ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

COMPUTER SCIENCE, COMPUTER ENGINEERING AND CONTROL

УДК 004.4
doi:10.21685/2072-3059-2021-1-1

Разработка комплексного подхода к программной самоадаптации и универсальный метод синтеза адаптивных программных систем

А. М. Бершадский[1], А. С. Бождай[2], Ю. И. Евсеева[3], А. А. Гудков[4]

[1,2,3,4]Пензенский государственный университет, Пенза, Россия

[1,2]bam@pnzgu.ru, [3]shymoda@mail.ru, [4]alexei.gudkov@gmail.com

Аннотация. *Актуальность и цели.* Широта применения современных прикладных программных систем, высокие требования к их производительности и надежности делают актуальной проблему создания универсального теоретического аппарата самоадаптации программного обеспечения прикладного назначения. Предложен комплексный подход к программной самоадаптации, интегрирующий методы самоадаптации на основе трассировки вычислительного процесса и наблюдения за информационной средой. Применение такого подхода лежит в основе универсального метода синтеза адаптивных программных систем, применимого как для создания простейших утилит, так и сложных программно-технических комплексов. Созданное таким образом программное обеспечение (ПО), способно менять собственную структуру и поведение как на основе анализа собственной поведенческой информации, так и путем обработки отзывов, поступающих от конечных пользователей. *Материалы и методы.* Для разработки универсального метода синтеза адаптивных программных систем использовались технологии трассировки программного кода, моделирования изменчивости, интеллектуального анализа данных, латентно-семантического и дистрибутивно-статистического анализа, а также математический аппарат теории графов и семантических сетей. *Результаты.* Основным результатом работы является универсальный метод синтеза адаптивных программных систем, реализующий предложенный комплексный подход к программной самоадаптации. Требуемая комплексность достигнута благодаря программно-технической интеграции концепций трассировки вычислительного процесса и наблюдения за информационной средой. Универсальность метода обеспечивается его инвариантностью к предметным областям, а также возможностью масштабирования с точки зрения структурной и поведенческой сложности синтезируемого адаптивного ПО. *Выводы.* Разработанный подход к программной самоадаптации и универсальный метод синтеза адаптивного ПО при-

менимы к широкому кругу прикладных областей и задач. В итоге получаемые адаптивные системы имеют расширенный жизненный цикл, обусловленный способностью самостоятельно реорганизовываться (без перекомпиляции исходного кода) под изменяющиеся требования предметной области. Это позволяет добиться существенного экономического эффекта за счет минимизации временных и материальных затрат на сопровождение, необходимые системные остановки для внесения изменений и вторичное тестирование.

Ключевые слова: самоадаптация, адаптивные программные системы, программная инженерия, моделирование изменчивости, интеллектуальный анализ данных, трассировка программного кода

Финансирование: исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-07-00408.

Для цитирования: Бершадский А. М., Бождай А. С., Евсеева Ю. И., Гудков А. А. Разработка комплексного подхода к программной самоадаптации и универсальный метод синтеза адаптивных программных систем // Известия высших учебных заведений. Поволжский регион. Технические науки. 2021. № 1. С. 3–12. doi:10.21685/2072-3059-2021-1-1

Development of a comprehensive approach to software self-adaptation and a universal method for synthesis of adaptive software systems

A.M. Bershadskiy[1], A.S. Bozhday[2], Yu.I. Evseeva[3], A.A. Gudkov[4]

[1,2,3,4]Penza State University, Penza, Russia

[1,2]bam@pnzgu.ru, [3]shymoda@mail.ru, [4]alexei.gudkov@gmail.com

Abstract. *Background*. Actual problem of creating a universal theoretical apparatus for software self-adaptation exists due to the widespread use of modern applied programs and high requirements for their performance and reliability. The article proposes an integrated approach to software self-adaptation that integrates adaptation methods based on tracing the computational process and information environment monitoring. The application of this approach underlies the universal method for the synthesis of adaptive programs, applicable both for creating the simplest utilities and complex software systems. The programs created in this way is capable of changing its own structure and behavior, both based on the analysis of its own behavioral information, and by processing feedback from end users. *Materials and methods*. To develop a universal method for the synthesis of adaptive software systems, we used the technologies of tracing program code, modeling variability, data mining, latent semantic and distributive statistical analysis, as well as the mathematical apparatus of graph theory and semantic networks. *Results*. The main result of the work is a universal method for the synthesis of adaptive systems that implements the proposed comprehensive approach to software self-adaptation. The required complexity was achieved due to the integration of a computational process trace and information environment monitoring concepts. The method universality is ensured by its invariance to subject areas, as well as by the ability to scale in terms of the structural and behavioral complexity of synthesized adaptive software.

