UDC 004.3
Doi: 10.31772/2587-6066-2020-21-3-296-302

For citation: Aab A. V., Galushin P. V., Popova A. V., Terskov V. A. Mathematical model of reliability of information processing computer appliances for real-time control systems. *Siberian Journal of Science and Technology.* 2020, Vol. 21, No. 3, P. 296–302. Doi: 10.31772/2587-6066-2020-21-3-296-302

Для цитирования: Математическая модель надёжности аппаратно-программных комплексов обработки информации для систем управления реального времени / А. В. Ааб, П. В. Галушин, А. В. Попова, В. А. Терсков // Сибирский журнал науки и технологий. 2020. Т. 21, № 3. С. 296–302. Doi: 10.31772/2587-6066-2020-21-3-296-302

MATHEMATICAL MODEL OF RELIABILITY OF INFORMATION PROCESSING COMPUTER APPLIANCES FOR REAL-TIME CONTROL SYSTEMS

A. V. Aab[1], P. V. Galushin[2], A. V. Popova[1*], V. A. Terskov[1]

[1]Reshetnev Siberian State University of Science and Technology
31, Krasnoyarskii rabochii prospekt, Krasnoyarsk, 660037, Russian Federation
[2]Siberian Law Institute of Ministry of Internal Affairs of the Russian Federation
20, Rokossovsky st., Krasnoyarsk, 660131, Russian Federation
*E-mail: anastasiya.popowa@mail.ru

One of the main characteristics of computer appliances for processing real-time information is reliability.

The reliability of software is understood as the property of this software to perform specified functions, maintaining its characteristics within the established limits under certain operating conditions.

Software reliability is determined by its reliability and recoverability.

Reliability of software is a property to maintain its performance when using it for processing information in the information system. The reliability of the software is estimated by the probability of its operation without failures under certain environmental conditions during a given observation period.

The development of real-time systems requires a large amount of resources for design and testing. One of the solutions to this problem is mathematical modeling of computer appliances. This allows more flexible design of real-time systems with the specified reliability, taking into account the limitations on price and development time, and also opens the possibility of more flexible optimization of computer appliances for real-time control systems.

To develop a mathematical model of the reliability of computer appliance for real-time systems, it is necessary to take into account the provision of a given level of reliability, with reasonable development costs.

There are many methods for improving software reliability, but the most promising and effective methods are redundancy, which is achieved using N-version programming.

To increase the reliability of the hardware of the computer appliance, it is also necessary to use redundancy and redundancy, which includes multiprocessor and provision of different buses and independent RAM.

This paper discusses existing approaches to improving the reliability of hardware and software, proposes a model of reliability of a computer appliance, which is understood as the product of the probability of failure-free operation of hardware and the probability of error-free operation of software.

In addition, new formulas are proposed for the steady state probabilities of the hardware states of a multiprocessor computer appliance with heterogeneous processors, which give the same result as the existing ones, but require fewer computations.

The paper concludes with a question about the possibility of optimizing the reliability of computer appliances based on the developed model, and indicates optimization methods that can be used to solve this problem.

Keywords: reliability, software reliability, real-time systems, mathematical model, multiversion programming.

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ НАДЁЖНОСТИ АППАРАТНО-ПРОГРАММНЫХ КОМПЛЕКСОВ ОБРАБОТКИ ИНФОРМАЦИИ ДЛЯ СИСТЕМ УПРАВЛЕНИЯ РЕАЛЬНОГО ВРЕМЕНИ

А. В. Ааб[1], П. В. Галушин[2], А. В. Попова[1*], В. А. Терсков[1]

[1]Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева
Российская Федерация, г. Красноярск, 660037, просп. им. газ. «Красноярский рабочий», 31
[2]Сибирский юридический институт МВД России
Российская Федерация, г. Красноярск, 660131, ул. Рокоссовского, 20
*E-mail: anastasiya.popowa@mail.ru

Одной из главных характеристик аппаратно-программных систем обработки информации реального времени является надёжность.

Под надёжностью программного обеспечения (ПО) понимается свойство этого обеспечения выполнять заданные функции, сохраняя свои характеристики в установленных пределах при определённых условиях эксплуатации.

Надёжность ПО определяется его безотказностью и восстанавливаемостью.

