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

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 аппаратнопрограммных 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.

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

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

Введение. Одной из главных характеристик аппаратно-программных систем обработки информации реального времени является надёжность [1–3].

Под надёжностью программного обеспечения (далее также – 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 (ИС). Безотказностью программного обеспечения оценивается вероятность его работы без отказов при определённых условиях внешней среды в течение заданного периода наблюдения.

Разработка систем реального времени требует большого количества ресурсов на проектирование и тестирование. Одно 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 открывается возможность более гибкой оптимизации аппаратно-программных комплексов систем управления реального времени.

Подобные модели с развитием вычислительных мощностей, эволюционных алгоритмов и нейронных сетей обретают всё большую актуальность.

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

Надёжность программного обеспечения. Надёжность архитектуры программного обеспечения включает как надёжность центрального ядра системы, так и надёжность индивидуальных компонентов, предоставляемых пользователю. Сбой отдельного компонента может привести к неработоспособности этого и, возможно, других компонентов ПО. Однако, это не должно приводить к неработоспособности всей системы в целом. Тщательный анализ программной архитектуры позволяет выявить компоненты, ошибки в которых оказывают наиболее существенное влияние на надёжность системы. Как правило, это компоненты, наиболее часто используемые или архитектурно связанные с множеством других компонентов. Существует много методов повышения надёжности программного обеспечения [4-5], но в настоящее время только подход мультиверсионного отказоустойчивого программирования является возможной альтернативой методам тестирования и верификации программ, обеспечивая высокий уровень надёжности функционирования критичного программного обеспечения [6]. При этом важно понимать, что верификация не гарантирует корректности, так как сами спецификации и/или системы верификации (как и любое другое ПО) могут содержать ошибки.

Использование систем поддержки принятия решений при мультиверсионном программировании позволяет уделять основное внимание качеству требований на этапе формирования надёжного ПО. Однако улучшение характеристик надёжности ПО с использованием избыточности требует дополнительных временных и финансовых ресурсов.

Поэтому основной вопрос на этом этапе заключается в том, каким образом, используя избыточность в архитектуре ПО, максимизировать надёжность и снизить стоимость разработки. Данная область включает в себя методы многокритериального принятия решений, ориентирующиеся на задачи с дискретным пространством решений. К настоящему времени разработано множество методологий многокритериальной поддержки принятия решений, учитывающих различные уровни информации о предпочтениях эксперта. Использование различных методов определения глубины мультиверсионности и многокритериального принятия решений при выборе архитектуры позволяет спроектировать программную систему, отвечающую предъявляемым требованиям.

В зависимости от количества и величины компонентов условные и безусловные вероятности сбоя, доступа, анализа и времени восстановления, а также и времени использования компонентов различны. Модель [7], приведенная ниже, может использоваться, чтобы оценить надёжность программного обеспечения для возможных архитектурных изменений, выбрать надёжную архитектуру из различных вариантов и имеет следующие обозначения:

1) M – число архитектурных уровней в архитектуре ПО;
2) N_j – число компонентов на уровне j, $j \in \{1,..,M\}$;
3) D_{ij} – множество индексов компонентов, зависящих от компонента i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$;
4) F_{ij} – событие сбоя, произошедшего в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$;
5) PU_{ij} – вероятность использования компонента i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$;
6) PF_{ij} – вероятность появления сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$;
7) PL^{ij}_{nm} – условная вероятность появления сбоя в компоненте m на уровне n при появлении сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, $n \in \{1,..,N_m\}$, $m \in \{1,..,M\}$;
8) TA_{ij} – относительное время доступа к компоненту i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, определяемое как отношение среднего времени доступа к компоненту i на уровне j к числу сбойных компонентов на малых уровнях архитектуры за одно и то же время;
9) TC_{ij} – относительное время анализа сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, определяемое как отношение среднего времени анализа сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, к числу сбойных компонентов на всех уровнях архитектуры, анализируемых в одно и то же время;
10) TE_{ij} – относительное время устранения сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, определяемое как отношение среднего времени восстановления в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, к числу сбойных компонентов на всех уровнях архитектуры, в которых происходит устранение сбоев в одно и тоже время;
11) TU_{ij} – относительное время использования компонента i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, определяемое как отношение среднего времени использования компонента i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, к числу компонентов на всех уровнях архитектуры, используемых в одно и то же время;
12) TR – среднее время простоя системы в большой архитектуре ПО реального времени, определяемое как время, в течение которого система не может выполнять свои функции;
13) *MTTF (Mean Time to Failure)* – среднее время появления сбоя в большой архитектуре ПО реального времени, определяемое как время, в течение которого сбоев в системе не происходит.

