ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

ИНФОРМАТИКА

INFORMATICS

ТОМ 20
VOL. 20

4 | 2023

ОТ РЕДАКЦИИ

В журнале «Информатика» публикуются оригинальные и обзорные статьи, описывающие результаты фундаментальных и прикладных исследований специалистов академического и вузовского профиля в области информатики и информационных технологий.

Основной целью журнала является публикация наиболее значимых новых результатов в указанной области. Приветствуются статьи, описывающие заключительные результаты научных проектов и диссертационных исследований, открывающие новые направления исследований, которые находятся на стыке информатики и других наук.

Журнал рассчитан на широкий круг специалистов в области информатики и информационных технологий.

Основные разделы журнала:

- биоинформатика;
- математическое моделирование;
- защита информации и надежность систем;
- информационные технологии;
- логическое проектирование;
- обработка сигналов, изображений, речи, текста и распознавание образов;
- автоматизация проектирования;
- интеллектуальные системы.

Префикс DOI: 10.37661

Индексирование:

Высшей аттестационной комиссией Республики Беларусь журнал «Информатика» был включен в список научных изданий для опубликования результатов диссертационных исследований.

В декабре 2017 г. включен в базу данных Российского индекса научного цитирования (РИНЦ). С помощью инструментов и сервисов, доступных на платформе eLIBRARY (раздел «Личный кабинет»), можно самостоятельно корректировать список своих публикаций и цитирований в РИНЦ.

В июле 2017 г. включен в базу журналов открытого доступа Directory of Open Access Journals (DOAJ).

С помощью поисковых систем Google Scholar, WorldCat, Соционет можно получить свободный доступ к полному тексту научных публикаций журнала.

Адрес редакции:
ул. Сурганова, 6, к. 305, г. Минск, 220012, Беларусь
Тел. +375 (017) 351 26 22

Editorial address:
Surganova str., 6, of. 305, Minsk, 220012, Belarus
Phone +375 (017) 351 26 22

E-mail: rio@newman.bas-net.by
https://inf.grid.by/jour

THE EDITOR'S NOTE

The journal "Informatics" is a scientific publication in computer sciences and information technologies which reviews the results in basic and applied research of scientists from the universities and scientific centers.

The journal focuses on the most significant and modern papers of research projects results and PhD/DSc thesis in computer sciences.

The journal is edited for the specialists in IT and computer sciences research and application.

The main sections of the journal:

- bioinformatics;
- mathematical modeling;
- information protection and system reliability;
- information technology;
- logical design;
- signal, image, speech, text processing and pattern recognition;
- computer-aided design;
- artificial intelligence methods.

DOI Prefix: 10.37661

Indexation:

the journal "Informatics" is in the list of scientific publications recommended by the Higher Attestation Commission of the Republic of Belarus for scientists to publish the results of PhD/DSc research.

In December 2017 the journal was included in the database of the Russian Science Citation Index (RISC) and provides free access to reviewed electronic scientific paper, improving scientific information traffic and also raising quotation of works of the authors (please use https://elibrary.ru or section for authors https://elibrary.ru_author_tools).

In July 2017 included in the database of open access journals Directory of Open Access Journals (DOAJ).

Using the Google Scholar, WorldCat, Соционет search engine, you can get free access to full text of scientific publications of magazine.

ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

ОБЪЕДИНЕННЫЙ ИНСТИТУТ ПРОБЛЕМ ИНФОРМАТИКИ
НАЦИОНАЛЬНОЙ АКАДЕМИИ НАУК БЕЛАРУСИ

ИНФОРМАТИКА

Informatika

Том 20, № 4, октябрь-декабрь 2023

Ежеквартальный научный журнал

Издается с января 2004 г.

Учредитель и издатель – государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси» (ОИПИ НАН Беларуси)

ИНФОРМАТИКА

Том 20, № 4, октябрь-декабрь 2023

Ответственный за выпуск *Мойсейчик Светлана Сергеевна*
Редактор *Гончаренко Галина Борисовна*
Компьютерная верстка *Бутевич Ольга Борисовна*

Сдано в набор 20.11.2023. Подписано в печать 18.12.2023. Формат 60×84 1/8. Бумага офсетная. Гарнитура Таймс. Ризография. Усл. печ. л. 11,8. Уч.-изд. л. 11,6. Тираж 40 экз. Заказ 11.

Государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси».
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/274 от 04.04.2014. ЛП № 02330/444 от 18.12.13. Ул. Сурганова, 6, 220012, Минск, Беларусь.

ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

THE UNITED INSTITUTE OF INFORMATICS PROBLEMS
OF THE NATIONAL ACADEMY OF SCIENCES OF BELARUS

INFORMATICS

Vol. 20, no. 4, October-December 2023

Published quarterly

Issued since January 2004

Founder and publisher – State Scientific Institution "The United Institute of Informatics Problems of the National Academy of Sciences of Belarus" (UIIP NASB)

INFORMATICS

Vol. 20, no. 4, October-December 2023

Issue Head *Sviatlana S. Maiseichyk*
Editor *Halina B. Hancharenka*
Computer Imposition *Volha B. Butsevich*

Sent for press 20.11.2023. Output 18.12.2023. Format 60×84 1/8. Offset paper. Headset Times. Riesography. Printed sheets 11,8. Publisher's signatures 11,6. Circulation 40 copies. Order 11.

State Scientific Institution "The United Institute of Informatics Problems of the National Academy of Sciences of Belarus".
Certificate on the state registration of the publisher, manufacturer, distributor of printing editions no. 1/274 dated 04.04.2014. License for the press no. 02330/444 dated 18.12.13.
6, Surganov Str., 220012, Minsk, Belarus.

ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

СОДЕРЖАНИЕ

ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

CONTENTS

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ
LOGICAL DESIGN

УДК 004.33.054
https://doi.org/10.37661/1816-0301-2023-20-4-7-23

Оригинальная статья
Original Paper

Формальная модель описания и условия обнаружения связных неисправностей взаимного влияния запоминающих устройств

В. Н. Ярмолик✉, Д. В. Деменковец, В. В. Петровская, А. А. Иванюк

Белорусский государственный университет информатики и радиоэлектроники,
ул. П. Бровки, 6, Минск, 220013, Беларусь
✉*E-mail: yarmolik10ru@yahoo.com*

Аннотация

Цели. Целью работы являются разработка и анализ формальной модели описания сложных связных неисправностей взаимного влияния запоминающих устройств и формулировка необходимых и достаточных условий их обнаружения. Актуальность данных исследований заключается в том, что современные запоминающие устройства, характеризующиеся большим объемом хранимых данных и изготовленные по новейшим технологическим нормам, отличаются проявлением в них сложных разновидностей неисправностей.

Методы. Результаты исследования основаны на классической теории и практике однократных маршевых тестов (*March tests*) запоминающих устройств. В частности, в работе используются формальные математические модели описания неисправностей памяти и показывается их ограниченность для представления связных неисправностей взаимного влияния. Главная идея предлагаемого авторами подхода заключается в применении нового формального описания подобных неисправностей, ключевым элементом которого является использование ролей, выполняемых ячейками запоминающего устройства, участвующими в неисправности.

Результаты. Определены три основные роли, которые выполняют ячейки связной неисправности взаимного влияния, а именно: роль агрессора (*A*), роль жертвы (*V*), а также роль, включающая роли жертвы и агрессора (*B*), выполняемые двумя ячейками одновременно по отношению друг к другу. Показано, что сценарий реализации ролей ячеек неисправности памяти определяется применяемым маршевым тестом и в первую очередь используемой им адресной последовательностью обращения к ячейкам. Приведена процедура построения формальной модели связной неисправности, основу которой составляют роли, выполняемые ячейками, входящими в неисправность, и сценарий, задаваемый тестом. На базе нового формального описания связных неисправностей взаимного влияния сформулировано утверждение, определяющее необходимые и достаточные условия обнаружения подобных неисправностей. Показывается наличие необнаруживаемых неисправностей взаимного влияния и определяется возможность их обнаружения в рамках многократных маршевых тестов. Проведенные экспериментальные исследования подтвердили справедливость сформулированных положений статьи. На базе классического примера связной неисправности взаимного влияния показано выполнение необходимых и достаточных условий ее обнаружения однократным маршевым тестом.

Заключение. Результаты исследований подтверждают, что предложенная формальная математическая модель описания связных неисправностей взаимного влияния позволяет идентифицировать их покрытие маршевыми тестами. В рамках предложенной модели определяются необходимые и достаточные условия обнаружения связных неисправностей взаимного влияния маршевыми тестами, покрывающими одиночные связные неисправности.

Ключевые слова: тестирование вычислительных систем, неисправности взаимного влияния, связные неисправности, маршевые тесты запоминающих устройств, адресная последовательность

Для цитирования. Формальная модель описания и условия обнаружения связных неисправностей взаимного влияния запоминающих устройств / В. Н. Ярмолик [и др.] // Информатика. – 2023. − Т. 20, № 4. – С. 7–23. https://doi.org/10.37661/1816-0301-2023-20-4-7-23

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 05.08.2023
Подписана в печать | Accepted 27.09.2023
Опубликована | Published 29.12.2023

Formal description model and conditions for detecting linked coupling faults of the memory devices

Vyacheslav N. Yarmolik✉, Denis V. Demenkovets, Vita V. Petrovskaya, Alexander A. Ivaniuk

Belarusian State University of Informatics and Radioelectronics,
st. P. Brovki, 6, Minsk, 220013, Belarus
✉*E-mail: yarmolik10ru@yahoo.com*

Abstract

Objectives. The aim of the work is to develop and analyze a formal model for describing complex linked coupling faults of memory devices and to formulate the necessary and sufficient conditions for their detection. The relevance of these studies lies in the fact that modern memory devices, characterized by a large amount of stored data and manufactured according to the latest technological standards, are distinguished by the manifestation of complex types of faults in them.

Methods. The presented results are based on the classical theory and practice of march tests (*March tests*) of memory devices. In particular, the paper uses formal mathematical models for describing memory faults and shows their limitations for representing complex linked coupling faults. The main idea of the approach proposed by the authors is based on the use of a new formal description of such faults, the key element of which is the introduction of roles performed by the cells involved in the fault.

Results. Three main roles are defined that cells of the complex linked coupling faults perform, namely the role of the aggressor (*A*), the role of the victim (*V*), as well as the role of both the victim and the aggressor (*B*), performed by two cells simultaneously in relation to each other. It is shown that the scenario for the implementation of the roles of memory failure cells is determined by the marching test used, and, first of all, by the address sequence used to access the cells. The procedure for setting a formal model of a linked fault is given, the basis of which is the roles performed by the cells included in the fault and the scenario specified by the test. A statement is given that determines, on the basis of a new formal description of linked coupling faults, the necessary and sufficient conditions for the detection of such faults. The presence of undetectable linked coupling faults is shown, and the conditions for their detection are formulated using multiple March tests. The conducted experimental studies have confirmed the validity of the formulated provisions of the article. On the basis of the classical example of a linked coupling fault, the fulfillment of necessary and sufficient conditions for its detection by a single march test is shown.

Conclusion. The results of the research confirm that the proposed formal mathematical model for describing linked coupling faults makes it possible to determine their detection by marching tests. Within the framework of the proposed model, the necessary and sufficient conditions for detecting linked coupling faults by marching tests that detect single coupled faults are determined.

Keywords: testing of computing systems, coupling faults, linked faults, march tests of memory devices, address sequence

For citation. Yarmolik V. N. *Formal description model and conditions for detecting linked coupling faults of the memory devices*. Informatika [*Informatics*], 2023, vol. 20, no. 4, pp. 7−23 (In Russ.).
https://doi.org/10.37661/1816-0301-2023-20-4-7-23

Conflict of interest. The authors declare of no conflict of interest.

Введение. Запоминающие устройства занимают доминирующее место среди аппаратной составляющей современных вычислительных систем. Спрос на высокоскоростную память с высокой степенью интеграции и низким энергопотреблением растет беспрецедентными темпами. Это связано с тем, что облачные вычисления, искусственный интеллект и телекоммуникационные технологии пятого поколения (5G) среди прочего позиционируются как основные направления четвертой промышленной революции. Для обеспечения характеристик современной памяти вычислительных систем, отвечающей требованиям новых технологических достижений, неизбежно возрастают необходимость и значимость тестирования запоминающих устройств [1]. Главной задачей тестирования памяти является обнаружение ее неисправных состояний, которые описываются различными моделями неисправностей [2–4].

Среди большого множества неисправностей запоминающих устройств выделяют неисправности взаимного влияния (*coupling faults, CF*), в которых участвуют две запоминающие ячейки. К разновидностям таких неисправностей относятся [2–8]:

1. Инверсные неисправности взаимного влияния (*inversion coupling faults, CFin*). В неисправности $CFin(a_i, a_j)$ участвуют ячейки a_i и a_j с адресами $i \neq j$. Ячейка a_i оказывает воздействие (влияние) на ячейку a_j и называется агрессором (*aggressor, A*), а ячейка a_j, на которую оказывается влияние, называется жертвой (*victim, V*). При наличии данной неисправности логический переход из 1 в 0 ($\downarrow$) или из 0 в 1 ($\uparrow$) в агрессоре a_i приводит к инверсии ($\updownarrow$) логического значения в ячейке-жертве a_j, что описывается двумя моделями инверсных неисправностей $CFin(a_i, a_j)$, а именно: $\langle\uparrow, \updownarrow\rangle$ и $\langle\downarrow, \updownarrow\rangle$ [2, 3]. Для представления соотношения адреса i агрессора и j жертвы в адресном пространстве памяти используют символы $\wedge$ и $\vee$. Это позволяет определить четыре различные инверсные неисправности $CFin(a_i, a_j) = \{\wedge\langle\uparrow, \updownarrow\rangle, \wedge\langle\downarrow, \updownarrow\rangle, \vee\langle\uparrow, \updownarrow\rangle, \vee\langle\downarrow, \updownarrow\rangle\}$. Согласно классическим определениям символ $\wedge$ означает, что адрес агрессора меньше адреса жертвы ($i < j$), а символ $\vee$, наоборот, больше ($i > j$). Известно, что однократные маршевые тесты реализуют обращение к ячейкам памяти путем последовательного перебора всех их адресов. Поэтому соотношение $i < j$ сопоставляет не только значения адресов ячеек a_i и a_j, но и время обращения к ним. При реализации элемента маршевого теста с возрастающей последовательностью адресов ($\Uparrow$) и соотношением адресов $i < j$ первоначально реализуется обращение к ячейке a_i и только затем, через некоторое время, к ячейке a_j.

2. Неисправности прямого действия (*idempotent coupling faults, CFid*). При наличии данной неисправности во время логического перехода из 1 в 0 или из 0 в 1 во влияющей ячейке a_i происходит принудительная установка определенного логического значения 0 или 1 в ячейке-жертве a_j [2, 3]. Различают восемь неисправностей прямого действия $CFid(a_i, a_j) = \{\wedge\langle\uparrow,0\rangle, \wedge\langle\uparrow,1\rangle, \wedge\langle\downarrow,0\rangle, \wedge\langle\downarrow,1\rangle, \vee\langle\uparrow,0\rangle, \vee\langle\uparrow,1\rangle, \vee\langle\downarrow,0\rangle, \vee\langle\downarrow,1\rangle\}$ [2, 3].

3. Статические неисправности взаимного влияния (*state coupling faults, CFst*). Переход зависимой ячейки в какое-либо состояние $a_j \in \{0,1\}$ происходит при определенном значении $a_i \in \{0,1\}$ влияющей ячейки. Возможны восемь неисправностей $CFst(a_i, a_j) = \{\wedge\langle0,0\rangle, \wedge\langle0,1\rangle, \wedge\langle1,0\rangle, \wedge\langle1,1\rangle, \vee\langle0,0\rangle, \vee\langle0,1\rangle, \vee\langle1,0\rangle, \vee\langle1,1\rangle\}$ [2, 3].

Различают одиночные (однократные) неисправности взаимного влияния, в качестве которых может быть любая из приведенных выше моделей неисправностей, и кратные (многократные), включающие некоторое подмножество одиночных неисправностей взаимного влияния [2]. В множестве кратных неисправностей выделяют несвязные кратные (*unlinked multiple*) неисправности взаимного влияния. Для таких неисправностей конкретная ячейка памяти участвует только в одной неисправности взаимного влияния, входящей в рассматриваемую кратную неисправность. В общем случае несвязная n-кратная неисправность взаимного влияния включает четное количество $r = 2n$ ячеек памяти, половина из которых являются агрессорами, а половина – жертвами. Таким образом, каждая ячейка однократной неисправности *CF*, входящей в n-кратную, имеет только одну роль, а именно агрессора (A) или жертвы (V).

В силу чрезвычайно большой емкости запоминающих устройств в настоящее время широко применяются и по-прежнему разрабатываются тесты с временной сложностью, линейно зависящей от емкости памяти. Подобные тесты имеют общее название «маршевые тесты» (*March tests*) [1–8] и применяются для случая бит-ориентированной памяти, каждая ячейка которой

хранит один бит. В общем случае маршевый тест состоит из конечного числа маршевых элементов (*march elements*) [2, 3]. В свою очередь каждый маршевый элемент содержит символ, определяющий порядок формирования адресной последовательности (*address sequence*) ячеек запоминающего устройства, задающей порядок обращения к ячейкам. Символ $\Uparrow$ указывает на последовательный перебор адресов памяти по возрастанию (в прямом порядке), а символ $\Downarrow$ – по убыванию (в обратном порядке). Маршевый элемент содержит последовательность операций чтения (*read, r*) и записи (*write, w*), заключенных в круглые скобки и разделяемых точкой с запятой. Переход к следующей ячейке согласно адресной последовательности осуществляется только после выполнения всех операций с текущей ячейкой в соответствии с маршевым элементом [1–6]. Например, простейший маршевый элемент, применяемый во многих тестах, имеет вид $\Uparrow(r0,w1)$. В соответствии с этим элементом согласно адресной последовательности из каждой ячейки читается 0 и записывается 1. Каждая операция чтения сопряжена со сравнением прочитанного значения с ожидаемым (0).

Обнаружение неисправностей запоминающих устройств как результат применения маршевого теста основано на получении прочитанных из его ячеек неверных значений, отличных от эталонных (ожидаемых). Таким образом, фиксируется неисправное состояние памяти и при необходимости проводится диагностическая процедура [2–4, 7, 8].

