Paradigms and graphical-analytical tools for building simulation tools and forming the architecture of a combined approach to studying the dynamic properties of systems with parallelism are described. An extension of the formal language of Petri nets is presented, which has greater modeling power than WF nets. The properties of hierarchical Petri nets are used to synthesize a holistic model.

Discrete-event modeling and modeling of dynamic systems, which allow reflecting the quantitative and qualitative characteristics of the elements of the systems under study, served as the basis for the combined approach to the simulation of systems with parallelism. On their basis, graphic-analytical tools are proposed that provide the ability to describe the modeled system, adhering to the principle of structural similarity. They have dynamic simulations that make it easy to visually analyze and correct the model. Also, the proposed toolkit provides for the analysis of the dynamic properties of the model, which makes it possible to identify accumulated phenomena that can lead to unpredictability of the system's functioning.

A conceptual model for the synthesis and analysis of systems with parallelism is proposed, which provides for the construction of the components of the model based on the architecture. Their step-by-step analysis and the formation of an integral model of the software system are carried out using a network representation, according to the matrix description of which invariants are calculated. The analysis of invariants allows one to obtain the dynamic properties of the model and determine the localization of structures that lead to critical situations when they are detected.

The architecture of the combined approach to the simulation of systems with parallelism is built, which provides the study of their dynamic properties to improve the reliability of the functioning of software systems

Keywords: architecture of the combined approach, formal languages of Petri nets, dynamic modeling tools

UDC 004.942

DOI: 10.15587/1729-4061.2021.239212

COMBINED APPROACH ARCHITECTURE DEVELOPMENT TO SIMULATION MODELING OF SYSTEMS WITH PARALLELISM

Oksana Suprunenko
PhD, Associate Professor
Department of Software for Automated Systems
Bohdan Khmelnytsky National University of Cherkasy
Shevchenka blvd., 81, Cherkasy, Ukraine, 18031
E-mail: ra-oks@vu.cdu.edu.ua

Received date 29.06.2021
Accepted date 16.08.2021
Published date 30.08.2021

How to Cite: *Suprunenko, O. (2021). Combined approach architecture development to simulation modeling of systems with parallelism. Eastern-European Journal of Enterprise Technologies, 4 (4 (112)), 74–82. doi: https://doi.org/10.15587/1729-4061.2021.239212*

1. Introduction

In the context of the growing complexity of software systems [1], the task is to make sound design decisions at the stages of design, debugging and maintenance of software projects. The complexity of software, in particular the structural and functional complexity, due to the presence of numerous parallel and competing processes that interact in the operation of the software product. For studying the static and dynamic properties of software systems with parallelism, one of the most powerful research tools is simulation. Its means make it possible to describe the systems under study at various levels of abstraction, taking into account their quantitative and qualitative characteristics [2].

When describing and analyzing software systems, several models are often built [3], reflecting various aspects of the system under study. When the conclusions made on the basis of these models are contradictory or incomplete, it becomes necessary to build a combined model, combines the means of various methodologies of simulation modeling [3, 4]. The toolkit, on the basis of which it is possible to build combined models, combines several methodologies, poorly developed [5, 6]. So, when designing software systems, models are used that describe various aspects of their construction and functioning. For example, UML diagrams [7], which are used to represent and analyze design decisions, allow describing structural aspects in diagrams of classes, packages, components, etc. Behavioral aspects are described by diagrams of activity and automata, as well as diagrams of use cases. The dynamic aspects of interaction are represented in sequence diagrams, communication, interaction, and synchronization. But the absence of dynamic simulation tools, as well as the inconsistency of the levels of detail, make it impossible to analyze the dynamic properties of components and the entire system, and is a problem in software design [6]. Therefore, the urgent task is to develop a unified instrumental and methodological basis for combined simulation models for the study of complex systems with parallel and competing processes.

2. Literature review and problem statement

When designing software tools, a set of diagrams and diagrams is used [7], for example UML diagrams, each of which reflects certain aspects of the software being developed. But the specificity of diagrams, which does not provide for visual dynamic simulation when analyzing the functioning of a unified software model, makes it difficult to evaluate and correct a software project. This may lead to not detecting a certain part of errors or design solutions, which in the future may cause the emergence of unpredictable functioning of the software product. Therefore, unresolved issues are related to the means of representing and analyzing the general model of a software system, primarily due to the large dimension of this problem and the heterogeneity of diagrams within one approach, and sometimes the use of diagrams developed in dif-

ferent approaches. But in the works of Hoare and Dijkstra it is noted that software must have «well-defined behavior» [7] and can be analyzed as mathematical objects. This idea is used in [8] to model workflow control based on WorkFlow interpretation of Petri nets (WF nets). But WF networks have significant limitations that do not allow them to be fully used in modeling and analyzing models of software systems. Therefore, it is advisable to conduct research on the development of an approach and tools for modeling and analysis of systems with parallelism, which include software systems.

To determine the basis for the development of an approach to modeling software systems, let's consider simulation modeling, to which the toolkit based on WF networks belongs. Simulation modeling allows an order of magnitude, and sometimes more, to improve the efficiency of solving applied problems [5]. According to modern research, simulation has four paradigms [6]: simulation of dynamic systems, discrete-event (process-oriented) simulation, system dynamics, and agent-based simulation. In the simulation of dynamic systems, the system under study is described by a set of state variables and algebraic analogs of differential equations of various types, assigned for the variables and describing their transformations in time. Variables most often have physical meaning and are continuous. Simulation of dynamic systems refers to the concept of quality simulation [4].

Discrete-event modeling allows one to describe the main states of the system and the conditions for the transition from current to next states, taking into account complex cause-and-effect relationships [8]. This simulation paradigm also makes it possible to reproduce the possible actions of the system in the middle and in the environment at successive points in time. It is used in applied industries, transport systems, etc. [4].

Using the system-dynamic approach, one describes the accumulation of characteristics of the studied processes, characterized by nonlinearity, which is due to the effects of delay [4]. In models of system dynamics, generalized quantitative characteristics of flows act as variables, which determines the belonging of system dynamics to the concept of quantitative modeling. System-dynamic models are used to solve the problems of strategic management of large organizational systems; they are used only at the macro level and do not have the means for a detailed description of complex systems. In contrast to system dynamics [4], discrete-event modeling tools (Fig. 1) allow describing the cause-and-effect relationships between model elements both at the meta-level and at the macro- and micro-level of modeling [4, 9]. These tools reflect the quantitative and qualitative characteristics of the elements of the system, translating them into associative-numerical form.