Безотказность ПО – это свойство сохранять работоспособность при использовании его для обработки информации в информационной системе. Безотказностью программного обеспечения оценивается вероятность его работы без отказов при определённых условиях внешней среды в течение заданного периода наблюдения.

Разработка и проектирование систем реального времени требует большого количества ресурсов на проектирование и тестирование. Одним из решений данной проблемы является математическое моделирование аппаратно-программных комплексов. Это позволяет более гибко проектировать системы реального времени с заданной надёжностью с учётом ограничений по цене и времени разработки, а также открывает возможность более гибкой оптимизации аппаратно-программных систем реального времени.

Для разработки математической модели надёжности аппаратно-программного комплекса систем реального времени необходимо учитывать обеспечение заданного уровня надёжности при целесообразных затратах на разработку.

Существует много методов повышения надёжности программного обеспечения, но наиболее перспективный и эффективный метод – это избыточность, которая достигается за счёт использования мультиверсионного программирования.

Для повышения надёжности аппаратной части комплекса также необходимо использовать избыточность и резервирование, что включает в себя мультипроцессорность и обеспечение разных шин и независимой оперативной памяти.

В данной статье рассматриваются существующие подходы к повышению надёжности аппаратного и программного обеспечения, предлагается модель надёжности аппаратно-программного комплекса, которая понимается как произведение вероятности безотказной работы аппаратного обеспечения и вероятности безошибочной работы программного обеспечения.

Кроме того, предлагаются новые формулы для вероятностей состояний аппаратного обеспечения многопроцессорного вычислительного комплекса с разнородными процессорами в установившемся режиме, дающие тот же результат, что существующие, но требующие меньше вычислений.

В заключении статьи ставится вопрос о возможности оптимизации надёжности аппаратно-программных комплексов на основе построенной модели, указываются методы оптимизации, которые могут быть использованы при решении данной задачи.

Ключевые слова: надёжность, программное обеспечение, системы реального времени, математическая модель, мультиверсионное программирование.

Introduction. Reliability is one of the main characteristics of real-time hardware and software systems for processing information [1–3].

Reliability of software is understood as the ability of this software to perform specified functions, while maintaining its characteristics within established limits under certain operating conditions.

The reliability of the software is determined by its faultlessness and recoverability. Faultlessness of software is a property to maintain operability when using it to process information in an information system (IS). The faultlessness of the software is the probability of its operation without failures under certain environmental conditions during a given observation period.

Real-time systems development requires a lot of design and testing resources. One of the solutions to this problem is mathematical modeling of hardware and software systems. This allows for more flexible design of real-time systems with a given reliability, taking into account the limitations on price and development time, and also opens up the possibility of more flexible optimization of hardware and software complexes of real-time control systems.

With the development of processing power, evolutionary algorithms and neural networks, such models are becoming increasingly relevant.

To develop a mathematical model of the reliability of the hardware and software complex of real-time systems, it is necessary to take into account the provision of a given level of reliability with reasonable development costs.

Reliability of the software. The reliability of the software architecture includes both the reliability of the central system core and the reliability of the individual components provided to the user. Failure of an individual component can lead to the inoperability of this and, possibly, other software components. However, this should not lead to the inoperability of the entire system as a whole. A thorough analysis of the software architecture allows to identify the components, the errors in which have the most significant impact on the reliability of the system. Generally, these are the components most commonly used or architecturally related to many other components. There are a lot of methods for increasing the reliability of software [4; 5], but currently only the multiversion fault-tolerant programming approach is a possible alternative to testing and verification methods, providing a high level of reliability of critical software functioning [6]. It is important to realize that verification does not guarantee correctness, since the specifications and / or verification systems themselves (like any other software) may contain errors.

The use of decision support systems in multiversion programming allows us to focus on the quality of requirements at the stage of creating reliable software. However, improving software reliability characteristics using redundancy requires additional time and financial resources. Therefore, the main question at this stage is how, using redundancy in the software architecture, to maximize reliability and reduce development costs. This area includes methods of multicriteria decision making, focusing on problems with a discrete decision space. Taking into account different levels of information about the expert's preferences, a variety of methodologies for multicriteria decision support have recently been developed. The use of various methods for determining the depth of multiversion and multi-criteria decision-making when choosing an architecture makes it possible to design a software system that meets the requirements.