Keywords: self-adaptation, adaptive software systems, software engineering, variability modeling, data mining, code tracing

Acknowledgments: the research was financed by the Russian Foundation for Basic Research within the scientific project No. 18-07-00408.

For citation: Bershadskiy A.M., Bozhday A.S., Evseeva Yu.I., Gudkov A.A. Development of a comprehensive approach to software self-adaptation and a universal method for synthesis of adaptive software systems. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences.* 2021;1:3–12. (In Russ.). doi:10.21685/2072-3059-2021-1-1

Введение

Современное общество сложно представить в отрыве от программных систем, поддерживающих его информационные потребности. Прикладное программное обеспечение (ПО) различных типов используется на предприятиях, в банках, коммерческих и государственных структурах. Степень интеграции программно-аппаратных комплексов с протекающими в социально-экономических системах процессами столь велика, что сбои в их работе приводят к тяжелым последствиям – вплоть до полной остановки функционирования таких систем.

Высокая сложность современных прикладных программных систем способствует тому, что ресурсозатраты на их разработку и сопровождение оказываются достаточно велики, а последствия возможных сбоев в работе могут быть критичны. Решением проблемы может стать наделение прикладного ПО способностью к самоадаптации.

Предлагаемый авторами комплексный подход к созданию самоадаптивных программных систем позволяет учитывать весь спектр информации, имеющей отношение к программной системе, для ее последующей самоадаптации: информацию о внутреннем строении системы, информацию о предметной области и динамике ее изменения, мнения конечных пользователей. Особенностью подхода также является то, что данная информация обрабатывается системой автоматически, без участия программиста или пользователя, и последующая самоадаптация также реализуется без человеческого вмешательства.

1. Синтез адаптивных программных систем на основе концепции трассировки вычислительного процесса

Идея применения трассировки вычислительного процесса для построения самоадаптивного ПО была предложена авторами в работах [1, 2]. Основной целью такой трассировки является автоматическое отслеживание потока управления программы с последующим созданием паттернов поведения системы. Анализ и последующее преобразование обнаруженных паттернов позволят оптимизировать поведение системы при различных адаптивных сценариях.

Основные положения концепции самоадаптации на основе трассировки вычислительного процесса формулируются следующим образом:

1. Использование программных трасс для моделирования поведения системы на определенном временном промежутке. Под трассами понимаются протоколы выполнения программного кода.

2. Выбор минимального размер используемых трасс путем фиксации в них только информации об условных переходах.

3. Возможность оптимизации исходного кода системы путем анализа записанных трасс.

Второй пункт в данном списке успешно реализуется путем применения технологии Intel Processor Trace [3]. Получаемые с ее помощью трассы содержат только информацию о том, как выполнялись условные переходы.

Для создания универсального метода синтеза самоадаптивного ПО необходима строгая математическая формализация процесса трассировки и последующего анализа записанных трасс. Отслеживая, какие последовательности программных инструкций выполняются наиболее часто, можно выявить устойчивые тенденции в поведении системы. В качестве средства такой математической формализации предлагается использовать теорию графов. Задача поиска наиболее часто встречающихся трасс в таком случае сводится к задаче поиска часто встречающихся подграфов. Каждый подграф в таком наборе будет представлять собой формальное представление программной трассы.

Граф представляет собой пару множеств $G=(V,E)$, где V – множество вершин, $E\subseteq V\times V$ – множество ребер. Назовем графом выполнения такой ориентированный граф G с множеством вершин $V=V(G)$, где каждая вершина является абстракцией линейного блока инструкций, а каждая дуга (из множества дуг $E(G)\subseteq V(G)\times V(G)$) описывает переход между блоками инструкций.