Agent-based modeling intended for the study of decentralized systems, its difference is the consideration of system objects as active entities with their inherent behavior. Also, agent-based models are adaptive, which makes it possible to display the adaptation of the model to changes in external conditions [4]. Models in this paradigm of simulation modeling are built according to the «bottom-up» principle, therefore, when constructing them, it is more difficult to take into account the general characteristics of the system being modeled, in particular when designing a software system.

From common tools of simulation modeling, a special group is formed by the graph of tools [2], which allow to describe the system at a certain point in time and simulate its behavior under the influence of external factors. These tools relate to the paradigm of discrete-event modeling and allow to describe the states into which the system passes under the influence of certain factors, the characteristics of these states and the transition conditions. Another paradigm of simulation modeling [4] – modeling dynamic systems – allows modeling a limited number of states (Fig. 1), which is a limiting factor in modeling systems with parallelism. On the other hand, messages of discrete-event modeling and modeling of dynamic systems allow to combine graphical-analytical properties in tools. They combine a graphical display of the dynamics of the functioning of the modeled system with an unambiguous mathematical description of its structure and dynamic features, which is used to determine and localize the unpredictable behavior of the model. Thus, the toolkit created on the basis of these two paradigms allows reproducing the behavior of the model over time and calculating the dynamic characteristics of the model. This approach helps to identify accumulated phenomena that manifest themselves only under certain conditions, but can lead to unpredictable behavior, the identification of which is a difficult problem when analyzing models of parallel and distributed software systems [9].

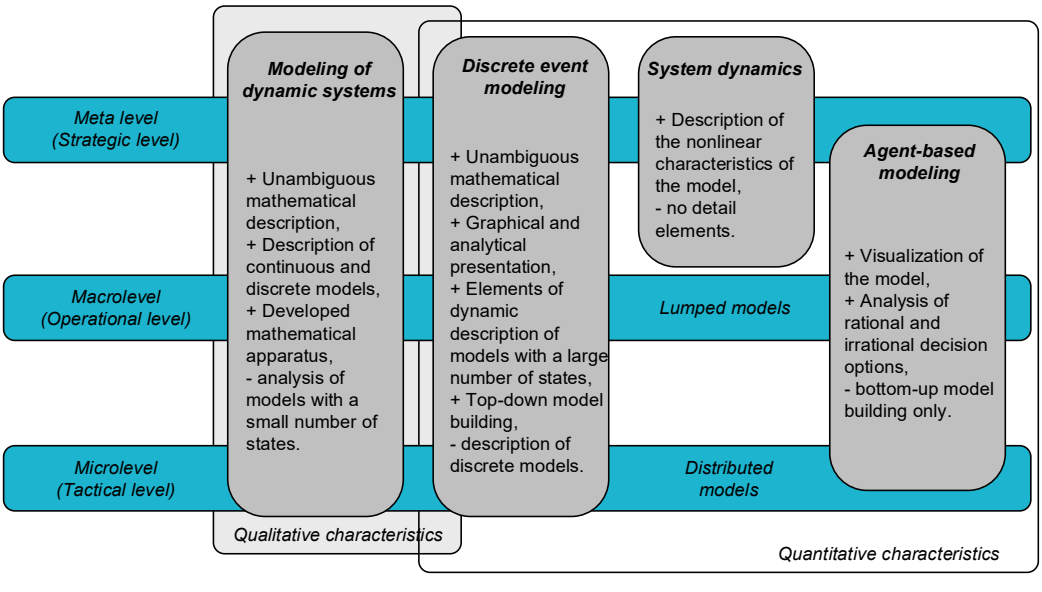

Fig. 1. Advantages and disadvantages of simulation paradigms for problems of modeling systems with parallelism

One of the representatives, built on a combination of discrete-event modeling and modeling of dynamic systems, is the theory of Petri Nets (PN) [10]. It allows not only to describe causal relationships in models with parallel and competing processes, but also to conduct research on models [6, 11]. Petri nets combine a convenient graphical representation with a mathematically rigorous description of the model [12, 13], which determines their use in the automated analysis of the constructed model (critical properties, exceptional situations).

An important characteristic of a software system is the predictability of behavior and the performance of all its elements. To achieve all its characteristics, limited Petri nets are used, to which safe, estimation Petri nets, WF nets (Work Flow Petri Nets, WF) and temporary Petri nets belong [14]. For these interpretations and modifications of Petri nets such properties as liveliness (reachability), preservation and predictebility are proved [8]. For the arrangement of the constituent systems with parallelism into a single model, the proposed nested [15] and interacting [16] Petri nets. But the mechanisms for simulating dynamic characteristics in nested Petri nets are rather complicated. For example, when simulating a model as a whole, it is difficult to control active streams of events, since it is necessary to move from one level of detail to another. For interacting Petri nets, in particular for their most studied WF-interpretation, the liveliness condition has not been proven [16].

In [17], an approximate proof of resource bisimulation for one-counter Petri nets was proposed, which was developed in [18] for Petri nets with λ-transitions. This idea can be used to reduce the dimension of a parallelism model to explore it analytically. But the use of λ-transitions in the tools for graphical display of the dynamics of the functioning of the model negate the advantages of visual analysis. Therefore, in the developed approach and modeling tools, it is necessary to provide a description of the auxiliary functions of the model, which are usually described by λ-transitions, in an explicit form.

In [19], the lack of time tracking is attributed to the disadvantages of modeling corporate applications with Petri nets, but time is of great importance in imitating the functioning of these systems. Also, the authors of [19] noted as a disadvantage of applying the rules of Petri nets of free choice, when several transitions can be activated simultaneously. To solve these problematic moments, the authors of [19] propose a Petri nets with Markov chains and queuing theory. But the potential of analytical research of simulation models of the theory of Petri nets is not considered.

Some modifications of bounded Petri nets have been well studied, but need adaptation to describe software systems. Thus, the most common WF networks [11] are used to describe workflows, but they do not allow solving all the problems that stand in the analysis of models of software systems. In particular, they do not support constructions like «critical section access control» and «bounded buffer resource transfer» [14] in the producer/consumer problem. This is due to the absence of controlling vertices of the place, which must be marked up in the initial marking, which contradicts the definition and basic properties of WF networks [8].