Depending on the number and size of the components, the conditional and unconditional probabilities of failure, access, analysis and recovery time, as well as the usage time of the components, are different. The model [7], given below, can be used to assess the reliability of software for possible architectural changes, select a reliable architecture from various options and has the following designators:

1) M – number of architectural levels in software architecture;

2) N_j – number of components at level $j, j \in \{1, ..., M\}$;

3) D_{ij} – set of component indices depending on the component i at level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}$;

4) F_{ij} – a failure that occurred in a component i at level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}$;

5) PU_{ij} – probability of using component i at level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}$;

6) PF_{ij} – probability of a failure in a component i at level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}$;

7) PL^{ij}_{nm} – the conditional probability of a failure in component m at level n when a failure occurs in a component i at level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}, n \in \{1, ..., N_m\}$, $m \in \{1, ..., M\}$;

8) TA_{ij} – relative access time to the component i at level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}$, defined as the ratio of the average access time to component i at level j to the number of failed components at small levels of architecture for the same time;

9) TC_{ij} – relative time to analyze failure in component i at level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}$, defined as the ratio of the average time of failure analysis in component i at level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}$, to the number of faulty components at all levels of the architecture analyzed at the same time;

10) TE_{ij} – the relative time to resolve a failure in component i at level j, $i \in \{1, ..., N_j\}$, $j \in \{1, ..., M\}$, defined as the ratio of the average recovery time in component i at the level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}$, to the number of failed components at all levels of the architecture, in which failures are eliminated at the same time;

11) TU_{ij} – relative usage time of component i at level j, $i \in \{1, ..., N_j\}, j \in \{1, ..., M\}$, defined as the ratio of the average use time of component i at the level j, $i \in \{1, ..., N_j\}$, $j \in \{1, ..., M\}$, to the number of components at all levels of the architecture used at the same time;

12) TR –average system downtime in a large real-time software architecture, defined as the time during which the system is unable to perform its functions;

13) *MTTF (Mean Time to Failure)* –the average time to failure in a large real-time software architecture, defined as the time during which no system failures occur.

In the architecture of real software, the average time for the appearance of a failure, that is, the time during which the software functions correctly, is [7]:

$$MTTF = \sum_{j=1}^{j=M} \sum_{i=1}^{i=Nj} \{PU_{ij} \times (1 - PF_{ij}) \times [TU_{ij} +$$

$$+ \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=Nm} [(1 - PL^{ij}_{nm})[TU_{nm} + \sum_{l \in Dnm} [(1 - PL^{nm}_{lm}) \times TU_{lm}]]] +$$

$$+ \sum_{k \in Dij} [(1 - PL^{ij}_{kj})[TU_{kj} + \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=Nm} [(1 - PL^{kj}_{nm}) \times$$

$$\times [TU_{nm} + \sum_{l \in Dnm} [(1 - PL^{nm}_{lm}) \times TU_{lm}]]]]]]\}.$$

The average downtime (recovery) of the software is [7]:

$$TR = \sum_{j=1}^{j=M} \sum_{i=1}^{i=Mj} \{PU_{ij} \times PF_{ij} \times [(TA_{ij} + TC_{ij} + TE_{ij}) +$$

$$+ \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=Nn} [PL^{ij}_{nm} \times [(TA_{nm} + TC_{nm} + TE_{nm}) +$$

$$+ \sum_{l \in Dnm} [PL^{nm}_{lm} \times (TA_{lm} + TC_{lm} + TE_{lm})]]]$$

$$\sum_{k \in Dij} [PL^{ij}_{kj} \times [(TA_{kj} + TC_{kj} + TE_{kj}) +$$

$$+ \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=Nm} [PL^{kj}_{nm}[(TA_{nm} + TC_{nm} + TE_{nm}) +$$

$$+ \sum_{l \in Dnm} [PL^{nm}_{lm} \times (TA_{lm} + TC_{lm} + TE_{lm})]]]]]]\}$$

Average system downtime and average failure time can be used to predict the reliability of software in general. For the case of continuous operation of complex software (and for a real-time system, this is the most likely mode of software operation), the reliability of the software can be estimated using the availability factor S, calculated by the following formula:

$$S = \frac{MTTF}{MTTF + TR}.$$

The availability factor can be interpreted as the probability of correct software functioning.