В архитектуре реального ПО среднее время появление сбоя, то есть время, в течение которого программное обеспечение функционирует корректно, равно [7]:

$$MTTF = \sum_{j=1}^{j=M} \sum_{i=1}^{i=N_j} \{PU_{ij} \times (1 - PF_{ij}) \times [TU_{ij} + \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=N_m}$$
$$[(1 - PL_{nm}^{ij}) \times [TU_{nm} + \sum_{l \in D_{nm}} [(1 - PL_{lm}^{nm}) \times TU_{lm}]]] + \sum_{k \in D_{ij}} [(1 - PL_{kj}^{ij})$$
$$\times [TU_{kj} + \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=N_m} [(1 - PL_{nm}^{kj}) \times [TU_{nm} + \sum_{l \in D_{nm}} [(1 - PL_{lm}^{nm}) \times TU_{lm}]]]]]]\}$$

Среднее время простоя (восстановления) программного обеспечения равно [7]:

$$TR = \sum_{j=1}^{j=M} \sum_{i=1}^{i=M_j} \{PU_{ij} \times PF_{ij} \times [(TA_{ij} + TC_{ij} + TE_{ij}) + \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=N_n}$$
$$[PL_{nm}^{ij} \times [(TA_{nm} + TC_{nm} + TE_{nm}) + \sum_{l \in D_{nm}} [PL_{lm}^{nm} \times (TA_{lm} + TC_{lm} + TE_{lm})]]]$$
$$\sum_{k \in D_{ij}} [PL_{kj}^{ij} \times [(TA_{kj} + TC_{kj} + TE_{kj}) + \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=N_m} [PL_{nm}^{kj} [(TA_{nm} + TC_{nm} + TE_{nm}) +$$
$$+ \sum_{l \in D_{nm}} [PL_{lm}^{nm} \times (TA_{lm} + TC_{lm} + TE_{lm})]]]]]]\}$$

Среднее время простоя системы и среднее время сбоя могут быть использованы для предсказания надёжности программного обеспечения в целом. Для случая непрерывной эксплуатации сложного программного обеспечения (а для системы реального времени именно такой режим эксплуатации программного обеспечения является наиболее вероятным) надёжность программного обеспечения можно оценить с помощью коэффициента готовности *S*, вычисляемого по следующей формуле:

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

Коэффициент готовности можно интерпретировать как вероятность корректного функционирования ПО.

Основной подход к увеличению надёжности программного обеспечения, к которому предъявляются повышенные требования по непрерывности и корректности функционирования, – мультиверсионная разработка, то есть создание независимыми друг от друга разработчиками нескольких версий компонента программного обеспечения, соответствующих одним и тем же спецификациям, но отличающихся реализациями. В архитектуре программного обеспечения должен быть предусмотрен механизм формирования общего результата работы данного программного компонента на основе результатов работы каждой отдельной версии. В [8] приводятся два основных метода построения архитектуры при использовании мультиверсионной разработки, которые сокращённо называются *NVP* и *RB*. При построении мультиверсионного компонента из *K* версий методом мультиверсионного программирования (*NVP*, *N*-version programming) для любого *K* надёжность равна:

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

где p^{v}_{ij} – вероятность сбоя алгоритма голосования для компонента i на уровне j, Z_{ij} – множество версий этого компонента, а p^{k}_{ij} – вероятность сбоя версии $k \in Z_{ij}$.

При построении мультиверсионного компонента из K версий методом блока восстановления (*RB*, recovery block) надёжность равна:

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

где p^{AT}_{ij} – вероятность безотказной работы приемочного теста для компонента i на уровне j, p^{k}_{ij} – вероятность сбоя версии $k \in Z_{ij}$.

Две последние формулы могут быть использованы для вычисления вероятностей отказов компонентов программного обеспечения:

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

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

Каждый процессор и шина выходят из строя в некоторые случайные моменты времени, после чего начинается их восстановление. Будем предполагать, что потоки отказов и восстановлений являются простейшими. Интенсивность отказов шин обозначим как ν_0, интенсивность потока восстановления шин – μ_0. Интенсивность потоков отказов процессоров обозначим ν_i, а интенсивность восстановления процессоров i-го типа – μ_i.