When working with a model in the process of its analysis, it is necessary to show localization and the cause of a conflict situation, which requires appropriate means. When designing software systems [6, 7], the description and visual analysis of design solutions is carried out in diagrams and diagrams. But the absence of a dynamic component in them, which makes it possible to reproduce the dynamic features of functioning in the analysis, complicates the situation. Therefore, in the formation of tools for dynamic modeling of systems with parallelism, it is necessary to improve the tools for simulation modeling.

Thus, tools for the description and analysis of models of software systems that allow processing large-dimensional models require further development. It is necessary to form an approach that combines the means of graphical construction of models of systems with parallelism, their analysis and correction on a single representation. For simulation and visual analysis, the WF network needs to be expanded and the mechanisms of visual analysis of models need to be improved.

3. The aim and objectives of research

The aim of research is to develop the architecture of a combined approach to the simulation modeling of systems with parallelism. This will make it possible to determine the dynamic characteristics of the components and the system with parallelism as a whole to improve the reliability of the functioning of software systems.

To achieve the aim, the following objectives have been set:

– to describe the features and limitations of formal languages of Petri nets for the construction of tools for the combined approach of simulation modeling of systems with parallelism;

– to analyze the properties of bounded Petri nets to analyze the dynamic properties of simulation modeling of systems with parallelism;

– to form the principles and conceptual model for the development of the architecture of a combined approach to the simulation modeling of systems with parallelism.

4. Materials and methods of research

Developing an approach and tools for modeling systems with parallelism requires identifying the potential and limitations of existing simulation tools. On their basis, a combined approach to modeling software systems with parallelism is formed, which, in a single presentation, takes into account the specifics of describing the structure and analyzing the functioning of the systems under study.

In the theoretical study, the paradigms of simulation modeling were used, which made it possible to form the basis for a combined approach to modeling systems with parallelism based on modeling dynamic systems and discrete-event modeling. The theory of formal languages of Petri nets is used to determine the potential and limitations of interpretations and modifications of Petri nets, used in the development of simulation tools for systems with parallelism. The theory of computer systems has made it possible to formalize the main stages of the study of models of software systems with parallelism. Algebra of processes and a component-oriented approach to the design of software systems provided an opportunity to build architecture of a combined approach to the simulation modeling of systems with parallelism.

Research hypothesis: the extension of WF nets with the properties of limited interpretations and modifications of Petri nets will allow building structural-like models of systems with parallelism, taking into account qualitative and quantitative parameters. The matrix representation of constituent systems with parallelism bounded Petri nets can be

used to determine the dynamic properties of network models. Analysis of the markup during simulation will improve the efficiency of the analysis of the dynamics of the functioning of network models.

5. The results of the study of the soil and the development of the architecture of the combined approach for the simulation modeling of systems with parallelism

5. 1. Formal languages of Petri nets, their features and limitations

Petri nets (PN) generate a class of formal Petri net languages. The language of Petri nets $\gamma=(PN, \Sigma, \sigma, s_0, M_F)$ is represented by a set of strings over the alphabet S of symbols denoting a finite set of transition vertices by the function $\sigma: T\to\Sigma$, as well as the initial state s_0 and the set of final states M_F. Such a Petri net is called marked.

The alphabet of the languages of Petri nets is connected with the vertices of the transitions, so the marked function assigns the vertices of the transitions to the symbols of the alphabet of the formal language: $\sigma: T\to\Sigma$. The sequence of starting transitions of the net generates a string of the language of Petri nets. Depending on the set of final markings, formal languages of L, G, T and P-type Petri nets are distinguished [14].

The languages of Petri nets are based on the principle of strict monotony [20], since if the transition vertex is permissible for marking μ_i, and it will be allowed for marking $\mu_j>\mu_i$, and if $\mu_i \xrightarrow{t} \mu'_i$ even $\mu_j \xrightarrow{t} \mu'_j$, then $\mu'_j \geq \mu'_i$. This indicates the finiteness of the covering tree for the Petri net, as well as development of problems of coverage (liveliness), support of the governing state and limitation.

The modification of WF nets is described by the class of formal languages of L-type Petri nets (in the language of completely finite Petri nets) [14], which is the most limited. The class of L-type languages is closed in relation to such important types of composition for modeling as concatenation, union, intersection, inversion, parallel composition and regular substitution [20]. Therefore, Petri nets, described by formal L-type languages, are promising for creating tools for modeling software systems.

But in Petri nets, which are described by L-type languages, of the model firing from an arbitrary markup that corresponds to the intermediate state of the target system is not provided, as well as safe access to the critical section and similar constructions is not specified. Classes of languages of Petri nets are closely related to each other, in particular, any language of G-type is included in the class of languages of L-type $G\subseteq L$ intermediate markings. G-type languages (a class of free languages for Petri nets) provide for working with a set of any initial markings, and also have no restrictions on final markings. Also, the G-type class reflects the hierarchical Petri nets [10], which allow to obtain a model by merging its individual fragments. This is an important property that is used in the design of software systems. An extension of the language of L-type Petri nets by elements of the G-type language is the L_G-language describing the Petri net $PN=(P, T, K, Q, R)$ with the addition of the transition vertices $\sigma: T\to\Sigma$, initial marking μ_0, intermediate marking μ' and a set of final markings M_F, such that

$$L_{L_G} = \left\{ \begin{matrix} \sigma(t_j)\in\Sigma^* \,\middle|\, t_j\in T \wedge \\ \wedge\left(\delta(\mu_0,t_j)\vee\delta(\mu',t_j)\right)\in M_F \end{matrix} \right\}. \quad (1)$$

Thus, language of L_G-type Petri nets makes it possible to carry out modeling, starting with any marking that refers to a finite set of final markings of the M_F net. Also Petri nets, which are described by L_G-type languages, can have a finite set of control vertices, the marking of which has to be restored at the end of the model simulation session. Such models can be converted to models, described by formal L-type languages, but such models will violate the structural similarity in relation to the simulated systems, complicate their analysis. Also, the converted models will have a significantly larger number of elements, increasing the complexity of calculating their dynamic properties.

5. 2. Properties of bounded Petri nets for analyzing the dynamics of systems with parallelism

Bounded Petri nets include [14] safe, estimation and temporary Petri nets, WF nets (Work Flow Petri Nets, WF). The properties of Petri nets are divided into static and dynamic [13]. Static properties of Petri nets associated with the structural features of their construction. When analyzing the static properties of a subclass of estimation Petri nets – control Petri nets – the structure of a Petri net is considered together with control functions [13].

