УДК 004.052.32
Doi: 10.31772/2712-8970-2021-22-3-459-467

Для цитирования: Повышение надежности программного обеспечения для распределенных систем управления / О. Д. Стрелавина, С. Н. Ефимов, В. А. Терсков, М. А. Лихарев // Сибирский аэрокосмический журнал. 2021. Т. 22, № 3. С. 459–467. Doi: 10.31772/2712-8970-2021-22-3-459-467.

For citation: Strelavina O. D., Efimov S. N., Terskov V. A., Likharev M. A. Increasing software reliability of a distributed control systems. *Siberian Aerospace Journal.* 2021, Vol. 22, No. 3, P. 459–467. Doi: 10.31772/2712-8970-2021-22-3-459-467.

Повышение надежности программного обеспечения для распределенных систем управления

О. Д. Стрелавина, С. Н. Ефимов*, В. А. Терсков, М. А. Лихарев

Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева
Российская Федерация, 660037, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
*E-mail: efimov@bk.ru

Рассматривается подход для оценки и улучшения основных параметров эффективности вычислительной сети. Для распределенных систем управления надежность, при обеспечении требуемой производительности, является главным критерием. Для повышения надежности функционирования вычислительной сети вводится как аппаратная, так и программная избыточность. Для обеспечения программной избыточности разрабатываются новые версии для тех модулей программного обеспечения (ПО), в которых возможны программные сбои. Рассматривается применение методов N-версионного программирования и блока восстановления для введения программной избыточности, а также оцениваются затраты на разработку сетевого ПО с учетом мультиверсионности. Для реализации предлагаемого подхода приводится математическая модель оценки надежности ПО, которая учитывает архитектуру программного обеспечения вычислительной сети и затраты на его разработку. На основе данной модели создана программная система для проведения исследования программной надежности вычислительной сети, с помощью которой можно находить зависимость надежности сетевого программного обеспечения (СПО) от количества версий одного из выделенных программных модулей. Сравнение динамики изменения показателей надежности СПО и трудовых затрат специалистов на его разработку указывает на достаточное количество новых версий для тех модулей СПО, программную надежность которых необходимо повысить на этапе проектирования. Делается вывод о значимости как определения параметра трудозатрат на разработку СПО, так и его использования при проектировании вычислительной сети, в которой надежность повышается методом программной избыточности.

Ключевые слова: надежность вычислительной сети, надежность программного обеспечения, программная избыточность, модель надежности, трудозатраты.

Increasing software reliability of a distributed control systems

O. D. Strelavina, S. N. Efimov*, V. A. Terskov, M. A. Likharev

Reshetnev Siberian State University of Science and Technology
31, Krasnoyarskii rabochii prospekt, Krasnoyarsk, 660037, Russian Federation
*E-mail: efimov@bk.ru

The article considers a method of assessing and improving main parameters of the computer network efficiency. Reliability is the main criteria for ensuring the required performance of distributed control systems. To improve reliability of the computer network hardware and software redundancy are being used. Software redundancy requires new versions to be developed for software modules in which failures are likely to occur. The article considers the N-version programming and recovery block as methods of introducing software redundancy and, taking the need to develop multiple versions of the same software module into account, estimates the costs of network software development. To implement the proposed approach article presents mathematical reliability model that takes into consideration the architecture of a computer network software and the labor costs that its development is going to require. This model becomes a basis for a software created to research computer network software reliability, which allows to find the dependance of network software reliability on the number of one of its software module versions. Comparison of the changes dynamics of reliability indicators and labor intensity of software development indicated a sufficient amount of software module versions that need to be developed. The article concludes by pointing out the importance of determining the labor intensity of network software development and of its usage in the design of a computer networks in which reliability is being increased through software redundancy.

Keywords: computer network reliability, software reliability, software redundancy, reliability model, labor intensity.

Введение

Качество любой вычислительной сети (ВС) можно оценить, используя ее основные характеристики. К ним относятся полнота выполняемых функций, производительность, пропускная способность, надежность, безопасность, прозрачность, масштабируемость и универсальность [1]. Ни один из этих критериев нельзя однозначно назвать самым важным, но среди них можно выделить несколько наиболее весомых. Одним из таких критериев является надежность [2–5].

