ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 3, No. 1, 2018

HIGH-PERFORMANCE SOFTWARE FOR DESIGNING COMPLEX CYBER-PHYSICAL SYSTEMS ON THE PARALLEL COMPUTERS

Yuriy Klushyn

Lviv Polytechnic National University, 12, Bandera str., Lviv, 79013, Ukraine.
Author's e-mail*: vladina_t@ukr.net*

Submitted on 01.12.2018

Abstract: **To study the effective functioning and behavior of parallel computing systems (which may be an integral part of the Cyber-Physical System), a high-performance software package based on mathematical models, methods and algorithms for stochastic modeling has been developed at the design stage. This software package completely solves the design problem – the parameters of a computing system have been calculated: its computational power, the average value of task execution time or various tasks on homogeneous resources of a parallel computing system, the distribution function of the task execution time. Based on the analysis of the parameters obtained, as well as indicators of the reliability of the system, the configuration of a parallel computing system has been selected or the possibility of using a previously selected computing system to perform the task has been considered.**

Index Terms: **Cyber-Physical Systems, parallel computing systems, a set of interrelated works, multilevel stochastic modeling, Markov process, distribution function of a random variable.**

I. INTRODUCTION

At present, the integration between computing and physical resources leads to the creation of complex computing systems with distributed parameters. Such systems are managed or controlled by computing resources that are integrated into the Internet. Such systems are called Cyber – Physical Systems (CPS) – these are systems consisting of various natural objects, artificial subsystems, and control computers that allow such an education to be represented as a single entity. CPS ensures close communication and coordination between computational and physical resources. Computers monitor and control physical processes using such a feedback loop, where what happens in physical systems affects computations and vice versa [1–3].

The predecessors of CPS can be considered as embedded real-time systems, distributed computing systems, automated control systems for technical processes and objects.

One of the main components of the Cyber – Physical System is its computational resource, which controls the second component of the Cyber-Physical System – objects that can be both natural and artificial. Therefore, when creating one or another cyber-physical system, the task is to choose the right control system, which includes the total computing resource CPS. In this regard, when using parallel

computer systems (CS) in Cyber-Physical Systems, the problem of a priori evaluation of the "suitability" of such CSs for solving a specific set of tasks for the required time becomes obvious.

With regard to parallel CS, this problem is called the prediction of the execution time of complex software systems; the latter are usually defined by graph models and are considered as complexes of interrelated jobs (CIJ) – tasks and / or their parallel-sequential fragments (subtasks, processes, program modules) [4-8].

It is important to note that the execution time of each program module (job) and CIJ as a whole is considered here as a random variable – through an arbitrary number of logical ramifications in the module program, cycles of indefinite length, random interaction between processes and external memory access, conflicts on shared resources, parallel CS and others. In this connection, the use of well-known exact methods for estimating the time of execution, for example, scheduling theory methods, to solve the prediction problem in the indicated stop is unacceptable.

The development of accurate mathematical models and algorithms for analyzing the functioning of parallel CS at a CIJ, which the user sets, with a random execution time of each job (process) would solve the actual problem of reliable analytical evaluation of the runtime of each specific CIJ at the CS a priori – before choosing the structure and configuration of parallel CS or to the detailed design of CIJ programs.

Due to the above features, the behavior of complex parallel computing systems can be studied through simulation (statistical) modeling, which is usually used for comparative analysis of alternative architectural and structural solutions when designing various CS nodes, as well as for assessing the accuracy of mathematical modeling. The issues of creating and applying simulation methods are well covered in the literature [9–10]. The advantage of simulation is the ability to analyze the job of the CS with almost any degree of detail. However, the study

of the CS methods of simulation in the general case is a time-consuming and complex process. The cost of creating simulation models and modeling, even when using specialized languages, is quite large. To study the effective functioning and behavior of parallel CS at the stage of their design or the choice of their structure, and configuration for the intended field of application, mathematical modeling is most often used. There are deterministic and stochastic (probabilistic) models. Deterministic models are used in assessing the value of relatively simple performance parameters of the CS, and therefore the possibility of using them to assess the effectiveness of the CS is very limited. Computer implementation of models of complex systems has its own specific features. Thus, due to the large number of system components, the resource consumption of computational procedures increases and the requirements for RAM for storing data structures are increased. The presence of long-distance links critically affects the possibilities of extensive parallelization of computations by formal methods and requires the use of specific decomposition algorithms. Finally, the inclusion of long-distance links in the system makes it difficult to parallelize by formal methods. The complexity of parallel CS requires the use of a set of interconnected models for their description [11–14]. As a result, the creation of a computational experiment toolkit with complex systems requires not only the development of optimal (for a given computational architecture) parallel algorithms, but also the construction of hierarchical computations. Such schemes define the process of interaction between simultaneously or sequentially executed computing modules, each of which, in turn, can job in parallel on one or more computing complexes of various architectures.