When checking the dynamic properties of Petri nets, the functioning of the model is considered and the analysis of complete and partial sequences of transitions is carried out. Dynamic characteristics include [8, 13] liveliness, reachability, safety (boundedness), preservation, conflictlessness (defectlessness for WF networks).

In the works of Heck [21], the properties of liveliness and safety were considered, in particular, the concept of liveliness was associated with the absence of dead locks, and the concept of safety was associated with the absence of unpredictable functioning of the model. The problems of dead locks and unpredictable behavior in network models of production systems have been associated with free choice constructs. That is, when the vertex of the place is the input one up to several vertices of transitions, a conflict situation arises, which was solved [21] by providing a choice of any of the vertices of the transitions for subsequent fired.

Definition 1. A Petri net is live when it does not have non-living markings for any possible sequences of firing the vertex of transitions σ_i, that is $\mu_i \notin M_{max}$.

The problem of proving the reachability property can be reduced to proving liveliness [21]. The proof of the reachability property is important, since most of the problems of analyzing Petri nets can be reduced to it [14].

Definition 2. The reachability problem for a Petri net PN with a marking μ is to determine the reachability of a marking μ', that is, $\mu'\in M(PN, \mu)$.

The reachability of the marking μ_n with some initial marking $\mu 0$ is ensured by the existence of successively marked transition vertices $\sigma=t_1, t_2, t_3, ..., t_n$, as a result of which the marking μ_n is achieved, which can be written as $\mu_0[\sigma>\mu_n$. The set of all possible reachable markings with μ_0 $M(PN, \mu_0)$ corresponds to the set of all possible sequences $W(PN, \mu_0)=\{\sigma_1, \sigma_2, ..., \sigma_s\}$.

The safe property of Petri nets concerns the limitation of the number of markers that can be at the top of a place at the same time. The safe condition for safe Petri nets requires that there are no more than one marker at the site vertex, that is, $W(PN, \mu_0)=\{\sigma_1, \sigma_2, ..., \sigma_s\}$. Safety Petri nets are used to describe in detail parallel processes, for example, in the case when it is necessary to control so that the next set of processes

can occur only after the end of the previous one. Safety in the class of safe Petri nets is a special case of the boundedness property, which provides for the establishment of a certain restriction on the maximum number of markers that are simultaneously in the investigated vertex of the place.

Definition 3. A vertex of a place $p_\varepsilon \in P$ of a Petri net with initial marking μ_0 is k-bounded (k-safe) if $\forall \mu' \in M(PN, \mu_0)$: $\mu'(p_\varepsilon) \le k$.

In safe Petri nets, all vertices of a place are 1-bounded. In the estimated Petri nets, vertices of places with different restrictions are allowed; such a network is called k-bounded [13] behind the top of the place, which has the largest restriction $k=\max(k_1, k_2,..., k_n)$. The boundedness property of Petri nets makes it possible to model with their help processes that depend on quantitative indicators, for example, the accumulation of certain resources.

The reachability problem can be formulated as the reachability problem for a dead-end marking, let's demonstrate this using the example of a model built by an estimation Petri net (Fig. 2, *a*). With the initial marking $\mu_0=[1, 0, 0, 0]^T$, transitions in the sequence $\sigma_1=t_1, t_2, t_3, t_2, t_4$, will be triggered sequentially, which will allow restoring the original marking. With the initial marking $\mu_{01}=[5, 2, 1, 1]^T$, the transition firing sequence is:

$$\sigma_{01} = t_0,t_2,t_2,t_3,t_2,t_1,t_0,t_2,t_2,t_3,t_0,t_0,t_2,t_0,t_1,t_0,t_0,$$
$$t_1,t_3,t_0,t_1,t_2,t_2,t_2,t_1,t_2,t_1,t_1,t_2,t_1,t_3,t_2,t_1,t_2,t_1,t_2,$$
$$t_0,t_3,t_2,t_0,t_2,t_1,t_1,t_1,t_2,t_2,t_2,t_2,t_2,t_2,t_2,t_2,t_2.$$

will lead to the marking $\mu_{1t}=[0, 0, 0, 31]$, which corresponds to the dead lock state, since with such marking, neither the vertex of the transition to the model can be fired. Thus, it is important to analyze the options for the functioning of the model and identify signs of a dead lock, especially in cases of implicit dead lock (traps). This problem can be solved by calculating the T- and P-invariants based on the basic equation of Petri nets [22], as well as analyzing the rank of the incidence matrix of the model.

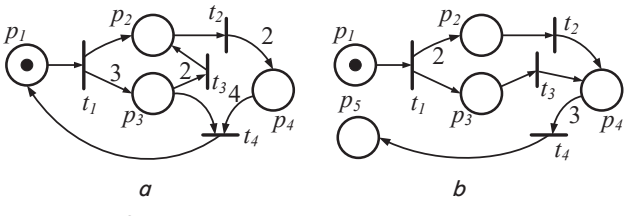

Fig. 2. Model based on the estimated Petri net

The solution to the system of linear Diophantine equations [22]: $W^T X=0$ is the vector X called the T-invariant. It ensures the existence of sequences of transitions $\sigma=t_1, t_2,..., t_n$ from the marking μ_0, which leads to the same initial marking. The T-invariant, which covers all elements with nonzero values, corresponds to the properties of liveliness (repeatability) and raechability. P-invariants find a decoupling of the system of linear Diophantine equations: $WP=0$, when all elements of P-invariants are covered with nonzero values, a correspondence is established to the boundedness and preservation properties of the model.

To describe the structure of the network, the incidence matrix W is used, which consists of columns that correspond to the vertices of transitions and rows that correspond to the vertices of places. When calculating the rank of the matrix W, the complete or incomplete controllability of the model is determined (provided that the T- and P-invariants are completely covered by nonzero values). If the rank of the incidence matrix is less than the smallest of the cardinalities of the sets of transition vertices $|T|$ or tops of places $|P|$ rang $(W)<\min(|T|, |P|)$, then the model is not completely controllable, otherwise, if these indicators are equal, the model will be completely controllable.