Надежность – это способность ВС безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью [6–8]. При низком уровне надежности ВС будет затронут и функционал систем, отвечающих за показатели других критериев качества. Поэтому обеспечение надежности ВС является приоритетной задачей при построении компьютерной сети [9].

Полностью искоренить возможность возникновения сбоев невозможно, а потому обеспечение надежности заключается в уменьшении количества ошибок, с которыми может столкнуться пользователь при функционировании сети. Одним из наиболее проверенных и зарекомендовавших себя способов повышения надежности является введение избыточности [10; 11].

В аппаратной части ВС избыточность служит для борьбы с периодически выходящими из строя процессорами и шинами. Аппаратная избыточность вводится путем резервирования процессорных элементов и шин интерфейса. Необходимое количество продублированных аппаратных компонентов разнится и зависит от интенсивности отказов и времени восстановления. Расчет оптимального количества этих компонентов является главной сложностью введения аппаратной избыточности.

Программную избыточность нельзя достигнуть с помощью дублирования – ошибки, возникающие в программных модулях, имеют внутреннюю природу [12], что приводит к появлению тех же ошибок в идентичных копиях. Поэтому вместо копий необходимо создавать новые версии, отличающиеся друг от друга языком программирования, разработавшими их программистами, использованными алгоритмами. Благодаря внутренним отличиям разных версий вероятность возникновения аналогичных сбоев минимизируется [13; 14].

Программная избыточность не применяется ко всей программе или пакету программ – ее используют для повышения надежности модулей, которые критически важны для функционирования всей сети в целом или к которым чаще всего обращаются пользователи и другие модули [15]. Как и в случае с аппаратной избыточностью, количество вводимых версий программных модулей нужно подбирать для каждой отдельной сети, что в сочетании с трудозатратами на разработку новых версий указывает на нетривиальную задачу эффективного использования программной избыточности. Для проведения исследования по данной проблеме используем математическую модель для оценки надежности ПО.

Модель оценки надежности ПО

На надежность СПО влияют его иерархические уровни – зависимость программных модулей друг от друга может привести к тому, что сбой в одном модуле распространится по архитектуре всего СПО [16]. Модель, описывающая надежность СПО, должна учитывать влияние иерархии программных модулей на сбои и время простоя [17].

Обозначения, применяемые в модели:

1) M – количество архитектурных уровней в архитектуре СПО;

2) N_j – количество модулей на уровне j, $j \in \{1, \ldots, M\}$;

3) D_{ij} – множество индексов модулей, зависящих от модуля i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$;

4) F_{ij} – событие сбоя, произошедшего в модуле i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$;

5) PU_{ij} – вероятность использования модуля i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$;

6) PF_{ij} – вероятность появления сбоя в модуле i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$;

7) PL_{nm}^{ij} – условная вероятность появления сбоя в модуле m на уровне n при появлении сбоя в модуле i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$, $n \in \{1, \ldots, N_m\}$, $m \in \{1, \ldots, M\}$;

8) TA_{ij} – относительное время доступа к модулю i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$, определяемое как отношение среднего времени доступа к модулю i на уровне j к числу сбойных модулей на малых уровнях архитектуры за одно и тоже время;

9) TC_{ij} – относительное время анализа сбоя в модуле i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$, определяемое как отношение среднего времени анализа сбоя в модуле i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$, к числу сбойных модулей на всех уровнях архитектуры, анализируемых в одно и тоже время;

10) TE_{ij} – относительное время устранения сбоя в модуле i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$, определяемое как отношение среднего времени восстановления в модуле i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$, к числу сбойных модулей на всех уровнях архитектуры, в которых происходит устранение сбоев в одно и тоже время;

11) TU_{ij} – относительное время использования модуля i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$, определяемое как отношение среднего времени использования модуля i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, \ldots, M\}$, к числу модулей на всех уровнях архитектуры, используемых в одно и тоже время;

12) Z_{ij} – множество версий модуля i, на уровне j, $k = 1, \ldots, K$;

13) T_{ij} – трудоемкость разработки модуля i на уровне j;

14) T_{ij}^k – трудоемкость разработки версии k модуля i на уровне j, $k \in Z_{ij}$ в чел-часах;

15) NVX_{ij} – трудоемкость разработки приемочного теста (для RB) или алгоритма голосования (для NVP);

16) T_s – общая трудоемкость сети;