В рамках маршевых тестов проблема обнаружения как одиночных, так и кратных несвязных неисправностей взаимного влияния считается решенной [2–4]. Для этого существуют эффективные тесты, хорошо зарекомендовавшие себя на практике [2–6]. Маршевые тесты, обнаруживающие неисправности взаимного влияния, предполагают однократное их применение для тестирования запоминающих устройств со стандартным начальным состоянием запоминающих ячеек, как правило нулевым, и стандартной счетчиковой адресной последовательностью [2, 3].

Проблема обнаружения связных неисправностей взаимного влияния (*linked coupling faults, LCF*) однократными маршевыми тестами в силу многообразия и сложных механизмов проявления таких неисправностей остается практически открытой и требует дальнейших исследований [3–8].

В настоящей статье предлагается формальная модель связных неисправностей взаимного влияния, описывающая роли всех ячеек, участвующих в неисправности, и сценарий их проявления в соответствии с маршевым тестом и его адресной последовательностью. Определяются необходимые и достаточные условия обнаружения таких неисправностей в рамках однократных тестов, и оценивается эффективность их обнаружения многократными тестами.

Связные неисправности взаимного влияния. В структуре классических моделей неисправностей памяти связные неисправности занимают одно из самых видных мест в силу сложности их обнаружения существующими маршевыми тестами памяти [2–6]. Под связными неисправностями понимают совокупность одиночных неисправностей различных типов, включающие в себя общие ячейки, которые участвуют в нескольких неисправностях одновременно. Отметим, что связная неисправность может включать одиночные неисправности одного и того же типа.

Рассматривая случай связной неисправности взаимного влияния, состоящей из n однократных *CF*, отметим, что в подобной неисправности участвуют $r < 2n$ ячеек памяти. Пример такой неисправности *LCF*1, состоящей из двух ($n = 2$) однократных неисправностей взаимного влияния, приведен на рис. 1 [2–4]. На этом рисунке ячейки памяти a_i, a_j и a_k с адресами $i < j < k$ представлены последовательно слева направо во временной зависимости обращения к ним в процессе выполнения прямой фазы маршевого теста. Дуги означают отношение между ячейками в рамках однократной неисправности взаимного влияния и проводятся от агрессора к жертве. Обозначения *LCF*1 и *LCF*2 относятся к одной и той же *LCF*-неисправности, механизмы проявления которой различны и зависят от временной последовательности формирования адресов участвующих в ней ячеек.

Рис. 1. Связная неисправность взаимного влияния для $n = 2$

Fig. 1. Linked coupling fault for n = 2

Из рис. 1 видно, что в *LCF*1 и *LCF*2 участвуют $r = 3$ ячейки. Они включают по две однократные неисправности *CF*, каждая из которых состоит из двух ячеек. Приведенный пример неисправности *LCF*1 представлен в большом числе литературных источников и позиционируется как пример необнаруживаемых неисправностей в рамках классических маршевых тестов [2–4]. Действительно, если предположить, что адреса ячеек a_i, a_j и a_k формируются в последовательности *i, j, k,* ячейки a_i и a_k участвуют в неисправности $\wedge\langle\uparrow, \updownarrow\rangle$ и, кроме того, ячейки a_j и a_k образуют аналогичную неисправность $\wedge\langle\uparrow, \updownarrow\rangle$, то будем иметь случай необнаруживаемой неисправности *LCF*1. Подчеркнем, что факт необнаружения этой неисправности относится к однократным маршевым тестам, использующим адресную последовательность, для которой выполняется условие генерирования сначала адреса *i*, затем *j* и последним *k*. Тогда очевидно, что последовательное проявление двух одиночных неисправностей, входящих в связную неисправность *LCF*1 (рис. 1), приведет к последовательному двукратному инвертированию состояния ячейки a_k. В результате состояние ячейки a_k при обращении к ней будет всегда верным и, соответственно, неисправность *LCF*1 является необнаруживаемой. Рассмотренный пример связной неисправности *LCF*1 приводится в качестве аргумента о необнаружении связных неисправностей вообще, хотя, в частности, связная неисправность, представленная на рис. 1 и состоящая, например, из двух неисправностей $\wedge\langle\uparrow,0\rangle$ или $\wedge\langle\uparrow,1\rangle$, будет обнаруживаемой. Более того, при использовании однократного маршевого теста, в котором адреса формируются во временной последовательности *i*, затем *k* и, наконец, *j*, неисправность *LCF*1, обозначенная как *LCF*2, является обнаруживаемой независимо от того, какие две неисправности *CF* образуют *LCF*2.

Таким образом, можно заключить, что связная неисправность взаимного влияния *LCF* зависит от количества *n* одиночных неисправностей взаимного влияния и их разновидностей, а также от числа *r* ячеек, участвующих в *LCF*. Для общего случая важным является также количество ролей (*A, V*) и их вид для каждой ячейки *LCF*. Поясним это на примере неисправности *LCF*1, для которой ячейки a_i и a_j играют роль *A* (агрессора), и для обеих ячеек эта роль одна. В то же время ячейка a_k имеет две роли *V* (жертвы). В общем случае понятие «роль» определяет взаимодействие между двумя ячейками, поэтому у ячейки a_k две роли *V* как результат взаимодействия с a_i и a_j в рамках одиночных неисправностей *CF*, составляющих *LCF*1.

Абстрагируясь от реалистичных и нереалистичных связных неисправностей *LCF*, множества которых в основном определяются конструктивными и технологическими особенностями памяти, сформулируем для общего случая *LCF* ряд положений. Во-первых, предположив, что в *LCF* участвуют *r* ячеек, оценим минимальное $\min(n)$ и максимальное $\max(n)$ количество *n* одиночных неисправностей взаимного влияния, образующих *LCF*. Отметим, что *LCF* состоит из одиночных *CF* и каждая из *r* ячеек входит как минимум в одну *CF*. Важным фактором для получения граничных значений $\min(n)$ и $\max(n)$ является следующее утверждение.

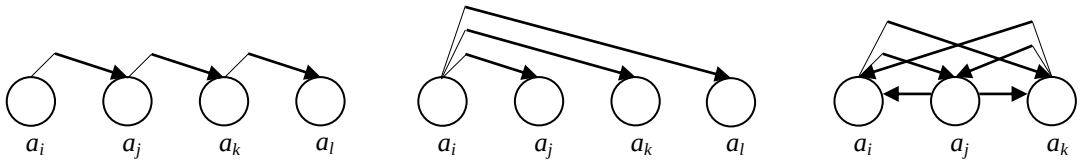
Утверждение 1. *Если неисправность* $CF(a_i, a_j)$ *входит в связную неисправность LCF, состоящую из r ячеек, то количество ролей для одной из ячеек* a_i *или* a_j*, образующих CF, должно быть не менее двух, а их общее количество для обеих ячеек неисправности CF лежит в диапазоне от 3 до* $4r - 6$.

Справедливость данного утверждения следует из того, что если каждая из ячеек a_i и a_j, образующих одиночную *CF*, имеет только одну роль, то данная *CF* не входит в *LCF*. Такая неисправность не связана с другими неисправностями *CF*, так как у ячеек, входящих в данную неисправность, отсутствуют роли, связывающие ее с другими ячейками памяти, и, соответст-

венно, с другими неисправностями *CF*. Таким образом, наличие хотя бы у одной из ячеек a_i и a_j неисправности $CF(a_i, a_j)$ больше чем одной роли обеспечивает связь между другими одиночными *CF*, образующими *LCF*. Следовательно, минимальное количество ролей для неисправности *CF*, входящей в *LCF*, равняется трем. Верхняя оценка количества ролей для ячеек участвующей в *LCF* неисправности $CF(a_i, a_j)$, равная $4r - 6$, достигается в том случае, когда каждая из двух ячеек неисправности выполняет роли и агрессора, и жертвы по отношению к остальным $r - 2$ ячейкам. Еще две роли возможны между самими ячейками a_i и a_j неисправности *CF*. Тогда максимальное число ролей ячеек неисправности *CF*, входящей в *LCF*, определяется как $2(r - 2) + 2(r - 2) + 2 = 4r - 6$.

Минимальное количество $\min(n)$ одиночных *CF*, образующих *LCF*, равняется $r - 1$. Это следует из утверждения 1, что соответствует минимальному количеству ролей у каждой *CF*, две из которых описывают саму *CF*, а третья определяет связь с другой *CF*. Примером может быть случай, когда в каждой последующей неисправности *CF* ячейка-агрессор одновременно является и жертвой, агрессором для которой выступает ячейка-жертва текущей неисправности. Таким образом, значение $\min(n) = r - 1$ также достижимо для второго крайнего случая, когда одна из r ячеек *LCF* является агрессором для остальных $r - 1$ ячеек. Здесь также имеем $r - 1$ одиночных неисправностей *CF*, одна из ячеек каждой из которых выполняет более двух ролей, в данном случае $r - 1$ ролей. Примеры подобных неисправностей *LCF*3 и *LCF*4 для $r = 4$ приведены на рис. 2.

Максимальное количество $\max(n)$ одиночных неисправностей *CF*, образующих *LCF*, достигается для случая, когда любая возможная пара ячеек из r ячеек образует две *CF*. Соответственно, первая ячейка выполняет роль *A*, вторая *V*, и наоборот. Таким образом, $\max(n) = 2 \cdot \binom{r}{2} = r(r-1)$.



*LCF*3 *LCF*4 *LCF*5

Рис. 2. Связные неисправности взаимного влияния для $r = 4$ и $r = 3$
Fig. 2. Linked coupling faults for r = 4 and r = 3

Пример *LCF*, состоящей из максимального количества одиночных *CF* для $r = 3$, приведен на рис. 2 как неисправность *LCF*5.

Таким образом, можно заключить, что для неисправности *LCF*, состоящей из r ячеек, количество n одиночных неисправностей *CF*, образующих эту неисправность, принадлежит диапазону

$$\min(n) = r - 1 \le n \le \max(n) = r(r-1). \quad (1)$$

Полученные предельные значения для количества n неисправностей *CF*, входящих в *LCF*, соответствуют аналогичным значениям для количества ребер в связных помеченных ориентированных графах, состоящих из r вершин. Пользуясь теорией графов, можно оценить число разновидностей неисправностей *LCF*, состоящих из r запоминающих ячеек, как общее число простых помеченных ориентированных графов, имеющих r вершин. Это число $2^{r(r-1)}$ растет экспоненциально от количества r ячеек и может быть использовано в качестве верхней оценки числа разновидностей *LCF*.

Отметим, что значения min(n) и max(n) количества n одиночных *CF*, образующих *LCF* (1), включающую r ячеек памяти, были получены без учета вида неисправностей *CF*, их расположения в памяти и специфики взаимного влияния друг на друга. Поэтому приведенные оценки количества конфигураций *LCF* лишь ориентировочно указывают на огромное число *LCF*. Поясним это на примере минимального числа (r = 3) ячеек памяти, участвующих в *LCF*, и минимального количества $n = r - 1 = 2$ одиночных *CF*, описывающих неисправности *LCF*. Учитывая конкретный вид *CF* (а именно, это неисправность *CFin, CFid* либо *CFst*) и их количество, равное 20 (см. предыдущий раздел), можно заключить, что две неисправности *CF*, в которых участвуют три ячейки, образуют 400 различных *LCF*. Количественная оценка 400 приведена только для одной из всевозможных неисправностей *LCF*, которые могут быть в случае конкретных r = 3 физических ячеек памяти с учетом их взаимного расположения.

Из рассмотренного примера о количестве неисправностей *LCF*, в которых участвуют только три ячейки, видно, что число таких неисправностей чрезвычайно велико. Даже в этом случае (небольшого количества ячеек, участвующих в *LCF*) число самих *LCF* и их многообразие не позволяют проводить анализ каждой из таких неисправностей на предмет ее обнаружения или необнаружения, что было возможным для одиночных *CF* [2, 4].

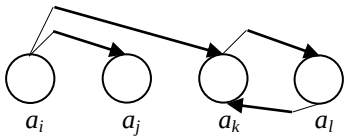
Необходимые и достаточные условия обнаружения *LCF*. Проблема обнаружения неисправностей *CF*, а также кратных неисправностей *CF* достаточно долго являлась основным стимулом разработки новых маршевых тестов [2]. Широко известные тесты *March C* и *March C–*, которые разрабатывались для обнаружения всевозможных одиночных неисправностей *CFid*, оказались весьма эффективными как по обнаруживающей способности неисправных состояний памяти в целом, так и по их невысокой временной сложности [2–6]. В рамках однократных маршевых тестов был разработан тест *March M*, в котором впервые реализовались условия обнаружения связных неисправностей и в первую очередь – двукратных связных неисправностей с различными конфигурациями [9]. Констатируя наличие необнаруживаемых неисправностей *LCF* однократными маршевыми тестами, были предложены более эффективные тесты, такие как *March LR* и *March LA* [10, 11]. В данных тестах обеспечивались условия обнаружения реально возможных неисправностей *LCF* в рамках рассматриваемых моделей связных неисправностей, а также технологических и конструктивных особенностей тестируемой памяти [10, 11]. Различные модификации тестов *March LR* и *March LA* эффективно используются для обнаружения динамических несвязных неисправностей [12], тестирования динамических запоминающих устройств [13] и реализации встроенного самотестирования памяти [14, 15].

В качестве условий обнаружения различных разновидностей *LCF* определялись необходимые множества маршевых элементов и их последовательность в тесте, которые обеспечивали обнаружение всех неисправностей рассматриваемого вида [2, 4]. Например, условием обнаружения всех одиночных *CFid* и всех связных двукратных *LCFid* является наличие в тесте следующих маршевых элементов: $\Uparrow\{ra, w\bar{a}, \ldots, wa, \ldots\}$, $\Uparrow\{r\bar{a}, wa, \ldots, w\bar{a}, \ldots\}$, $\Downarrow\{ra, w\bar{a}, \ldots, wa, \ldots\}$, $\Downarrow\{r\bar{a}, wa, \ldots, w\bar{a}, \ldots\}$ [4]. В свою очередь, условия наличия необходимых маршевых элементов в тесте формулировались на основании условий активизации конкретной неисправности и условий ее обнаружения. Подобные условия касаются ячеек памяти, непосредственно входящих в неисправность, и описывают последовательность действий, необходимых для обнаружения неисправности. В случае простейшей *CF* вида $\wedge\langle\uparrow,0\rangle$, где ячейка a_i является агрессором ($\uparrow$), а ячейка a_j – жертвой (0), необходимо последовательное во времени выполнение следующих условий. Во-первых, необходима первоначальная установка обеих ячеек в начальное состояние, а именно ячейки a_i в нулевое состояние, а ячейки a_j – в единичное. Затем выполняется условие активизации неисправности, которое заключается в выполнении изменения состояния из 0 в 1 в ячейке a_i, и, наконец, условие обнаружения неисправности как результата чтения нулевого содержимого ячейки a_j и сравнения его с эталонным (предварительно установленным в ячейке a_j) значением, равным 1.

Для установления условий обнаружения сложных, многочисленных и разнообразных конфигураций связных неисправностей взаимного влияния *LCF*(a_i, a_j, a_k, …, a_z), включающих r ячеек a_i, a_j, a_k, …, a_z с упорядоченным соотношением адресов $i < j < k < \ldots < z$, введем их фор-

мальное описание. Основой такого описания являются роли каждой из ячеек рассматриваемой *LCF* по отношению ко всем остальным ячейкам, входящим в описание неисправности. Аналогично, как и в простейшем случае *CF*, две ячейки a_i и a_j, $i \neq j$, могут влиять друг на друга, т. е. одна быть жертвой (*V*), а вторая – агрессором (*A*). В примерах моделей неисправностей *LCF*1, *LCF*2, *LCF*3, *LCF*4 и *LCF*5 роли каждой из ячеек указываются связями между ними. Ячейка с исходящей дугой – агрессор по отношению к ячейке, к которой направлена дуга. Для общего случая в рамках связных неисправностей, объединяющих две одиночные *CF*, в которые входят те же две ячейки a_i и a_j, введем третью роль (*both*, *B*) ячейки a_i, означающую две роли *A* и *V* по отношению к ячейке a_j. Соответственно, ячейка a_j по отношению к a_i также будет иметь две роли *V* и *A*, которые для нее будут обозначаться тем же символом *B*. Отсутствие связи (ролей) между ячейками a_i и a_j будем обозначать символом (-). Таким образом, в описании *LCF* будут представлены все ячейки, участвующие в ней, и все роли каждой из них.