Обобщенно математическая модель поведения адаптивной системы будет описываться кортежем

$$M=(F,D), \tag{1}$$

где $F=(X,A)$ – базовый (статический) граф выполнения, абстрагирующий все возможные блоки инструкций программы и все возможные переходы между ними, при этом каждой вершине из множества $X_i=\{x_1,x_2,...,x_m\}$ присвоена метка, определяемая последовательностью инструкций, связанных с данной вершиной: $x_i=(c,b)$ (здесь c – номер вершины, b – связанный с ней блок последовательности линейных инструкций); $D=\{G_1,G_2,...,G_i,...,G_n\}$, (здесь $G_i=(V_i,E_i)$ – граф выполнения программы на i-м запуске, при этом $G_i\subseteq F$, $V_i\subseteq X$, $E_i\subseteq A$).

Существующие методы решения задачи о поиске частых подграфов обычно построены на основе общего подхода, включающего три этапа:

1. Формирование кандидатов – составление списка подграфов, которые будут рассматриваться на последующих этапах.
2. Отсечение неподходящих кандидатов.
3. Подсчет, сколько раз каждый из конечных кандидатов встречается среди множества графов, поступивших на вход алгоритма.

В качестве примера можно привести решения на базе Apriori-алгоритма, подробное описание которых можно найти, например, в работе [4].

Процесс самоадаптации программной системы на основе технологии трассировки вычислительного процесса будет включать в себя следующие этапы:

1. Трассировка выполнения программы на множестве запусков.
2. Сбор полученных трасс и их графовая формализация, построение математической модели поведения системы в соответствии с формулой (1).

3. Обработка множества формализованных трасс *D* таким образом, чтобы номера вершин в них соответствовали номерам вершин из графа *F*.

4. Поиск часто встречающихся подграфов в множестве графов, построенных на основе трасс и являющихся частью математической модели.

5. Нахождение в частых подграфах участков, соответствующих операторам ветвления, которые при всех запусках программы сработали одинаковым образом, т.е. всегда были истинны или всегда были ложны.

6. Оптимизация исходного кода программы: удаление инструкций, связанных с вычислением найденных на предыдущем этапе условий, и замена условных переходов на безусловные. Пример оптимизации иллюстрирует псевдокод, представленный в табл. 1.

Таблица 1

Оптимизация кода на основе анализа трасс

Код до модификации	**Код после модификации**

Вычисление 1
Вычисление условия
if (условие истинно)
{
Вычисление 2
}
else
{
Вычисление 3
}
	Вычисление 1
	if (1*)
	{
	Вычисление 2
	}
	else
	{
	Вычисление 3
	}
	* или 0, в зависимости от того, было условие истинным или ложным при запусках программы

Следует учитывать, что после внесения изменений работоспособность программы может быть нарушена. Поэтому должна быть предусмотрена возможность возврата программы к исходному состоянию и возможность предварительного запроса пользователю о необходимости модификации.

Применение на практике указанного метода самоадаптации позволит повысить производительность программной системы за счет сокращения количества выполняемых процессором инструкций, связанных с вычислением условий.

2. Синтез адаптивных программных систем на основе наблюдения за информационной средой

Метод самоадаптации на основе трассировки вычислительного процесса не позволяет автоматически реорганизовать систему в ответ на изменения в предпочтениях ее пользователей. Проблему можно решить, разработав метод синтеза, центральной идеей которого будет наблюдение за окружающей информационной средой.

Перечислим основные принципы такого наблюдения:

1. Для представления структуры системы используются специализированные сущности, абстрагирующие отдельные ее компоненты (модели, программные функции, параметры и т.д.), – характеристики.

2. Каждое состояние системы можно представить отдельным набором характеристик.

3. Формирование новых системных состояний путем использования информации, поступающей из внешней информационной среды.

Полное множество всех возможных состояний системы назовем поведенческой структурой и будем описывать ее с помощью модели характеристик. Модель характеристик – это разновидность моделей морфологического множества, визуально изображаемая как дополненное некоторыми новыми типами отношений И/ИЛИ-дерево [5]. На рис. 1 представлен пример модели. Характеристики модели (вершины дерева) представляют собой абстракции определенных программных компонентов (фрагментов программного кода, 3D-моделей, некоторых параметров и т.д.). С помощью отношений модели формируются потенциально допустимые состояния системы [6].