Вероятность $P_{j0,j1,\ldots,jN}$ нахождения системы в состоянии, в котором j_0 шин интерфейса исправны и участвуют в вычислительном процессе, а $(m_0 - j_0)$ неисправны и восстанавливаются, j_1 процессоров 1-го типа исправны и участвуют в вычислительном процессе, а $(m_1 - j_1)$ неисправны и восстанавливаются, ..., j_N процессоров N-го типа исправны и участвуют в вычислительном процессе, а $(m_N - j_N)$ неисправны и восстанавливаются, определяется по следующей формуле [9-10]:

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

где $k = \sum_{i=0}^{N} m_i$, а суммирование в знаменателе ведётся по всем возможным значениям индексов $j_0,\ldots,j_N$ (это же будет подразумеваться во всех последующих аналогичных формулах). Кроме того, используется следующее обозначение:

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

Формула для вероятностей состояний в стационарном режиме может быть несколько упрощена. Во-первых, мы можем сократить $k!$ в числителе и знаменателе, а также объединить произведения в числителе и знаменателе:

$$P_{j_0,j_1,\ldots,j_N} = \frac{\prod_{i=0}^{N} \frac{m_i!}{(m_i - j_i)!j_i!}\rho_i^{j_i}}{\sum_{j_0,\ldots,j_N} \prod_{i=0}^{N} \frac{m_i!}{(m_i - j_i)!j_i!}\rho_i^{j_i}},$$

Теперь перепишем знаменатель так, чтобы сначала выполнялось суммирование, а потом – вычисление произведения:

$$P_{j_0,j_1,\ldots,j_N} = \frac{\prod_{i=0}^{N} \frac{m_i!}{(m_i - j_i)!j_i!}\rho_i^{j_i}}{\prod_{i=0}^{N} \sum_{j_i=0}^{m_i} \frac{m_i!}{(m_i - j_i)!j_i!}\rho_i^{j_i}},$$

Каждая из сумм в знаменателе теперь представляет собой произведение биномиального коэффициента и соответствующей степени фиксированной величины, и поэтому может быть упрощена с использованием формулы бинома Ньютона следующим образом [11]:

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

Таким образом, окончательно получаем следующее выражение для вероятностей состояний аппаратного обеспечения в установившемся режиме:

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

Упрощённые формулы содержат меньше величин, чем исходные, и, следовательно, позволяют избежать выполнения некоторых ненужных операций (например, умножения и числителя, и знаменателя на $k!$, что, очевидно, не оказывает влияние на значение выражения) и требуют выполнения существенно меньшего количества операций для вычисления значения знаменателя, чем исходные формулы, приведённые в работах [9–10].

Зная минимальную конфигурацию аппаратного обеспечения, необходимую для выработки программным обеспечением системы реального времени управляющего воздействия в рамках требуемых заказчиком интервалов, можно вычислить надёжность аппаратной части. Действительно, если количество безотказно функционирующих шин оперативной памяти и процессоров превышает минимально требуемое, то «излишние» компоненты программного обеспечения можно рассматривать как резерв. Тогда надёжность аппаратного обеспечения системы управления реального времени, понимаемая как вероятность обеспечения минимальной требуемой производительности аппаратного обеспечения, может быть вычислена как сумма вероятностей нахождения аппаратного обеспечения в состояниях, в которых количество безотказно функционирующих шин оперативной памяти и процессоров превышает минимальное требуемое количество:

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

где M_0 – минимальное требуемое для обеспечения заданной производительности число исправных шин памяти, M_i – минимальное требуемое для обеспечения заданной производительности число исправных процессоров i-го типа.

Формула для G_{cp} подразумевает вычисление вероятностей различных состояний аппаратного обеспечения с различными значениями индексов $j_0, j_1, \dots j_N$. При этом не целесообразнее повторять вычисления, которые уже были выполнены. Например, знаменатель всех формул имеет одно и то же значение, а, следовательно, его достаточно вычислить один раз.

Кроме того, так как требуется вычисление состояний с последовательными значениями индексов, то можно воспользоваться следующими рекуррентными формулами для степеней и биномиальных коэффициентов, следующих из определений степени и факториала:

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

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

Данные соотношения позволяют вычислять вероятности для состояний с последовательными значениями индексов с минимальным количеством дополнительных операций:

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

Из этого соотношения элементарными преобразованиями можно получить и формулы «уменьшения индекса»:

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

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

Так как наиболее простой вид формулы для вероятностей состояний принимают в случаях, когда все шины и процессоры исправны, либо когда все они не исправны, то вычисления G_{cp} целесообразно начинать с вероятности состояния, в котором все шины оперативной памяти и процессоры исправны:

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

а затем, используя формулы «уменьшения индексов», последовательно вычислить остальные необходимые вероятности состояний.