Calculation of invariants for the model (Fig. 2, *a*) had the following results: $T_{2a}=[1, 2, 1, 1]^T$; $P_{2a}=[5, 2, 1, 1]^T$; the rank of the incidence matrix: rang (rang $(W_{2a})=3<\min(|T_{2a}|,|P_{2a}|)=4$. Thus, although the T and P-invariants are covered by completely nonzero elements, the rank of the incidence matrix is less the value $|T|$ than maybe a conflict situation that can be caused by the construction of free choice from the vertex of the place p_3. So, with appropriate marking, the markers from the top of the place p_3 with a free choice of further path can activate the top of the transition t_4 and never start the top of the transition t_3. To eliminate unpredictable functioning, it is necessary to rebuild the model so as to preserve the number of triggering of the transition vertices (active actions), that is, the value of the T-invariant $T_{2b}=[1, 2, 1, 1]^T$. An example of the rebuilt model is shown in Fig. 2, *b*, where the structure of the model is refined to ensure double activation of the top of the transition t_2 with a single actuation of the top of the transition t_3 and the structure of free choice, which can lead to a dead lock, is eliminated. When calculating the invariants and the rank of the incidence matrix, the following results were obtained:

$$P_{2b} = \begin{bmatrix} 3 & 1 & 1 & 1 & 3 \end{bmatrix}^T;$$

$$W_{2b} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 2 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ \text{rang}(W_{2b}) = 4.$$

Thus, the invariants are completely covered by nonzero values, as well as rang $(W_{2b})=4=\min(|T_{2b}|,|P_{2b}|)$, which indicates complete controllability and compliance with the properties of liveliness, reachability, safety and preservation.

One of the important characteristics of bounded Petri nets, which ensure its reliable functioning, is preservation. Each vertex of the place of Petri nets when modeling applied systems is k-bounded, that is, it has a certain maximum markup value that can be achieved in it, taking into account the number of marks that remained from the previous steps and the number of marks that are added at the current stage: $\#\left(p_\eta^0, I_R(t_h)\right) + \#\left(p'_\eta, I_R(t_h)\right) \le \xi$.

Naturally, absolute preservation is achieved in automatic Petri nets, in which each transition has one input and one output arc [10], the number of markers in such a network does not increase and does not decrease during its operation. Also, absolute preservation can be achieved provided that the same number of input and output arcs are provided for each vertex of the Petri net transition.

Definition 4. A Petri net PN with initial marking μ_0 is called strictly preservation if for all its markers μ' the following condition is satisfied:

$$\sum_{p_i \in P} \mu'(p_i) = \sum_{p_i \in P} \mu(p_i). \quad (2)$$

This condition imposes very strict restrictions, but when modeling most applied problems, the number of markers in

the model can increase and decrease during its operation. Therefore, the preservation property of such models is often evaluated not by the strict notion of preservation, but by the weighted vector $\bar{\omega}$ of positive integers ($\omega_i \in Z^+$) defined for each marking. Vector elements allow to display the number of marks that are simultaneously moving in the Petri net. For example, in Fig. 3, *a* a Petri net is presented, in which it is possible to balance the number of markers in each marker, such a Petri net can be called loosely persisted or persisted with respect to the weighting vector $\omega=\{\omega_1, \omega_2,..., \omega_n\}$.

Definition 5. Petri net PN with initial marking μ_0 is called preserved with respect to the weighting vector $\omega=\{\omega_1, \omega_2,..., \omega_n\}$, where $n=|P|$ and $\omega_i \in Z^+$, if the condition is satisfied for all markings μ':

$$\sum_{i=\{1,2,...n\}} \omega_i \mu'(p_i) = \sum_{i=\{1.2,...,n\}} \omega_i \mu(p_i). \quad (3)$$

For the Petri nets persisted with respect to the weighting vector $\bar{\omega}$, it is important that the initial and final markings coincide [21], which can be clearly demonstrated when transforming it into a closed Petri net.

An example of Petri nets preserved with respect to the weighting vector, which is able to restore the original marking countless times, is a closed Petri net, an example of which is shown in Fig. 3, *b*. The closed non-strictly preserved Petri net restores the original marking in six steps, that is, $\mu_0=\mu_6$.

The weighing vectors shown in Fig. 3, *a*, *b* for variants of loosely preserved Petri nets, allow maintaining constant the product $\omega_i\mu_i$, which determines the preservation property of the reduced models. For automated verification of this property, the variant of the closed network model (Fig. 3, *b*) has advantages, since it is easier to implement.

The languages of Petri nets are closed with respect to the finite substitution [13, 14], which makes it possible to use analyzed and debugged local models and construction primitives for modeling complex systems, in the models of which a certain vertex of a place can be replaced by a corresponding local model or a certain template [14]. Replacement is performed by merging the initial vertex of the place of the embedded area, for example, the vertex with the vertex of the place p_3, into which the final substitution is performed, and the final vertex of the place, the segments with a duplicate of the same vertex p_{31} (Fig. 4).

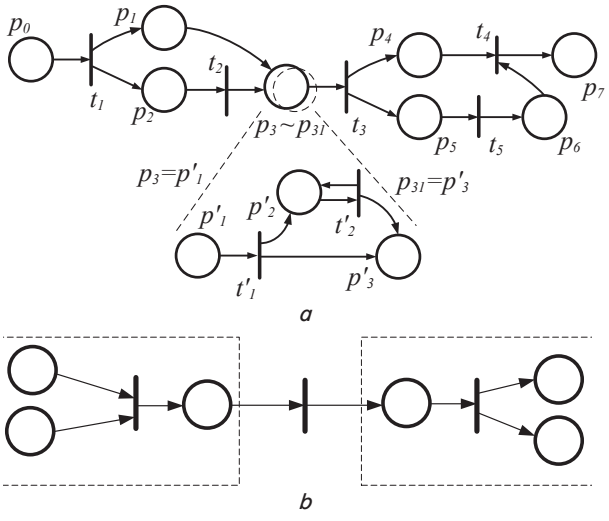

Fig. 4. Variants of assembling local models:
a – final substitution into the vertex of the place p_3;
b – serial connection

Also, when assembling debugged models, the design shown in Fig. 4, *b*, if this corresponds to the logic of building the model. When the embedded submodel has an initial vertex of a location vertex, and its final vertex is a transition vertex (as the p_0–t_6 model in Fig. 3, *b* is open), then it is embedded according to the accepted method of constructing Petri nets on a linear section between the corresponding transition vertex and the subsequent adjacent location vertex.

5. 3. Conceptual model and combined approach to simulation modeling of systems with parallelism

In the formation of the architecture of the combined approach to the simulation modeling of systems with parallelism (Fig. 5), the theoretical foundations were two paradigms of simulation – discrete-event simulation and simulation of dynamic systems.