Thus, from the point of view of the parallel performance optimization problem, high-performance software systems for modeling complex systems can be themselves interpreted as complex systems. This naturally complicates the process of designing and developing such software.

A complex system can be considered in parts; the behavior of each part is detailed in the corresponding range of variability. Therefore, in the simplest case, the design system is a software system that integrates several applications that interact through input and output data flows.

There are mechanisms for dynamic control of computational processes that support the execution of a task at a given point in time [15], for example, the division of computations into "necessary" ("critical") and "optional" processes, an apparatus of inaccurate calculations, multivariate programming with several versions of program modules, where versions are selected dynamically against the time remaining until the end of the specified task completion time.

These and some other mechanisms for the dynamic control of computations have a priori estimates (statically) of the possibilities of performing tasks and their fragments during fixed time intervals.

Thus, the solution of the mathematical problem of predicting the execution time of a CIJ underlies the design methodology of complex systems on parallel computers, where the main parameters are the maximum execution time of a given task, as well as the system reliability parameters.

II. STATEMENT OF THE PROBLEM

The purpose of the article is a method for constructing a high-performance software complex for designing complex systems on parallel computers in full, based on existing mathematical models, methods and algorithms for multilevel stochastic modeling, which are used to calculate the average execution time for specific complex tasks (defined by a set of interrelated jobs – CIJ) parallel computing systems [10–17]. The software package described in the article solves the problem of design in full. That is, the parameters of the computing system are calculated: the computing power of the parallel system, the average value of the execution time of the CIJ on homogeneous resources of the parallel computing system, and also the distribution function of a random variable — the execution time of a set of interrelated jobs of the parallel computing system. Based on the analysis of the parameters obtained, as well as reliability indicators (for example, technical resource, service life, probability of failure-free operation, availability factor) of the system, the configuration of the parallel computing system is selected or the suitability of the selected computing system for solving this problem is considered.

The approach is based on the mathematical method of a multilevel stochastic simulation of the performance of CIJ [18] on parallel computers, which is superior in accuracy to the simulation of the same processes, since it allows to calculate the distribution function of the CIJ runtime. Calculation time according to the method [18] is an order of magnitude less than that required for simulation.

III. THE METHOD OF MULTILEVEL STOCHASTIC MODELING

One of such methods that most accurately solves the problem of predicting CIJ execution time is the method of multilevel stochastic modeling [18], which reduces the number of states of Markov process with an insignificant (by a few percent) increase in the prediction error of CIJ performance in parallel CS.

The essence of the method of circularly stochastic modeling is as follows.

The graph G of a given CIJ (Fig. 1) is described by the table of the connectivity of its vertices (Table 1). The Table contains N rows (by the number of vertices of the graph G), each of which indicates the numbers of jobs (vertices) that are the predecessors and successors of this job.

The process of CIJ performance is represented by a mathematical model (Fig. 2) in the form of a single-phase queuing system (CS) with $k \geq 2$ of the same type servicing

devices (**CD**) with buffer **B** for jobs ready for execution (current front *F*), the latter come from pool **P** containing initial state of n jobs.