В формальном описании $LCF(a_i, a_j, a_k, \ldots, a_z)$ последовательно представляется информация в виде набора символов -, *A*, *V* и *B* о всех ячейках $a_i, a_j, a_k, \ldots, a_z$. При описании ячеек также сохраняется их последовательность $a_i, a_j, a_k, \ldots, a_z$, т. е. на первой позиции всех описаний приводится информация о ячейке a_i, затем a_j и т. д. Эта последовательность определяется временем обращения к ячейкам, которое для однократных маршевых тестов соответствует последовательности их адресов. В качестве примера рассмотрим неисправность *LCF*6, представленную как в графическом виде, так и в виде нового формального описания (рис. 3).



$$LCF6(a_i, a_j, a_k, a_l) = \{\langle a_i, A, A, -\rangle; \langle V, a_j, -, -\rangle; \langle V, -, a_k, B\rangle; \langle -, -, B, a_l\rangle\}$$

Рис. 3. Связная неисправность взаимного влияния *LCF*6 и ее описание
*Fig. 3. Linked coupling fault LCF*6 *with description*

В приведенном примере *LCF*6 ячейки a_k и a_l, входящие в эту неисправность, образуют две простейшие *CF* и выступают по отношению друг к другу в обеих ролях, что обозначено символом *B* в описании *LCF*6 (рис. 3). Как уже пояснялось, символ *B* означает, что у ячейки есть две роли, а какую именно она реализует, определяется сценарием, который в данном случае задается последовательностью обращения к ячейкам в рамках маршевого теста. В зависимости от прямой ⇑ адресации, когда адреса соотносятся как $i < j < k < \ldots < z$, или обратной ⇓, когда $i > j > k > \ldots > z$, описание *LCF*6 представляется следующим образом:

$$\begin{aligned} &\Uparrow\{\langle a_i, A, A, -\rangle; \langle V, a_j, -, -\rangle; \langle V, -, a_k, A\rangle; \langle -, -, V, a_l\rangle\}, \\ &\Downarrow\{\langle a_i, A, A, -\rangle; \langle V, a_j, -, -\rangle; \langle V, -, a_k, V\rangle; \langle -, -, A, a_l\rangle\}. \end{aligned} \quad (2)$$

Как отмечалось ранее, упорядоченное соотношение адресов ячеек, участвующих в неисправности *LCF*, соответствует временной последовательности обращения к ним в рамках однократных маршевых тестов с фиксированной адресной последовательностью. Очевидно, что соотношение значений физических адресов ячеек, участвующих в *LCF*, и соотношение времен обращения к этим ячейкам могут быть различными. Примером подобной ситуации является неисправность, показанная на рис. 1. Аналогично неисправности *LCF*1 и *LCF*2, приведенные на рис. 1, могут быть описаны во введенной нотации следующим образом:

$$\begin{aligned} &LCF1 = LCF(a_i, a_j, a_k) = \{\langle a_i, -, A\rangle; \langle -, a_j, A\rangle; \langle V, V, a_k\rangle\}; \\ &LCF2 = LCF(a_i, a_k, a_j) = \{\langle a_i, A, -\rangle; \langle V, a_k, V\rangle; \langle -, A, a_j\rangle\}. \end{aligned} \quad (3)$$

Таким же образом описываются любые из возможных неисправностей *LCF*. Рассмотрим примеры данных описаний для ранее приведенных *LCF*:

$$LCF3(a_i, a_j, a_k, a_l) = \{\langle a_i, A, -, -\rangle; \langle V, a_j, A, -\rangle; \langle -, V, a_k, A\rangle; \langle -, -, V, a_l\rangle\};$$
$$LCF4(a_i, a_j, a_k, a_l) = \{\langle a_i, A, A, A\rangle; \langle V, a_j, -, -\rangle; \langle V, -, a_k, -\rangle; \langle V, -, -, a_l\rangle\};$$
$$LCF5(a_i, a_j, a_k) = \{\langle a_i, B, B\rangle; \langle B, a_j, B\rangle; \langle B, B, a_k\rangle\}.$$

Как и для *LCF*6 (2), неисправность *LCF*5 представляется в виде $\Uparrow\{\langle a_i, A, A\rangle; \langle V, a_j, A\rangle; \langle V, V, a_k\rangle\}$, $\Downarrow\{\langle a_i, V, V\rangle; \langle A, a_j, V\rangle; \langle A, A, a_k\rangle\}$.

Основным достоинством формальной модели описания конфигураций связных неисправностей является ее компактная запись, содержащая полную информацию о всех одиночных *CF*, образующих *LCF*. Например, из описания *LCF*1 следует, что данная связная неисправность образована из двух одиночных *CF*, а именно $CF(a_i, a_k) = \{\langle a_i, A\rangle; \langle V, a_k\rangle\}$ и $CF(a_j, a_k) = \{\langle a_j, A\rangle; \langle V, a_k\rangle\}$.

Упорядоченное обращение к ячейкам $a_i, a_j, a_k, \ldots, a_z$ неисправности $LCF(a_i, a_j, a_k, \ldots, a_z)$ при реализации маршевого элемента теста определяется временной последовательностью генерирования их адресов $i, j, k, \ldots, z$. Эта последовательность является важным элементом формального описания *LCF*, что, например, видно из описаний *LCF*1 и *LCF*2 (3) (см. рис. 1). Действительно, для последовательности адресов i, j, k используем описание неисправности *LCF*1, а для i, k, j – описание *LCF*2 (3). Позиция конкретной ячейки, предположим ячейки a_k, в ее описании так же, как и само описание в формальной модели *LCF*, соответствует временному порядку обращения к этой ячейке при реализации маршевого элемента теста. Например, в формальной модели $LCF1=\{\langle a_i, -, A\rangle; \langle -, a_j, A\rangle; \langle V, V, a_k\rangle\}$ описание ячейки a_k находится на третьей позиции, так как адрес k генерируется тестом после формирования адресов i и j. В самом описании $\langle V, V, a_k\rangle$ ячейки a_k ее обозначение также приведено на третьей позиции.

Таким образом, в основе новой формальной модели *LCF* лежат три роли, которые выполняют ячейки связной неисправности взаимного влияния, а именно роль агрессора (A), роль жертвы (V), а также роли жертвы и агрессора (B), выполняемые двумя ячейками одновременно по отношению друг к другу. В реальности предложенная модель описывает сценарий реализации ролей ячеек неисправности памяти, который определяется применяемым маршевым тестом и в первую очередь используемой в этом тесте адресной последовательностью обращения к ячейкам. Отметим, что для тестов с максимальной покрывающей неисправности *CF* способностью сценарий будет задаваться только адресной последовательностью, что и отображается в формальной модели *LCF*, как это видно на примере *LCF*1 и *LCF*2.

Для каждой ячейки $a_i, a_j, a_k, \ldots, a_z$ неисправности *LCF* в ее описании различают левую и правую часть, причем в левой части представлены те ячейки неисправности, которые активизируются до активизации текущей ячейки, а в правой части – после ее активизации. Под активизацией понимается обращение к ячейке памяти в соответствии с маршевым элементом теста и выполнение соответствующих операций записи и чтения.

Рассмотренная формальная модель *LCF* позволяет сформулировать условия обнаружения подобных неисправностей однократными маршевыми тестами.

Утверждение 2. *Необходимыми и достаточными условиями обнаружения связной неисправности взаимного влияния* $LCF(a_i, a_j, a_k, \ldots, a_z)$ *являются:*

1. Применение маршевого теста с произвольной адресной последовательностью, обнаруживающего всевозможные одиночные неисправности взаимного влияния CF.

2. Наличие в формальной модели неисправности $LCF(a_i, a_j, a_k, \ldots, a_z)$*, которая соответствует адресной последовательности теста, хотя бы одной ячейки* a_l, $l \in \{i, j, k, \ldots, z\}$*, в правой, либо левой, либо в обеих одновременно частях описания которой присутствует только одна роль V.*

Как показывалось ранее, произвольная неисправность $LCF(a_i, a_j, a_k, \ldots, a_z)$ представляет собой композицию одиночных неисправностей *CF*, все множество которых обнаруживается маршевыми тестами, например, такими, как *March LA* и *March LR* [10, 11]. Отметим, что всевоз-

можные одиночные неисправности *CF* гарантированно обнаруживаются указанными тестами только в случае, когда они не входят в связную неисправность *LCF*. В случае когда одиночные неисправности *CF* входят в *LCF*, указанные тесты либо обнаруживают эти неисправности по неверному состоянию ячейки агрессора, которая является жертвой в другой неисправности, либо гарантированно обеспечивают условие их активизации, изменив состояние ячейки-жертвы, с последующим обнаружением неверного состояния. Действительно, в случаях когда ячейка-агрессор не может выполнить условие активизации, наличие неисправного состояния памяти (наличие *LCF*) будет обнаружено по неверному состоянию такой ячейки. Это объясняется тем, что тесты, обнаруживающие одиночные *CF*, обеспечивают условия активизации всех подобных неисправностей путем задания необходимого начального состояния ячейки-агрессора (0, 1) или выполнения в ней одного из переходов (↑, ↓) (см. неисправности *CFin, CFid* и *CFst*) [2]. Таким образом, все ячейки-агрессоры неисправностей *CF*, входящие в *LCF*, либо будут являться причиной их обнаружения, став жертвами в рамках других *CF*, либо реализуют влияние на свои ячейки-жертвы. Изменения в ячейках-жертвах по отношению к первоначальным эталонным значениям в большинстве случаев будут обнаружены данными тестами [2, 4]. Это следует из свойств тестов, обнаруживающих все множество одиночных неисправностей взаимного влияния [2]. Исключением являются случаи маскирования, которые возникают при влиянии нескольких агрессоров на одну ячейку-жертву, когда, например, один агрессор изменяет эталонное значение жертвы на противоположное, а второй агрессор возвращает первоначальное значение ячейки-жертвы. Пример маскирования приведен на рис. 1 и описан формальной моделью *LCF*1 (3). Как видно, в данном случае у ячейки a_k две роли жертвы ($\langle V, V, a_k \rangle$) (3) и обе находятся в одной (левой) части ее описания. В случае когда ячейка-жертва выполняет только одну роль жертвы, т. е. у нее есть только один агрессор, перед обращением к этой ячейке маскирование невозможно. Более того, наличие двух ролей жертвы у ячейки одновременно, но в обеих частях формального описания означает, что содержимое данной ячейки будет прочитано только после однократного изменения ее состояния как результата проявления *CF*. Соответственно, *LCF* будет обнаружена. Подобный случай иллюстрируется примером неисправности *LCF*2, в формальном описании которой присутствуют две роли жертвы ($\langle V, a_k, V \rangle$) ячейки a_k, в данном случае одна роль V в правой, а вторая роль V в левой части формального описания.

Следует отметить, что механизм маскирования является весьма многообразным и характерен не только для запоминающих устройств, однако в рамках данной статьи рассматривается случай неисправностей *LCF* запоминающих устройств и возможное маскирование неисправностей *CF*, входящих в них [16].

Гарантированное обнаружение любой неисправности взаимного влияния *CF* безотносительно того, входит она либо не входит в связную неисправность *LCF*, обеспечивается анализом состояния ячейки-жертвы путем выполнения операции чтения ее состояния после реализации влияния на нее только одного агрессора (см. утверждение 2). Пример реализации подобного сценария описан формальной моделью *LCF*3, где анализ состояния ячейки-жертвы a_j ($\langle V, a_j, A, - \rangle$) выполняется после оказания влияния на нее только ячейки-агрессора a_i. Наличие хотя бы одной ячейки a_l, $l \in \{i, j, k, \ldots, z\}$, формальной модели $LCF(a_i, a_j, a_k, \ldots, a_z)$, в описании которой в правой, либо левой части, либо в обеих частях одновременно присутствует только одна роль жертвы, обеспечивает обнаружение данной $LCF(a_i, a_j, a_k, \ldots, a_z)$, что является необходимым и достаточным условием обнаружения *LCF* тестом, покрывающим одиночные *CF*.

Более сложное поведение запоминающего устройства происходит в случае наличия в нем неисправностей *LCF*, включающих *CF*, ячейки которых выполняют роль *B*, а также если проявление самих неисправностей *LCF* носит лавинообразный характер.

Сложные неисправности *LCF*. Под сложными неисправностями *LCF* будем понимать неисправности, включающие *CF*, ячейки которых выполняют роль *B*, или неисправности, проявление которых носит лавинообразный характер. Содержание сложных неисправностей *LCF* поясним на примерах *LCF*7, *LCF*8 и *LCF*9 (рис. 4).

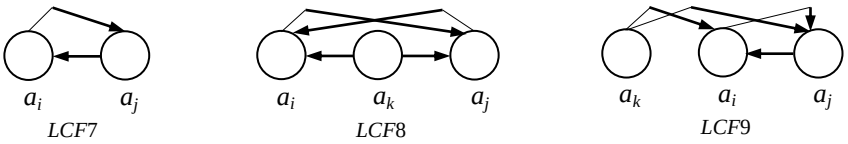


Рис. 4. Сложные связные неисправности взаимного влияния
Fig. 4. Complex linked coupling faults

Простейшим примером сложной *LCF* является *LCF*7, в которую входят две *CF*, образованные двумя ячейками a_i и a_j. Формальное описание неисправности $LCF7(a_i, a_j) = \{\langle a_i, B,\rangle; \langle B, a_j\rangle\}$ в зависимости от адресации $\Uparrow\{\langle a_i, A\rangle; \langle V, a_j\rangle\}$, $\Downarrow\{\langle a_i, V\rangle; \langle A, a_j\rangle\}$ показывает его соответствие условиям утверждения 2, что свидетельствует об ее обнаружении. Причем эта неисправность будет обнаруживаться в случае проявления одной из двух неисправностей $CF(a_i, a_j)$ или $CF(a_j, a_i)$ при стандартной реализации маршевого элемента теста и отсутствии лавинообразного характера проявления неисправностей.

Следует отметить, что чаще всего сложные неисправности *LCF* проявляются (активизируются) за счет активизации только одного из агрессоров *CF*, входящих в эту неисправность. Подобные ситуации объясняются физическими особенностями памяти. Как отмечалось в ряде литературных источников [9, 10, 12, 13], реалистичные неисправности *CF* образуются из ячеек, связанных общими шинами, на которые подаются электрические сигналы. Соответственно, при наличии сигналов, инициирующих изменение в ячейке-агрессоре, и происходит влияние на его жертву, которая, будучи потенциальным агрессором, не оказывает влияния на свою жертву.

Лавинообразный характер *LCF* возникает тогда, когда активизация одной *CF*, входящей в *LCF*, вызывает активизацию другой либо других *CF*, принадлежащих этой же *LCF*. Данный эффект заключается в том, что в активизированной ячейке-жертве одной неисправности *CF*, которая выполняет роль агрессора в другой *CF*, происходит изменение, активизирующее эту неисправность.

Структура сложных неисправностей *LCF*, а также сценарий их обнаружения, определяемый маршевым тестом, может приводить к различным эффектам их лавинообразного проявления. Специфика лавинообразного характера неисправностей во многом определяется и физическими характеристиками памяти. Возможна различная глубина лавинообразного характера проявления неисправности *LCF*, определяемая числом последовательно во времени активизированных *CF*. Эффект гонок сопряжен с процессом активизации нескольких *CF*, которые приводят к взаимоисключающим действиям на ячейку-жертву. Этот же эффект влияет на последовательность активизации других *CF*, входящих в *LCF*. Режим автогенерации, как правило затухающей, также является возможным эффектом, когда ячейка, входящая в *LCF*, многократно изменяет свое состояние на противоположное.

В случае неисправности *LCF*7 возможен лавинообразный характер ее проявления, когда при реализации элемента маршевого теста активизируется $CF(a_i, a_j)$, т. е. в ячейке-агрессоре a_i проводится операция записи, в результате которой изменяется состояние в ячейке-жертве a_j. В зависимости от вида другой $CF(a_j, a_i)$ как результат изменения состояния ячейки a_j, в данном случае агрессора, может произойти очередное изменение состояния a_i. При определенных условиях, например, если в *LCF*7 входят неисправности $CF(a_i, a_j) = \langle\uparrow, \updownarrow\rangle$ и $CF(a_j, a_i) = \langle\uparrow, \updownarrow\rangle$ и присутствует лавинообразный эффект, при начальных значениях $a_i = a_j = 0$ состояние ячейки a_i останется неизменным. Таким образом, при лавинообразном проявлении неисправности *LCF*7 отличными от ожидаемых будут состояния обеих ячеек, что обеспечивает обнаружение этой неисправности.

Лавинообразный характер проявления одиночных *CF*, входящих в *LCF*, может приводить к маскированию состояния ячеек запоминающего устройства. В конечном счете маскирование состояния ячейки может привести к маскированию неисправности *LCF*.

В табл. 1 приведены последовательные состояния ячеек неисправностей *LCF*, включающих только $CFin = \langle\uparrow, \updownarrow\rangle$, как результат применения маршевого элемента $\Uparrow(r0,w1)$. Каждая из неисправностей описывается как $LCF7 = LCF(a_i, a_j) = \{\langle a_i, B\rangle; \langle B\, a_j\rangle\}$, $LCF8 = LCF(a_i, a_k, a_j) = \{\langle a_i, V, B\rangle; \langle A, a_k, A\rangle; \langle B, V, a_j\rangle\}$, $LCF9 = LCF(a_k, a_i, a_j) = \{\langle a_k, A, A\rangle; \langle V, a_i, B\rangle; \langle V, B, a_j\rangle\}$.

Таблица 1
Диаграмма состояний ячеек, входящих в *LCF*7, *LCF*8 и *LCF*9

Table 1
*Cells state diagram included in LCF*7, *LCF*8 *and LCF*9

Неисправность *Malfunction*	*LCF*7	*LCF*7*	*LCF*8	*LCF*8*	*LCF*9	*LCF*9*
	a_i, a_j	a_i, a_j	a_i, a_k, a_j	a_i, a_k, a_j	a_k, a_i, a_j	a_k, a_i, a_j
Начальное состояние	0, 0	0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
⇑(*r*0,*w*1)	0/1, 0/**1**	0/1/0, 0/**1**	0/1, 0, 0/1 1/0, 0/1, 1/0 0/1, 1, 0/1	0/1/0, 0, 0/1 0/1, 0/1, 1/0/**1**	0/1, 0/1, 0/1 1, **1**, 1	0/1, 0/1/0, 0/1 1, 0/1, 1/0 1, 1/**0**, 0/1
Эталонное состояние	1, 1	1, 1	1, 1, 1	1, 1, 1	1, 1, 1	1, 1, 1

Для нулевого начального состояния ячеек *LCF* в табл. 1 приведены последовательности изменений их состояний как результат проявления неисправностей *LCF*7, *LCF*8 и *LCF*9 в случае стандартного процесса их активизации и обнаружения. Лавинообразный характер проявления указанных неисправностей приведен в столбцах табл. 1, помеченных как *LCF*7*, *LCF*8* и *LCF*9*. Последовательность адресов, генерируемых тестом, соответствует последовательности индексов обозначений ячеек. Например, для неисправности *LCF*8* согласно маршевому элементу ⇑(*r*0,*w*1) первоначально выполняется обращение к ячейке a_i, затем к ячейке a_k и, наконец, к ячейке a_j. В содержательной части таблицы приводится последовательность изменений состояния ячеек как результат действий маршевого элемента и проявления рассматриваемой неисправности. Продолжая пример неисправности *LCF*8*, первое обращение будет выполнено к ячейке a_i, из которой будет прочитано значение 0 и записана 1 (0/1). Такое изменение (↑) состояния ячейки a_i активизирует неисправность $CF(a_i, a_j) = \langle\uparrow, \updownarrow\rangle$, что приведет к инвертированию состояния ячейки a_j (0/1). Выполнение перехода из 0 в 1 (↑) в ячейке a_j, в свою очередь, активизирует неисправность $CF(a_j, a_i) = \langle\uparrow, \updownarrow\rangle$, приводящую к лавинообразному инвертированию состояния ячейки a_i (1/0). Окончательно после обращения к ячейке a_i состояние ячеек примет вид a_i, a_k, a_j = 0/1/0, 0, 0/1 = 0, 0, 1. Символы **0** и **1**, приведенные в табл. 1, свидетельствуют об отличии состояний ячеек от ожидаемых, что означает обнаружение соответствующей *LCF* при выполнении операции чтения из этих ячеек и прекращение выполнения теста.