17) B_{ij} – дихотомическая переменная, принимающая значение 1 (тогда $NVP_{ij} = 0$, $RB_{ij} = 0$), если в программном модуле не используется программная избыточность, иначе равна 0;

18) NVP_{ij} – дихотомическая переменная, принимающая значение 1 (тогда $B_{ij} = 0$, $RB_i = 0$), если в программном модуле используется программная избыточность по методу *N*-версионного программирования, иначе равна 0;

19) RB_{ij} – дихотомическая переменная, принимающая значение 1 (тогда $B_{ij} = 0$, $NVP_{ij} = 0$), если в программном модуле используется программная избыточность по методу блока восстановления, иначе равна 0;

20) TR – среднее время простоя СПО ВС, определяемое как время, в течение которого система не может выполнять свои функции;

21) $MTTF$ – среднее время появления сбоя в СПО ВС, определяемое как время, в течение которого сбоев в системе не происходит;

22) S – коэффициент готовности СПО ВС;

23) T_s – общая трудоемкость реализация СПО.

Среднее время сбоя сетевого программного обеспечения вычислительной сети равно:

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

$$+\sum_{(m=1)\&(m\neq j)}^{M}\sum_{n=1}^{N_m}\left(\big(1-PL_{nm}^{ij}\big)\times\left(TU_{nm}+\sum_{l\in D_{nm}}\Big(\big(1-PL_{lm}^{nm}\big)\times TU_{im}\Big)\right)\right)+$$

$$+\sum_{k\in D_{ij}}\left(\big(1-PL_{kj}^{ij}\big)\times\left(TU_{kj}+\sum_{(m=1)\&(m\neq j)}^{M}\sum_{n=1}^{N_m}\Big(\big(1-PL_{nm}^{kj}\big)\times\big(TU_{nm}+\right.\right.$$

$$\left.\left.+\sum_{l\in D_{nm}}\Big(\big(1-PL_{lm}^{nm}\big)\times TU_{lm}\Big)\big)\big)\big)\right)\Big]\Big\}.$$

Среднее время простоя сетевого программного обеспечения вычислительной сети равно:

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

$$+\sum_{(m=1)\&(m\neq j)}^{M}\sum_{n=1}^{Nm}\left(PL_{nm}^{ij}\times\left(\big(TA_{nm}+TC_{nm}+TE_{nm}\big)+\sum_{l\in D_{nm}}\Big(PL_{lm}^{nm}\times\big(TA_{lm}+TC_{lm}+TE_{lm}\big)\Big)\right)\right)\times$$

$$\times\sum_{k\in D_{ij}}\Big[PL_{kj}^{ij}\times\Big[\big(TA_{kj}+TC_{kj}+TE_{kj}\big)+$$

$$+\sum_{(m=1)\&(m\neq j)}^{M}\sum_{n=1}^{N_m}\left(PL_{nm}^{kj}\times\left(\big(TA_{nm}+TC_{nm}+TE_{nm}\big)+\sum_{l\in D_{nm}}\Big(PL_{lm}^{nm}\times\big(TA_{lm}+TC_{lm}+TE_{lm}\big)\Big)\right)\right)\Big]\Big]\Big]\Big\}.$$

Обе эти формулы учитывают иерархию модулей и потому являются универсальными для любого СПО с архитектурными уровнями. По ним также определяется коэффициент готовности программной части ВС:

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

Сами по себе данные показатели рассматривают надежность СПО ВС лишь в ее начальном состоянии. В [18] описаны два основных метода введения мультиверсионности: *NVP* (*N*-version programming) и *RB* (recovery block).

NVP подразумевает, что все версии программы выполняются параллельно, а результат их работы определяется с помощью алгоритма голосования [19]. Надёжность мультиверсионного модуля *i* на уровне *j*, построенного из *K* версий методом мультиверсионного программирования для любого *k*, равна

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

где p_{ij}^{v} – вероятность безотказной работы алгоритма голосования; p_{ij}^{k} – вероятность безотказной работы версии $k \in Z_{ij}$.