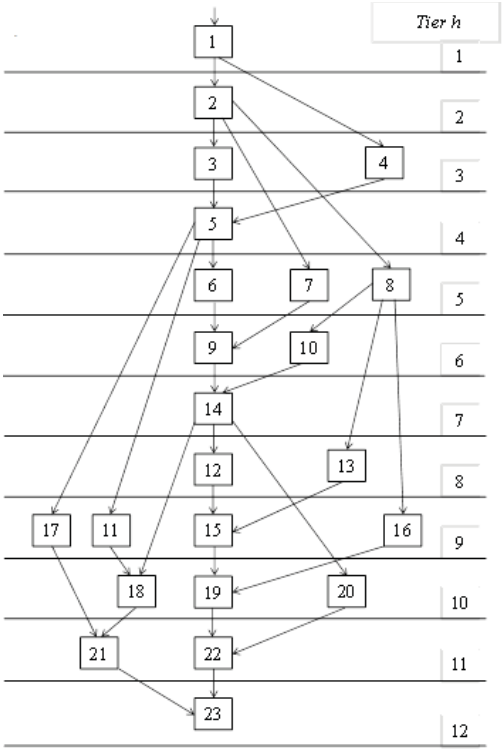

Fig. 1. Count G of a given CIJ

The vertices of the graph *G (A, H)*, which defines the given CIJ, are divided by the number of vertices into layers (tiers), the number of which is determined by the number of vertices of the critical path in this graph (Fig. 1).

Each layer is a set of parallel jobs that can be performed on the **CD** at the same time.

Note that splitting the vertices of the graph into tiers can be ambiguous.

For the mathematical model of the process *X (t)*, this means that the pool of jobs **P** is divided by the number of layers into *R* pools, where *R* is the length of the critical path in the column of CIJ.

The model operates in continuous time. Random time t_i the service of any job a_j is assumed to be distributed exponentially with the parameter $\mu_i = 1 / M[t_i]$, but in our case the value of j remains different for different *jobs*, depending on the specific values of $M[t_i]$. At the initial moment of time, one application arrives from the pool **P** in the system (buffer **B** and **CD** servicing devices) – "initial" operation a_1, which immediately begins to be serviced at one of the **CD**. Upon completion of the service a_1 (in the general case, a_i) in the **CD**, this job leaves the system "transmitting" its number *j* to the pool **P**, from which the successors of the job a_i , which turned out to be ready for execution (that is, all of them predecessors) the numbers of these jobs are uniquely determined from the table of connectedness of the vertices of the graph *G* and belong to a certain subset of layers from h to *h + l-1* ("view window"), moreover:

– the value of *h* can vary from *1* to *R*;

– *l* is the number of layers in the "view window" – can vary from *1* to *R*;

– *h*-number of the initial layer in the "viewing window";

– *h + l-1*-number of the final layer in the "viewer".

Table 1

Table of connectivity of vertices of graph G (Fig. 1)

Number job a_j	Job - Prede-cessors $\{a_{jP}\}_j$	Job – Successors $\{a_{jS}\}_j$	M[t_j]	μ_j	Rank r_j	Connec-tedness b_j
1	–	2.4	150	0.0067	12	2
2	1	3.7.8	30	0.0333	11	3
3	2	5	160	0.0063	10	1
4	1	5	80	0.0125	10	1
5	3.4	6.11.17	70	0.0143	9	3
6	5	9	120	0.0083	8	1
7	2	9	170	0.0059	8	1
8	2	10.13.16	50	0.0200	8	3
9	6.7	14	130	0.0077	7	1
10	8	14	150	0.0067	7	1
11	5	18	100	0.0100	4	1
12	14	15	160	0.0063	5	1
13	8	15	100	0.0100	5	1
14	9.10	12.18.20	40	0.0250	6	3
15	12.13	19	30	0.0333	4	1
16	8	19	170	0.0059	4	1
17	5	21	100	0.0100	3	1
18	11.14	21	70	0.0143	3	1
19	15.16	22	130	0.0077	3	1
20	14	22	80	0.0125	3	1
21	17.18	23	120	0.0083	2	1
22	19.20	23	50	0.0200	2	1
23	-	-	40	0.0250	1	0

Thus, at any given time in the system there may be not all CIJ jobs ready for execution (as in [18]), but only those of them that are in the "viewing window" from l adjacent tiers, and this is the "window" moves along the CIJ column one tier, as soon as the job of layer *h* is completed.

Note that for *l = R*, the lamellar simulation reduces to a direct stochastic simulation of the performance of the CIJ as a whole [13].

Within each "viewing window", the system operates according to the algorithm described in [18].

The operation of the CS is described by Markov process terminating *X (t)* over a set of states $X = \{(m; \vec{i}_w; \vec{j}_n)\}$; $\vec{i}_w$ includes the numbers of jobs ready for execution and waiting in buffer **B**, $\vec{j}_n$ includes the numbers of jobs that are being serviced in the **CD**.