Наличие символа **1** для неисправностей *LCF*7 и *LCF*7* свидетельствует об их обнаружении, а отсутствие символов **0** и **1** в случае неисправности *LCF*8 говорит об ее необнаружении, что также следует из ее формального описания, в котором не выполняются условия утверждения 2. В то же время неисправность *LCF*9 является обнаруживаемой, хотя физически это та же неисправность *LCF*8 с измененным сценарием обращения к ячейкам, входящим в эту неисправность. Лавинообразный характер проявления неисправности *LCF*9* после обращения к ячейке a_k сказывается на состоянии ячеек, участвующих в неисправности, и имеет вид a_k, a_i, a_j = 0/1, 0/1/0, 0/1 = 1, 0, 1 (см. табл. 1). В данном случае возможно и другое поведение ячеек памяти, а именно a_k, a_i, a_j = 0/1, 0/1, 0/1/0 = 1, 1, 0, входящих в неисправности *LCF*9*. Это поведение определяется эффектом гонок.

Как показывалось ранее, роль ячейки *B*, входящей в *LCF*, включает две роли, каждая из которых реализуется в одной из двух фаз теста, а именно прямой либо обратной, что эквивалентно наличию одной роли жертвы в правой либо левой части описания *LCF*. Наличие двух ролей, а именно *V* и *B*, в правой либо левой части описания хотя бы одной ячейки, входящей в *LCF*, также гарантирует обнаружение этой неисправности, так как в одной из фаз теста роль *B* эквивалентна роли *A*. Соответственно, этот случай сводится к случаю наличия одной роли жертвы. Таким образом, можно заключить, что наличие в формальной модели $LCF(a_i, a_j, a_k, \ldots, a_z)$, которая соответствует сценарию адресной последовательности теста, хотя бы одной ячейки a_l, $l \in \{i, j, k, \ldots, z\}$, в описании которой в правой или левой части присутствует только одна из ролей жертвы *V* или *B* либо обе одновременно, является необходимым и достаточным условием обнаружения данной неисправности.

Возвращаясь к неисправности, приведенной на рис. 1 для случая последовательности адресов *i, j, k* теста, обнаруживающего одиночные связные неисправности, и ее описанию $\{\langle a_i, -, A\rangle; \langle -, a_j, A\rangle; \langle V, V, a_k\rangle\}$, можно констатировать необнаружение этой неисправности, так как не выполняются условия утверждения 2. В то же время применение других адресных последовательностей (других сценариев) может обеспечить ее обнаружение, как это видно из всевозможных описаний данной неисправности в зависимости от временной последовательности обращения к ячейкам, входящим в данную неисправность:

$$\begin{aligned}
&LCF(a_i, a_j, a_k) = \{\langle a_i, -, A\rangle; \langle -, a_j, A\rangle; \langle V, V, a_k\rangle\};\\
&LCF(a_i, a_k, a_j) = \{\langle a_i, A, -\rangle; \langle V, a_k, V\rangle; \langle -, A, a_j\rangle\};\\
&LCF(a_j, a_i, a_k) = \{\langle a_j, -, A\rangle; \langle -, a_i, A\rangle; \langle V, V, a_k\rangle\};\\
&LCF(a_j, a_k, a_i) = \{\langle a_j, A, -\rangle; \langle V, a_k, V\rangle; \langle -, A, a_i\rangle\};\\
&LCF(a_k, a_i, a_j) = \{\langle a_k, V, V\rangle; \langle A, a_i, -\rangle; \langle A, -, a_j\rangle\};\\
&LCF(a_k, a_j, a_i) = \{\langle a_k, V, V\rangle; \langle A, a_j, -\rangle; \langle A, -, a_i\rangle\}.
\end{aligned} \qquad (4)$$

Очевидно, что для двух случаев из шести в выражениях (4) выполняются положения утверждения 2, а именно для $LCF(a_i, a_k, a_j)$ и $LCF(a_j, a_k, a_i)$, что обеспечивает необходимые и достаточные условия их обнаружения.

Многократные маршевые тесты для обнаружения *LCF*. Невысокая покрывающая способность связных неисправностей *LCF* классическими однократными маршевыми тестами объясняется последовательной процедурой доступа к ячейкам памяти и неизменным сценарием реализации теста [16]. Анализируя пример шести формальных описаний (4) неисправности *LCF*1, показанной на рис. 1, можно отметить, что только два из шести сценариев позволят обнаружить эту неисправность. Соответственно, неисправность будет обнаружена только путем изменения сценария и многократного применения теста. Сценарием в данном случае будет адресная последовательность маршевого теста, обнаруживающего все множество одиночных неисправностей взаимного влияния.

Идея многократного тестирования памяти рассматривалась в рамках исчерпывающего, псевдоисчерпывающего и неразрушающего тестирования памяти [2, 4, 16]. Во всех случаях необходимым являлось повторение маршевого теста для различных сценариев, определяемых степенями свободы, которые присущи маршевым тестам [17, 18]. Показано, что маршевые тесты памяти обнаруживают сложные неисправности путем их многократного применения для различных адресных последовательностей и (или) начального состояния запоминающих ячеек [3, 20]. Важным результатом многократного тестирования запоминающих устройств является обнаружение всевозможных их неисправных состояний. Однако достижение 100%-й полноты покрытия может потребовать большого числа (кратности) применения теста.

Для случая изменяемой адресной последовательности, определяющей сценарий для $LCF(a_i, a_j, a_k, \ldots, a_z)$, всегда существует временная последовательность обращения к ячейкам $a_i, a_j, a_k, \ldots, a_z$, для которой выполняются условия утверждения 2. Примером этому может быть неисправность *LCF*1 (4). Для нее существуют две из шести последовательностей адресов ячеек a_i, a_j и a_k, для которых эта неисправность будет обнаруживаемой. В общем случае для любого неисправного состояния памяти существует множество адресных последовательностей, для которого это состояние памяти будет обнаружено однократным тестом [3, 4, 16]. Соответственно, при случайной процедуре выбора адресной последовательности отношение *p* количества адресных последовательностей, для которых неисправность является обнаруживаемой, к общему числу последовательностей будет определять вероятность обнаружения этой неисправности однократным тестом.

Если предположить, что однократное применение маршевого теста дает возможность обнаруживать неисправности *LCF* с вероятностью $p_1 = p$, то *f*-кратное его использование при произвольных (случайных) адресных последовательностях позволяет достичь значения вероятности обнаружения p_f с помощью выражения [3, 21]

$$p_f = 1-(1-p_1)^f,\ f = 1, 2, 3, \ldots. \qquad (5)$$

Очевидно, что с ростом f значение вероятности p_f стремится к единице. Соответственно, полнота покрытия таких неисправностей стремится к 100 %.

Для неисправности *LCF*1 с учетом (4) $p_1 = 2/6 = 0{,}333\ldots$.

Экспериментальные исследования. Для оценки эффективности обнаружения неисправностей *LCF* и подтверждения полученных результатов было разработано программное средство, которое моделирует процесс тестирования памяти с использованием маршевых тестов и осуществляет визуализацию процесса тестирования, а также анализ обнаруживаемых тестом неисправностей запоминающих элементов памяти.

Входными данными программного средства являются:

– модель памяти, задаваемая количеством ячеек;

– модели неисправностей памяти и их количество;

– маршевый тест;

– кратность f маршевого теста;

– адресная последовательность для каждой итерации теста.

Выходными данными программного средства являются:

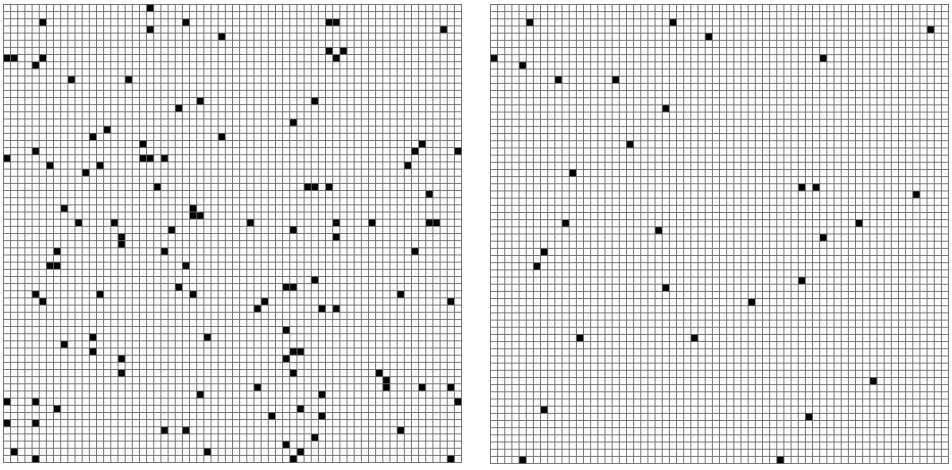
– графическое отображение алгоритма выбранного маршевого теста;

– визуальное отображение результатов тестирования;

– отчеты об обнаруженных неисправностях.

Программное средство разработано на языке C#. В качестве основы логики работы моделей неисправностей *LCF* был использован поведенческий паттерн «наблюдатель» (*observer*).

На рис. 5, *a* показан пример массива ячеек, выполняющих роль жертв в исходных неисправностях *LCF*1, включающих три ячейки, а на рис. 5, *b* – результат работы программного средства после двух итераций маршевого теста *March LR*. На первой итерации была использована адресная последовательность, представляющая собой отраженный код Грея, а на второй – последовательность АнтиГрея.



a) *b)*

Рис. 5. Пример визуализации процесса тестирования с применением разработанного программного средства: *a)* массив ячеек, выполняющих роль жертв; *b)* количество ячеек жертв после двух итераций теста

Fig. 5. An example of visualization of testing process using the developed software tool: a) an array of cells acting as victims; b) number of victim cells after two test iterations

Как видно из рис. 5, *b*, количество ячеек жертв необнаруженных *LCF* и, соответственно, самих неисправностей после реализации двукратного теста *March LR* существенно уменьшилось.

Для оценки вероятности обнаружения неисправности *LCF*1 многократными маршевыми тестами был проведен следующий эксперимент. В программном средстве была смоделирована память размером 8 бит и установлена связная неисправность взаимного влияния *LCF*, в которой адреса ячеек памяти расположены в соотношении $i < j < k$. В рассматриваемую *LCF* входят неисправности $CF(a_i, a_k) = \langle\uparrow, \updownarrow\rangle$ и $CF(a_j, a_k) = \langle\uparrow, \updownarrow\rangle$.

Для обнаружения неисправности *LCF* многократно выполнялся маршевый тест *March LA*. В каждой новой итерации теста использовалась случайно выбранная адресная последовательность из всех возможных ранее сгенерированных 40 320 последовательностей для памяти размером 8 бит. Было выполнено 10 000 экспериментов, каждый из которых заключался в обнаружении смоделированной ранее неисправности *f*-кратным маршевым тестом. В табл. 2 представлены значения вероятности обнаружения неисправности *LCF*1 в зависимости от кратности *f* маршевого теста в виде ее экспериментальной оценки (эксп.) и значения, вычисленные согласно выражению (5) (теор.).

Таблица 2
Вероятность p_f обнаружения неисправности *LCF*1 *f*-кратным тестом *March LR*

Table 2
*Probability p_f of LCF*1 *fault detection by f-run test March LR*

f		1	2	3	4	5	6	7	8	9	10	11	12
p_f	Теор.	0,3333	0,5555	0,7037	0,8024	0,8683	0,9122	0,9414	0,9609	0,9739	0,9826	0,9884	0,9922
	Эксп.	0,3300	0,5521	0,6985	0,7982	0,8669	0,9104	0,9402	0,9592	0,9745	0,9829	0,9886	0,9919

Результат эксперимента показал, что вероятность обнаружения неисправности *LCF*1 однократным маршевым тестом *March LR* составляет 0,33. С увеличением кратности теста *f* вероятность обнаружения неисправности стремится к 1, что соответствует выражению (5) и подтверждает правильность выбора авторами данной математической модели.

Заключение. Предложенная авторами формальная математическая модель описания связных неисправностей взаимного влияния позволила сформулировать необходимые и достаточные условия обнаружения таких неисправностей. Обобщенный характер формальной модели исключает необходимость анализа каждой конкретной конфигурации связной неисправности. Несомненным достоинством новой формальной модели является ее модифицируемость за счет изменения адресной последовательности маршевого теста. Сценарий поведения ячеек при тестировании памяти определяется адресной последовательностью однократного маршевого теста. Перспективным представляется применение предложенной модели связных неисправностей взаимного влияния для их идентификации и диагностической локализации. Очевидна применимость данной модели и для других видов связных неисправностей запоминающих устройств, а также для широкого класса кодочувствительных неисправностей.

Вклад авторов. *В. Н. Ярмолик* предложил формальную модель описания связных неисправностей взаимного влияния и сформулировал условия их обнаружения, *Д. В. Деменковец* и *В. В. Петровская* приняли участие в экспериментальных исследованиях и анализе полученных результатов, *А. А. Иванюк* обобщил и проанализировал полученные результаты.

Список использованных источников

1. Lee, K. Coverage re-evaluation of memory test algorithms with physical memory characteristics / K. Lee, J. Kim, S. Baeg // IEEE Access. – 2021. – Vol. 9. – P. 124632–124639.

2. Goor, A. J. Testing Semiconductor Memories, Theory and Practice / A. J. Goor. – Chichester, UK : John Wiley & Sons, 1991. – 536 p.

3. Ярмолик, С. В Маршевые тесты для тестирования ОЗУ / С. В. Ярмолик, А. П. Занкович, А. А. Иванюк. – Saarbrücken, Germany : LAP Lambert Academic Publishing, 2012. – 302 с.

4. Неразрушающее тестирование запоминающих устройств / В. Н. Ярмолик [и др.]. – Минск : Бестпринт, 2005. – 230 с.

5. Cascaval, P. Efficient march test for 3-coupling faults in random access memories / P. Cascaval, S. Bennett // Microprocessors and Microsystems. – 2001. – Vol. 24, no. 10. – P. 501–509.

6. Caşcaval, P. March test algorithm for unlinked static reduced three-cell coupling faults in random-access memories / P. Caşcaval, D. Caşcaval // Microelectronics J. – 2019. – Vol. 93, iss. C. – Art. 104619.

7. Cockburn, B. E. Synthesized transparent BIST for detecting scrambled pattern-sensitive faults in RAMs / B. E. Cockburn, Y. F. Sat // Proc. of the IEEE Intern. Test Conf., Washington, DC, USA, 21–25 Oct. 1995. – Washington, DC, USA, 1995. – P. 23–32.

8. Cockburn, B. E. Deterministic tests for detecting single V-coupling faults in RAMs / B. E. Cockburn // J. of Electronic Testing: Theory and Applications. – 1994. – Vol. 5, no. 1. – P. 91–113.

9. Mikitjuk, V. G. RAM testing algorithm for detection multiple linked faults / V. G. Mikitujk, V. N. Yarmolik // Proc. of the 1996 European Design and Test Conf. (ED&TC'96), Paris, France, 11–14 Mar. 1996. – Paris, France, 1996. – P. 435–440.

10. March LR: a test for realistic linked faults / A. J. Goor [et al.] // Proc. of the 14th VLSI Test Symp., Princeton, NJ, USA, 28 Apr. – 01 May 1996. – Princeton, NJ, USA, 1996. – P. 272–280.

11. March LA: a test for linked memory faults / A. J. Goor [et al.] // Proc. of the 1997 European Design and Test Conf. (ED&TC'97), Paris, France, 17–20 Mar. 1997. – Paris, France, 1997. – P. 627.

12. Modified March MSS for unlinked dynamic faults detection / L. W. Ying [et al.] // Proc. of the IEEE 20th Student Conf. on Research and Development (SCOReD), Bangi, Malaysia, 08–09 Nov. 2022. – Bangi, Malaysia, 2022. – P. 68–72.

13. Chou, C.-W. Testing inter-word coupling faults of wide I/O DRAMs / C.-W. Chou, Y.-X. Chen, J.-F. Li // Proc. of the 2015 IEEE 24th Asian Test Symp., Mumbai, India, 22–25 Nov. 2015. – Mumbai, India, 2015. – P. 22–25.

14. Manasa, R. Implementation of BIST technology using March-LR algorithm / R. Manasa, R. Verma, D. Koppad // Proc. of the 2019 4th Intern. Conf. on Recent Trends on Electronics Information Communication & Technology (RTEICT), Bangalore, India, 17–18 May 2019. – Bangalore, India, 2019. – P. 1208–1212.

15. Implementation of minimized March SR algorithm in a memory BIST controller / A. Z. Jidin [et al.] // J. of Engineering and Technology. – 2022. – Vol. 13, no. 2. − P. 1−14.

16. Ярмолик, В. Н. Контроль и диагностика цифровых устройств ЭВМ / В. Н. Ярмолик. – Минск : Наука и техника, 1988. – 240 с.

17. Sokol, B. Address sequence for march tests to detect pattern sensitive faults / B. Sokol, S. V. Yarmolik // Proc. of 3rd IEEE Intern. Workshop on Electronic Design Test and Applications (DELTA'06), Kuala Lumpur, Malaysia, 17–19 Jan. 2006. – Kuala Lumpur, Malaysia, 2006. – P. 354–357.

18. Sokol, B. Impact of the address changing on the detection of pattern sensitive faults / B. Sokol, I. Mrozek, V. N. Yarmolik // Information Processing and Security Systems. – London : Springer Science + Business Media, Inc., 2005. – P. 217–226.

19. Yarmolik, S. V. Address sequences and backgrounds with different Hamming distance for multiple run March tests / S. V. Yarmolik // IEEE Intern. J. of Applied Mathematics and Computer Science. – 2008. – Vol. 18, no. 3. − P. 329−339.