При подходе *RB* мультиверсионность вводится через добавление нескольких версий вычислительного модуля, создание приёмочного теста, проверяющего работу версий, и подпрограммы, которая, опираясь на результаты теста, либо принимает результат работы модуля, либо выбирает другую версию и перезапускает вычисление [19]. Надежность такого модуля:

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

где p_{ij}^{AT} – вероятность безотказной работы приемочного теста для модуля *i*, $i = 1, \ldots, N$ на уровне $j, j = 1, \ldots, M$; p_{ij}^{k} – вероятность сбоя версии $k \in Z_{ij}$.

Для любого из подходов вероятность сбоя будет рассчитываться, как

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

При введении программной избыточности всегда встает вопрос: сколько именно версий нужно вводить? Возможность применения мультиверсионности ограничена множеством факторов. Одним из наиболее весомых из них является трудоемкость, отражающая трудозатраты на постройку сети, напрямую зависящие от количества версий программных модулей, которые необходимо разработать. Для отражения этого ограничения в модели есть формула [20]:

$$T_s = \sum_{j=1}^{M}\sum_{i=1}^{N_i}\left[B_{ij}T_{ij} + \left(NVP_{ij} + RB_{ij}\right) \times \left(NVX_{ij} + \sum_{k \in Z_{ij}}^{K} T_{ij}^{k}\right)\right].$$

При исследовании ВС на этот показатель можно опираться для нахождения оптимального количества версий для каждого программного модуля. Исследовать сеть можно разными способами, но у всех них одна суть: трудоемкость T_s должна стремиться к минимуму, а коэффициент готовности S – к максимуму.

Исследование надежности СПО

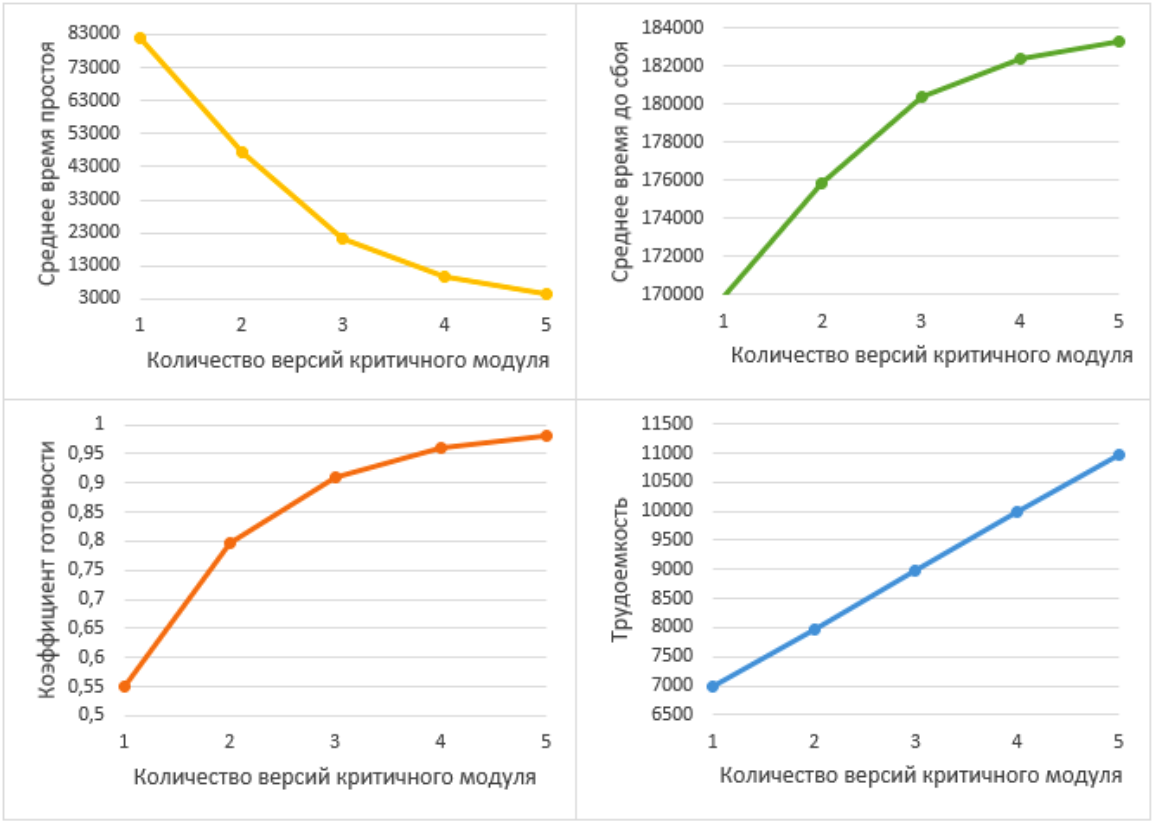
На основе рассмотренной модели разработана программная система, позволяющая рассчитывать надежность СПО и проводить исследования по указанным входным данным.

