УДК 519.872

Сибирский журнал науки и технологий. 2017. Т. 18, № 4. С. 773–778

МОДЕЛЬ ФУНКЦИОНИРОВАНИЯ ПРОГРАММНОЙ СИСТЕМЫ НА ОСНОВЕ GERT-СЕТИ

Т. А. Панфилова, И. А. Панфилов*, В. В. Золотарев, И. В. Ковалев, Е. А. Сопов

Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева
Российская Федерация, 660037, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
*E-mail: crook_80@mail.ru

Предлагается оригинальный подход к оценке надежности программного обеспечения с помощью модели GERT-сети. Такой подход позволяет моделировать надежность программных комплексов, состоящих из нескольких взаимодействующих программных компонентов. В качестве исходных данных для оценки надежности используются значения оценок надежности отдельных блоков. Такие оценки могут назначаться экспертом, а могут быть получены в результате исследований самого программного комплекса. Были проведены экспериментальные исследования с программной системой «Протокол безопасного обмена данными». Были получены статистические оценки надежности функционирования отдельных программных блоков. С помощью предложенной модели была оценена общая надежность всего программного комплекса.

Предлагается подход к моделированию надежной архитектуры программного комплекса, основанный на идее мультиверсионного программирования. Рассмотрены два различных способа реализации мультиверсионности – NVP и RB.

Задача выбора надежной архитектуры сформулирована в виде задачи многокритериальной смешанной оптимизации с алгоритмически заданными целевыми функциями. Критериями задачи являются общий коэффициент готовности программного комплекса и трудоемкость, которая также зависит от количества и состава программных компонентов комплекса. Задача решается многокритериальным генетическим алгоритмом. Были рассмотрены различные подходы к решению задач многокритериальной оптимизации. Для решения задачи был реализован генетический алгоритм с переменной длиной хромосом, позволяющий кодировать программные архитектуры, различающиеся по количеству и составу компонентов.

В результате применения генетического алгоритма были получены различные варианты программных архитектур разрабатываемого комплекса, отличающиеся от исходной повышенной надежностью. При этом алгоритм предлагал реализовывать множество версий лишь для тех программных компонентов, которые были недостаточно надежны.

Ключевые слова: надежность программного обеспечения, GERT-сети, генетический алгоритм.

Siberian Journal of Science and Technology. 2017, Vol. 18, No. 4, P. 773–778

THE OPERATIONAL MODEL OF A SOFTWARE SYSTEM BASED ON THE GERT-NETWORK

T. A. Panfilova, I. A. Panfilov*, V. V. Zolotarev, I. V. Kovalev, E. A. Sopov

Reshetnev Siberian State University of Science and Technology
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russian Federation
*E-mail: crook_80@mail.ru

The study proposes an original approach to estimate the reliability of software using the GERT-network model. The approach allows simulating the reliability of software complexes consisting of several interacting software components. As the initial data for reliability evaluation, the values of individual blocks' reliability estimates are used. Such estimations can be assigned by the expert, but also can be obtained as the result of the investigation of the software complex. In this paper, experimental studies were carried out for the "Secure Data Exchange Protocol" software system. Statistical estimates of the reliability for the individual program blocks operating were obtained. Using the proposed model, the overall reliability of the entire software package was estimated.

The article also proposes an approach to modeling a reliable software architecture based on the idea of multiversion programming. The article considers two different ways to implement the multiversion of NVP and RB.

The problem of choosing a reliable architecture is formulated in the form of a multi-objective mixed-typed optimization problem with algorithmically defined objective functions. The criteria to the problem are the overall availability of the software complex and its complexity, which also depends on the number and composition of software components. The problem is solved using a multi-objective genetic algorithm. In the study, various approaches to solving multi-objective optimization problems were considered. A genetic algorithm with a variable length of chromosomes was implemented, which allows encoding software architectures that differ in the number and composition of components.

As a result of the genetic algorithm application, various versions of software architectures of the software complex were obtained, which have better reliability. At the same time, the algorithm has proposed to implement multiply versions only for those software components that were not sufficiently reliable.

Keywords: software reliability, GERT, genetic algorithm.

Введение. Конкуренция на рынке производителей программного обеспечения (ПО) сейчас высока, как никогда. Сокращение сроков разработки ПО, необходимость адаптации программ под различные устройства, необходимость учета программной совместимости и повышенные требования к информационной безопасности – вот неполный перечень причин, почему вопрос о надежности программного обеспечения стоит так остро.