The conceptual apparatus of the approach is made up of parallel and competing processes, formal languages of L-type and G-type Petri nets, which describe bounded Petri nets. Also referred to the concepts are PN-patterns and tools, which are a combination of safe, estimation and control Petri nets for building design features of the model and interconnections. Let's call each subordinate model a submodel, and the model into which submodels are gradually assembled – a partial model of the system. The sequential collection of submodels into a partial model with a convolution of submodel elements will be called reduction.

To analyze the constructed components of the model (submodels) and the model as a whole, simulation modeling and a matrix representation of their structure [13], the method of invariants [22] are used. To determine the dynamic properties of the investigated submodels and partial models of the system with parallelism, which are formed at each stage of model collection, the invariants and the rank of the incidence matrix are calculated and analyzed.

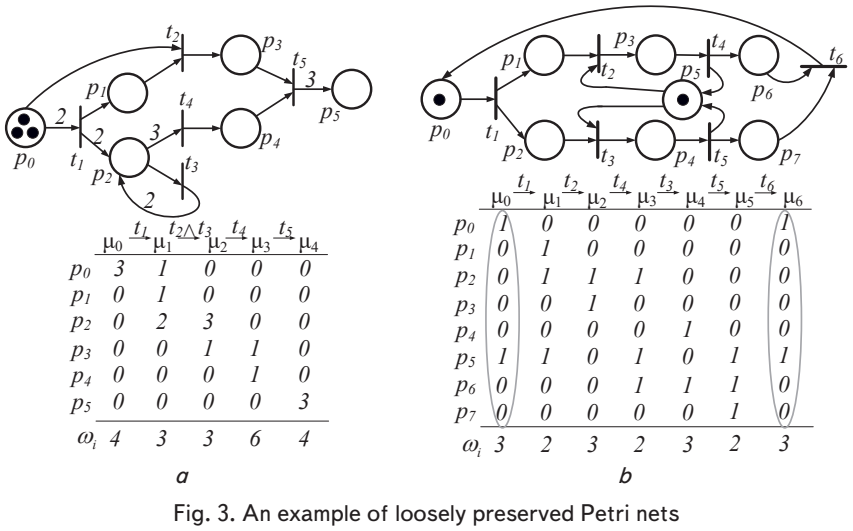

Fig. 3. An example of loosely preserved Petri nets

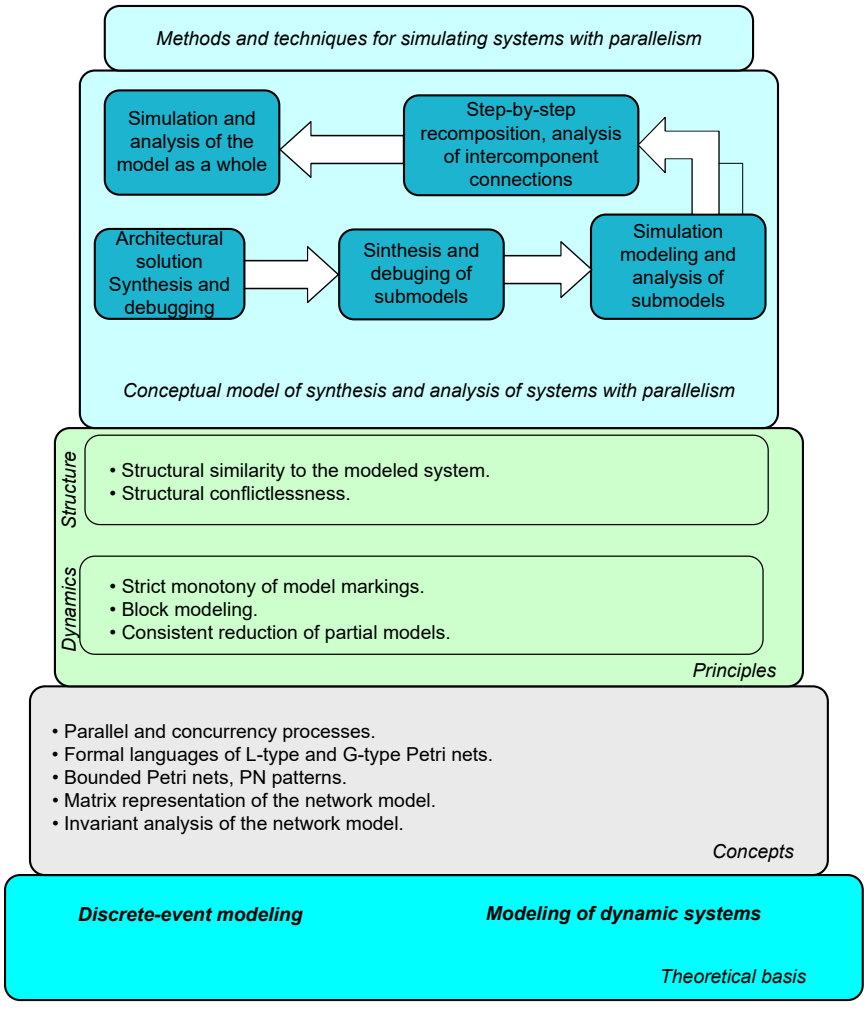

Fig. 5. Architecture of the combined approach to the simulation modeling of systems with parallelism

The elements of the combined approach to modeling systems with parallelism are two groups of principles. The first group includes principles related to the structural structure of the model:

– principles of structural similarity;

– the principle of structural conflictlessness [23].

The second group includes the principles that determine the features of modeling and analysis of the dynamic properties of the model:

– the principle of strict monotony of model markings;

– the principle of block modeling;

– the principle of sequential reduction of partial models.

Thus, the principle of structural similarity greatly simplifies the verification of a model in a mixed mode (analytically and visually), the principle of structural conflict-free allows to correct errors in building a model based on a list of rules [23]. The principle of strict monotonicity of markings makes it possible to unambiguously describe the functioning of the model and check important properties of the system, such as controllability and boundedness. The principle of block modeling, used in the algebra of processes [24], hierarchical networks [13] and the component-oriented approach to the design of software systems, allows the process of building and researching a simulation model of a system with parallelism in stages. The principle of reduction of partial models allows one to reduce the dimension of the investigated model for its analysis by the method of invariants.