The main approach to increasing the reliability of software, which is subject to increased requirements for continuity and correct functioning, is multiversion development, that is, the creation by independent developers of several versions of a software component that correspond to the same specifications, but differ in implementations. The software architecture should provide for a mechanism for forming the overall result of the operation of this software component based on the results of the work of each individual version. In [8], two main methods of building architecture when using multiversion development are given, which are abbreviated as *NVP* and *RB*.

When building a multiversion component from *K* versions by the multiversion programming method (*NVP*, *N*-version programming) for any *K*, the reliability is equal to:

$$R_{ij} = p_{ij}^{v}\left(1 - \prod_{k \in Z_{ij}}^{K} (1 - p_{ij}^{k})\right),$$

where p^{v}_{ij} – probability of failure of the voting algorithm for component *i* at level *j*, Z_{ij} – many versions of this component, and p^{k}_{ij} is probability of failure of version $k \in Z_{ij}$.

When building a multiversion component from *K* versions using the recovery block (RB, recovery block) method, the reliability is:

$$R_{ij} = \sum_{k \in Z_{ij}}^{K} p_{ij}^{k} p_{ij}^{AT} \prod_{l=1}^{k-1}\left((1 - p_{ij}^{l}) p_{ij}^{AT} + p_{ij}^{l}(1 - p_{ij}^{AT})\right),$$

where p^{AT}_{ij} – the probability of failure-free operation of the acceptance test for component *i* at level *j*, p^{k}_{ij} is probability of failure of version $k \in Z_{ij}$.

The last two formulas can be used to calculate the probabilities of failure of software components:

$$PF_{ij} = 1 - R_{ij}.$$

Hardware reliability. To increase the reliability of the hardware part of the complex, it is also necessary to use redundancy and reservation, which includes multiprocessing and the provision of different buses and independent RAM.

Each processor and bus fail at some random points in time, after which they begin to recover. Let us assume that the flows of failures and recoveries are the simplest. The bus failure rate is denoted as ν_0, the bus recovery rate is μ_0. The rate of processor failure streams is denoted by ν_i, and the rate of recovery of processors of the *i* type is μ_i.

Probability $P_{j0,j1,\dots,jN}$ finding the system in a state in which j_0 шин интерфейса исправны и участвуют в вычислительном процессе, а $(m_0 - j_0)$ неисправны и восстанавливаются interface buses are in good working order and participate in the computational process, and $(m_0 - j_0)$ are faulty and are being recovered, j_1 type 1 processors are operational and participate in the computational process, and $(m_1 - j_1)$ faulty and are recovering, ..., j_N processors of *N* type are proper and participate in the computational process, and $(m_N - j_N)$ are faulty and are recovered, is determined by the following formula [9; 10]:

$$P_{j_0, j_1, \dots, j_N} = \frac{\dfrac{k!}{\prod_{i=0}^{N} j_i!} \prod_{i=0}^{N} \dfrac{m_i!}{(m_i - j_i)!} \rho_i^{j_i}}{\sum\limits_{j_0, \dots, j_N} \dfrac{k!}{\prod_{i=0}^{N} j_i!} \prod_{i=0}^{N} \dfrac{m_i!}{(m_i - j_i)!} \rho_i^{j_i}},$$

where $k = \sum_{i=0}^{N} m_i$, and the summation in the denominator is carried out over all possible values of the indices $j_0, \dots, j_N$ (this will be implied in all subsequent similar formulas). In addition, the following designation is used:

$$\rho_i = \frac{\nu_i}{\mu_i}.$$

The formula for the probabilities of states in a stationary mode can be simplified. First, we can shorten *k*! in the numerator and denominator, and also combine the works in the numerator and denominator:

$$P_{j_0, j_1, \dots, j_N} = \frac{\prod_{i=0}^{N} \dfrac{m_i!}{(m_i - j_i)! j_i!} \rho_i^{j_i}}{\sum\limits_{j_0, \dots, j_N} \prod_{i=0}^{N} \dfrac{m_i!}{(m_i - j_i)! j_i!} \rho_i^{j_i}}.$$