Важнейшим фактором, определяющим качество современного программного продукта, является надежность его функционирования. Этой проблемой занимаются исследователи и производители ПО по всему миру. Программные сбои, приводящие к остановке производств или обслуживания клиентов, ошибки проектировщиков, приводящие к утечкам клиентских данных, – эти проблемы несут колоссальные финансовые и репутационные риски как крупным, так и малым компаниям. Одной из самых известных катастроф, вызванных ошибкой программного обеспечения, была гибель ракетоносителя Ariane 5 в 1996 году [1].

В то же время производители программного обеспечения вынуждены сокращать сроки разработки ПО, использовать унифицированные решения для обеспечения совместимости программных систем, привлекать к разработке удаленных специалистов. То есть, несмотря на высокий запрос на надежное ПО, разработчики вынуждены экономить.

Самый эффективный способ обеспечить надежность функционирования программного обеспечения – это проведение работ на этапе проектирования ПО. Выбор надежной архитектуры программ позволит избежать большинства трудностей и сократить затраты на устранение проблем надежности в будущем. Однако у современных исследователей нет единого подхода даже к определению термина «надежность программного обеспечения». Одни вкладывают в это понятие невосприимчивость к внешним воздействиям, другие – простоту и прозрачность программных решений, исключающую возможность возникновения ошибок [2].

Причин возникновения ошибок и сбоев в программных системах также великое множество: проблемы, связанные с проектированием, аппаратным обеспечением, поведением пользователей в системе, внешними воздействиями на систему. Отсюда и разнообразие подходов к обеспечению надежности. Это требует развития модельно-алгоритмического обеспечения, методов и средств выбора надёжного варианта ПО.

Модели оценки надежности программной системы. В данной работе применяется мультиверсионный подход к обеспечению программной избыточности [3]. Простое дублирование компонентов, как при аппаратном резервировании, недопустимо, так как в отличие от аппаратуры программные ошибки имеют внутреннюю природу и при дублировании не исчезнут. При введении программной избыточности возникновение сбоя в функционально эквивалентных модулях (версиях) на одних и тех же входных данных происходит в различных точках его исполнения, что снижает вероятность общей ошибки системы.

В исследовании используются модели надежности программного обеспечения, использующие для получения оценок различные источники данных. Одна из моделей – коэффициент надежности архитектуры ПО [4]:

$$R = \sum_{j=1}^{M} \sum_{i=1}^{N_j} PU_{ij} \cdot R_{ij}, \qquad (1)$$

где M – число уровней архитектуры ПО; N_j – число компонентов на уровне j, j = 1, ..., M; PU_{ij} – вероятность того, что компонент i на уровне j, $i \in \{1, \ldots, N_j\}$, $j \in \{1, ..., M\}$, будет использоваться; R_{ij} – надежность компонента i уровня j.

Значения надежности отдельных компонентов ПО R_{ij} могут быть получены в результате статистической обработки результатов тестирования или могут быть присвоены экспертом. В обоих случаях для расчёта надежности программной системы в целом придется использовать вероятности задействования программных компонентов, оценить которые экспертам непросто.

В рамках данного исследования предлагается метод получения количественных оценок надежности для отдельных программных компонентов и для программных комплексов. В качестве метода моделирования предложены GERT-сети [5].

На рис. 1, где представлен пример фрагмента работы программы для ЭВМ, состояние «1» соответствует работе модуля ввода данных, состояние «2» – это следующий по очередности модуль информационной системы, переход к нему возможен с вероятностью p_1; состояние «3» соответствует системному сбою, который приводит к остановке работы всей программы; состояние «4» описывает работы модуля по корректировке вводимых данных, после которых возможен переход к состоянию «2», «3» или к состоянию «5», соответствующему вызову модуля по ручной корректировке вводимых данных. При этом $p_1 + p_2 + p_3 = 1$ и $p_4 + p_5 + p_6 = 1$. Состояния «1» и «4» носят вероятностный, а состояния «2», «3», «5» – детерминированный характер.

Моделирование процесса функционирования программного обеспечения GERT-сетью позволяют ответить на следующие вопросы:

1) какова вероятность того, что работа программы приведет к системному сбою;

2) какова вероятность того, что программа продолжит штатную работу;

3) чему равны математическое ожидание и дисперсия времени, необходимого для продолжения штатной работы.