The conceptual model of synthesis and analysis of systems with parallelism (Fig. 5) reflects a step-by-step process of building, testing and debugging a model with parallelism. At the beginning of the process of synthesis and analysis of a model of a system with parallelism, an architectural solution is built that reflects a general idea of the system at the time of setting the problem and will be refined. The next step is the construction and adjustment of submodels, which allows to obtain model components that consist of a relatively small number of elements and allow to prepare submodels for the analysis of dynamic properties on a network and matrix representation. Based on the configured submodels, their simulation is carried out, which allows to check the logic of the model's functioning, as well as the presence of potentially dangerous structures, for example, dead locks and traps.

A dead lock state in the model arises if the final markup is not achieved and, at the same time, there are no prerequisites for triggering any transition. It can occur, for example, when one process requires a resource, is currently being used by another process, or when there are insufficient resources to initiate a particular process. The trap will not lead to an explicit termination of the functioning of the model or to a state of dead lock, but the final result of the simulation will not be achieved.

After simulation, the submodels are analyzed on a matrix representation using the invariants method, which allows to check the main dynamic properties of submodels. Also, with invariant analysis, one can detect signs of potentially dangerous scenarios for the functioning of the model, for example, such as a hidden dead lock, the analysis of which is demonstrated on the model shown in Fig. 2.

At the fourth stage, to obtain a model of a system with parallelism, a phased recomposition of submodels into a partial model is carried out. At the same time, the analysis of inter-component connections is carried out, which is carried out on the matrix representation of the partial model using the method of invariants and the rank of the incidence matrix. The difference between this stage and the second stage is not only the object – a partial model, as opposed to a submodel, but also the need to convolve the elements of the partial model with the wrapping method in order to reduce the dimension of the model for invariant analysis.

The last fifth composite conceptual model provides for the simulation of a model of a system with parallelism as a whole, while the convolution of subordinate models is also used, provides the ability to analyze the network model and its matrix representation by the method of invariants. If certain elements need to be checked against the background of simulating the entire system, they can be detailed.

6. Discussion of the research results of modeling tools and analysis of systems with parallelism

The paradigms of simulation modeling are analyzed. It is proposed to create an approach to modeling software systems based on a combination of two paradigms of simulation modeling: discrete-event modeling and modeling of dynamic systems. The means of discrete-event modeling are represented by bounded Petri nets, on the basis of which local models of design solutions are built, analyzed in a simulation experiment. Modeling dynamic systems allows to get an unambiguous mathematical description of the network model in matrix form and calculate the value of invariants, the analysis of which allows to determine the dynamic characteristics of the model.

The features and limitations of formal languages of Petri nets are described, which made it possible to estimate the limitations of the characteristics of formal languages of L-type for describing models of software systems. This made it possible to define the properties necessary for complementing L-type languages as G-type properties for representing design decisions. In particular, the extended language L_G-type (1) has no restrictions on the initial and intermediate markings, which refer to a finite set of final markings. Thus, the limitation of WF networks [11], described by L-type languages, was extended. Selected interpretations of bounded Petri nets for constructing combined modeling tools for systems with parallelism that have greater modeling power than WF nets. These are safe, estimation and control Petri nets that allow to overcome the drawbacks indicated in [19]. Also, the proposed Petri nets can have a finite set of controlling vertices, it allows describing parallel and competing processes that are formed during the operation of software applications.

The properties of bounded Petri nets are analyzed, described by the class of formal languages of the L_G-type, such as liveness, reachability, boundedness and preservation. In particular, the reachability property can be used to define dead-end markings (Fig. 2). The complete controllability of the model is evidenced by the coverage of all elements with T- and P-invariants by nonzero values together with the equality of the rank of the incidence matrix of the minimum dimension of the cardinality of the sets of vertices of transitions of the vertices of places. To ensure the predictability of the functioning of the model, the property of loose preservation is applied (3). The preservation of bounded Petri nets with respect to the finite substitution allows them to be used to form local models, which are then assembled (Fig. 4) into an integral model of the software system.

Principles for a combined approach to the simulation modeling of systems with parallelism have been formulated. They allow to analyze the model at all stages of working with it, and also lay the foundation for a conceptual model for the synthesis and analysis of systems with parallelism. Unlike partially matched elements of analysis [6, 7] and modifications of Petri nets with unproven properties or hidden elements [15–18], the proposed conceptual model allows to combine the necessary means of synthesis and analysis of a model with parallelism. These are means of an explicit graphical description of the model, its analytical representation, which allows to identify unforeseen situations and determine the structures of the model that cause. This conceptual model makes it possible to propose and test design solutions that will ensure the operability and predictability of the functioning of the system under study.

On the basis of the above-mentioned components, the architecture of the combined approach for the step-by-step construction of the components of the system model with parallelism is formed, and on their basis the formation of an integral model of the software system is presented. This approach takes into account the limitations of the dimension of local models up to 40–45 elements of the largest set of the sets T and P, determines the maximum dimension of the incidence matrix. For the formation of an integral model of the system and its analysis, the recomposition of partial models is provided.

The disadvantages of the combined approach should be attributed to the rather laborious construction of submodels and a model of the software system. To reduce this disadvantage, it is possible to form design templates that are often found in models of software systems.

The prospects of this study are the addition of tools with elements of temporary Petri nets, also refer to bounded Petri nets, as well as automated generation of templates to correct areas of the model with unpredictable behavior.

7. Conclusions

1. It is proposed on the basis of a combination of two paradigms of simulation modeling: discrete-event modeling and modeling of dynamic systems to build a combined approach and tools for modeling systems with parallelism. The analysis of the formal languages of Petri nets made it possible to determine the features and limitations of the L-type language for constructing tools for modeling systems with parallelism. Bounded Petri nets, which are described by the formal language of the L_G-type, allow to expand the modeling power in comparison with the tools based on the L-type languages, it is important when modeling software systems. In particular, the proposed tools allow to start simulation modeling from any markup, refers to a finite set of final markings. It is also possible to create a finite number of control vertices, the layout of which has to be restored at the end of the model simulation session.

2. The technique for identifying a conflict situation on the example of a hidden dead lock using the invariant method and analysis of the incidence matrix is presented. It allows to analyze the causes of the dead lock, propose a correction to the model and check its dynamic properties. So, if a critical situation can arise in the model, the coverage of the components of the T- and P-invariants by nonzero elements is not enough, it is necessary that the rank of the incidence matrix is equal to the smallest dimension with the cardinality of the sets of transition vertices $|T|$ or tops of places $|P|$.