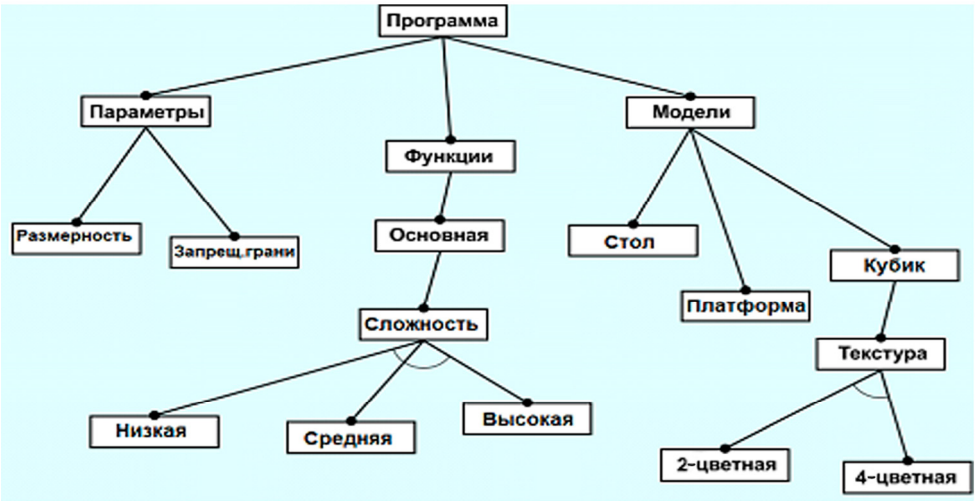

Рис. 1. Пример модели характеристик

Представленный на рис. 1 практический пример модели характеристик описывает адаптивное поведение приложения, реализующего тестирование пространственного интеллекта по методу «Кубики Коса» [7]. Приведенная модель описывает изменчивость двух параметров – функции генерации матрицы, по которой генерируется тестируемый кубиками узор, и количество используемых в узоре цветов.

Метод самоадаптации на основе наблюдения за информационной средой включает в себя следующие основные этапы:

1. Формирование исходного множества естественноязыковых документов, в которых содержатся описания пользовательских предпочтений в отношении желаемой функциональности системы.

2. Формирование множества основных терминов документов. Реализуется с целью сокращения размерности исходного пространства документов. Этап включает в себя 5 базовых шагов:

1) преобразование каждого документа в форму, не содержащую стоп-слов (знаков препинания, союзов, предлогов и т.д.;

2) преобразование всех слов в документах к исходным формам – леммам;

3) определение части речи для каждого слова;

4) все слова, кроме существительных, глаголов и прилагательных, исключаются из документов;

5) на основе преобразованных документов формируется матрица X, в которой число строк m – это общее количество документов, а число столбцов n – количество терминов.

На основе частоты встречаемости слов формируются весовые коэффициенты матрицы. Последний этап работы алгоритма включает в себя два подэтапа:

2.1. Определение похожих документов. Необходимо понизить размерность исходной матрицы документов, для чего целесообразно воспользоваться ее сингулярным разложением:

$$X = YCW^T, \tag{2}$$

где матрицы Y и W являются ортогональными, а C – диагональной. Главная диагональ матрицы C полностью сформирована из сингулярных чисел матрицы X. Из матрицы Y возьмем k первых столбцов, а из матрицы W – k первых строк, в результате чего получим матрицу $X' = Y' C' W'$, представляющую собой приближение исходной матрицы документов X с рангом k. Матрица $J = C' W'$ размерностью $k \times n$ будет применяться для выявления схожих документов.

2.2. Далее на основе значений матрицы J реализуется кластеризация множества документов. Для этого можно использовать, например, метод k-средних или метод нечеткой кластеризации C-средних [8].

3. Выделение характеристик в кластерах, полученных на предыдущем этапе метода. В контексте поставленной задачи каждую характеристику можно понимать как устойчивое словосочетание в тексте документа. Для определения устойчивых словосочетаний можно воспользоваться следующей формулой:

$$C = \frac{t_1 + t_2}{2f\left(t^1 t^2\right)}, \tag{3}$$

где t_1 и t_2 – количество отличных друг от друга пар со словами, составляющими словосочетание; $f\left(t^1 t^2\right)$ – число появлений словосочетания, в котором оба слова встречаются совместно. Качество пары $\left(t^1 t^2\right)$ обратно пропорционально значению C.