Данный подход по расчету надежности оказывается применим для оценки программных систем, состоящих из нескольких связанных исполняемых программных модулей, когда передача данных из модуля в модуль и активация следующего программного модуля происходят автоматически, без участия пользователя. Такая схема предполагает не только последовательное исполнение модулей, но и варианты с ветвлением дерева исполняемых модулей. Однако если речь идет об участии некоторого пользовательского интерфейса, где инициация того или иного программного модуля зависит от решения пользователя или наборов внешних данных, то моделирование процесса функционирования с помощью GERT-сетей в описанном режиме не представляется возможным.

Получение экспериментальных значений надежности. Для апробации рассмотренных моделей были проведены экспериментальные исследования с программным комплексом «Протокол безопасного обмена данными» (ПБОД) [6]. Логически данный комплекс состоит из пяти программных модулей и двух хранилищ данных. Каждый модуль представлен 5–8 функциональными блоками, исполняемыми последовательно. Были проведены испытания ПБОД, составлены протоколы испытаний. В результате экспериментов были получены статистические оценки надежности функционирования 14-ти функциональных блоков, входящих в различные программные модули комплекса. В табл. 1 представлены количественные значения полученных оценок. Значения коэффициентов готовности для функциональных блоков были рассчитаны по методике, предложенной в [7].

Расчет надежности функционирования программного комплекса. Требовалось получить оценки надежности функционирования как отдельных программных модулей, так и всего комплекса ПБОД в целом. Для этого процесс функционирования каждого модуля был представлен GERT-сетью. Для статистически оцененных компонентов программных модулей в алгоритме были использованы полученные функции плотностей распределения и их характеристики (табл. 1), для остальных компонентов были предложены экспертные оценки. Пример модели функционирования для программного модуля № 5 представлен на рис. 2.

Расчетное значение коэффициента надежности архитектуры ПО согласно методике, предложенной в [7], составило $R5 = 0{,}7589$. Вычисление коэффициента надежности архитектуры ПО по формуле (1) для программного модуля № 5 дало результат $R5 = 0{,}75945375$. Расхождение составило $E5 = 0{,}00055375$.

Для оставшихся программных модулей ПБОД с помощью GERT-сетей были рассчитаны значения коэффициентов готовности R_i. Расхождения с экспертным методом также составили незначительные величины. По формуле (1) был рассчитан коэффициент надежности архитектуры R всего ПБОД (табл. 2).

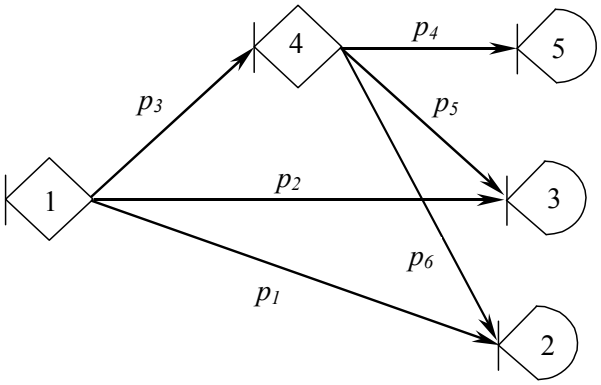

Рис. 1. Пример GERT-сети

Fig. 1. The example of the GERT-network

Таблица 1

Моменты и производящие функции моментов

№ п/п	Тип распределения	$M_E(s)$	Математическое ожидание	Второй момент
1	Экспоненциальное	$\left(1-\frac{s}{a}\right)^{-1}$	3	0,16
2	Нормальное	$e^{sm+(1/2)s^2\sigma^2}$	100	0
3	Константа	n	5	0
4	Экспоненциальное	$\left(1-\frac{s}{a}\right)^{-1}$	95	10
5	Экспоненциальное	$\left(1-\frac{s}{a}\right)^{-1}$	0	0

Окончание табл. 1

№ п/п	Тип распределения	$M_E(s)$	Математическое ожидание	Второй момент
6	Экспоненциальное	$\left(1-\frac{s}{a}\right)^{-1}$	0	0,1
7	Нормальное	$e^{sm+(1/2)s^2\sigma^2}$	5	0,05
8	Нормальное	$e^{sm+(1/2)s^2\sigma^2}$	1000	10
9	Нормальное	$e^{sm+(1/2)s^2\sigma^2}$	100	6
10	Нормальное	$e^{sm+(1/2)s^2\sigma^2}$	1000	20
11	Нормальное	$e^{sm+(1/2)s^2\sigma^2}$	45	0,5
12	Нормальное	$e^{sm+(1/2)s^2\sigma^2}$	20	10
13	Константа	n	20	0
14	Экспоненциальное	$\left(1-\frac{s}{a}\right)^{-1}$	5	0,15