However, now all the job that goes into $\vec{i}_w; \vec{j}_n$ belongs to the layers from *h* to *h + l-1*. This means the following: until all the job of the *h* – th layer has been completed, the job of

the layers following the layer $h + l\text{-}1$ cannot be performed. After all the job of the h – th layer has been completed, the total number of layers in the "viewing window" is again brought to l (unless many layers of R have been exhausted) and the algorithm described above is repeated. Thus, the feature of the CS functioning here is that in the system at any moment of time there can only be jobs related to layers with current numbers from h to $h + l\text{-}1$.

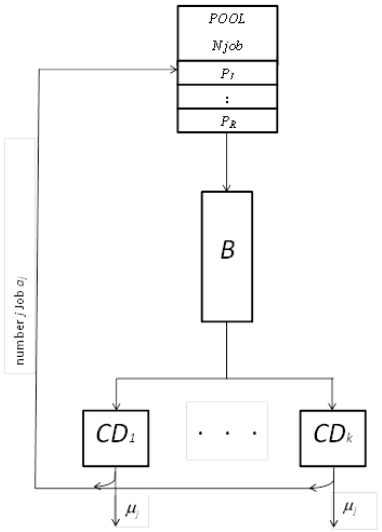

Fig. 2. Mathematical model

We will assume that a certain discipline (criterion) K [14] is given for an unambiguous selection of jobs from buffer B for maintenance in the **CD**, for example, the choice of job with the smallest number, therefore when the **CD** is released, the jobs according to the given criterion K have the highest priority (dispatch rule) [13–18].

To illustrate the functioning of the model (Fig. 2) when performing CIJ according to Fig. 1, we assume that the system contains two **CD**, that is, $k = 2$, and it can only have two layers at each time, that is, $l = 2$.

The predecessors of each job a_i are indicated here only if they belong to the previous $l\text{-}1$ layers. For example, for job a_{11}, which belongs to the 9 – th layer (denoted by a_{11} (9)), Table 1 indicates that it does not have predecessors (although in fact this is the job of $a_{5(4)}$): for $l = 2$, it is 4-th and 9-th tiers cannot simultaneously be in the "view window" and in the system.

Similarly, the successors of each job a_i are indicated only if they belong to the following $l\text{-}1$ tiers. For CIJ in Fig. 1 at the end of service $a_{1\,(1)}$ belonging to the first layer and pool **P1**, are ready to perform the job $a_{2\,(2)}$ and $a_{4\,(3)}$ of the second and third layers with **P2** and **P3** (see Table 1), which enter the system, and in this case directly on the **CD**. If the first of these jobs was $a_{4(3)}$, then the system does not receive new job, since $a_{5\,(5)}$ is the successor of job $a_{4(3)}$ is not ready for execution (job $_{3(3)}$ was not done), and there are only the job of the 2 – nd tier is $a_{2(2)}$. If the execution of $a_{2(2)}$ is completed (the only one in the **P2** pool), then the system can receive the job of the two following tiers (with $l = 2$), that is, with **P3** and **P4**, but in this case only one job comes $a_{3(3)}$ – successor of job $a_{2(2)}$. Now the system will have two jobs of the 3 – rd layer – $a_{3(3)}$ and $a_{4(3)}$. When one of these jobs ($a_{3(3)}$ or $a_{4(3)}$) is completed, no new jobs enter the system, since in both cases job $a_{5\,(4)}$ is not ready to be performed. Note that both jobs of the 3rd tier – $a_{3\,(3)}$ and $a_{4(3)}$ (from **P3**) have not yet been performed. The System will receive job from the 5th tier (from **P5**), in particular $a_{7(5)}$ and $a_{8(5)}$. For the mathematical model in Fig. 4 this means that job $a_{7(5)}$ and $a_{8(5)}$ do not have predecessors, job $a_{3(3)}$ has only one successor – job $a_{5(4)}$ (see Table 1).

The algorithm for determining the state of the process when performing a specific CIJ is described in [18].

IV. BLOCK – DIAGRAM OF THE SOFTWARE PACKAGE

The program complex consists of six modules:

– module for setting the structure of the graph;

– module for determining the states of the BMP;

– module of transformation of the matrix Q to a block triangular form;

– module for calculating the average time;

– module for calculating the distribution function;

– module for selecting a configuration of a parallel CS.

In Fig. 3 a block diagram of the connections of all the modules listed is shown.