Для проведения исследования возьмем СПО, состоящее из 10 модулей. Предположим, что все модули, за исключением одного, работают без сбоев (т. е. имеют надежность равную 1). Общая надежность ПО будет определяться надежностью критичного модуля со значением 0,55. В него вводится программная избыточность методом *NVP*, трудоемкость разработки новых версий этого модуля берется за 1000.

В таблице представлены результаты зависимости показателей надежности СПО от версий одного из его модулей, а на рисунке наглядно отображена динамика изменения параметров.

Результаты исследования СПО

Количество версий критичного модуля	Среднее время простоя	Среднее время до сбоя	Коэффициент готовности	Трудоемкость
1	82115	169838	0,5500	6981
2	47509	175844	0,7975	7981
3	21373	180379	0,9089	8981
4	9619	182420	0,9590	9981
5	4340	183336	0,9815	10981



Динамика изменения параметров ПО ВС

Computer network software parameters' change dynamics

Заключение

Полученные графики позволяют определить то количество новых версий выделенного модуля, при котором рост стоимости разработки нового элемента ПО начинает значительно превышать рост программной надежности ПО. В какой-то момент введение новых версий перестает быть целесообразным, так как затраты начинают перевешивать получаемую отдачу.

В выбранной для примера ситуации надежность всего СПО зависит от одного его модуля. При проектировании реального сетевого программного обеспечения таких модулей будет значительно больше, что многократно усложнит задачу исследования. Это подчеркивает значимость параметра трудоемкости, который определяется моделью наряду с надежностью СПО. Сравнивая рост трудозатрат с ростом программной надежности сети, можно увидеть, в какой момент следует остановить добавление новых версий программных модулей.

Таким образом, был рассмотрен подход, позволяющий как оценивать, так и улучшать такие параметры СПО, как надежность и трудозатраты. Для этого на основе математической модели оценки надежности и трудозатрат на разработку было создано программное обеспечение для исследования зависимости надежности СПО и трудозатрат на его разработку от количества версий для проблемных, с точки зрения надежности, модулей ПО. Проанализирована динамика изменения параметров надежности сетевого программного обеспечения и трудозатрат на его разработку от количества версий одного из его программных модулей. Из полученных результатов можно сделать вывод о важности учета параметра трудоемкости при исследовании надежности сети, в которую вводится программная избыточность.

Библиографические ссылки

1. Кузин А. В. Компьютерные сети. М. : Форум: Инфра-М, 2011. 192 с.

2. Макарук Р. В., Гиляров В. Н. Нечёткие модели и программный комплекс для анализа характеристик вычислительной сети // Научные ведомости белгородского государственного университета. Серия: экономика. Информатика. 2013. № 22. С. 161–166.

3. Ефимов С. Н., Тынченко В. В., Тынченко В. С. Проектирование вычислительной сети эффективной архитектуры для распределенного решения сложных задач // Вестник СибГАУ. 2007. № 3 (16). С. 15–19.

4. Ефимов С. Н. Оценка надежности распределенных автоматизированных систем управления технологическим процессом // Промышленные АСУ и контроллеры. 2011. № 9. С. 9–13.

5. Расулов М. М. Оценка надежности программного обеспечения // Актуальные научные исследования в современном мире. 2020. № 6 (62). С. 112–116.

6. Ложков А. В. Методика оценки надежности вычислительной сети // Научные записки молодых исследователей. 2014. № 4. С. 28–31.

7. Гуров С. В., Половко А. М. Основы теории надежности. СПб. : БХВ-Петербург, 2006. 704 с.

8. Бржозовский Б. М., Мартынов В. В., Схиртладзе А. Г. Диагностика и надежность автоматизированных систем. М. : ТНТ, 2013. 352 с.

9. Воротникова Т. Ю. Исследование развития вопроса повышения надежности программного обеспечения // Globus. 2019. № 11 (44). С. 42–45.