20. Mrozek, I. Multi-run Memory Tests for Pattern Sensitive Faults / I. Mrozek. – Cham : Springer International Publishing AG, 2019. – 135 p.

21. Построение и применение маршевых тестов для обнаружения кодочувствительных неисправностей запоминающих устройств / В. Н. Ярмолик [и др.] // Информатика. – 2021. – № 1(18). – С. 25–42.

References

1. Lee K., Kim J., Baeg S. Coverage re-evaluation of memory test algorithms with physical memory characteristics. *IEEE Access*, 2021, vol. 9, pp. 124632–124639.

2. Goor A. J. *Testing Semiconductor Memories, Theory and Practice*. Chichester, UK, John Wiley & Sons, 1991, 536 p.

3. Yarmolik S. V., Zankovich A. P., Ivaniuk A. A. Marshevue testu dlya testirovaniya OZU. *March Tests for Memory Testing*. Saarbrücken, Germany, LAP Lambert Academic Publishing, 2012, 302 p. (In Russ.).

4. Yarmolik V. N., Murashko I. A., Kummert A., Ivaniuk A. A. Nerazrushayuschee testirovanie zapominayuschih ustroistv. *Non-Destructive Storage Testing*. Minsk, Bestprint, 2005, 230 p. (In Russ.).

5. Cascaval P., Bennett S. Efficient march test for 3-coupling faults in random access memories. *Microprocessors and Microsystems*, 2001, vol. 24, no. 10, pp. 501–509.

6. Caşcaval P., Caşcaval D. March test algorithm for unlinked static reduced three-cell coupling faults in random-access memories. *Microelectronics Journal*, 2019, vol. 93, iss. C, art. 104619.

7. Cockburn B. E., Sat Y. F. Synthesized transparent BIST for detecting scrambled pattern-sensitive faults in RAMs. *Proceedings of the IEEE International Test Conference, Washington, DC, USA, 21–25 October 1995*. Washington, DC, USA, 1995, pp. 23–32.

8. Cockburn B. E. Deterministic tests for detecting single V-coupling faults in RAMs. *Journal of Electronic Testing: Theory and Applications*, 1994, vol. 5, no. 1, pp. 91–113.

9. Mikitjuk V. G., Yarmolik V. N. RAM testing algorithm for detection multiple linked faults. *Proceedings of the 1996 European Design and Test Conference (ED&TC'96), Paris, France, 11–14 March 1996*. Paris, France, 1996, pp. 435–440.

10. Goor A. J., Gaydadjiev G. N., Yarmolik V. N., Mikitujk V. G. March LR: a test for realistic linked faults. *Proceedings of the 14th VLSI Test Symposium, Princeton, NJ, USA, 28 April – 01 May 1996.* Princeton, NJ, USA, 1996, pp. 272–280.

11. Goor A. J., Gaydadjiev G. N., Yarmolik V. N., Mikitujk V. G. March LA: a test for linked memory faults. *Proceedings of the 1997 European Design and Test Conference (ED&TC'97), Paris, France, 17–20 March 1997*. Paris, France, 1997, p. 627.

12. Ying L. W., Hussin R., Ahmad N., Fook L. W., Jidin A. Z. Modified March MSS for unlinked dynamic faults detection. *Proceedings of the IEEE 20th Student Conference on Research and Development (SCOReD), Bangi, Malaysia, 08–09 November 2022*. Bangi, Malaysia, 2022, pp. 68–72.

13. Chou C.-W., Chen Y.-X., Li J.-F. Testing inter-word coupling faults of wide I/O DRAMs. *Proceedings of the 2015 IEEE 24th Asian Test Symposium, Mumbai, India, 22–25 November 2015*. Mumbai, India, 2015, pp. 22–25.

14. Manasa R., Verma R., Koppad D. Implementation of BIST technology using March-LR algorithm. *Proceedings of the 2019 4th International Conference on Recent Trends on Electronics Information Communication & Technology (RTEICT), Bangalore, India, 17–18 May 2019*. Bangalore, India, 2019, pp. 1208–1212.

15. Jidin A. Z., Hussin R., Mispan M. S., Lee W. F., Zakaria N. A. Implementation of minimized March SR algorithm in a memory BIST controller. *Journal of Engineering and Technology*, 2022, vol. 13, no. 2, pp. 1−14.

16. Yarmolik V. N. Kontrol' i diagnostika cifrovyh ustrojstv jelektronno-vychislitel'nyh mashin. *Monitoring and Diagnostics of Digital Devices of Electronic Computers*. Minsk, Nauka i tehnika, 1988, 240 p. (In Russ.).

17. Sokol B., Yarmolik S. V. Address sequence for march tests to detect pattern sensitive faults. *Proceedings of 3rd IEEE International Workshop on Electronic Design Test and Applications (DELTA'06), Kuala Lumpur, Malaysia, 17–19 January 2006*. Kuala Lumpur, Malaysia, 2006, pp. 354–357.

18. Sokol B., Mrozek I., Yarmolik V. N. Impact of the address changing on the detection of pattern sensitive faults. *Information Processing and Security Systems*. London, Springer Science + Business Media, Inc., 2005, pp. 217–226.

19. Yarmolik S. V. Address sequences and backgrounds with different Hamming distance for multiple run March tests. *IEEE International Journal of Applied Mathematics and Computer Science*, 2008, vol. 18, no. 3, pp. 329−339.

20. Mrozek I. *Multi-run Memory Tests for Pattern Sensitive Faults*. Cham, Springer International Publishing AG, 2019, 135 p.

21. Yarmolik V. N., Levantsevich V. A., Demenkovets D. V., Mrozek I. *Construction and application of march tests for pattern sensitive memory faults detection*. Informatika [*Informatics*], 2021, vol. 18, no. 1, pp. 25–42 (In Russ.).

Информация об авторах

Ярмолик Вячеслав Николаевич, доктор технических наук, профессор, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: yarmolik10ru@yahoo.com

Деменковец Денис Викторович, магистр технических наук, старший преподаватель, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: demenkovets@bsuir.by

Петровская Вита Владленовна, магистр технических наук, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: vita.petrovskaya@gmail.com

Иванюк Александр Александрович, доктор технических наук, доцент, профессор кафедры информатика, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: ivaniuk@bsuir.by

Information about the authors

Vyacheslav N. Yarmolik, D. Sc. (Eng.), Prof., Belarusian State University of Informatics and Radioelectronics.
E-mail: yarmolik10ru@yahoo.com

Denis V. Demenkovets, M. Sc. (Eng.), Senior Lecture, Belarusian State University of Informatics and Radioelectronics.
E-mail: demenkovets@bsuir.by

Vita V. Petrovskaya, M. Sc. (Eng.), Belarusian State University of Informatics and Radioelectronics.
E-mail: vita.petrovskaya@gmail.com

Alexander A. Ivaniuk, D. Sc. (Eng.), Assoc. Prof., Prof. of Computer Science Department, Belarusian State University of Informatics and Radioelectronics.
E-mail: ivaniuk@bsuir.by

УДК 681.32
https://doi.org/10.37661/1816-0301-2023-20-4-24-37

Оригинальная статья
Original Paper

Моделирование дискретных управляющих систем с параллелизмом поведения

Д. И. Черемисинов, Л. Д. Черемисинова✉

Объединенный институт проблем информатики
Национальной академии наук Беларуси,
ул. Сурганова, 6, Минск, 220012, Беларусь
✉*E-mail: cld@newman.bas-net.by*

Аннотация

Цели. Рассматривается задача функциональной верификации устройств управления относительно спецификации на их проектирование. При решении задач реализации и тестирования дискретных систем приходится иметь дело с наличием параллелизма в поведении взаимодействующих объектов управления, что отображается также и в задании на проектирование устройств управления ими. Цель исследования заключается в разработке метода имитационного моделирования описаний дискретных систем, который позволяет динамически тестировать поведение таких систем на области, ограниченной их возможным функционированием.

Методы. В работе рассматривается класс систем управления с параллелизмом происходящих в них процессов, позволяющим линеаризовать их выполнение. Для задания спецификации таких систем управления предлагается использовать язык ПРАЛУ параллельных алгоритмов управления, в основе которого лежат сети Петри и который позволяет упорядочивать во времени события, происходящие в процессе работы устройства. Предлагается объектно-ориентированный подход к моделированию описания алгоритма управления на уровне транзакций. Для этого разработана модель TLM (Transaction-Level Modeling) описаний на языке ПРАЛУ устройств с параллелизмом поведения. Модель уровня транзакций описывает систему набором взаимодействующих процессов, которые выполняются параллельно и определяют ее поведение во времени.

Результаты. Определены ключевые понятия модели TLM для моделирования описаний алгоритмов управления на языке ПРАЛУ: структура данных, транзакции, процессы и барьерной механизм синхронизации параллельно выполняющихся процессов. Предложен метод преобразования описания алгоритма на языке ПРАЛУ в модель TLM, который основан на представлении операций языка в виде композиций элементарных операций, выполняющихся последовательно. Набор таких операций составляет базис алгоритмического разложения параллельного алгоритма на языке ПРАЛУ в программу на промежуточном языке, которая выполняется строго последовательно. Разработаны трансляторы этой программы на языки Verilog и C, результаты их компиляции представляют симуляторы поведения системы управления.

Заключение. Предложенный метод имитационного моделирования может быть использован при создании испытательного стенда для функциональной верификации схемной реализации устройств управления с параллелизмом поведения. При этом тестовые последовательности для верификации схемной реализации могут генерироваться динамически – в процессе моделирования описания алгоритма на языке ПРАЛУ непосредственно устройства управления или системы, включающей алгоритм управления и алгоритмы поведения управляемых объектов.

Ключевые слова: параллельный алгоритм, устройство управления, имитационное моделирование, функциональная верификация, модель TLM, язык ПРАЛУ

Для цитирования. Черемисинов, Д. И. Моделирование дискретных управляющих систем с параллелизмом поведения / Д. И. Черемисинов, Л. Д. Черемисинова // Информатика. – 2023. − Т. 20, № 4. – С. 24–37. https://doi.org/10.37661/1816-0301-2023-20-4-24-37

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 17.08.2023
Подписана в печать | Accepted 22.09.2023
Опубликована | Published 29.12.2023

Simulation of discrete control systems with parallelism of behavior

Dmitry I. Cheremisinov, Ljudmila D. Cheremisinova✉

The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, st. Surganova, 6, Minsk, 220012, Belarus
✉*E-mail: cld@newman.bas-net.by*

Abstract

Objectives. The problem of functional verification of control devices with respect to their design specification is considered. When solving the problems of implementing and testing of discrete systems, one has to deal with the presence of parallelism in the behavior of interacting control objects, which is also displayed in the assignment for designing control systems. The aim of the work is to develop a method for simulating descriptions of such systems, which allows their behavior testing dynamically on the area limited by their possible functioning.

Methods. The paper considers a class of control systems with parallelism of the processes occurring in them, which permits linearization of their execution. To specify the behavior of such control systems, it is proposed to use the PRALU language of parallel control algorithms, which is based on Petri nets and which allows to order events occurring during the device operation. An object-oriented approach to simulation of the description of the control algorithm at the transaction level is proposed. For this purpose, a TLM (Transaction-Level Modeling) model has been developed for describing the devices with behavior parallelism in PRALU language. The transaction level model describes a system as a set of interacting processes that run in parallel and determine the behavior of the system over time.

Results. The key concepts of the TLM model for simulating the descriptions of control algorithms in the PRALU language are defined: data structure, transactions, processes, and a barrier mechanism for synchronization of parallel processes. A method is proposed for transforming the description of an algorithm in the language into a TLM model, which is based on the representation of language operations as compositions of elementary operations that are performed sequentially. The set of these operations forms the basis for the algorithmic decomposition of a parallel algorithm in PRALU language into intermediate language program that is executed strictly sequentially. Translators of this program into the Verilog and C languages have been developed, the results of their compilation are simulators of the behavior of control system.

Conclusion. The proposed simulation method can be used to create a test bench for functional verification of the circuit implementation of control devices with behavior parallelism. In this case, test sequences for verifying the circuit implementation can be generated dynamically – in the process of simulating the description of the algorithm in the PRALU language directly the control device or system, which include the control algorithm and the algorithms of controlled objects behavior.

Keywords: concurrent algorithm, control device, simulation, functional verification, TLM model, PRALU language

For citation. Cheremisinov D. I., Cheremisinova L. D. *Simulation of discrete control systems with parallelism of behavior*. Informatika [*Informatics*], 2023, vol. 20, no. 4, pp. 24−37 (In Russ.). https://doi.org/10.37661/1816-0301-2023-20-4-24-37

Conflict of interest. The authors declare of no conflict of interest.

Введение. Рост сложности и количества требований к функционалу проектируемых в настоящее время управляющих систем увеличил трудоемкость не только их проектирования, но и верификации. Время, требуемое для верификации, растет даже значительно быстрее, чем функциональная сложность проектов. Тестирование становится одной из наиболее важных и необходимых стадий проверки схемных реализаций или программного обеспечения. По разным оценкам, в настоящее время на этап тестирования и отладки приходится до 60–80 % общих затрат на разработку управляющих систем [1], в отдельных случаях на его долю приходится до 90 % затрат.

На этапе функциональной верификации устанавливается, реализует ли спроектированное устройство желаемое поведение, т. е. работает ли оно согласно установленным в его спецификации требованиям. Самым распространенным подходом к верификации в настоящее время является имитационное моделирование (simulation-based verification), для проведения которого необходима тестовая система. Эта система представляет собой специализированную программную среду, которая решает три основные задачи: генерацию тестовой последовательности, проверку правильности поведения модели тестируемого компонента и оценку полноты тестирования. Тест представляет собой последовательность наборов значений сигналов, подаваемых на вход тестируемой схемы, и наборов значений сигналов, ожидаемых на выходе [2]. Проверка правильности поведения тестируемой схемы выполняется в процессе ее моделирования на тестовой последовательности и сравнения полученных результатов с эталонными, заданными исходной спецификацией.

Качество тестирования напрямую зависит от используемых тестовых последовательностей. В основе традиционно применяемых в практике тестирования методов построения тестовых последовательностей лежит их ручная разработка, случайная и ограниченно случайная (constrained-random) генерация тестов. Хотя, несмотря на простоту, случайные тесты и позволяют быстро обнаруживать значительное число ошибок в проекте, однако при этом нельзя определить, насколько полно они покрывают область работы тестируемого устройства. Более выигрышной представляется [3] стратегия тестирования с использованием знаний о функциональности проектируемого устройства, которая предполагает направленную генерацию тестов (directed tests approach).

Существующие в настоящее время тестовые системы, предназначенные для автоматизации функционального тестирования, позволяют формализовать описание тестируемых устройств, генерировать тестовые последовательности и оценивать правильность поведения аппаратной реализации устройств в процессе их моделирования. Вместе с тем генерируемые тесты не привязаны жестко к этой реализации. В настоящее время стандартом для разработки тестовых систем является универсальная методология верификации UVM (Universal Verification Methodology) [4], которая представляет собой библиотеку языка программирования SystemVerilog с открытым исходным кодом, позволяющую создавать гибкие компоненты тестовых систем. UVM ориентирована на верификацию систем трансформационного типа [5], реализующих классические алгоритмы, которые по заданным исходным данным вычисляют результат за конечную последовательность шагов.

Наиболее развитыми подходами и средствами разработки функциональных тестов в настоящее время являются технологии AVM (Advanced Verification Methodology) [6] компании Mentor Graphics и OVM (Open Verification Methodology, URL: https://verificationacademy.com/archive/ovm-white-paper) компаний Mentor Graphics и Cadence Design Systems на основе языков SystemVerilog или SystemC. При разработке тестовых систем в рамках этих технологий предполагается: объектно-ориентированный подход, моделирование на уровне транзакций (TLM, Transaction-Level Modeling) [7] и направленная генерация псевдослучайных стимулов (constraint-random generation – генерация на основе ограничений). Можно упомянуть также академическую разработку – технологию UniTESK (Unified TEsting and Specification tool Kit) [8] Института системного программирования РАН. В этой работе основной упор делается на автоматизацию процесса генерации тестовых последовательностей на основе автоматных моделей устройств управления. Тестовые воздействия находятся в процессе обхода графа состояний конечного автомата, моделирующего тестируемую систему.

В настоящей работе рассматривается задача верификации систем реактивного типа [5], особенность которых (в отличие от систем трансформационного типа) заключается в непрерывном (и в общем случае бесконечном) обмене сигналами с внешней средой в процессе их функционирования. Наиболее известной моделью реактивных систем является конечный автомат [9, 10], который широко используется для описания протоколов. Однако наряду с традиционно организованными системами, реализующими чисто последовательное поведение, существует ряд систем, в которых выразительных средств аппарата конечных автоматов оказывается недостаточно. Важнейшим свойством таких систем является присущий им параллелизм происходящих в них процессов.

Проблема верификации на основе моделей для устройств с параллелизмом поведения еще недостаточно изучена. Наиболее известными моделями такого типа являются сеть конечных автоматов, система помеченных переходов (labelled transition systems, LTS) [11] и цветные сети Петри [12]. Модели на основе сетей Петри позволяют описать системы с «истинным параллелизмом», в которых некоторые из процессов, происходящих в одном и том же компоненте системы, могут выполняться параллельно и независимо друг от друга.

Большинство известных подходов к генерации тестовых последовательностей для систем, поведение которых задано на языках, в основе которых лежит аппарат сетей Петри, основаны на построении графа достижимых состояний. Тестовые наборы генерируются по этому графу достижимости путем его обхода [13, 14] аналогично тому, как это делается для случая конечных автоматов [15]. Недостатком такого метода является экспоненциальный рост размера пространства возможных состояний системы. В результате граф достижимости сталкивается с проблемой взрыва числа состояний, что негативно влияет на производительность методов тестирования сложных систем.