V. THE MODULE SETS THE STRUCTURE OF THE GRAPH CIJ

The module for specifying the structure of a CIJ graph is used to enter input data using the algorithm described in Chapter 3, as well as in [18]. As the initial data are used:

– the table of connectivity of the vertices of the graph, which describes the graph G of a given CIJ (Table 1, columns 1-3.)

– parameters characterizing the given CIJ (see below);

– the parameter of parallel CS – the number k of service devices (computers of the CS)

– the cr parameter, which specifies the scheduling criterion;

– l is the number of layers in the "view window" – can vary from 1 to R;

– h-number of the initial layer in the "viewing window";

– $h + l\text{-}1$-number of the final layer in the "viewer".

– D– probability of performing CIJ at time given $T_{max.}$

The parameters that characterize a given CIJ are:

– N is the total number of all jobs given by the CIJ (or the initial number of applications in the pool **P**);

– μ_i – the intensity of service of each job (application) (Table. 1 column 5)

– b_i is the degree of connectedness of each job of this CIJ, which determines the number of successors (subset $\{a_{JS}\}_j$) of job a_i (Table 1, column 3)

– r_j is the rank of job a_j in a given CIJ.

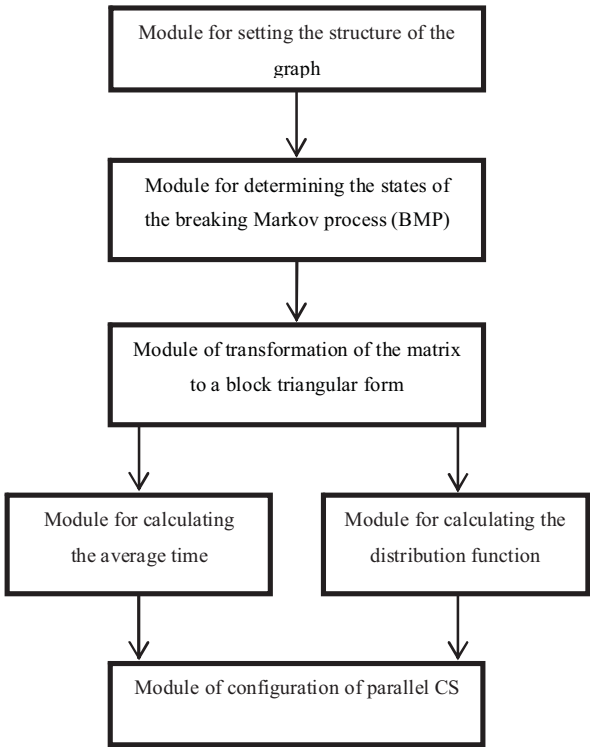

Fig. 3. Block is a diagram of a software system for designing complex systems on the parallel computers.

The *cr* parameter, which specifies the dispatching criteria, can take the following values:

– 0 – scheduling by rank criterion: "r_j" (Table 1, column 6);

– 1 – scheduling by the criterion of rank of connectedness of the vertices "r_i / b_i";

– 2 – scheduling according to the criterion "the choice of job with the lowest number";

– 3 – scheduling according to the criterion "the choice of job with the highest number".

VI. MODULE FOR DETERMINING THE STATE OF THE BMP WHEN PERFORMING A SPECIFIC CIJ

The module implements the algorithms described in Section 3, as well as in [18].

Set the initial state of the system.

State of the system:
Pool job = N-1
Job buffer = 0
Job CD = 1 (i.e. job a1)
Number of States = 1
Current state = 1

The module uses three job arrays:

– the array *Q* (*s, s*), which defines the matrix *Q* of the transition intensity of the considered terminating Markov process;

– arrays *X1* () *X2* (), elements of which are the state of the system X_p (*m, i_w, j_n*).

It should be noted that with tier stochastic modeling, the dimension of the array *Q (s, s)* with $l = R$ can be estimated by the formula (1). This is advisable for CIJ with a small amount of job, when the number of states of weapons of mass destruction is deliberately small. For large dimensions of the model, there is a sharp increase for computation required for processing the matrix *Q*, and hence, an increase in the cost of computer memory. So for the graph shown in Fig. 1, the number of states in accordance with formula 1 is equal to *233*, in reality, $S = 134$ with $l = R$; $S = 63$ with $l = 2$ (Table 1).

$$S^* = 2 + \sum_{l=1}^{k} C_{N-2}^{l}, \qquad (1)$$

k – the number of **CD**$

$$k = \overline{1, N-2}; \; C_{N-2}^{l} = \frac{(N-2)!}{(N-2-l)!l!}.$$

Therefore, to reduce the amount of computer memory used, it is desirable to determine the actual dimension of the array *Q* (*s, s*), which is one of the results of the implementation of the algorithms for determining the states of the BMP [18-20] in this software module. An additional opportunity to save computer memory when implementing the proposed methods is a special numbering and state order in the process of constructing the matrix *Q* (*s, s*) [14–18].