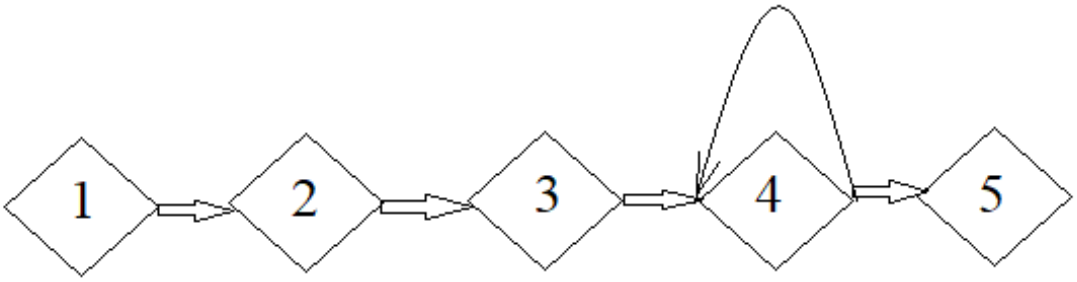

Рис. 2. Модель функционирования программного модуля № 5: 1 – блок доступа к адресной информации; 2 – блок доступа к базе входных данных; 3 – входной интерфейс блока; 4 – блок проверки формата; 5 – блок модификации программного кода

Fig. 2. The model of operation of software module № 5 (1 – Addressing information access block, 2 – Input data access block, 3 – Block input interface, 4 – Format checker block, 5 – Block of modification in the software)

Таблица 2

Результаты расчетов

Номер модуля	Коэффициентов готовности	Ошибка
*R*1	0,87256001	0,00023374
*R*2	0,70564758	0,00065225
*R*3	0,90590065	0,00044870
*R*4	0,65729975	0,00070024
*R*5	0,75945375	0,00055375
ПБОД *R*	0,62007891	

Проектирование надежного варианта программного комплекса. Для обеспечения надежности функционирования программного комплекса был использован мультиверсионный подход. В работе рассмотрено два распространенных подхода в мультиверсионном программном обеспечении: первый подход – мультиверсионное программирование (N-version Programming, NVP), второй подход – блок восстановления (Recovery Block, RB) [8]. Был разработан многокритериальный генетический алгоритм, позволяющий определить эффективные с точки зрения надежности и стоимости разработки программные блоки, подлежащие мультиверсионному дублированию, способ реализаций мультиверсий (NVP или RB), а также количество версий каждого блока.

Задача выбора надежной архитектуры сформулирована в виде задачи многокритериальной смешанной оптимизации с алгоритмически заданными целевыми функциями. Критериями задачи являются общий коэффициент готовности программного комплекса и трудоемкость, которая также зависит от количества и состава программных компонентов комплекса:

$$T_s = \sum_{j=1}^{M} \sum_{i=1}^{N_j} \left(B_{ij} T_{ij} + \left(NVP_{ij} + RB_{ij} \right) \cdot \left(NVX_{ij} + \sum_{k \in Z_{ij}}^{K_{ij}} T_{ij}^{k} \right) \right) \quad (2)$$

при ограничении $Z_{ij} \geq 1$, где M – количество архитектурных уровней в архитектуре ПО; N_j – количество компонентов на уровне j, $j \in \{1, ..., M\}$; K_{ij} – глубина программной избыточности компонента i на уровне j; Z_{ij} – множество версий компонента i на уровне j; T_{ij} – трудоемкость разработки компонента i на уровне j; T^k_{ij} – трудоемкость разработки версии k компонента i на уровне j, $k \in Z_{ij}$, чел. ч; NVX_{ij} – трудоемкость разработки среды исполнения версий (приемочного теста для RB – или алгоритма голосования для NVP); B_{ij} – бинарная переменная, принимающая значение 1 (тогда $NVP_{ij} = 0$, $RB_{ij} = 0$), если в программном компоненте не используется программная избыточность, иначе равна 0; NVP_{ij} – бинарная переменная, принимающая значение 1 (тогда $B_{ij} = 0$, $RB_{ij} = 0$), если в программном компоненте введена программная избыточность методом NVP, иначе равна 0; RB_{ij} – бинарная переменная, принимающая значение 1 (тогда $B_{ij} = 0$, $NVP_{ij} = 0$), если в программном компоненте введена программная избыточность методом RB, иначе равна 0; T_s – общая трудоемкость реализации программной системы.