Now let's rewrite the denominator so that first the summation is performed, and then the multiplication:

$$P_{j_0, j_1, \dots, j_N} = \frac{\prod_{i=0}^{N} \dfrac{m_i!}{(m_i - j_i)! j_i!} \rho_i^{j_i}}{\prod_{i=0}^{N} \sum\limits_{j_i=0}^{m_i} \dfrac{m_i!}{(m_i - j_i)! j_i!} \rho_i^{j_i}}.$$

Each of the sums in the denominator is now the multiplication of the binomial coefficient and the corresponding power of a fixed value, and therefore can be simplified using the Newton binomial formula as follows [11]:

$$\sum_{j_i=0}^{m_i} \frac{m_i!}{(m_i - j_i)! j_i!} \rho_i^{j_i} =$$

$$= \sum_{j_i=0}^{m_i} \frac{m_i!}{(m_i - j_i)! j_i!} \rho_i^{j_i} \cdot 1^{m_i - j_i} = (\rho_i + 1)^{m_i}.$$

Thus, we finally obtain the following expression for the probabilities of the hardware states in the steady state:

$$P_{j_0, j_1, \dots, j_N} = \frac{\prod_{i=0}^{N} \dfrac{m_i!}{(m_i - j_i)! j_i!} \rho_i^{j_i}}{\prod_{i=0}^{N} (\rho_i + 1)^{m_i}}.$$

Simplified formulas contain fewer values than the original ones, and, therefore, allow to avoid performing some unnecessary operations (for example, multiplying both the numerator and the denominator by *k*!, which obviously does not affect the value of the expression) and require significantly fewer operations to calculate the value of the denominator than the original formulas given in [9; 10].

Knowing the minimum hardware configuration required for the real-time system software to generate control action within the intervals required by the customer, the reliability of the hardware can be calculated. Indeed, if the number of trouble-free operating memory buses and processors exceeds the minimum required, then "redundant" software components can be considered as a reserve. Then the reliability of the hardware of the real-time control system, understood as the probability of ensuring the minimum required performance of the hardware, can be calculated as the sum of the probabilities of finding the hardware in states in which the number of trouble-free operating memory buses and processors exceeds the minimum required number:

$$G_{\text{cp}} = \sum_{\substack{M_0 \le j_0 \le m_0 \\ \dots \\ M_N \le j_N \le m_N}} P_{j_0, j_1, \dots, j_N},$$

where M_0 is the minimum number of proper memory buses required to ensure a given performance, M_i is the minimum number of serviceable processors of the i type required to ensure the specified performance.

The formula for G_{cp} involves calculating the probabilities of different states of hardware with different values of the indices $j_0, j_1, \dots, j_N$. In this case, it is not advisable to repeat the calculations that have already been performed. For example, the denominator of all formulas has the same value, and, therefore, it is sufficient to calculate it once.

In addition, since it is required to compute states with sequential values of the indices, one can use the following recurrent formulas for the degrees and binomial coefficients, following from the definitions of the degree and factorial:

$$\rho_i^{j_i+1} = \rho_i^{j_i} \cdot \rho_i \ ,$$

$$\frac{m_i!}{(m_i - j_i - 1)!(j_i + 1)!} = \frac{m_i!}{(m_i - j_i)! j_i!} \cdot \frac{m_i - j_i}{j_i + 1} .$$

These ratios allow to calculate the probabilities for states with sequential index values with a minimum number of additional operations:

$$P_{j_0, \dots j_i + 1, \dots, j_N} = P_{j_0, \dots j_i, \dots, j_N} \cdot \frac{m_i - j_i}{j_i + 1} \rho .$$

From this ratio, elementary transformations can be used to obtain the formula for "decreasing the index":

$$P_{j_0, \dots j_i, \dots, j_N} = P_{j_0, \dots j_i + 1, \dots, j_N} \cdot \frac{j_i + 1}{(m_i - j_i)\rho},$$

$$P_{j_0, \dots j_i - 1, \dots, j_N} = P_{j_0, \dots j_i, \dots, j_N} \cdot \frac{j_i}{(m_i - j_i + 1)\rho} .$$

Since the simplest form of the formula for the probabilities of states is taken in cases when all buses and processors are in proper order, or when all of them are in bad order, it is advisable to start calculating G_{cp} with the probability of a state in which all buses of the RAM and processors are working correctly:

$$P_{m_0, m_1, \dots, m_N} = \frac{\prod_{i=0}^{N} \rho_i^{m_i}}{\prod_{i=0}^{N} (\rho_i + 1)^{m_i}}$$

and then, using the "decreasing indices" formulas, sequentially calculate the remaining required state probabilities.