VII. CONVERSION RATE MATRIX TO BLOCK TRIANGULAR FORM

This software module not only transforms the matrix Q of the intensities of transitions to a block triangular form, but also forms this matrix in a compressed form, which simplifies and speeds up the execution of algorithms for obtaining the numerous characteristics of the system under study.

To bring the matrix to a triangular form, it is necessary and sufficient to order the elements of the state vector *X*. The numbering and ordering of states, as in [18], is created by decreasing or not increasing the values of the parameter m (the current number of jobs in the pool P), so that the value m for each state $X_p \in X$ would appear no more than in any of the previous (with a smaller number) states. States with the same values of m are ordered by decreasing or not increasing values of $c = w + n$ (the number of jobs in the system); if two or more states are characterized by the same values of m and c, then they are numbered arbitrarily.

To reduce the amount of used memory and accelerate the calculation, a special representation of the lower triangular matrix *Q* of dimension $n * n$ in the form of two arrays was used:

– array *wq* (*m, p*) values of zero elements in the row of the matrix *Q,*

– array *iq* (*m, n*) of numbers of nonzero elements in the row of the matrix *Q*.

Since in the matrix Q all diagonal elements are nonzero, in the array *iq* (*m, n*) the first column is used to indicate the type of row in which the nonzero elements described in it are located. The module also uses an array *num* (*p*) of the number of nonzero elements in rows of various types and an array *vec (n*) of the probability vector of the initial states of a system of dimension *s*.

VIII. THE MODULE CALCULATES THE AVERAGE RUNTIME CIJ

With the help of this module, the average value of CIJ T execution time on homogeneous resources of parallel CS is calculated. In this case, the module uses one job array *T (s, 4*), in which the current value T is remembered.

IX. THE DISTRIBUTION FUNCTION CALCULATION MODULE

With the help of this module, the distribution function of a random variable T is calculated – the execution time of a complex of interrelated jobs of a parallel CS. The algorithm for finding the distribution function of a random variable is described in [20].

X. MODULE OF CONFIGURATION OF PARALLEL CS

Based on the results obtained, as well as the system reliability parameters, the configuration of a parallel computing system is selected that satisfies all the necessary requirements for its operation, or the suitability of a previously selected computing system for solving a specific problem is considered.

XI. CONCLUSIONS

Based on the analysis of mathematical models, methods and algorithms for cyclic stochastic modeling, as well as research, a high-performance software package was created, designed to solve the problem of designing complex parallel computing systems in full. This software package calculates the parameters of the computing system: the average time of task execution on homogeneous resources of a parallel computing system with a given probability, and also calculates the distribution function of the task execution time in a parallel computing system. Based on the analysis of the parameters obtained, as well as reliability indicators (for example, technical resource, service life, probability of failure-free operation, availability ratio) and the computing power of the system, the system configuration is selected or the suitability of the previously selected computing system is considered. The computational resource obtained in this way can be one of the components of a complex cyber-physical system and allows you to control various objects of this system in real time.

The use of the method of multilevel stochastic modeling in the developed software package makes it possible to regulate the amount of RAM used in computing resources. So, for $l = R$, the total amount of computer memory used for processing the transition intensity matrix, which corresponds to performing a set of interrelated jobs with $N < 100$ jobs, about 260 kilobytes, and for $l = 2$, about 120 kilobytes.

REFERENCES

[1] Tsvetkov V. Ya., Alpatov A. N. Problems of distributed systems // Prospects of science and education – 2014. – No. 6. – P. 31–36.

[2] Khaitan et al., "Design Techniques and Applications of Cyber Physical Systems: A Survey", IEEE Systems Journal, 2014.

[3] Rad, Ciprian-Radu; Hancu, Olimpiu; Takacs, Ioana-Alexandra; Olteanu, Gheorghe (2015). "Smart Monitoring of Potato Crop: A Cyber-Physical System Architecture Model in the Field of Precision Agriculture". Conference Agriculture for Life, Life for Agriculture. 6: 73–79.