10. Шубинский И. Б. Надежные отказоустойчивые информационные системы. Методы синтеза. Ульяновск : Печатный двор, 2016. 547 с.

11. Грузенкин Д. В., Камысов С. С. Применение программной избыточности для повышения надежности программного обеспечения // Новая наука: От идеи к результату. 2016. № 9. С. 9–11.

12. Наумов А. А., Айдинян А. Р. Надежность программного обеспечения и методы ее повышения // Инженерный вестник Дона. 2018. № 2 [Электронный ресурс]. URL: http://ivdon.ru/ru/magazine/archive/n2y2018/4946 (дата обращения: 10.05.2021).

13. Ковалев П. В. Определение надежности мультиверсионного программного обеспечения с использованием методов анализа сетей // Вестник СибГАУ. 2009. № 1-2. С. 56–59.

14. Поздняков Д. А. Компонентная программная архитектура мультиверсионных систем обработки информации и управления : дис. канд. техн. наук. Красноярск, 2006. 126 с.

15. Тынченко В. В., Царев Р. Ю. К вопросу оценки надежности программного обеспечения с многоуровневой архитектурой [Электронный ресурс] // Современные проблемы науки и образования. 2015. № 2-1. URL: http://science-education.ru/ru/article/view?id=20878 (дата обращения: 18.04.2021).

16. Караванов А. В., Иванов Н. Д. Архитектура программного обеспечения для высоконадежных систем // Космические аппараты и технологии. 2018. № 2. С. 100–104.

17. Русаков М. А. Многоэтапный анализ архитектурной надежности в сложных информационно-управляющих системах : дис. канд. техн. наук. Красноярск, 2005. 168 с.

18. Новой А. В. Система анализа архитектурной надежности программного обеспечения : дис. канд. техн. наук. Красноярск, 2011. 131 с.

19. Ковалев И. В., Новой А. В. Расчет надежности отказоустойчивых архитектур программного обеспечения // Вестник СибГАУ. 2007. № 4. С. 14–17.

20. Шеенок Д. А. Многокритериальная оптимизация отказоустойчивой программной архитектуры специализированными эволюционными алгоритмами: дис. канд. техн. наук. Красноярск, 2013. 84 с.

References

1. Kuzin A. V. *Komp'yuternye seti* [Computer networks]. Moscow, Forum: Infra-M Publ., 2011, 192 p.

2. Makaruk R. V. [Fuzzy models and a software package for analyzing the characteristics of a computer network]. *Nauchnye vedomosti belgorodskogo gosudarstvennogo universiteta. Seriya: ekonomika. informatika,*. 2013, No. 22, P. 161–166 (In Russ.).

3. Efimov S. N., Tynchenko V. V., Tynchenko V. S. [Design of computing network with efficient architecturefor complex problems distributed solving]. *Vestnik SibGAU*. 2007. No. 3, P. 15–19 (In Russ.).

4. Efimov S. N. [The industrial distributed control system reliability estimation]. *Promyshlennye ASU i kontrollery.* 2011, No. 9, P. 15–19 (In Russ.).

5. Rasulov M. M. [Software reliability assessment]. *Aktual'nye nauchnye issledovaniya v sovremennom mire.* 2020, No. 6, P. 112–116 (In Russ.).

6. Lozhkov A. V. [Methodology for assessing the reliability of a computer network]. *Nauchnye zapiski molodykh issledovateley.* 2014, No. 4, P. 28–31 (In Russ.).

7. Gurov S. V., Polovko A. M. *Osnovy teorii nadezhnosti* [Fundamentals of the theory of reliability]. St.Petersburg, BKhV-Peterburg Publ., 2006, 704 p.

8. Brzhozovskiy B. M., Martynov V. V., Skhirtladze A. G. *Diagnostika i nadezhnost' avtomatizirovannykh sistem* [Diagnostics and reliability of automated systems]. Moscow, TNT Publ., 2013, 352 p.

9. Vorotnikova T. Y. [Research of the development of increasing the software reliability issue]. *Globus*. 2019, No. 11, P. 42–45 (In Russ.).

10. Shubinskiy I. B. *Nadezhnye otkazoustoychivye informatsionnye sistemy. Metody sinteza* [Reliable fault-tolerant information systems. Synthesis methods]. Ulyanovsk, Pechatnyy dvor Publ., 2016, 547 p.