4. В каждом из выделенных кластеров определяются отношения между составляющими его характеристиками. На основе найденных соотношений строится семантическая сеть. Эффективным способом решения данной задачи будет дистрибутивно-статистический метод, подробно описанный в работах [9, 10]. Результатом работы метода будет матрица семантической связности устойчивых словосочетаний (в контексте нашей работы – характеристик). На основе полученной матрицы будет построена семантическая сеть, узлами

которой являются сами характеристики, а дуги показывают наличие семантической связи между ними.

5. Путем сопоставления полученных на предыдущем этапе семантических сетей с исходной моделью характеристик получаем множество конфигураций модели.

6. В процессе функционирования программной системы каждому выделенному в ней типу пользователей будет поставлена в соответствие некоторая конфигурация модели характеристик, что позволит сделать поведение программы более гибким по отношению к пользователю.

Предложенный метод позволит системе самостоятельно выявлять и исправлять собственные объективные недостатки в процессе функционирования, а также подстраиваться под отдельные категории пользователей.

Заключение

Существует немало подходов к разработке адаптивного ПО, однако ни один из них не решает важной задачи – не учитывает в процессе самоадаптации весь спектр информации, с которой работает система (как внутренней, так и внешней). Для ее решения авторами были предложены две концепции самоадаптации программных систем – концепция самоадаптации на основе трассировки вычислительного процесса и концепция самоадаптации на основе наблюдения за информационной средой. На основе введенных концепций разработан универсальный метод синтеза адаптивного ПО. Концепция трассировки вычислительного процесса сосредоточена на поиске участков программного кода, которые функционируют неоптимально, и последующей автоматической оптимизации данных участков. Концепция самоадаптации на основе наблюдения за информационной средой позволяет программной системе самостоятельно перестраивать собственное поведение, согласуя его с нуждами пользователей. Комплексное применение методов, разработанных на основе предложенных концепций, позволит сделать поведение программной системы максимально гибким и оптимизировать ее производительность без вмешательства разработчиков.

Список литературы

1. Бершадский А. М., Бождай А. С., Евсеева Ю. И., Гудков А. А. Исследование и разработка методов динамического анализа кода для создания самоадаптивного программного обеспечения // Моделирование, оптимизация и информационные технологии. 2018. № 6. С. 108–120.
2. Бождай А. С., Евсеева Ю. И., Гудков А. А. Разработка самоадаптивного программного обеспечения на основе технологии трассировки вычислительного процесса // Известия высших учебных заведений. Поволжский регион. Технические науки. 2020. № 3. С. 26–35.
3. Enhance performance analysis with Intel Processor Trace. Performance explained easy. URL: https://easyperf.net/blog/2019/08/23/Intel-Processor-Trace (дата обращения: 20.10.2020).
4. Пыжов В. О., Куликов Г. С., Панов А. В. Задача поиска частых подграфов и алгоритмы ее решения // Актуальные вопросы современной науки. 2016. № 1. С. 74–83.
5. Schobbens P. E., Heymans P., Trigaux J. C. Feature diagrams: a survey and formal semantics // 14th IEEE International Requirements Engineering Conference (RE'06). Washington : IEEE Computer Society. 2011. P. 139–148.

6. Kang K. C. [et al.]. Feature-oriented domain analysis (FODA) : feasibility study. Pittsburgh : Software Engineering Institute, 1990. 161 p.
7. Stone M. Kohs Block Design Test // Test Critiques / ed. by D. J. Keyser, R. C. Sweetland. Kansas City : Test Corporation of America, 1985. P. 102–114.
8. Eibe F., Witter I. Data mining. Practical machine learning tools and techniques. The Morgan Kaufmann series in data management systems, 2005. 525 с.
9. Филиппович Ю. Н., Прохоров В. А. Семантика информационных технологий: опыты словарно-тезаурусного описания / предисл. А. И. Новикова. М. : Изд-во МГУП, 2002. 368 с.
10. Кобрин Р. Ю. Определение дифференциальных семантико-грамматических признаков терминов // Термин и слово : межвуз. сб. Н. Новгород : Нижегород. ун-т, 1997. С. 34–41.