[4] Bocharov P. L., Ignatushchenko V. V. Mathematical models and methods for evaluating the effectiveness of parallel computing systems on complexes of interrelated jobs // Tez. report international conf, "High-Performance Computing Systems in Management and Scientific Research," Alma-Ata, 1991, p. 6.

[5] Ignatushchenko V. V., Klushin Y. S. Prediction of the implementation of complex software systems on parallel computers: direct stochastic modeling // Automation and Remote Control. 1994. No. 12, p. 142–157.

[6] Khritankov A. S. Mathematical model of performance characteristics of distributed computing systems. Computer science, management, economics. JOBS OF MIPT. – 2010. – Vol. 2, No. 1 (5), p. 110–115.

[7] Ivutin A. N., Larkin E. V. Prediction of the execution time of the algorithm. Magazine. News of TSU. Technical science. Issue number 3/2013 C 301–315.

[8] Ivanov N. N. Mathematical prediction of reliable execution of sets of tasks with symmetric runtime distributions. Journal of Open Education, Issue No. 2–2 / 2011, p. 52–55.

[9] Kulagin V. P., Problems of parallel computing systems Perspectives of Science & Education. 2016. 1 (19) International Scientific Electronic Journal ISSN 2307–2334 (Online)

[10] Bondur V. G. Modern approaches to the processing of large flows of hyperspectral and multispectral aerospace information // Study of the Earth of their cosmos. 2014. No. 1. P. 4–17

[11] Salibekyan S. M., Panfilov P. B. Questions of automaton-network modeling of computer systems with data flow control // Information technologies and computer systems. 2015. No. 1. P. 3–9.

[12] Kulikov, I., Chernykh, I., Glinsky, B., Weins, D., Shmelev, A. Astrophysics simulation on RSC massively parallel architecture // Proc. 2015 IEEE/ACM 15th Int. Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015. IEEE Press, 2015.1131–1134.

[13] Boccara N. Modeling Complex Systems. NY: Springer, 2004. 397 p.

[14] Lublinsky B. Defining SOA as an architectural style. 9 January 2007. [Electronic resource]: .

[15] Ivanov S.V., Identification of Parametrically Connected Models of Complex Systems, Nauch.-tekhnich. we know SPSU ITMO. High-performance computing and computer modeling technologies. 2008. Vol. 54. pp. 100–107.

[16] Ivanov N. N., Ignatushchenko V. V., Mikhailov A. Y., Static prediction of the execution time of complexes of interrelated jobs in multiprocessor computing systems, Avtomat. and Telemekh., 2005, issue 6, 89–103.

[17] Ignatushchenko V. V., Klushin Y. S. Prediction of the implementation of complex software systems on parallel computers: direct stochastic modeling // Automation and Remote Control. 1994. N12, p. 142–157.

[18] Klushin, Y. S. Improving the accuracy of estimating the execution time of folding software systems in multiprocessor computer systems for belt stochastic modeling. Bulletin of NU "Lviv Polytechnic" No. 881. Computer systems and netjobs. – Lviv: NU "LP", 2017.

[19] Klushin Y. S. reducing the number of states of the Markov process when executing complex software systems on parallel computers.

Scientific Bulletin of Chernivtsi University. Computer systems and components. 2016. T. 7. Vol. 2, pp. 53–62.

[20] Reibman A. L., Trivedi K. S. Numerical transient analysis of Markov models // Computers and Operations Research. 1988. Vol. 15. No. 1. P. 19–36.

[21] Preidunov Y. V. Development of mathematical models and methods for predicting the implementation of complex software systems on parallel computing systems. PhD thesis. M.: Inst. Of Problems of Management RAS, 1992.

Yuriy Klushyn works at the Computer Engineering Department of Lviv Polytechnic National University, Ukraine. He graduated from Lviv Polytechnic Institute with a degree in electrical engineering in 1983. In 1998, he received a Ph.D. in technical sciences from the Institute of Control Sciences of the Russian Academy of Sciences. He was recognized for his outstanding contribution to the development of special computer systems as a senior researcher in 1995.

He has scientific, academic and practical experience in the field of research and design of parallel computer systems, a proven contribution to the design methodology and the methodology for developing high-performance systems on the parallel computers. Mr. Klushyn is the author of more than 30 scientific articles and patents.