3. Principles and conceptual model for the development of the architecture of a combined approach to the simulation modeling of systems with parallelism are formed. The conceptual model allows to analyze the model of a software system using simulation tools. It also provides for the use of analytical representation, which creates the basis for automated model verification when building and simulating the functioning of software tools in conditions close to real ones. The principles concerning the structure of the model allow for self-checking when building the components of the model and visual analysis of sections of the model when identifying conflict situations. The principles relating to the dynamics of the model make it possible to provide an analysis of the dynamic properties of both the components of the model and private models and the model as a whole. This is important to

provide a holistic analysis of the software system model, especially when there are many parallel and competing processes. The possibility of applying in the combined approach the principles of constructing hierarchical Petri nets, which are described by formal languages of the G-type, makes it possible to comply with the requirements of resilience to changes and scalability for software systems. These properties are important in the development of modern software systems.

References

1. Stoian, V. A. (2008). Modeliuvannia ta identyfikatsiya dynamiky system iz rozpodilenymy parametramy. Kyiv: Kyivskyi universytet, 201.
2. Strogalev, V. P., Tolkacheva, I. O. (2008). Imitatsionnoe modelirovanie. Moscow: Izd-vo MGTU im. N.E. Baumana, 280.
3. Samarskiy, A. A., Mihaylov, A. P. (2001). Matematicheskoe modelirovanie. Idei. Metody. Primery. Moscow: Fizmatlit, 320.
4. Suprunenko, O. (2013). Paradigms of simulation modeling in studying complex parallel systems. Eastern-European Journal of Enterprise Technologies, 5 (4 (65), 63–67. Available at: http://journals.uran.ua/eejet/article/view/18353/16394
5. Sergienko, I. V. (2018). Mathematical and program modelling of complicated systems using supercomputer technologies. Visnik Nacional'noi' Academii' Nauk Ukrai'ni, 3, 39–48. doi: https://doi.org/10.15407/visn2018.03.039
6. Karpov, Yu. G. (2005) Imitatsionnoe modelirovanie sistem. Vvedenie v modelirovanie s AnyLogic 5. Sankt-Peterburg: BHV-Peterburg, 400.
7. Braude, E. (2004). Tekhnologiya razrabotki programmnogo obespecheniya. Sankt-Peterburg: Piter, 655. Available at: http://www.immsp.kiev.ua/postgraduate/Biblioteka_trudy/TekhnologiyaRazrabProgrBraude2004.pdf
8. Van Hee, K. (2002). Workflow management: models, methods, and systems. The MIT Press. doi: https://doi.org/10.7551/mitpress/7301.001.0001
9. Karpov, Yu. G. (2010). Model Checking. Verifikatsiya parallel'nyh i raspredelennyh programmnyh sistem. Sankt-Peterburg: BHV-Peterburg, 560.
10. Kuzmuk, V. V., Suprunenko, O. A. (2014). The means for the description of information flows in dynamic models of medical hardware-software systems. Theoretical & Applied Science, 7 (15), 11–18. doi: https://doi.org/10.15863/tas.2014.07.15.2
11. Van der Aalst, W. M. P. (2013). Business Process Management: A Comprehensive Survey. ISRN Software Engineering, 2013, 1–37. doi: https://doi.org/10.1155/2013/507984
12. Jensen, K., Rozenberg, G. (Eds.) (1991). High-level Petri Nets: Theory and Application. Springer, 724. doi: https://doi.org/10.1007/978-3-642-84524-6
13. Kuz'muk, V. V., Suprunenko, O. O. (2010). Modifitsirovannye seti Petri i ustroystva modelirovaniya parallel'nyh protsessov. Kyiv: Maklaut, 252.
14. Peterson, Dzh. (1984). Teoriya setey Petri i modelirovanie sistem. Moscow: Mir, 264.
15. Lomazova, I. A. (2004). Vlozhennye seti Petri: modelirovanie i analiz raspredelennyh sistem s obektnoy strukturoy. Moscow: Nauchniy mir, 208.
16. Lomazova, I. A. (2009). Adaptivnoe i dinamicheskoe modelirovanie potokov rabot na osnove vzaimodeystvuyuschih setey Petri. Metody i sredstva obrabotki informatsii. Trudy III Vserossiyskoy nauchnoy konferentsii. Moscow: Izdatel'skiy otdel fakul'teta vychislitel'noy matematiki i kibernetiki MGU, 32–37.
17. Bashkin, V. A. (2012). Approximating bisimulation in one-counter nets. Automatic Control and Computer Sciences, 46, 317–323. doi: https://doi.org/10.3103/s014641161207005x
18. Bashkin, V. A. (2017). On the Resource Equivalences in Petri nets with Invisible Transitions. Petri Nets and Software Engineering (PNSE'17). Zaragoza, 51–68.
19. Belusso, C. L. M., Sawicki, S., Roos-Frantz, F., Frantz, R. Z. (2016). A Study of Petri Nets, Markov Chains and Queueing Theory as Mathematical Modelling Languages Aiming at the Simulation of Enterprise Application Integration Solutions: A First Step. Procedia Computer Science, 100, 229–236. doi: https://doi.org/10.1016/j.procs.2016.09.147
20. Kuz'min, E. V., Sokolov, V. A. (2005). Vpolne strukturirovannye sistemy pomechennyh perekhodov. Moscow: Fizmatlit, 176.
21. Hack, M. (1975). Decision Problems for Petri Nets and Vector Addition Systems, Computation Structures Group Memo 95, Project MAC. Massachusetts Institute of Technology, Cambridge, Massachusetts, March 1974, pp. 79. revised as Memo 95-1, August 1974; Technical Memo 59, Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts, March 1975, pp. 7.
22. Murata, T. (1989). Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77 (4), 541–580. doi: https://doi.org/10.1109/5.24143
23. Suprunenko, O. O. (2019). Combined approach to simulation modeling of the dynamics of software systems based on interpretations of Petri nets. KPI Science News, 5-6, 43–53. doi: https://doi.org/10.20535/kpi-sn.2019.5-6.174596
24. Nesterenko, B. B., Novotarskiy, M. A. (2007). Algebra protsessov dlya modelirovaniya slozhnyh sistem s real'noy rabochey nagruzkoy. Reiestratsiya, zberihannia ta obrobka danykh, 9 (4), 49–59.