References

1. Bershadskiy A.M., Bozhday A.S., Evseeva Yu.I., Gudkov A.A. Research and development of methods for dynamic code analysis to create self-adaptive software. *Modelirovanie, optimizatsiya i informatsionnye tekhnologii = Modeling, optimization and information technology*. 2018;6:108–120. (In Russ.)
2. Bozhday A.S., Evseeva Yu.I., Gudkov A.A. Development of self-adaptive software based on the technology of tracing the computational process. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2020;3:26–35. (In Russ.)
3. *Enhance performance analysis with Intel Processor Trace. Performance explained easy*. Available at: https://easyperf.net/blog/2019/08/23/Intel-Processor-Trace (accessed 20.10.2020).
4. Pyzhov V.O., Kulikov G.S., Panov A.VThe problem of finding frequent subgraphs and the algorithm for solving it. *Aktual'nye voprosy sovremennoy nauki = Actual issues of contemporary science*. 2016;1:74–83. (In Russ.)
5. Schobbens P.E., Heymans P., Trigaux J.C. Feature diagrams: a survey and formal semantics. *14th IEEE International Requirements Engineering Conference (RE'06)*. Washington: IEEE Computer Society. 2011:139–148.
6. Kang K. C. [et al.]. *Feature-oriented domain analysis (FODA): feasibility study*. Pittsburgh: Software Engineering Institute, 1990:161.
7. Stone M. Kohs Block Design Test. *Test Critiques*. Kansas City: Test Corporation of America,1985:102–114.
8. Eibe F., Witter I. *Data mining. Practical machine learning tools and techniques. The Morgan Kaufmann series in data management systems*, 2005:525.
9. Filippovich Yu.N., Prokhorov V.A. *Semantika informatsionnykh tekhnologiy: opyty slovarno-tezaurusnogo opisaniya* = S*emantics of information technologies: experiments of dictionary-thesaurus description*. Moscow: Izd-vo MGUP, 2002:368. (In Russ.)
10. Kobrin R.Yu. Definition of differential semantic and grammatical features of terms. *Termin i slovo: mezhvuz. sb. = Term and word: intercollegiate collection*. Nizhny Novgorod: Nizhegorod. un-t, 1997:34–41. (In Russ.)

Информация об авторах / Information about the authors

Александр Моисеевич Бершадский
доктор технических наук, профессор, заведующий кафедрой систем автоматизированного проектирования, Пензенский государственный университет (Россия, г. Пенза, Красная, 40)

Aleksandr M. Bershadskiy
Doctor of engineering sciences, professor, head of the sub-department of computer aided design systems, Penza State University (40 Krasnaya street, Penza, Russia)

E-mail: bam@pnzgu.ru

Александр Сергеевич Бождай
доктор технических наук, доцент, профессор кафедры систем автоматизированного проектирования, Пензенский государственный университет (Россия, г. Пенза, Красная, 40)

E-mail: bam@pnzgu.ru

Aleksandr S. Bozhday
Doctor of engineering sciences, associate professor, professor of the sub-department of computer aided design systems, Penza State University (40 Krasnaya street, Penza, Russia)

Юлия Игоревна Евсеева
кандидат технических наук, доцент кафедры систем автоматизированного проектирования, Пензенский государственный университет (Россия, г. Пенза, Красная, 40)

E-mail: shymoda@mail.ru

Yuliya I. Evseeva
Candidate of engineering sciences, associate professor of the sub-department of computer aided design systems, Penza State University (40 Krasnaya street, Penza, Russia)

Алексей Анатольевич Гудков
кандидат технических наук, доцент кафедры систем автоматизированного проектирования, Пензенский государственный университет (Россия, г. Пенза, Красная, 40)

E-mail: alexei.gudkov@gmail.com

Aleksey A. Gudkov
Candidate of engineering sciences, associate professor of the sub-department of computer aided design systems, Penza State University (40 Krasnaya street, Penza, Russia)

Поступила в редакцию / Received 15.12.2020
Поступила после рецензирования и доработки / Revised 20.12.2020
Принята к публикации / Accepted 29.12.2020