Another opportunity for improvement opens up if the minimum configuration that provides a given performance includes fewer hardware components than half of the components present in the system. In this case, you can calculate the sum of the probabilities of states in which the performance is not sufficient to generate a control action within the given time constraints, and then calculate the required probability as the probability of the opposite event [12]:

$$G_{\text{cp}} = 1 - \sum_{\substack{0 \le j_0 < M_0 \\ \dots \\ 0 \le j_N < M_N}} P_{j_0, j_1, \dots, j_N} .$$

When calculating using this formula, one should start with the probability of a state in which all RAM buses and processors are faulty:

$$P_{0,0,\dots,0} = \frac{1}{\prod_{i=0}^{N} (\rho_i + 1)^{m_i}}$$

and then use the index increase formulas obtained above.

Reliability model of hardware and software complex. Finally, using the above considerations, we can combine software and hardware reliability models into a general reliability model of a multiprocessor hardware-software complex of a real-time control system with multiversion software.

Due to the abstract nature of software and the negligible probability of errors during its copying and distribution (in addition, the integrity of the software can be checked using mechanisms such as checksums), it can be considered that hardware and software failures occur independently.

Therefore, the probability of simultaneous failure-free operation of the hardware-software complex is equal to the product of the probabilities of failure-free operation of the software and hardware [12]:

$$P_F = G_{\text{cp}} \cdot S .$$

Conclusion. Thus, we have obtained a model for calculating the reliability of multiprocessor hardware and software systems of real-time systems with heterogeneous processors and multiversion software, using queuing theory and reliability theory, which allows to consider many architecture options in a short time and without significant costs typical for constructing experimental samples and reliability assessment by organizing trial operation.

The proposed mathematical model can be used to automate the design of multiprocessor hardware and software systems. In practice, they strive to ensure that the projected real-time control system has the highest possible reliability, provided that the costs of its creation and operation do not exceed the allocated funds. Thus, we come to the problem of conditional or multicriteria optimization, in which the objective function is expressed in terms of the probabilities of states calculated within the framework of the proposed model. Despite the fact that there are analytical expressions for the reliability indicators of a multiprocessor hardware-software complex, this optimization problem has a number of inconvenient features: the variables being optimized are discrete, the presence of a single extremum and properties convenient for optimization (convexity) is not guaranteed, and the volume of the search space is growing rapidly with an increase in the number of processor types.

When solving such optimization problems, evolutionary optimization methods, for example, a genetic algorithm [13], a probabilistic genetic algorithm [14] or an asymptotic probabilistic genetic algorithm [15], as well as some other "nature-inspired" optimization methods, for example, the particles swarm method [16]. The disadvantage of evolutionary optimization methods is that they have a large number of tunable parameters that can significantly affect the quality of the solutions found. Moreover, different sets of parameters can be effective for different optimization problems. Thus, the use of evolutionary optimization methods requires, as a rule, experimentation and the involvement of a specialist in the field of evolutionary optimization methods.

In order to exclude the stage of selecting an effective combination of parameters, self-tuning can be used [17; 18]. Self-adjusting optimization methods use several sets of parameters, computing resources (for simplicity, we can assume that this is the number of calculations of the objective function) which are distributed depending on the quality of the solutions obtained with their help. At the beginning of the optimization process, all parameter sets receive the same resources. Then the sets of parameters that generate the best solutions receive additional computing resources due to those that perform worse. There is a certain lower bound for the allocated resources, this is done so that any set of parameters can manifest itself in the future when conditions change, since different stages of optimization may require different values of parameters.

The study of the effectiveness of these optimization methods in solving the problem of optimizing the reliability of multiprocessor hardware and software complexes of real-time control systems with multiversion software and dissimilar processors is a possible direction for further research.