В настоящей работе рассматривается задача построения тестовой системы для верификации схемной (или программной) реализации устройства управления с параллелизмом поведения. Спецификация проектируемого устройства представляется на языке ПРАЛУ описания параллельных алгоритмов управления [5] и задает систему реактивного типа. Параллельный алгоритм на ПРАЛУ обладает свойством линеаризуемости [16]: результат параллельного выполнения операций алгоритма эквивалентен некоторому последовательному их выполнению. На языке ПРАЛУ можно также описать и поведение объекта, управляемого проектируемым устройством. В этом случае объект управления будет рассматриваться как часть тестового окружения. Тестовые последовательности формируются на основе описанных алгоритмов поведения устройства и объекта управления динамически – в процессе их функционирования. В работе показано, как при разработке тестовой системы для моделей устройств такого типа реализовать объектно-ориентированный подход к моделированию и симуляции на уровне транзакций. Данный подход широко используется в современных средствах моделирования цифровых устройств, так как он позволяет отделить процесс отладки системы и генерации тестовых последовательностей от ее реализации. Предложена модель TLM описаний на языке ПРАЛУ устройств с параллелизмом происходящих в них процессов. Результатом работы системы верификации устройства управления являются тест и программа, моделирующая поведение на этом тесте. Верификация реализации системы состоит в сравнении поведения, отображенного в результате моделирования, и реального эталонного поведения системы, заданного спецификацией на проектирование.

1. Язык описания алгоритма функционирования устройств с параллелизмом поведения. Проблема проектирования устройств управления является одной из важнейших при автоматизации производственных процессов в различных отраслях промышленности. При решении задачи реализации устройств управления приходится иметь дело с параллелизмом, присутствующим в объектах управления. Управление такими объектами заключается в обеспечении согласованной работы взаимодействующих компонентов, работающих параллельно и асинхронно. Параллелизм, присутствующий в объектах управления, отражается в функциональной модели цифровых устройств, управляющих данными объектами. Для цифровых устройств рассматриваемого класса характерно также и то, что управляющие воздействия и сигналы о состоянии объектов управления описываются булевыми переменными и лишь небольшой процент всей информации является числовым.

Для задания спецификации на проектирование устройств с параллелизмом поведения предлагается использовать язык ПРАЛУ [5] описания простых алгоритмов логического управления. Алгоритмы на этом языке представляются в виде причинно-временных зависимостей между событиями, происходящими в технической системе. Событие представляется в виде конъюнкции двоичных переменных алгоритма. Наступление некоторого события в системе происходит в результате выполнения одной из операций алгоритма. Основными операциями языка ПРАЛУ являются операции ожидания и действия.

Операция ожидания – k^{in} сводится к ожиданию момента времени, когда конъюнкция k^{in} примет значение 1. Операция действия $\rightarrow k^{out}$ выполняется путем присвоения переменным, образующим конъюнкцию k^{out}, значений, обращающих ее в 1. Окончание операции действия приводит к изменению состояния системы в пространстве переменных и является причиной возникновения некоторого нового события. Операции ожидания и действия могут интерпретироваться как опрос состояний датчиков объекта управления и выдача команд на исполнительную и сигнальную аппаратуру.

В одной из интерпретаций операций действия в языке предполагается, что все внутренние (если такие имеются в описании) и выходные (или управляющие) переменные сохраняют свои значения до тех пор, пока их не изменит какая-либо из операций действия.

Для задания порядка выполнения операций в алгоритмах на ПРАЛУ используется механизм переходов, аналогичный сети Петри. Описание перехода описывается цепочкой операций. Она начинается перечнем меток позиций, из которых запускается переход, состоит из последовательности операций языка и заканчивается перечнем меток позиций, в которые происходит переход после выполнения всех операций. Алгоритм управления на ПРАЛУ представляется неупорядоченной совокупностью цепочек вида μ_i: $l_i \rightarrow \nu_i$, где через l_i обозначен некоторый линейный алгоритм, составленный из операций языка; μ_i и ν_i – начальная и конечная метки, которыми служат непустые подмножества множества позиций (задаваемых натуральными числами) $M = \{1, 2, \ldots, n\}$. Порядок выполнения цепочек алгоритма управления в процессе его реализации определяется множеством N запуска [5], его текущие значения $N_t \subset M$. Среди предложений алгоритма выделяется одно – начальное, его метка заносится в множество N_0 перед началом реализации алгоритма.

В процессе реализации алгоритма управления цепочки запускаются независимо друг от друга. Если в некоторый момент времени для некоторой цепочки μ_i: $l_i \rightarrow \nu_i$ выполняется условие $\mu_i \subseteq N_t$ и реализуется событие k_i^{in}, с ожидания которого начинается цепочка l_i, то она запускается. После завершения цепочки состояние N_t заменяется на $N_{t+1} = (N_t \setminus \mu_i) \cup \nu_i$. Синтаксически параллельность алгоритма отражается в наличии начальных $|\mu_i| > 1$ меток цепочки (когда несколько процессов сливаются, порождая один процесс) и конечных $|\nu_i| > 1$ меток (когда запускается одновременно несколько независимых процессов). В описании алгоритма могут быть также цепочки μ_i: $l_i \rightarrow \nu_i$ и μ_j: $l_j \rightarrow \nu_j$ с одинаковыми начальными метками $\mu_i = \mu_j$ (случай $\mu_i \cap \mu_j \neq \varnothing$, но $\mu_i \neq \mu_j$ не допускается). Такие цепочки являются связанными и не могут запускаться одновременно: их запуск зависит не только от текущей разметки сети (задаваемой множеством запуска N_t), но и от некоторых внешних событий. По определению связанные цепочки должны начинаться с операций ожидания несовместных событий: соответствующие им конъюнкции k_i^{in} и k_j^{in} должны быть попарно ортогональны.

Алгоритм на ПРАЛУ описывает замкнутую систему, если все события в пространстве переменных вызываются реализацией операций действия самого алгоритма. В случае незамкнутых систем можно выделить подмножество внешних (входных или условных) переменных алгоритма, значения которых задаются внешней средой.

Ниже приведен пример алгоритма управления на языке ПРАЛУ, который описывает незамкнутую систему. Значения входных переменных множества $\{x, y, z\}$, определяющих поведение системы, задаются извне:

1: $-y \rightarrow a\, c - \bar{y} \rightarrow b \rightarrow 2.3$
2: $-x \rightarrow \bar{a}\ \bar{b} \rightarrow 4.5$
3: $-\bar{x}y \rightarrow 6$

3: $-xy \rightarrow \overline{c} - \overline{x} \rightarrow c \rightarrow 3$
4: $- z \rightarrow a \rightarrow 7$
5: $- \overline{x}z \rightarrow b \rightarrow 8$
6.7.8: $\rightarrow \overline{a}\ \overline{b}\ \overline{c} \rightarrow$.

Кроме упомянутых операций ожидания и действия в расширенной версии языка – АЛУ – доступны их расширения, реализующие некоторые операции ограниченной арифметики, которые связаны со счетом событий и временными задержками. Для обеспечения реакции устройства на особые события введены также операции гашения, которые при реализации описания алгоритма управления дают возможность досрочно завершить некоторые или все параллельно выполняющиеся цепочки.

Язык алгоритмов управления обеспечивает возможность иерархического описания поведения системы, которое отражает структуру многокомпонентной системы и организацию взаимодействия отдельных ее частей. В частности, на этом же языке можно описать и поведение объекта, управляемого проектируемым устройством. Такое описание может рассматриваться как часть тестового окружения в процессе моделирования алгоритма функционирования устройства управления. Совместное описание устройства и объекта управления позволяет увеличить замкнутость моделируемой системы, сокращая число переменных, значения которых должны определяться извне.

2. Моделирование цифровых систем на уровне транзакций. В современных системах верификации цифровая система представляется в виде компонентов, поведение которых задается как набор параллельных взаимодействующих процессов. Взаимодействие процессов представляется как коммуникация, в которой актами коммуникации служат транзакции через общие структуры данных. Моделирование на уровне транзакций (TLM) представляет собой подход к моделированию обмена данными в цифровых системах, при котором организация связи между функциональными компонентами системы отделена от их реализации.

На уровне транзакций упор в большей степени делается на функциональность обмена данными между процессами и в меньшей – на их фактическую реализацию. Все выполняемые при этом операции изменения состояний, передачи данных и вычислений являются транзакциями. Каждая транзакция описывает преобразование данных, общих для взаимодействующих процессов. Транзакции создаются динамически в процессе моделирования в соответствии с заданными условиями. Произведенные ими изменения состояний данных становятся видимыми для остальных процессов после их завершения.

Модель уровня транзакций описывает компоненту системы набором процессов, которые определяют ее поведение и взаимодействуют одновременно. На уровне описания аппаратуры при моделировании для синхронизации процессов, как правило, используется потактовая смена значений сигналов системы. На уровне TLM генератор синхросигналов не применяется, и при моделировании процессы синхронизируются в моменты обмена сигналами между процессами, т. е. после выполнения транзакций.

Модели TLM используются в современных средствах верификации проектов аппаратуры. Такой подход позволяет отделить процесс отладки системы и генерации тестовых последовательностей от ее аппаратной реализации. Наиболее популярным языком моделирования уровня TLM является SystemC (стандарт IEEE 1666), который представляет собой расширение языка C++. Исполняемая программа, получаемая в результате компиляции модели на языке SystemC, реализует симулятор с интегрированными средствами управления имитацией. Параллелизм в SystemC имеет семантику чередования операций последовательных процессов. Параллельные процессы языка SystemC вводятся как потоки (threads), которые планируются для последовательного выполнения планировщиком SystemC на основе невытесняющего (non-preemptive) мультипрограммирования. Потоками являются последовательные процессы, имеющие общую память.

Модель многопоточности в SystemC обладает существенным недетерминизмом, и процессы языка SystemC имеют специальную организацию для устранения этого недетерминизма. Атомарные операции процессов, задаваемые транзакциями, линеаризуются в процессе их выполне-

ния таким образом, чтобы обеспечивать детерминизм результата выполнения операций над общими данными несколькими параллельными процессами. Линеаризуемость операций можно понимать и в том смысле, что результат параллельного выполнения любых операций эквивалентен их некоторому последовательному выполнению [16]. Выполнение линеаризуемой операции с точки зрения любого другого потока является мгновенным: операция либо не начата, либо завершена.

Синхронизация процессов модели TLM на уровне транзакций осуществляется барьерным механизмом [17]. Барьерами являются точки исходного кода, в которых каждый процесс должен приостановиться и подождать достижения барьера всеми параллельными процессами выполняемой группы потока. В модели TLM на языке SystemC точки барьера задаются вызовами функции wait (.). Изменения значений, запланированные в процессах группы, в общей структуре данных происходят мгновенно после достижения барьера всеми процессами.

В работе [5] было предложено характеризовать цифровые устройства по типу алгоритмического описания. Устройства, моделью которых являются классические алгоритмы (алгоритмы планирования), относятся к трансформационному типу. Их цель – вычисление некоторого результата по исходным данным посредством конечной последовательности шагов. Примерами таких систем являются процессоры, компиляторы языков программирования, веб-серверы. Модель TLM предложена для цифровых устройств трансформационного типа.

Функционирование реактивных систем заключается в непрерывном взаимодействии с окружающей средой: опросом и изменением ее состояния. Поведение реактивной системы задается алгоритмом управления (протоколом взаимодействия). Примерами таких устройств являются контроллеры периферийных устройств компьютера, подключаемые к общей шине, встроенные системы, устройства управления оборудованием. В последнее время термин «реактивная система» стал употребляться и для обозначения программных систем, в которых асинхронно обрабатываются потоки данных, объем которых не предопределен [18].

Алгоритмы управления реактивными системами не являются алгоритмами в смысле классической теории алгоритмов, тем не менее показано, что для формализации и моделирования таких алгоритмов также можно использовать подход TLM.

3. Модель TLM реактивных систем с параллелизмом поведения на языке ПРАЛУ. Алгоритмы на языке ПРАЛУ допускают интерпретацию моделью TLM. Для построения модели TLM алгоритма на языке ПРАЛУ требуется уточнить семантику операций ожидания и действия, определить понятие операции, транзакции, потока, барьера, а также уточнить определение параллельности выполнения операций. В работе используется модель параллелизма операций типа «чередование», в которой одновременность понимается как возможность упорядочивать операции произвольным образом. Суть принятого уточнения порядка выполнения операций состоит в доопределении частичного порядка на множестве операций, задаваемого исходным параллельным алгоритмом, до линейного порядка. В такой интерпретации алгоритм на ПРАЛУ обладает свойством линеаризуемости, т. е. результат параллельного выполнения операций алгоритма эквивалентен некоторому последовательному выполнению атомарных операций.

Ключевые моменты предлагаемой модели TLM описания алгоритма управления на языке ПРАЛУ состоят в следующем:

1. Структурой данных является вектор значений переменных алгоритма: внешних – входных (условных) и внутренних – выходных. Каждая компонента вектора соответствует одной из переменных и задает пару значений этой переменной: текущее и планируемое. Такое разделение значений одной и той же переменной вызвано необходимостью исключить в процессе вычислений изменение значений переменных до достижения барьера. Доступ к компонентам вектора данных осуществляется посредством операции установки значений переменных алгоритма, определяющей планируемые значения, и операции проверки значений условных переменных, читающей их текущие значения.

2. Транзакции представляются операциями ожидания и действия алгоритма, которые описывают события взаимодействия [19]. Эти операции рассматриваются в виде композиций некоторых элементарных операций, выполняемых последовательно. Реализация операции ожидания – k^{in} состоит в выполнении операции приостановки процесса и проверки текущих значений

переменных, указанных в конъюнкции k^{in}. Реализация операции действия $\rightarrow k^{out}$ заключается в установке планируемых значений переменных, указанных в конъюнкции k^{out}.

3. Цепочки операций языка ПРАЛУ интерпретируются как процессы модели TLM. Соответственно, последовательный процесс представляет собой совокупность последовательно выполняемых операций ожидания и действия алгоритма. Вводится понятие ветви, которая задает последовательный процесс, начинающийся с некоторой операции. Ветвь является динамическим объектом, который порождается операцией образования и уничтожается операцией ее прекращения. После образования ветвь может выполняться, а в процессе выполнения – приостанавливаться. Образование ветви заключается в занесении первой ее операции в очередь ветвей, готовых для выполнения (ОГ). Операция прекращения ветви состоит в удалении ее из ОГ. Операция приостановки ветви является аналогом функции wait(.) в языке SystemC, при ее реализации ветвь переносится из ОГ в очередь ветвей, ждущих выполнения (ОЖ). Введенные приведенным способом ветви аналогичны потокам в модели TLM на языке SystemC.

4. При моделировании алгоритма управления из ОГ последовательно извлекаются ветви и выполняются до приостановки. Выполнение ветви *G* приостанавливается, когда ее начальный фрагмент – k^{in} не может выполниться на множестве текущих значений переменных и тогда ветвь переносится в ОЖ или же он выполнен и тогда в ОЖ вносится ветвь, начинающаяся с операции, которая должна выполняться в ветви *G* следующей.

5. Синхронизация параллельно выполняемых ветвей осуществляется с помощью барьерного механизма. Достижение барьера фиксирует такты моделирования и изменения значений переменных. Точки барьера задаются операциями приостановки выполнения всех ветвей. Структура данных барьера синхронизации представлена в памяти очередями ветвей ОГ и ОЖ. Барьер достигнут, когда ОГ становится пустой. При достижении барьера: элементы из ОЖ переносятся в ОГ (ОЖ становится пустой); вводятся (если система незамкнута) новые значения входных (внешних) переменных в качестве планируемых; накопленные планируемые значения пересылаются в текущие для каждой компоненты вектора переменных; выгружаются значения выходных переменных и запускается выполнение первой ветви из ОГ.

Барьерная синхронизация считается довольно затратной в смысле памяти и времени выполнения. Синхронизация процессов занимает больше времени, чем вычисления, особенно в распределенных вычислениях. В модели TLM на языке ПРАЛУ точки барьера в явном виде не указываются (в отличие от модели TLM на языке SystemC), барьер формируется автоматически в процессе моделирования. Операции образования, прекращения и приостановки ветвей относятся к накладным расходам на организацию вычислений. Однако в модели TLM описаний на языке ПРАЛУ каждая из операций барьерного механизма может быть реализована в современных процессорах одной командой, в том числе и наиболее сложная из них – приостановка, которая является аналогом функции wait(.) в SystemC. Реализовать отдельный планировщик не требуется.

4. Реализация симулятора моделей TLM на языке ПРАЛУ. Преобразование описания алгоритма на языке ПРАЛУ в модель TLM осуществляется путем представления операций языка в виде композиций элементарных операций, которые выполняются строго последовательно. Набор этих операций (табл. 1) составляет базис алгоритмического разложения параллельных алгоритмов на языке ПРАЛУ.

Начальными состояниями ветвей TLM модели являются цепочки алгоритма на языке ПРАЛУ. Связанные цепочки (с одинаковыми метками) порождают разные ветви, и операция их запуска заменяется операцией запуска соответствующих ветвей. Например, в приведенном в разд. 1 алгоритме две связанные цепочки (третья и четвертая) с меткой $\mu = 3$ порождают две связанные ветви H3 и H4, и операция перехода $\rightarrow 3$ заменяется запуском этих двух ветвей. Однако выполняться в текущем такте может только одна из ветвей. Для правильного запуска группы связанных ветвей вводится переменная маски *w*. Значение $w = 1$ фиксирует факт выполнения условия запуска (наступление ожидаемого события) для одной из ветвей. Значение этой переменной опрашивается перед выполнением каждой из ветвей с тем, чтобы исключить выполнение остальных цепочек, если $w = 1$ (используется операция «Прекращение процесса по условию»).