11. Gruzenkin D. V., Kamysov S. S. [Application of software redundancy to increase software reliability]. *Novaya nauka: Ot idei k rezul'tatu*. 2016, No. 9, P. 9–11 (In Russ.).

12. Naumov A. A., Aydinyan A. R. [Software reliability and methods to improve it: Don's Engineering Bulletin]. *Nadezhnost' programmnogo obespecheniya i metody ee povysheniya. Inzhenernyy vestnik Dona.* 2018, No. 2. (In Russ.). Available at: http://ivdon.ru/ru/magazine/archive/n2y2018/4946 (accessed 10.05.2021).

13. Kovalev P. V. [The reliability research of n-version software using methods of network analysis]. *Vestnik SibGAU*. 2009, Vol. 22, № 1-2, P. 56–59 (In Russ.).

14. Pozdnyakov D. A. *Komponentnaya programmnaya arkhitektura mul'tiversionnykh sistem obrabotki informatsii i upravleniya. Dis. kand. tekhn. nauk.* [Component software architecture of multiversion information processing and control systems. Cand. techn. sci. diss.]. Krasnoyarsk, 2006, 126 p.

15. Tynchenko V. V., Tsarev R. Yu. [Toward the problem of evaluation of the reliability of software with multiple level architecture]. *K voprosu otsenki nadezhnosti programmnogo obespecheniya s mnogourovnevoy arkhitekturoy. Sovremennye problemy nauki i obrazovaniya.* 2015, No. 2-1 (In Russ.). Available at: http://science-education.ru/ru/article/view?id=20878 (accessed: 18.04.2021)

16. Karavanov A. V., Ivanov N. D [Software architecture for highly reliable systems]. *Kosmicheskie apparaty i tekhnologii*. 2018, No. 2, P. 100–104 (In Russ.).

17. Rusakov M. A. *Mnogoetapnyy analiz arkhitekturnoy nadezhnosti v slozhnykh informatsionno-upravlyayushchikh sistemakh. Dis. kand. tekhn. nauk* [Multi-stage analysis of architectural reliability in complex information management systems. Cand. techn. sci. diss.]. Krasnoyarsk, 2005, 168 p.

18. Novoy A. V. *Sistema analiza arkhitekturnoy nadezhnosti programmnogo obespecheniya. Dis. kand. tekhn. nauk.* [Software architectural reliability analysis system. Cand. techn. sci. diss.]. Krasnoyarsk, 2011, 131 p.

19. Kovalev I. V., Novoy A. V. [Software architecture for highly reliable systems]. *Vestnik SibGAU*. 2007, No. 4, P. 14–17 (In Russ.).

20. Sheenok D. A. *Mnogokriterial'naya optimizatsiya otkazoustoychivoy programmnoy arkhitektury spetsializirovannymi evolyutsionnymi algoritmami. Dis. kand. tekhn. nauk.* [Multi-criteria optimization of fail-safe software architecture by specialized evolutionary algorithms. Cand. techn. sci. diss.]. Krasnoyarsk, 2013, 84 p.

Стрелавина Олеся Денисовна – магистрант информатики и вычислительной техники; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. E-mail: strelavi@mail.ru.

Ефимов Сергей Николаевич – кандидат технических наук, доцент, доцент кафедры информационно-управляющих систем; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. E-mail: efimov@bk.ru.

Терсков Виталий Анатольевич – доктор технических наук, профессор, профессор кафедры информационно-управляющих систем; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. E-mail: terskovva@mail.ru.

Лихарев Михаил Андреевич – магистрант информатики и вычислительной техники; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. E-mail: misha.likharev@inbox.ru.

Strelavina Olesya Denisovna – magistrand of computer science; Reshetnev Siberian State University of Science and Technology. E-mail: strelavi@mail.ru.

Efimov Sergei Nikolaevich – Cand. Sc., associate professor of the department of information management systems; Reshetnev Siberian State University of Science and Technology. E-mail: efimov@bk.ru.

Terskov Vitaliy Anatol'evich – Dr. Sc., professor, professor of the department of information management systems; Reshetnev Siberian State University of Science and Technology. E-mail: terskovva@mail.ru.

Likharev Mikhail Andreevich – magistrand of computer science; Reshetnev Siberian State University of Science and Technology. E-mail: misha.likharev@inbox.ru.