Задача решается многокритериальным генетическим алгоритмом. В исследовании были рассмотрены различные подходы к решению задач многокритериальной оптимизации [9–12]. Для решения задачи был реализован генетический алгоритм (ГА) с переменной длиной хромосом, позволяющий кодировать программные архитектуры, различающиеся по количеству и составу компонентов [4]. Алгоритм был исследован на множестве тестовых задач [13], результаты исследования позволяют говорить о высокой эффективности предложенной модификации. Длина хромосомы в ГА зависит от количества архитектурных уровней M и количества N_i компонентов на i-м уровне. При этом сами M и N_i выступают как переменные задачи. Z_{ij} в хромосоме задают количество версий компонента i на уровне j. На рис. 3 приведен пример кодирования программной архитектуры в виде бинарной хромосомы.

Для ГА с переменной длиной хромосом использовался специальный оператор процентного скрещивания.

В результате применения генетического алгоритма были получены различные варианты программных архитектур разрабатываемого комплекса, отличающиеся от исходной повышенной надежностью. При этом алгоритм предлагал реализовывать множество версий лишь для тех программных компонентов, которые были недостаточно надежны. Результаты работы алгоритма приведены в табл. 3.

Заключение. В данном исследовании был предложен новый способ оценки надежности программных систем. Способ позволяет получать оценки надежности функционирования программной системы на основе оценок надежности функционирования ее компонентов. Был разработан стохастический алгоритм многокритериальной условной оптимизации, позволяющий находить надежные варианты архитектуры ПО с учетом трудозатрат на реализацию ПО и возможностью использования различных подходов к обеспечению отказоустойчивости. Применение в генетическом алгоритме хромосом переменной длины и оператора процентного скрещивания позволило существенно сократить пространство поиска. Разработанный алгоритм был применен для проектирования программной системы с высокими требованиями к надежности. В ходе проведенных с системой экспериментов были получены статистические оценки надежности отдельных ее компонентов, составлена модель функционирования программной системы. В результате применения разработанного генетического алгоритма были предложены архитектуры программной системы, позволяющие существенно улучшить надежность ее функционирования.

Благодарности. Работа поддержана Министерством образования и науки Российской Федерации, соглашение № RFMEFI57414X0126.

Acknowledgments. This work was supported by the Ministry of Education and Science of the Russian Federation, agreement № RFMEFI57414X0126.

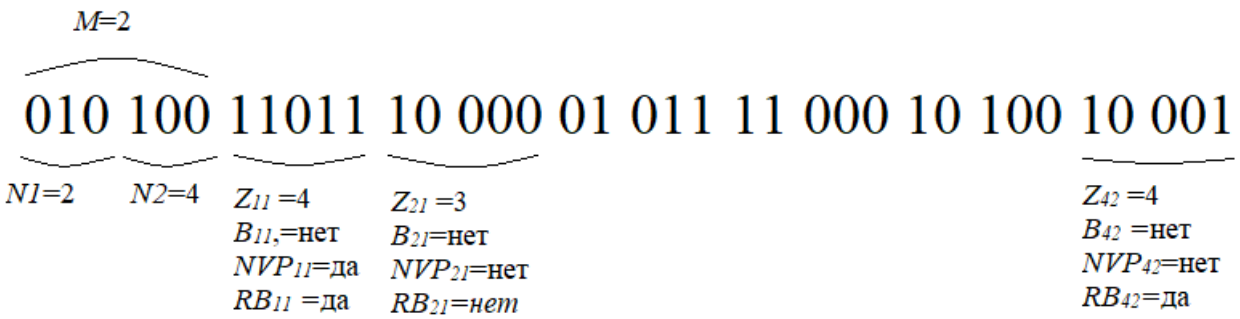

Рис. 3. Пример хромосомы в генетическом алгоритме

Fig. 3. The example of chromosome in the genetic algorithm

Таблица 3

Результаты работы генетического алгоритма

№ решения	Коэффициент готовности системы	Трудозатраты на модернизацию, чел. ч
1	0,92007891	489
2	0,87100579	367
3	0,95411304	523
4	0,96486553	597
5	0,90338740	454
Исходные значения	0,62007891	351

Библиографические ссылки

1. Taverna M. A. Ariane problems force ESA to examine Rosetta options // Aviation Week & Space Technology. 2003. T. 158, № 2. P. 403.