Ещё одна возможность для улучшения открывается, если минимальная конфигурация, обеспечивающая заданную производительность, включает меньше аппаратных компонент, чем половина компонент, присутствующих в системе. В этом случае можно вычислить сумму вероятностей состояний, в которых производительность не достаточна для выработки управляющего воздействия в рамках заданных временных ограничений, а затем вычислить требуемую вероятность как вероятность противоположного события [12]:

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

При вычислении по этой формуле следует начать с вероятности состояния, в котором все шины оперативной памяти и процессоры неисправны:

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

а затем использовать формулы увеличения индексов, полученные выше.

Модель надёжности аппаратно-программного комплекса. Наконец, мы можем, используя приведённые выше соображения, объединить модели надёжности программного обеспечения и аппаратного обеспечения в общую модель надёжности многопроцессорного аппаратно-программного комплекса системы управления реального времени с мультиверсионным программным обеспечением.

В силу абстрактного характера программного обеспечения и пренебрежимо малой вероятности ошибок при его копировании и распространении (кроме того, целостность программного обеспечения может быть проконтролирована с помощью таких механизмов как контрольные суммы), можно считать, что отказы аппаратной и программной части происходят независимо.

Поэтому вероятность одновременной безотказной работы аппаратно-программного комплекса равна произведению вероятностей безотказной работы программной и аппаратной частей [12]:

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

Заключение. Таким образом, мы получили модель расчета надёжности многопроцессорных аппаратно-программных комплексов систем реального времени с разнородными процессорами и мультиверсионным программным обеспечением, используя теорию массового обслуживания и теорию надёжности, которая позволяет рассматривать множество вариантов архитектуры за короткое время и без существенных затрат, характерных для построения опытных образцов и оценки надёжности путём организации опытной эксплуатации.

Предложенная математическая модель может быть использована для автоматизации проектирования многопроцессорных аппаратно-программных комплексов. На практике стремятся к тому, чтобы проектируемая система управления реального времени обладала наибольшей возможной надёжностью при условии, что затраты на её создание и эксплуатацию не превосходят выделенные средства. Таким образом, мы приходим к задаче условной или многокритериальной оптимизации, в которой целевая функция выражается через вероятности состояний, рассчитываемых в рамках предложенной модели. Несмотря на то, что для показателей надёжности многопроцессорного аппаратно-программного

комплекса существуют аналитические выражения, данная задача оптимизации обладает рядом неудобных особенностей: оптимизируемые переменные являются дискретными, наличие единственного экстремума и удобных для оптимизации свойств (выпуклость) не гарантировано, а объём пространства поиска быстро растёт с ростом количества типов процессоров.

При решении подобных задач оптимизации хорошо зарекомендовали себя эволюционные методы оптимизации, например, генетический алгоритм [13], вероятностный генетический алгоритм [14] или асимптотический вероятностный генетический алгоритм [15], а также некоторые другие «вдохновлённые природой» методы оптимизации, например, метод роя частиц [16]. Недостатком методов эволюционной оптимизации является наличие у них большого количества настраиваемых параметров, которые могут существенно влиять на качество найденных решений. Причём разные наборы параметров могут быть эффективны для различных задач оптимизации. Таким образом, использование эволюционных методов оптимизации требует, как правило, экспериментирования и привлечения специалиста в области эволюционных методов оптимизации.

Для того, чтобы исключить этап подбора эффективного сочетания параметров, может использоваться самонастройка [17-18]. Самонастраивающиеся методы оптимизации используют несколько наборов параметров, вычислительные ресурсы (для простоты можно считать, что это количество вычислений целевой функции) между которыми распределяются в зависимости от качества получаемых с их помощью решений. В начале процесса оптимизации все наборы параметров получают одинаковые ресурсы. Затем наборы параметров, которые порождают лучшие решения, получают дополнительные вычислительные ресурсы за счёт тех, которые показывают себя хуже. Существует некоторая нижняя граница для выделяемых ресурсов, это делается для того, чтобы любой набор параметров мог проявить себя в будущем, когда условия изменятся, так как различные этапы оптимизации могут требовать различных значений параметров.

Исследование эффективности указанных методов оптимизации при решении задачи оптимизации надёжности многопроцессорных аппаратно-программных комплексов систем управления реального времени с мультиверсионным программным обеспечением и разнородными процессорами является возможным направлением дальнейших исследований.

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

6. Avizienis A. The N-Version Approach to FaultTolerant 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, AddisonWesley, 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.

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' apparatnoprogrammnyh 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.

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.

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.