References

1. Buttazzo G. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. New York, NY, Springer, 2011. XVI+524 p.

2. Vasil'ev V. A., Legkov K. E., Levko I. V. [The real-time systems and applications]. *Informaciya i kosmos.* 2016, № 3, P. 68–70 (In Russ.).

3. CHerkesov G. N. *Nadezhnost' apparatno-programmnyh kompleksov* [Reliability of the computer appliances]. Spb., Piter Publ., 2005, 479 p.

4. Lipaev V. V. *Ekonomika proizvodstva programmnyh produktov* [The economics of the software engineering]. Moscow, SINTEG Publ., 2011, 358 p.

5. Zatuliveter Yu. S., Fishchenko E. A., Hodakovskij I. A. [The software methods for improving the reliability of structurally complex distributed computing and control processes]. *Nadezhnost'.* 2009, No. 1, P. 42–49 (In Russ.).

6. Avizienis A. The N-Version Approach to Fault-Tolerant Software. *IEEE Trans. Soft. Eng.* 1985, Vol. SE-11 (12), P. 1511–1517.

7. Kukartsev V. V., Sheenok D. A. [Optimization of the software architecture of logistics information systems]. *Logisticheskie sistemy v global'noj ekonomike.* 2013, No. 3, P. 138–145 (In Russ.).

8. Antamoshkin O. A., Degterev A. S., Rusakov M. A. et al. [The analysis of the reliability of computer appliances]. *Uspekhi sovremennogo estestvoznaniya*. 2005, No. 6, P. 44–45 (In Russ.).

9. Efimov S. N., Terskov V. A. *Rekonfiguriruemye vychislitel'nye sistemy obrabotki informacii i upravleniya* [The reconfigurable computing systems of information processing and control]. – Krasnoyarsk, KrIZHT IrGUPS Publ., 2013, 249 p.

10. Efimov S. N., Tyapkin V. N., Dmitriev D. D. et al. Methods of assessing the characteristics of the multiprocessor computer system adaptation unit. *ZHurnal Sibirskogo federal'nogo universiteta. Seriya: Matematika i fizika*. 2016, Vol. 9, No. 3, P. 288–295 (In Russ.).

11. Graham R. L., Knuth D. E., Patashnik O. Concrete Mathematics – A foundation for computer science. Reading, MA, USA, Addison-Wesley Professional, 1994, 657 p.

12. Ventcel' E. S., Ovcharov L. A. *Teoriya veroyatnostej i eyo inzhenernye prilozheniya* [Probability theory and its engineering applications]. Moscow, Vysshaya shkola Publ., 2000, 480 p.

13. Goldberg D. E. Genetic algorithms in search, optimization, and machine learning. Reading, MA, Addison-Wesley, 1989, 372 p.

14. Vorozheikin F. Yu., Gonchan T. N., Panfilov I. A. at al. Modified Probabilistic Genetic Algorithm for the Solution of Complex Constrained Optimization Problems. *Vestnik SibSAU.* 2009. No. 5 (26), P. 31–36.

15. Galushin P. V. [Design and evaluation of asymptotic probabilistic genetic algorithm]. *Zhurnal Sibirskogo federal'nogo universiteta. Seriya: Matematika i fizika.* 2012, No. 1(5), P. 49–56 (In Russ.).

16. Kovalev I. V., Solov'ev E. V., Kovalev D. I. et al. [Application of particle swarm optimization to design of N-version software composition]. *Pribory i sistemy. Upravlenie, kontrol', diagnostika*. 2013, No. 3, P. 1–6 (In Russ.).

17. Semenkin E., Semenkina M. Stochastic Models and Optimization Algorithms for Decision Support in Spacecraft Control Systems Preliminary Design. *Informatics in Control, Automation and Robotics, Lecture Notes in Electrical Engineering.* 2014, Vol. 283, P. 51–65.

18. Semenkin E., Semenkina M. Self-Configuring Genetic Programming Algorithm with Modified Uniform Crossover Operator. *Proceedings of the IEEE Congress on Evolutionary Computation.* June 10–15, 2012.

Библиографические ссылки

1. Buttazzo G. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. New York, NY, Springer, 2011. XVI+524 P.

2. Васильев В. А., Легков К. Е., Левко И. В. Системы реального времени и области их применения // Информация и космос. 2016. № 3. С. 68–70.