2. Надёжность информационных систем / Ю. Ю. Громов [и др.]. Тамбов : Изд-во ТГТУ, 2010. 160 с. ISBN 978-5-8265-0911-1.

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

4. Панфилова Т. А., Панфилов И. А. Формализация задачи выбора надежного варианта программного обеспечения // Вестник СибГАУ. 2008. № 2 (19). С. 26–28.

5. Ковалев И. В., Волкова Г. В. Автоматизированные системы управления / СибГТУ. Красноярск, 2006. 179 с.

6. Styugin M., Parotkin N. Multilevel decentralized protection scheme based on moving targets // International J. of Security and Its Applications. 2016. Vol. 10, iss. 1. Pp. 45–54.

7. Прямой и обратный алгоритм расчета стохастических сетей / Т. А. Панфилова [и др.] // Вестник СибГАУ. 2013. № 1 (47). С. 91–96.

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

9. Schaffer J. D. Multiple objective optimization with vector evaluated genetic algorithms // Proceedings of an International Conference on Genetic Algorithms and Their Applications / J. J. Grefenstette (Ed.). Pittsburgh, PA, 1985. P. 93–100.

10. Fonseca C. M., Fleming P. J. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. Part I. A unified formulation : Technical report 564 / University of Sheffield. Sheffield, UK, 1995.

11. Horn J., Nafpliotis N., Goldberg D. E. A niched Pareto genetic algorithm for multiobjective optimization // Proceedings of the First IEEE Conference on Evolutionary Computation. 1994. Vol. 1. P. 82–87.

12. Zitzler E., Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach // IEEE Transactions on Evolutionary Computation. 1999. Vol. 3, No. 4. Pp. 257–271.

13. Иванов И. А., Сопов Е. А. Исследование эффективности самоконфигурируемого коэволюционного алгоритма решения сложных задач многокритериальной оптимизации // Системы управления и информационные технологии. 2013. Вып. 51 (1.1). С. 141–145.

References

1. Taverna M. A. Ariane problems force ESA to examine Rosetta options. *Aviation Week & Space Technology*. 2003. Vol. 158, No. 2, P. 403.

2. Gromov Yu. Yu. *Nadezhnost' informatsionnykh sistem* [Reliability of information systems]. Tambov, Tambov State Technical University, 2010, 160 p.

3. Novoy A. V. *Sistema analiza arkhitekturnoy nadezhnosti programmnogo obespecheniya. Dis. kand. dok. tekhn. nauk* [System for analysis of architectural reliability of software. PhD. techn. sci. diss]. Krasnoyarsk, 2011, 131 p.

4. Panfilova T. A., Panfilova T. A. [Formalization Of Reliable Variant Of Software Choice Problem]. *Vestnik SibGAU*, 2008, No. 2 (19), P. 26–28 (In Russ.).

5. Kovalev I. V., Volkova G. V. *Avtomatizirovannye sistemy upravleniya* [Automated control systems]. Krasnoyarsk, SibSTU Publ., 2006, 179 p.

6. Styugin M., Parotkin N. Multilevel decentralized protection scheme based on moving targets. *International J. of Security and Its Applications*. 2016, Vol. 10, Iss. 1, P. 45–54.

7. Panfilova T. A., R. Yu. Tsarev, A. V. Shtarik, E. N. Shtarik, E. R. Khasanov, Panfilova T. A. [Direct and inverse algorithms of stochastic network calculation]. *Vestnik SibGAU*. 2013, No. 1 (47), P. 91–96 (In Russ.).

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

9. Schaffer J. D. Multiple objective optimization with vector evaluated genetic algorithms. In J. J. Grefenstette (Ed.), Proceedings of an International Conference on Genetic Algorithms and Their Applications, Pittsburgh, PA, 1985, P. 93–100.

10. Fonseca C. M., Fleming P. J. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. Part I: A unified formulation. Technical report 564, University of Sheffield, Sheffield, UK, January 1995.

11. Horn J., Nafpliotis N., Goldberg D. E. A niched Pareto genetic algorithm for multiobjective optimization. *In Proceedings of the First IEEE Conference on Evolutionary Computation*. 1994, Vol. 1, Piscataway, P. 82–87.

12. Zitzler E., Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*. 1999, Vol. 3, No. 4, P. 257–271.

13. Ivanov I. A., Sopov E. A. Investigation of the self-configured coevolutionary algorithm for complex multi-objective optimization problem solving. *Control Systems and Information Technology*. 2013, No. 51 (1.1), P. 141–145.