Таблица 1
Базовые элементарные операции
Table 1
Basic elementary operations

Операция *Operation*	Функция *Function*	Описание *Description*
П	Приостановка ветви	Текущая ветвь переносится из ОГ в ОЖ, начиная с операции, следующей за «П», и выбирается следующая ветвь из ОГ
?	Переход и запуск ветвей ?(H1, H2,…)	Ветви H1, H2,… вносятся в ОГ
A	Прекращение процесса по условию Aw	Прекращение реализации ветви, если $w = 1$
К	Остановка алгоритма	Конец работы алгоритма
S	Установка выходных переменных Sk^{out}	Устанавливаются планируемые значения переменных (в соответствии с конъюнкцией k^{out})
O	Проверка условных переменных Ok^{in}	Проверяется выполнение условия $k^{in} = 1$ на множестве текущих значений переменных
P	Установка переменной синхронизации Pr	Устанавливается единичное значение переменной синхронизации r
R	Сброс переменной синхронизации Rr	Устанавливается нулевое значение переменной синхронизации r
M	Установка маски Mw	Устанавливается единичное значение маски w для связанных цепочек
C	Сброс маски Cw	Устанавливается нулевое значение маски w для связанных цепочек

Аналогично для синхронизации слияния k параллельных цепочек алгоритма на ПРАЛУ используются k переменных синхронизации. Значения переменных синхронизации опрашиваются перед выполнением ветви, соответствующей слиянию этих k цепочек. Например, последняя цепочка алгоритма начинается меткой $\mu = 6.7.8$ и может выполняться, когда множество запуска $N_t \supseteq \{6, 7, 8\}$. Для фиксации данного факта вводятся три переменные синхронизации r_1, r_2 и r_3, которым присваивается значение 1 соответственно при переходах $\rightarrow 6$, $\rightarrow 7$ и $\rightarrow 8$. При запуске ветви H7, соответствующей последней цепочке алгоритма, производится опрос значений этих переменных посредством использования операции ожидания события $r_1 r_2 r_3 = 1$.

Операции действия $\rightarrow y_1 y_2 \ldots y_l$ соответствует команда установки значений (планируемых) переменных промежуточного языка: S$y_1, y_2, \ldots, y_l$. Операция ожидания – $x_1 x_2 \ldots x_l$ транслируется во фрагмент, состоящий из двух операций приостановки и проверки значений (текущих) переменных: П O$x_1, x_2, \ldots, x_l$. Операции ожидания – $x_1 x_2 \ldots x_l$, открывающей одну из связанных цепочек, соответствует фрагмент П Aw : O$x_1, x_2, \ldots, x_l$: Mw. Реализация операции перехода $\rightarrow p.q. \ldots r$ состоит в запуске всех ветвей с начальными метками $p, q, \ldots, r$.

Приведенный выше пример описания алгоритма управления на языке ПРАЛУ переводится в следующее описание на промежуточном языке:

H1 ПOy : Sa,c : ПО $\bar{y}$: Sb : ?2,3,4
H2 ПOx : S $\bar{a}$, $\bar{b}$: ?5,6
H3 ПAw_1 : Oy, $\bar{x}$: Mw_1 Cw_2 : Pr_1 : ?7
H4 ПAw_1 : Oy,x : Mw_1 S $\bar{c}$: ПО $\bar{x}$: Sc : ППCw_1 : ?4,3
H5 ПOz : Sa : Cw_2 : Pr_2 : ?7
H6 ПО $\bar{x},z$: Sb : Cw_2 : Pr_3 : ?7
H7 ПAw_2 : Or_1,r_2,r_3 : Mw_2 : Rr_1,r_2,r_3 : S $\bar{a}$, $\bar{b}$, $\bar{c}$: К

5. Моделирование описаний систем управления на уровне модели TLM на языке ПРАЛУ. Симуляторы электронных устройств интегрируют редактор описаний функционирования, механизм моделирования (процессор, представляющий собственно симулятор) и средство экранного отображения формы сигналов. В предлагаемой реализации симулятора описаний управляющих устройств механизм моделирования отделен от средства отображения формы сигнала, что позволяет интегрировать его с разными средствами отображения результатов модели-

рования. Программа процессора симуляции сохраняет результаты проведенного моделирования в специальной базе данных, форма представления которых согласована с формой представления данных средства экранного отображения сигналов. Это позволяет сохранять и анализировать результаты моделирования, а также сопоставлять результаты при использовании разных входных данных.

Основополагающим моментом при моделировании алгоритмов на языке ПРАЛУ является соглашение о длительности выполнения операций языка, в частности это касается операций действия. Данное соглашение существенно влияет на степень соответствия изменений сигналов, производимых симулятором и появляющихся на выходах программной или схемной реализации алгоритма. Реализация (так же, как и моделирование) алгоритмов выполняется в соответствии с некоторым предположением о длительности выполнения операций языка. Наиболее естественно принятое предположение об одинаковой длительности выполнения всех операций (и, в частности, операций действия). Один из путей повышения быстродействия вычислений состоит в предположении о нулевой длительности операций. В этом случае вычисления по одной ветви выполняются до тех пор, пока для их продолжения не потребуется изменение состояний условных переменных. Такое предположение означает, что приостановки выполнения ветвей будут происходить только при выполнении операций ожидания. Для схемной реализации алгоритмов управления более естественно предположение об одинаковой, но не нулевой длительности выполнения операций действия. В этом случае приостановка ветви будет производиться после выполнения операции действия, а не только тогда, когда операция ожидания не может выполниться на множестве текущих значений переменных.

Достижение барьера при моделировании алгоритма фиксирует такты работы устройства, а полученные изменения значений выходных переменных (отмеченные в векторе планируемых значений) соответствуют изменениям значений сигналов на выходах схемной реализации алгоритма управления при подаче на ее входы значений сигналов, соответствующих значениям в векторе текущих значений.

Внешнее поведение процессора, симулирующего алгоритм на языке ПРАЛУ, представляет собой циклы, называемые циклами сканирования. Каждый цикл состоит из ввода новых значений условных переменных – опроса датчиков системы; выполнения программы до достижения барьера и вывода текущих значений выходных переменных – выдачи команд на исполнительные механизмы (рис. 1). Длительность цикла сканирования непостоянна и зависит от выполняемого алгоритма.

Рис. 1. Циклы сканирования при симуляции алгоритма на языке ПРАЛУ

Fig. 1. Scan cycles when simulating the algorithm on the PRALU language

Продемонстрируем процесс моделирования работы приведенного в разд. 1 алгоритма на промежуточном языке. В табл. 2 показаны девять тактов моделирования этого алгоритма. Каждая из подтаблиц соответствует одному такту, порождаемому достижением барьера. Столбцы каждой из подтаблиц задают: выбираемые из ОГ ветви G_j в форме $H_i.O_{il}$ (здесь O_{il} – начальная операция ветви H_i), состояния очередей ОГ и ОЖ готовых и ждущих ветвей, векторы масок, текущих $\mathbf{t_i}$ и планируемых $\mathbf{p_i}$ значений переменных алгоритма. В первой строке каждой подтаблицы показаны состояния $ОГ_i$, $ОЖ_i$, векторов масок, текущих $\mathbf{t_i}$ и планируемых $\mathbf{p_i}$ значений переменных в начале соответствующего такта (после ввода значений внешних переменных). При переходе от i-го такта к (i+1)-му $ОЖ_i \Rightarrow ОГ_{i+1}$, $\mathbf{p_i} \Rightarrow \mathbf{t}_{i+1} = \mathbf{p}_{i+1}$. Вектор масок состоит из двух частей: первая часть задает значения масок w_i, вторая – значения переменных синхронизации r_i.

Векторы $\mathbf{t}_i$ и $\mathbf{p}_i$ отражают изменения всех переменных: сначала входных (условных), затем внутренних (если они есть) и выходных (управляющих). Производимые при выполнении операций изменения значений переменных выделяются жирным шрифтом. Положим, что перед началом моделирования все переменные x, y, z, a, b, c имеют нулевые значения: $\mathbf{t}_1 = \mathbf{p_1} = 0\,0\,0\,0\,0\,0$, переменные масок и синхронизации тоже: 0 0 000. При моделировании использовался следующий тестбенч, задающий изменение внешних (входных) переменных в начале тактов моделирования: **0 0 0**, 0 **1** 0, 0 **0** 0, **1 1** 0, 1 1 **1**, **0** 1 1, 0 1 **0**, 0 **0 1**, 0 0 **0**. Выходные реакции устройства управления отражают изменение значений управляющих переменных в конце каждого такта: 0 0 0, **1** 0 **1**, 1 **1** 1, **0 0 0**, **1** 0 0, 1 **1 1**, 1 1 1, 1 1 1, **0 0 0**.

Таблица 2
Такты моделирования алгоритма управления

Table 2
Cycles of the control algorithm simulation

Номер такта *Measure number*	G_j	Очередь готовых ветвей *The queue of ready branches*	Очередь ждущих ветвей *The queue of waiting branches*	Маски *Masks*	Текущие значения $\mathbf{t}_i$ *Current values* $\mathbf{t}_i$	Планируемые значения $\mathbf{p}_i$ *Planned values* $\mathbf{p}_i$
1		H1	∅	0 0 000	000 000	000 000
	H1	∅	**H1.O*y***			
2		H1.O*y*	∅	0 0 000	0**1** 0 000	0 1 0 000
	H1.O*y*	∅	**H1.O $\bar{y}$**			0 1 0 **1** 0 **1**
3		H1.O $\bar{y}$	∅	0 0 000	0**0** 0 101	000 101
	H1.O $\bar{y}$	**H2, H3, H4**∅	∅			000 1**1** 1
	H2	H3, H4	**H2.O*x***			
	H3	H4	H2.O*x*, **H3.Aw_1**			
	H4	∅	H2.O*x*, H3.Aw_1, **H4.Aw_1**			
4		H2.O*x*, H3.Aw_1, H4.Aw_1	∅	0 0 000	**1 1** 0 111	1 1 0 111
	H2.O*x*	H3.Aw_1, H4.Aw_1, **H5,H6**	∅			1 1 0 **0 0** 1
	H3.Aw_1	H4.Aw_1, H5, H6	**H3.Aw_1**			
	H4.Aw_1	H5,H6	H3.Aw_1, **H4.O $\bar{x}$**	**1** 0 000		1 1 0 0 0 **0**
	H5	H6	H3.Aw_1, H4.O $\bar{x}$, **H5.Oz**			1 1 0 000
	H6	∅	H3.Aw_1, H4.O $\bar{x}$, H5.Oz, **H6.O $\bar{x}z$**			1 1 0 000
5		H3.Aw_1, H4.O $\bar{x}$, H5.Oz, H6.O $\bar{x}z$	∅	1 0 000	1 1**1** 000	1 1 1 000
	H3.Aw_1	H4.O $\bar{x}$, H5.Oz, H6.O $\bar{x}z$	∅			
	H4.O $\bar{x}$	H5.Oz, H6.O $\bar{x}z$	**H4.O $\bar{x}$**			1 1 1 000
	H5.Oz	H6.O $\bar{x}z$	H4.O $\bar{x}$, **H7**	1**0** 0 1 0		1 1 1 **1** 0 0
	H6.O $\bar{x}z$	∅	H4.O $\bar{x}$, H7, **H6.O $\bar{x}z$**			
6		H4.O $\bar{x}$, H7, H6.O $\bar{x}z$	∅	1 0 0 1 0	**0** 1 1 1 0 0	0 1 1 1 0 0
	H4.O $\bar{x}$	H7, H6.O $\bar{x}z$	**H4.П**			0 1 1 1 0 **1**
	H7	H6.O $\bar{x}z$	H4.П, **H7**			
	H6.O $\bar{x}z$	∅	H4.П, H7	1**0** 0 1 1		0 1 1 1 **1** 1
7		H4.П, H7	∅	1 0 0 1 1	0 1**0** 1 1 1	0 1 0 1 1 1
	H4.П	H7,H4, H3	∅	**0** 0 0 1 1		
	H7	H4,H3	∅			
	H4	H3	**H7, H4.O*y,x***			
	H3	∅	H7, H4.O*y,x*	**0 0 1** 1 1		
8		H7,H4. O*y,x*	∅		0**0 1** 1 1 1	0 0 1 1 1 1
	H7	H4.O*y,x*	**H7.Aw_2**			
	H4.O*y,x*	∅	H7.Aw_2			
9		H7.Aw_2	∅	0 0 1 1 1	0 0**0** 1 1 1	0 0 0 1 1 1
	H7.Aw_2	∅	∅	0 0 0 0 0		0 0 0 **0 0 0**
				0 0 0 0 0	**0 0 0 0 0 0**	

Преобразование представления описания алгоритма функционирования устройства на промежуточном языке в программу симулятора выполняется однопроходным текстовым преобразователем, конвертирующим операторы промежуточного языка в вызовы процедур в синтаксисе языка С. Эффективность построенной таким образом программы можно оценить на примере реализации одной из самых трудоемких операций промежуточного языка – операции приостановки. Работа, выполняемая этой операцией, заключается в запоминании адреса следующего вызова процедуры в стеке данных и переходе к выполнению процедуры, адрес которой извлечен из стека возвратов. Компилятор С строит для осуществления этих действий фрагмент кода, состоящий всего из двух машинных команд.

Программа симулятора алгоритмов на ПРАЛУ сохраняет результат моделирования алгоритма в файловом формате VCD (Value Change Dump – дамп изменения значений) [20]. Формат VCD является частью стандарта IEEE для языка описания оборудования на языке Verilog. Историю изменения сигналов, хранящихся в базе в текстовом формате VCD, можно впоследствии просмотреть с помощью инструментов просмотра формы сигналов. Программа просмотра формы сигнала позволяет разработчику СБИС видеть изменение сигналов во времени и анализировать взаимосвязь выходных сигналов модели с входными сигналами. Имеющиеся средства просмотра формы сигналов (например, свободно доступная программа GTKWave, URL: https://gtkwave.sourceforge.net/), входящие в состав промышленных и свободно доступных САПР, позволяют не только отображать графики изменения сигналов, но и изменять масштаб временной последовательности, а также сравнивать и вычислять разницу между значениями сигнала, соответствующими двум точкам, отмеченным курсором на графике.

Заключение. Поведение встроенного устройства управления существенно зависит от объекта, которым оно управляет, и среды, в которой оно работает. Моделирование устройства управления с целью верификации должно производиться на области его запланированного функционирования. С использованием имитационного моделирования верификация схемной реализации устройства управления может быть выполнена двумя путями: динамически в процессе отладки описания алгоритма его функционирования и на тестовой последовательности, полученной в процессе моделирования этого описания. В статье предложены модель TLM для моделирования описаний алгоритмов управления на языке ПРАЛУ, а также метод преобразования описания алгоритма на языке ПРАЛУ в программу на промежуточном языке, которая выполняется строго последовательно. Разработаны трансляторы этой программы на языки Verilog и С, результаты компиляции которых на машинный язык представляют собой программы имитационного моделирования заданной системы управления. Исходными данными для программ имитационного моделирования служат тестовые последовательности.

Вклад авторов. *Л. Д. Черемисинова* предложила модель TLM реактивных систем с параллелизмом поведения, проанализировала полученные результаты и подготовила текст статьи. *Д. И. Черемисинов* разработал и программно реализовал метод преобразования описания алгоритма на языке ПРАЛУ в программу на промежуточном языке.

Список использованных источников

1. Слинкин, Д. И. Анализ современных методов тестирования и верификации проектов сверхбольших интегральных схем / Д. И. Слинкин // Программные продукты и системы. – 2017. – № 3(30). – С. 401–407.

2. Камкин, А. Обзор современных технологий имитационной верификации аппаратуры / А. Камкин, М. Чупилко // Программирование. – 2011. – № 3. – С. 42–49.

3. Kaner, C. What Is a Good Test Case? Software Testing Analysis & Review Conference (STAR) East / C. Kaner. – Mode of access: https://profinit.eu/wp-content/uploads/2016/03/WhatIsGoodTestcase.pdf. – Date of access: 10.02.2023.

4. The Open Source VHDL Verification Methodology. User’s Guide Rev. 1.2 / ed. J. Lewis. – Mode of access: https://www.doulos.com/knowhow/vhdl/the-open-source-vhdl-verification-methodology-osvvm/. – Date of access: 02.09.2023.

5. Закревский, А. Д. Параллельные алгоритмы логического управления / А. Д. Закревский. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1999. – 202 с.

6. Advanced Verification Methodology Cookbook / A. Rose [et al.] ; ed. M. Glasser. – Mentor Graphics Corporation, 2008. – 338 p.

7. Cai, L. Transaction level modeling: an overview / L. Cai, D. Gajski // First IEEE/ACM/IFIP Intern. Conf. on Hardware/ Software Codesign and Systems Synthesis, Newport Beach, CA, USA, 1–3 Oct. 2003. – Newport Beach, 2003. – P. 19–24.

8. Подход UniTesK к разработке тестов / В. В. Кулямин [и др.] // Программирование. – 2003. – № 6(29). – С. 25–43.

9. Lee, D. Principles and methods of testing finite state machine – a survey / D. Lee, M. Yannakakis // Proceedings of the IEEE. – 1996. – Vol. 84, no. 8. – P. 1090–1123.

10. Kanso, B. Compositional testing for FSM-based models / B. Kanso, O. Chebaro // Intern. J. of Software Engineering & Applications (IJSEA). – 2014. – Vol. 5, no. 3. – P. 9–23.

11. Ponce de Leon, H. Model-based testing for concurrent systems with labeled event structures / H. Ponce de Leon, S. H. Delphine Longuet // Software Testing, Verification & Reliability. – 2014. – Vol. 24, no. 7. – P. 558–590.

12. Питерсон, Дж. Теория сетей Петри и моделирование систем : пер. с англ. / Дж. Питерсон. – М. : Мир, 1984. – 264 с.

13. Zhu, H. A methodology of testing high-level Petri nets / H. Zhu, X. D. He // Information and Software Technology. – 2002. – Vol. 44. – P. 473–489.

14. I/O conformance test generation with colored Petri nets / J. Liu [et al.] // Applied Mathematics and Information Sciences. – 2014. – Vol. 8, no. 6. – P. 2695–2704.

15. Бурдонов, И. Б. Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай / И. Б. Бурдонов, А. С. Косачев, В. В. Кулямин // Программирование. – 2003. – № 5. – С. 11–30.

16. Herlihy, M. P. Linearizability: a correctness condition for concurrent objects / M. P. Herlihy, J. M. Wing // ACM Transactions on Programming Languages and Systems. – 1990. – Vol. 12, no. 3. – P. 463–492.

17. Solihin, Y. Fundamentals of Parallel Multicore Architecture / Y. Solihin. – CRC Press, 2015. – 494 p.

18. Halbwachs, N. Synchronous Programming of Reactive Systems / N. Halbwachs. – Springer-Verlag, 2010. – 192 p.

19. Черемисинов, Д. И. Проектирование и анализ параллелизма в процессах и программах / Д. И. Черемисинов. – Минск : Беларуская навука, 2011. – 300 с.