3. Черкесов Г. Н. Надежность аппаратно-программных комплексов. Спб. : Питер, 2005. 479 с.

4. Липаев В. В. Экономика производства программных продуктов. М. : СИНТЕГ, 2011. 358 с.

5. Затуливетер Ю. С., Фищенко Е. А., Ходаковский И. А. Программные методы повышения надеж-

ности структурно сложных распределенных вычислений и процессов управления // Надежность. 2009. №. 1. С. 42–49.

6. Avizienis A. The N-Version Approach to Fault-Tolerant Software // IEEE Trans. Soft. Eng, 1985. Vol. SE-11 (12). P. 1511–1517.

7. Кукарцев В. В., Шеенок Д. А. Оптимизация программной архитектуры логистических информационных систем // Логистические системы в глобальной экономике. 2013. № 3. С. 138–145.

8. Анализ надежности мультиверсионных архитектур аппаратно программных комплексов / О. А. Антамошкин, А. С. Дегтерев, М. А. Русаков и др. // Успехи современного естествознания. 2005. №. 6. С. 44-45.

9. Ефимов С. Н., Терсков В. А. Реконфигурируемые вычислительные системы обработки информации и управления. Красноярск : КрИЖТ ИрГУПС, 2013. 249 с.

10. Methods of assessing the characteristics of the multiprocessor computer system adaptation unit / Efimov S. N., Tyapkin V. N., Dmitriev D. D. и др. // Журнал Сибирского федерального университета. Серия: Математика и физика. 2016. Т. 9, № 3. С. 288–295.

11. Грэхем Р. Л., Кнут Д. Э., Паташник О. Конкретная математика. Математические основы информатики : 2-е изд. ; пер. с англ. М. : ООО «И.Д. Вильямс», 2010. 784 с.

12. Вентцель Е. С., Овчаров Л. А. Теория вероятностей и её инженерные приложения : 2-е изд. М. : Высшая школа, 2000. 480 с.

13. Goldberg D. E. Genetic algorithms in search, optimization, and machine learning. Reading, MA, Addison-Wesley, 1989. 372 p.

14. Vorozheikin F. Yu., Gonchan T. N., Panfilov I. A. at al. Modified Probabilistic Genetic Algorithm for the Solution of Complex Constrained Optimization Problems // Vestnik SibSAU. 2009. Iss. 5 (26). P. 31–36.

15. Галушин П. В. Разработка и исследование асимптотического вероятностного генетического алгоритма // Журнал Сибирского федерального университета. Серия: Математика и физика. 2012. № 1(5). С. 49–56.

16. Использование метода роя частиц для формирования состава мультиверсионного программного обеспечения / Ковалев И. В., Соловьев Е. В., Ковалев Д. И. и др. // Приборы и системы. Управление, контроль, диагностика. 2013. № 3. С. 1–6.

17. Semenkin E., Semenkina M. Stochastic Models and Optimization Algorithms for Decision Support in Spacecraft Control Systems Preliminary Design // Informatics in Control, Automation and Robotics, Lecture Notes in Electrical Engineering. 2014. Vol. 283. P. 51–65.

18. Semenkin E., Semenkina M. Self-Configuring Genetic Programming Algorithm with Modified Uniform Crossover Operator // Proceedings of the IEEE Congress on Evolutionary Computation. June 10–15, 2012.

Aab Andrey Vladimirovich – 2-year master's degree student; Reshetnev Siberian State University of Science and Technology.

Galushin Pavel Viktorovich – Cand. Sc., docent; Siberian Law Institute of Ministry of Internal Affairs of the Russian Federation.

Popova Anastasiya Valer'evna – 2-year master's degree student; Reshetnev Siberian State University of Science and Technology. E-mail: anastasiya.popowa@mail.ru.

Terskov Vitaly Anatolyevich – Dr. Sc., Professor; Reshetnev Siberian State University of Science and Technology.

Ааб Андрей Владимирович – магистр; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева.

Галушин Павел Викторович – кандидат технических наук, доцент; Сибирский юридический институт МВД России.

Попова Анастасия Валерьевна – магистр; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. E-mail: anastasiya.popowa@mail.ru.

Терсков Виталий Анатольевич – доктор технических наук, профессор; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева.