that differ from each other in colors in the initial labeling. A label with an arbitrary index should be removed from the site and returned when it is no more used. In a simple semaphore, labels of the same type must be used, and their number in the initial marking is equal to the number of resources.
7. It is necessary to add a counter-position to the sections that are critical in terms of execution time. To check the maximum execution time of the transitions of the critical section, you must add protective conditions. The counter must be provided with a transition with the opposite protective condition.
8. In case of mutual transition from or to the hierarchical Petri net, it is necessary to add empty spaces and corresponding transitions, which are connected in series.

The goal of [4] is to describe the method of (H)MSC diagrams into the Petri net translation and apply the method for formal analysis and verification of the properties of such diagrams. The algorithm result is the Petri net in a format compatible with the CPN Tools system. This net will be hierarchical if the source specification is specified using the HMSC diagram, or if the input MSC contains reference expressions.

Message sequence diagrams (MSC diagrams) are popular scripting language designed to formalize and analyze system requirements during the software design phase. In this paper, the syntax of the SDL-2010 language is used to describe declarations and expressions with data.

General description of the translation algorithm is considered. At the first stage, the input (H)MSC-diagram builds its internal representation, which will be called the partial order graph (H)MSC. For each event in the diagram, a node is created in the partial order graph, which stores information about the set of adjacent nodes, the process identifier and the structural structure to which the event belongs.

In stage 2, the partial order graph is processed. During processing, such actions as auxiliary graph nodes creation (input and output nodes for MSC embedded expressions), links unfolding (constructing partial order graphs for referential MSC), searching and processing alternatives with non-local choice occur.

At stage 3, the processed graph of partial order is transmitted to the Petri net (CPN). Each node of the graph corresponds to the transition to CPN. Each arc connecting two nodes in a partial order graph corresponds to a position and two oriented arcs connecting two transitions in the CPN. The orientation of the constructed network arcs coincides with the orientation of the arcs of the partial order graph. The event execution in MSC corresponds to the triggering of the corresponding transition in the resulting CPN. The start events of the MSC-diagrams correspond to transitions with input start sites with an initial marking of 1'(). The MSC-diagrams correspond to eventual transitions with output end places that do not contain outgoing arcs. All data types and variables declared in the MSC document are converted to the corresponding data types and CPN ML variables.

Each event from the base set of MSC elements is modeled by a single CPN transition. According to the MSC standard, the event of receiving a message is preceded by the event of its sending. Each message is modeled by two transitions in CPN. The order of transitions corresponding to the events of sending and receiving a message is observed with the help of a place that connects these transitions and ensures the correct sequence between their operations.

Predicative (guarding) conditions are used to check in the process of executing the truth diagram of a predicate specified in the MSC data language. Such conditions can be placed at the beginning of an inline expression or MSC diagram.

If the test condition is false, then the execution of chart events that follow the protection condition is terminated. In the process of translation, each predicate condition is transformed into a single transition of a network with a trigger function that calculates the value of the established predicate.

The article describes the translation of the basic elements of the MSC, MSC elements with data, structural elements. The resulting CPN size estimation is also proposed.

The article [5] proposes formal semantics for mapping the business process modeling (BPMN) notation to color Petri nets. The choice of CPN is due to the presence of formal verification methods, analysis of the space of states and invariants. And the concept of colored markers allows you to model the process data.

The elements analyzed are limited to the core BPMN, and include the following types. Common elements that are used in several types of diagrams (for example, “Process”, “Interaction”, “Choreography”). Common elements provide model developers with the ability to display additional information about the process, such as operations, error detection, and resources. The CPN ML declaration, corresponding to the standard BPMN 2.0 structure diagram, is considered. These declarations allow converting BPMN elements and their attributes one-on-one to Petri Net. Gateways are used to control the actions flows. The article uses the matrix display of various types of gateways to the corresponding Petri nets. BPMN tasks are matched with CPN ML ads. The CPN ML color set is created for each type of BPMN task in order to cover all of its optional attributes. In BPMN, an action can have attributes that define its optional behavior, such as looping and parallel loops on objects. The article proposes replacing them with a flow control construct using various types of gateways. A hierarchical CPN is used to represent the BPMN sub process. Each subprocess is represented as an embedded Petri net. A reusable task is a task definition that can be invoked from any process using a call operation. In CPN, a reusable task and its challenge is modeled using nested Petri nets.

The article [6] considers only a limited subset of the BPMN notation used to model orchestration processes. The nodes of the diagram are the control flow objects, including: operations, logical operators, and events. The difference of the proposed approach in the rejection of behavioral equivalence, which will simplify the display method.

The proposed approach distinguishes between nodes that produce a change in the management object, leading to a change in its state and nodes that do not change it, but route it. Operations transform the process object, they are associated with transitions. The arcs define the order in which the process operations are executed.

Logical branching and merging operators, such as AND, OR, are modeled by an equivalent Petri net. The BPMN notation allows for an “abbreviated” notation when one graphic element combines an operation and a logical operator at the same time. For analysis, such elements are expanded into a separate operation and a logical operator. Cyclic execution of the operation is modeled by a pair of transitions. The first is responsible for carrying out the work, and the second is for checking the condition.

Start, end and intermediate events placed in the stream are modeled by the Petri Net transition. Events attached to the boundaries of operations are modeled by the logical operators “OR”/“AND” depending on whether the event stops the flow of control or not.

Based on the review, we can conclude that a single approach is used to translate various types of diagrams into the Petri net: sequential transformation of the elements of the diagrams into equivalent subsets of the Petri nets. At the same time, the existing methods of translating graphic diagrams are specialized and aimed at working with one/two graphic languages.

3 Diagram to Inhibitory Petri Net Translation

Consider the task of translating some diagram into the inhibitor Petri net. The input parameter is a diagram, the diagram model has the form [7–12]:

$$G = (V, E, TV, TE),$$

Where

V is vertices set,
E is connections set, $E \subset (V \times V)$
TV is vertices types set,
TE is connections types set.

Restriction on the input data is assumed that the diagram may contain only one type of directional communication, which corresponds to an edge in Petri nets.

At the output, an inhibitor Petri net is obtained, the model of which has the form:

$N = (P, T, F)$, где
P is positions set,
T is transitions set,
F is arcs set, $F \subset (P \times T \times ING) \cup (T \times P \times ING)$, $ING = \{0, 1\}$ is an inhibitory arc attribute.

The input diagram vertices are translated on the Petri nets that implement the equivalent function. Inhibitor Petri net is Turing complete, which allows to implement any computable function on it. The unidirectional communication diagram is translated into the corresponding arc of the Petri net.

Displaying algorithm is the following.

Step 1. Defining the function of translating the top of the diagram into the extended Petri net.

For each type of diagram, it is necessary to define the function of translating the top of the diagram into the extended Petri net model ($Ftrans : V \rightarrow VN$). Extended Petri net model has the following view:

$$VN = (V, N, PIN, TOUT),$$

where

V is a diagram vertex,
N is Petri net,
PIN is input positions tuple,
$TOUT$ is output positions tuple.

For each incoming connection v there must be an input position in the Petri net *vn[N]* and for each outgoing connection there must be a transition.

Step 2. Formation of a set of extended Petri nets.

For each vertex v ∈ V of the diagram g ∈ G translation to the extended Petri net $vn \in VN$ is formed and AVN set is formed:

$$AVN = \{Ftrans(v) \mid v \in V\}$$

Step 3. Consolidation of multiple extended Petri nets into the output Petri nets.

Let *pn* be the empty Petri net $(\emptyset, \emptyset, \emptyset)$. Copy all the arcs, positions and transitions of all extended Petri nets into pn:

$$pn[P] = \{p | \exists a \in AVN,\ p \in a[N][P]\},$$
$$pn[T] = \{t | \exists a \in AVN,\ t \in a[N][T]\},$$
$$pn[F] = \{f | \exists a \in AVN,\ f \in a[N][F]\}$$

Step 4. Translation of diagram connections.

Based on each connection of the $e \in E$ of the diagrams, we connect the positions and transitions of the output Petri net. For each connection, the initial and final vertices and the corresponding extended Petri nets $vn1 \in VN$, $vn2 \in VN$ are found. From the output transitions tuple for *v1* the transition corresponding to the chosen connection is found, from the input positions tuple for *v2* the position corresponding to the selected connection is found. Selected output transition and input position are connected with an arc:

$$pn[F] = pn[F] \cup \{(vn\_in[TOUT][e],\ vn\_out[PIN][e], 0)\,|\,\forall e \in E,$$
$$\exists\, vn\_in \in AVN,\ \exists vn\_out \in AVN,\ vn\_in[V] = e1, \quad vn\_out[V] = e2\}.$$

Figure 1 shows diagram translation example. Petri nets *P1, P2, P3, P4* match vertices *V1, V2, V3, V4*. They are interconnected with arcs *A1, A2, A3, A4, A5* corresponding connections *E1, E2, E3, E4, E5*. For *V1* and *V2* connection the arc is created, outgoing from the transition *t1* of the network *P1* and entering the position *p1* of the network *P2*.

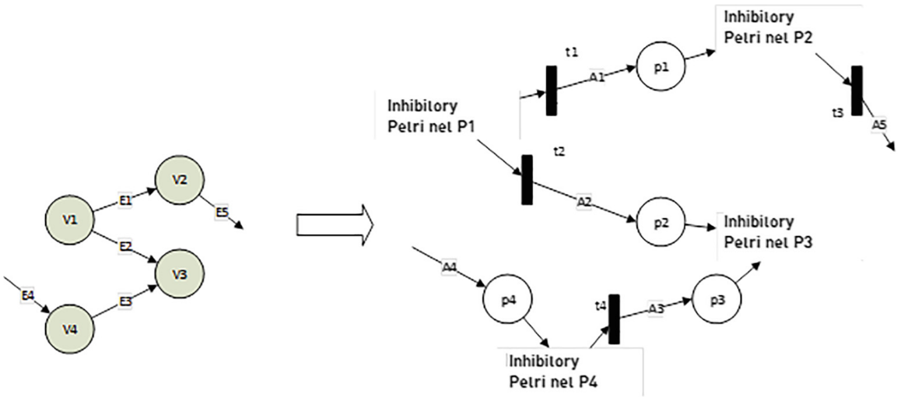


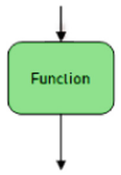
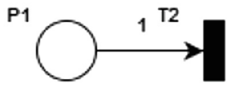
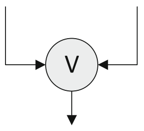
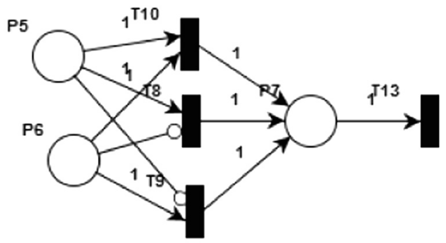
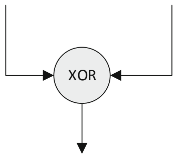
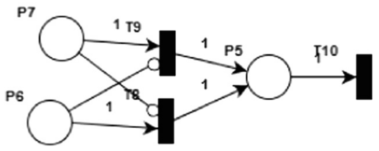
Fig. 1. Diagram to Petri net translation

4 EPC-Diagram Management Stream to Petri Net Translation

Event Chain Process (EPC) is type of diagrams used for modeling, analysis and reorganization of business processes. At the same time, EPC-diagrams can be used to simulate the behavior of individual parts of the system when implementing functions and serve as a replacement for traditional flowcharts (behavioral modeling). The EPC method was developed by Augusto-Wilhelm Scheer in the early 1990s.

Consider broadcasting EPC charts. The model for the EPC notation consists of many types of vertices TV = {Event, Function, Information, Document, File, Cluster, Object Set, Message, Product, Organizational Unit, Position, Artist, Location, Application, Module, AND, OR, XOR, Purpose, Term}, and many types of connections TE = {Control flow, Organizational flow, Resource flow, Information flow, Information service flow, Inventory flow}. The following types of vertices are directly responsible for the control flow: Event, Function, AND, OR, XOR; and communication type: Control flow. Thus, the flow of control of the EPC-diagram fits the limitations of one type of communication. We define the function of translating the top of the diagram into the extended Petri net model (see Table 1).

Table 1. EPC-diagrams elements to extended Petri nets translation

Element	Graphical symbol	Extended Petri net
Function		 PIN = (P1) TOUT= (T2)
OR operator (merging)		 P5 1 T10 1 1 T8 1 P7 1 T13 P6 1 1 T9 PIN = (P5, P6) TOUT= (T13)
XOR operator (merging)		 PIN = (P7, P6) TOUT= (T10)

For each diagram element, Petri nets are built in accordance with Table 1, and interconnected.

Since this approach uses semantic formalization of semi-formal languages due to the specialization of these languages, there are limitations.

It is necessary to abandon some of the features of the translated language, simplify it, give it new features, etc., which will have a positive impact on the formalization. Bring the language to a state where ambiguity disappears.

5 Implementation

The proposed method is implemented on the.NET Framework 4.5 platform for translating EPC diagrams from Microsoft Visio 2017 to the Petri network for the Platform Independent Petri Net Editor.

The algorithm of the program consists of the following steps.

- connect to a running copy of Microsoft Visio;
- read the EPC-diagrams elements and relationships between them;
- consistently convert each element of the EPC-diagrams into an equivalent extended Petri net in accordance with the translation function of the diagram vertices;
- connect extended Petri nets into a single Petri net based on the original EPC diagram links;
- save the received Petri net in the Platform Independent Petri Net Editor format;
- show the received Petri net on the screen.

The developed tool has the following features: translation of EPC-diagrams built in Microsoft Visio, including the highlighted part of the diagram; output of the received Petri net to the screen with support for saving the image to a file, automatic positioning of elements, scaling and printing; save the Petri file in Platform Independent Petri Net Editor format.

The Fig. 2 shows the diagram in the visual language EPC, taken for translation example. In the Fig. 3 you can see the resulting equivalent of the original EPC diagrams in Petri.

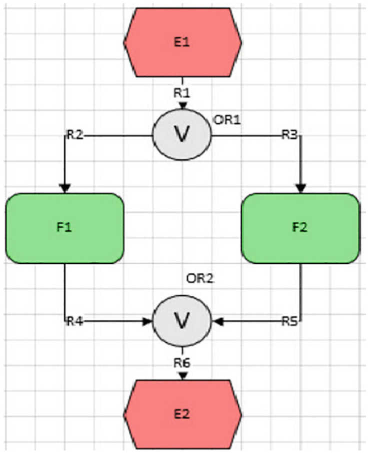


Fig. 2. The source EPC diagram in MS Visio

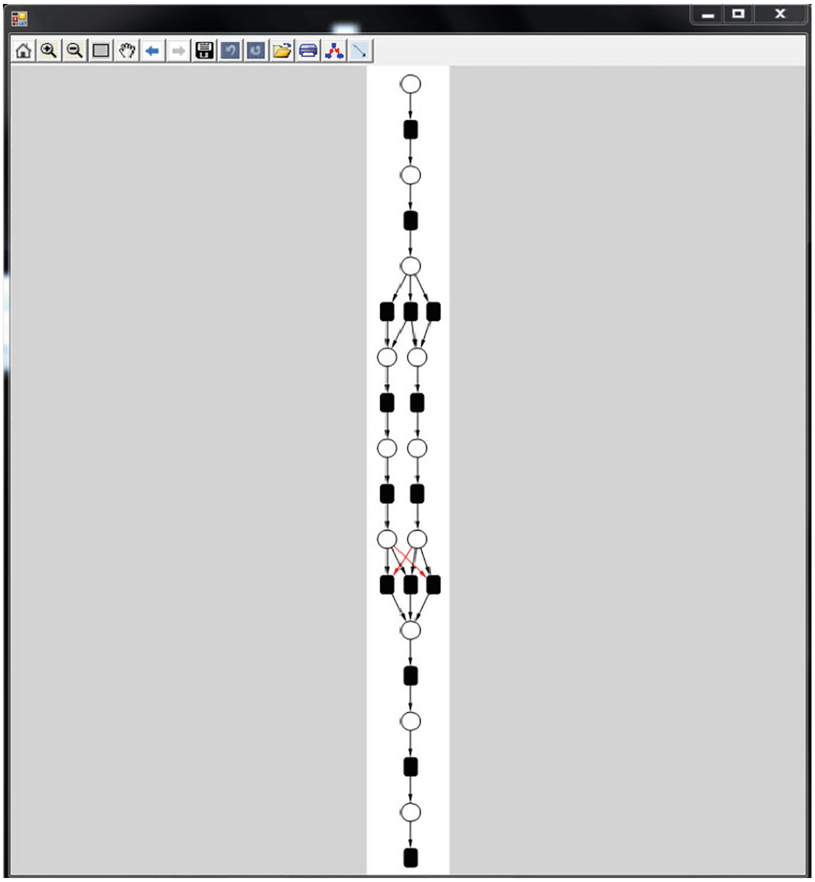


Fig. 3. The result diagram on Petri net

6 Conclusion

The article deals with the task of translating various types of diagrams into the inhibitor-Petri nets. A universal method of translation of a diagram with one type of directional communication is described, the models of a diagram, an inhibitor Petri net, and a translation algorithm are described. The application of the method for translating the flow of control of an EPC diagram consisting of 8 types of elements into a Petri net is shown, and examples of the translation of various diagrams are given. As an experiment, the algorithm was tested in other most well-known visual languages of business processes, such as BPMN, IDEF3, and successfully achieved its goal.

Acknowledgements. The study was supported by the Russian Foundation for Basic Research No. 17-07-01417 and the regional support of the Russian Foundation for Basic Research, grant No. 18-47-730032.

References

1. Markov, A.V., Romannikov, D.O.: Algorithm for automatic translation of the activity diagram into the Petri net. Rep. Acad. High. Educ. Russ. Fed. **1**, 104–112 (2014)
2. Voevoda, A.A., Markov, A.V.: CAD methods for complex systems based on the combined use of UML-diagrams and Petri nets. Modern technologies. Syst. Anal. Model. **2**(42), 110–115 (2014)
3. Markov, A.V.: Software development by sharing UML diagrams and Petri nets (overview). Scientific papers oft he Novosibirsk Stet Technical University **1**(71), 96–131 (2013)
4. Chernenok, S.A., Nepomnyaschiy, V.A.: Analysis and verification of MSC-diagrams of distributed systems using expanded Petri nets. Model. Anal. Inf. Syst. **6**(21), 94–106 (2014)
5. Ramadan, M., Elmongui, H.G., Hassan, R.: BPMN formalisation using coloured petri nets. In: Proceedings of the 2nd GSTF Annual International Conference on Software Engineering & Applications (SEA) (2011)
6. Fedorov, I.G.: The method of displaying the executable model of a business process on the Petri net. Stat. Econ. **4**, 179–183 (2013)
7. Afanasyev, A.N., Voit, N.N., Ukhanova, M.E., Ionova, I.S., Epifanov, V.V.: Analysis of the design and technological workflows in a large radio-technical enterprise. Radiotechnics **6**, 49–58 (2017)
8. Afanasyev, A.N., Voit, N.N.: Intelligent agent-based system for analysis of design work flow models. Autom. Control. Process. **4**(42), 52–61 (2015)
9. Afanasyev, A.N., Voit, N.N., Gainullin, R.F., Brigadnov, S.I., Horodov, V.S., Sharov, O.G.: RV grammatics metacompiler. Proc. Ulyanovsk State Tech. Univ. **4**(76), 48–52 (2016)
10. Afanasyev, A.N., Voit, N.N.: Grammar-algebraic approach to the analysis and synthesis of diagrammatic models of hybrid dynamic flows of design works. Inf. Meas. Control Syst. **12** (15), 69–78 (2017)
11. Afanasyev, A.N., Voit, N.N., Ukhanova, M.E.: Monitoring and analysis of denotative and significative semantic errors of diagrammatic workflow models in the design of automated systems. Radiotechnics **6**, 84–92 (2018)
12. Voit, N.N.: Methods and tools for automating workflow design. Inf. Meas. Control Syst. **11** (14), 84–89 (2018)