20. Dtrgeron, J. Writing Test Benches & Functional Verification of HDL Models 2gd Edition / J. Dtrgeron. – Springer, 1993. – 508 p.

References

1. Slinkin D. I. *Analysis of modern methods for testing and verification of vlsi projects software products and systems*. Programmnyye produkty i sistemy [*Software Products and Systems*], 2017, no. 3(30), pp. 401–407 (In Russ.).

2. Kamkin A., Chupilko M. *Overview of modern technologies for simulation verification of equipment*. Programmirovaniye [*Programming*], 2011, no. 3, pp. 42–49 (In Russ.).

3. Kaner C. What is a good test case? Software Testing Analysis & Review Conference (STAR) East. Available at: https://profinit.eu/wp-content/uploads/2016/03/WhatIsGoodTestcase.pdf (accessed 10.02.2023).

4. Lewis J. (ed.). *Open Source VHDL Verification Methodology. User's Guide Rev. 1.2*. Available at: https://www.doulos.com/knowhow/vhdl/the-open-source-vhdl-verification-methodology-osvvm/ (accessed 02.09.2023).

5. Zakrevskiy A. D. Parallel'nyye algoritmy logicheskogo upravleniya. *Parallel Logic Control Algorithms*. Minsk, Institut tehnicheskoj kibernetiki Nacional'noj akademii nauk Belarusi, 1999, 202 p. (In Russ.).

6. Rose A., Fitzpatrick T., Rich D., Foster H. *Advanced Verification Methodology Cookbook*. In M. Glasser (ed.). Mentor Graphics Corporation, 2008, 338 p.

7. Cai L., Gajski D. Transaction level modeling: an overview. *First IEEE/ACM/IFIP International Conference on Hardware/ Software Codesign and Systems Synthesis, Newport Beach, CA, USA, 1–3 October 2003*. Newport Beach, 2003, pp. 19–24.

8. Kulyamin V. V., Petrenko A. K., Kosachev A. S., Burdonov I. B. *UniTesK approach to test development*. Programmirovaniye [*Programming*], 2003, no. 6(29), pp. 25–43 (In Russ.).

9. Lee D., Yannakakis M. Principles and methods of testing finite state machine – a survey. *Proceedings of the IEEE*, 1996, vol. 84, no. 8, pp. 1090–1123.

10. Kanso B., Chebaro O. Compositional testing for FSM-based models. *International Journal of Software Engineering & Applications (IJSEA)*, 2014, vol. 5, no. 3, pp. 9–23.

11. Ponce de Leon H. Delphine Longuet S. H. Model-based testing for concurrent systems with labeled event structures. *Software Testing, Verification & Reliability*, 2014, vol. 24, no. 7, pp. 558–590.

12. Peterson J. L. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, Englewood Cliffs, 1981, 290 p.

13. Zhu H., He X. D. A methodology of testing high-level Petri nets. *Information and Software Technology*, 2002, vol. 44, pp. 473–489.

14. Liu J., Ye1 X., Zhou J., Song X. I/O conformance test generation with colored Petri nets. *Applied Mathematics and Information Sciences*, 2014, vol. 8, no. 6, pp. 2695–2704.

15. Burdonov I. B., Kosachev A. S., Kulyamin V. V. *Irredundant algorithms for traversing directed graphs. Deterministic case*. Programmirovaniye [*Programming*], 2003, no. 5, pp. 11–30 (In Russ.).

16. Herlihy M. P., Wing J. M. Linearizability: a correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, 1990, vol. 12, no. 3, pp. 463–492.

17. Solihin Y. *Fundamentals of Parallel Multicore Architecture*. CRC Press, 2015, 494 p.

18. Halbwachs N. *Synchronous Programming of Reactive Systems*. Springer-Verlag, 2010, 192 p.

19. Cheremisinov D. I. Proyektirovaniye i analiz parallelizma v protsessakh i programmakh. *Design and the Analysis of Parallelism in Processes and Programs.* Minsk, Belaruskaja navuka, 2011, 300 p. (In Russ.).

20. Dtrgeron J. *Writing Test Benches & Functional Verification of HDL Models 2gd Edition*. Springer, 1993, 508 p.

Информация об авторах

Черемисинов Дмитрий Иванович, кандидат технических наук, доцент, ведущий научный сотрудник, Объединенный институт проблем информатики Национальной академии наук Беларуси.
E-mail: cher@newman.bas-net.by

Черемисинова Людмила Дмитриевна, доктор технических наук, профессор, главный научный сотрудник, Объединенный институт проблем информатики Национальной академии наук Беларуси.
E-mail: cld@newman.bas-net.by

Information about the authors

Dmitry I. Cheremisinov, Ph. D. (Eng.), Assoc. Prof., Leading Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: cher@newman.bas-net.by

Ljudmila D. Cheremisinova, D. Sc. (Eng.), Prof., Chief Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: cld@newman.bas-net.by

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ
INTELLIGENT SYSTEMS

УДК 004.05; 004.942
https://doi.org/10.37661/1816-0301-2023-20-4-38-47

Оригинальная статья
Original Paper

Метод структурно-параметрической адаптации «умного города» к цифровой экономике

К. А. Радкевич[1]✉, С. В. Кругликов[2]

[1]ОАО «Гипросвязь»,
ул. Сурганова, 24, Минск, 220012, Беларусь
✉E-mail: radkevich@giprosvjaz.by

[2]Объединенный институт проблем информатики
Национальной академии наук Беларуси,
ул. Сурганова, 6, Минск, 220012, Беларусь

Аннотация

Цели. Взаимодействие «умного города» с цифровой экономикой можно раскрыть и проанализировать в рамках применения метода структурно-параметрической адаптации, который позволяет адаптировать параметры модели системы и ее структуру для обеспечения оптимальной работы в условиях изменяющейся внешней среды. Рассмотреть концепцию «умного города» можно исходя из принципов структурно-параметрической адаптации, таких как интероперабельность, децентрализация, виртуализация, возможности работы в режиме реального времени, модульности и ориентации на услуги. В рамках структурно-параметрического анализа «умный город» рассматривается как сложная многоуровневая киберфизическая система.

Методы. Используются методы структурно-параметрической адаптации и структурно-параметрической оптимизации.

Результаты. Разработаны общий вид алгоритма структурно-параметрической адаптации системы и математическая модель задачи структурно-параметрической оптимизации систем «умного города».

Заключение. Для решения задач построения, оптимизации и адаптации структуры системы «умного города» необходимо опираться на технические требования к системе и возможности используемой инфраструктуры. Для этого может также применяться метод структурно-параметрической адаптации.

Ключевые слова: структурно-параметрическая адаптация, принципы структурно-параметрической адаптации, «умный город», киберфизическая система, информационная система, математическая модель

Благодарности. Исследование выполнялось в рамках НИР ГПНИ «Исследование, обоснование и определение путей (направлений) развития создаваемых в Республике Беларусь систем «умного» города (региона)» (№ гос. регистрации 20220061).

Для цитирования. Радкевич, К. А. Метод структурно-параметрической адаптации «умного города» к цифровой экономике / К. А. Радкевич, С. В. Кругликов // Информатика. – 2023. − Т. 20, № 4. – С. 38–47. https://doi.org/10.37661/1816-0301-2023-20-4-38-47

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 11.09.2023
Подписана в печать | Accepted 02.11.2023
Опубликована | Published 29.12.2023

Method of structural-parametric adaptation of "smart city" to digital economy

Kseniya A. Radkevich[1]✉, Sergey V. Kruglikov[2]

[1]JSV "Giprosvjaz",
st. Surganova, 24, Minsk, 220012, Belarus
✉E-mail: radkevich@giprosvjaz.by

[2]The United Institute of Informatics Problems of the National Academy of Sciences of Belarus,
st. Surganova, 6, Minsk, 220012, Belarus

Abstract

Objectives. The interaction between a "smart city" and digital economy can be explored and analyzed within the framework of the structural-parametric adaptation method. This method allows the adaptation of the parameters and structure of the system model to ensure optimal functioning in the conditions of changing external environment. The concept of a "smart city" can be examined based on the principles of structural-parametric adaptation, such as interoperability, decentralization, virtualization, real-time operation, modularity, and service orientation. Within the scope of structural-parametric analysis, a "smart city" is regarded as a complex multi-level cyber-physical system.

Methods. Structural-parametric adaptation methods and structural-parametric optimization methods are employed.

Results. A general form of the algorithm for structural-parametric adaptation of the system and a mathematical model of the problem of structural-parametric optimization of "smart city" systems have been developed.

Conclusion. To address the challenges of constructing, optimizing, and adapting the structure of a "smart city" system, it is necessary to consider the technical requirements of the system and the capabilities of the infrastructure used, and to apply the structural-parametric adaptation method.

Keywords: structural-parametric adaptation, principles of structural-parametric adaptation, "smart city", cyber-physical system, information system, mathematical model

Acknowledgements. The research was conducted within the framework of the scientific research project "Investigation, justification, and determination of ways (directions) for the development of "smart city" systems in the Republic of Belarus" (Registration No. 20220061).

For citation. Radkevich K. A., Kruglikov S. V. *Method of structural-parametric adaptation of "smart city" to digital economy*. Informatika [*Informatics*], 2023, vol. 20, no. 4, pp. 38−47 (In Russ.).
https://doi.org/10.37661/1816-0301-2023-20-4-38-47

Conflict of interest. The authors declare of no conflicts of interest.

Введение. Научный подход к проектированию сложных адаптивных систем, например системы «умного города», предполагает применение научных методов познания. Одним из направлений исследования «умных городов» является структурно-параметрический подход, включающий структурно-параметрический синтез, структурно-параметрическую адаптацию и структурно-параметрическую имплементацию. В процессе проектирования и разработки систем «умного города» можно использовать подход декомпозиции поставленной задачи, разложения и определения структуры с конкретными качественными и количественными показателями, анализа и сравнения имеющейся инфраструктуры и иных компонентов с запросами заказчика, выработки альтернативных решений, адаптации разрабатываемой модели системы к существующим реалиям, а также последующего синтеза системы и ее имплементации во времени и пространстве с постоянной ориентацией на изменяющиеся условия.

Определение принципов структурно-параметрической адаптации «умного города». «Умный город» выступает в роли концепции как встраивающийся в существующую экосистему цифровой экономики и подчиняющийся ее основным тенденциям развития, так и наоборот – как видоизменяющий данную экосистему в ходе своего развития. Анализ динамики городов [1] демонстрирует, что представление о городах как о структурах, сохраняющих постоянное равновесие, является поверхностным, поскольку в большей мере данное состояние отражает лишь физические свойства города. В связи с тем что современная городская среда нуждается в постоянном развитии и активной адаптации к цифровой реальности, возникает необходимость оценки места «умного города» в экосистеме цифровой экономики страны.

В рамках подхода структурно-параметрической адаптации целесообразно рассмотрение «умного города» как киберфизической и мультиагентной системы. Современные города становятся сложной киберфизической системой, в которой вся физическая инфраструктура каждой подсистемы и растущее количество виртуальных структур, встроенных в виртуальный мир, объединены в единую сеть [2].

Принципы структурно-параметрической адаптации «умного города» как ключевого элемента экосистемы цифровой экономики могут быть следующими:

Принцип интероперабельности предполагает способность двух или более сетей, систем, устройств, приложений или компонентов обмениваться и легко использовать информацию безопасно, эффективно и практически не доставляя неудобств пользователю. Это взаимодействие киберфизических и социальных систем должно включать в себя интеллектуальные здания, инженерные коммуникации, интегрированную транспортную систему, уличные элементы и сооружения, больницы, учебные заведения, логистические центры, транспортно-пересадочные узлы, бизнес-центры, торговые объекты, энергоцентры и т. п.

Интероперабельность (совместимость) включает решение проблем, связанных с отсутствием согласованности в существующих стандартах «умного города», механизмов для сравнения и гармонизации инициатив по стандартизации, гармонизации и согласованности между существующими архитектурными решениями.

Сложность технологической экосистемы «умного города» вызвана множеством его компонентов как киберфизической системы. Однако общую структуру работы с данными можно выразить в нескольких категориях: зондирование, управление данными, объединение данных, обработка и визуализация [3].

Принцип виртуализации – это структурно-параметрический принцип, который предполагает использование виртуальных моделей городской инфраструктуры и услуг для улучшения и оптимизации городской жизни. Включает в себя разработку трехмерных моделей городских объектов, которые создаются на основе собранных данных о городской инфраструктуре и услугах, позволяющих создавать и тестировать новые проекты, не вмешиваясь в реальные структуры города. Трехмерные модели дают возможность использовать собранные данные для прогнозирования деятельности и поведения субъектов «умного города».

Принцип децентрализации заключается в том, что управление и контроль за различными системами и процессами в городе распределяются между узлами, которые находятся ближе к месту их использования [4]. Такой подход к управлению обеспечивает более эффективное приме-

нение ресурсов и возможность быстрого реагирования на изменения в окружающей среде. Каждая система может быть управляема с помощью своего собственного узла, который принимает решения на основе данных, полученных от различных источников.

Принцип ориентации на услуги обусловлен наличием в «умном городе» множества субъектов услуг. «Умный город» должен обеспечивать наличие вертикальных и горизонтальных связей между субъектами, а также реализовывать эти связи на всех этапах жизненного цикла продуктов и услуг. Для осуществления принципа ориентации на услуги необходимо создание цифровой инфраструктуры, которая позволяет обеспечивать доступность услуг и взаимодействие граждан с государственными и коммерческими службами в режиме онлайн.

Принцип модульности – один из ключевых принципов структурно-параметрической адаптации «умного города» – позволяет создавать гибкие и масштабируемые системы, состоящие из отдельных модулей, которые способны адаптироваться к различным изменениям в окружающей среде и потребностям городских жителей. Каждый из этих модулей выполняет конкретную функцию и может быть заменен или модифицирован без нарушения работы всей системы.

Принцип надежности определяет модель «умного города» как надежную систему, которая должна обеспечивать безопасность и конфиденциальность данных, а также защиту от возможных угроз безопасности [5]. Так, например, соображения безопасности исходят из того, что технологии связи, используемые для эффективной связи в «умных городах», подвержены различным проблемам безопасности. Конфиденциальность же исходит из множества подключенных устройств в «умных городах», что требует использования надежных инструментов криптографии для защиты устройств и данных от злоумышленников.

Принцип эволюционирования можно рассматривать с точки зрения усложнения цифровых технологий, составляющих «умный город». Концепция «умного города» получила свое начало еще в конце прошлого века с возникновением продуктов таких крупных корпораций, как CISCO и IBM (Generation 1.0), и сегодня перешла к этапу, называемому Generation 5.0: искусственный интеллект, робототехника, интернет вещей, 6G и т. п. [6]. Постоянная эволюция «умного города» в целом и технологий, применяемых в данной концепции, требует наличия гибких подходов к построению «умных городов» и их постоянному развитию.

Принцип устойчивости целесообразно рассматривать в связи с принципом эволюционирования. Устойчивость городов отражает способность городских систем (включая социоэкологические и социотехнические сети) сохранять или быстро возвращаться к исходным функциям, быстро трансформировать системы для текущей или будущей адаптации [7].

Человеческие ресурсы выступают в качестве одного из ключевых компонентов «умного города», так как составляющие элементы «умного города» должны базироваться не только на цифровых решениях и средствах их регулирования и управления, но и на высококвалифицированных кадрах [1]. Данный принцип предполагает также наличие профильных организационных структур, ответственных за развитие «умного города».

Метод структурно-параметрической адаптации «умного города». В связи с большим количеством разработанных информационных систем (ИС) и средств для принятия решений особенностью развития и построения систем «умного города» является возможность применения метода структурно-параметрической адаптации уже созданных моделей систем к целям развития и внедрения для решения конкретных прикладных задач.

Применение метода структурно-параметрической адаптации обусловлено тем, что требования и возможности у различных городов и регионов разные, в связи с чем возникают определенные сложности:

– проблематичность адаптации системы без участия самих разработчиков;

– цели пользователей могут не совпадать с целями разработчиков системы;

– необходимое информационное обеспечение системы может отличаться от того, которым владеет пользователь.

К причинам, побуждающим изменять существующие ИС, в особенности такие, как «умный город», можно отнести изменение требований к идентификации и авторизации, точности опи-

сания отдельных процессов, необходимость в расширении круга решаемых задач как потребность введения новых ресурсов и сервисов. Поэтому в практике внедрения и применения таких систем возникают проблемы модификации и адаптации структуры модели системы, отдельных ее блоков, включения в модель новых модулей, позволяющих реализовывать практические цели пользователя, а также определения необходимой информации и параметров, входящих в модель. Процесс решения данных проблем можно отнести к адаптивной структурно-параметрической процедуре. Адаптивная методология в данном случае понимается как система, способная изменяться (автоматически или полуавтоматически) с целью обеспечения устойчивых показателей регионального развития при изменении внешних условий [8]. Массовое внедрение и использование информационных технологий на основе концепции интернета вещей [9] позволяют современным городам выявлять, интерпретировать и удовлетворять потребности государственного и частного секторов.

Одним из основных этапов проектирования систем «умного города» и их элементов являются исследования по выбору и обоснованию критерия качества (оптимальности) проектных решений. Исходными данными для этих исследований служат результаты целевого, ресурсного и проектного анализов. Критерий качества – это объектный показатель, имеющий обычно количественное выражение, позволяющий определить степень соответствия систем их функциональному (целевому) назначению, т. е. мера достижения поставленной цели. Рассматривать только один критерий при проектировании системы «умного города» недостаточно, так как это приводит к необъективной оценке системы. Иными словами, не учитываются такие важные характеристики системы, как ее быстродействие, удобство и др. Для учета этих характеристик необходимо ввести в задачу структурно-параметрической адаптации дополнительные критерии. При правильном выборе ограничений для второстепенных критериев задача будет решена без необходимости многократного последовательного прохода по всем критериям. Основные этапы структурно-параметрической адаптации системы показаны на рис. 1.

Структурно-параметрический синтез «умного города». Под структурно-параметрическим описанием объекта понимается такое его описание, которое показывает, из каких подсистем, блоков, агрегатов и деталей состоит данный объект, как эти компоненты соединены и взаимодействуют между собой, каковы их весовые, габаритные характеристики и т. п. Структурно-параметрический синтез моделей ИС, в том числе «умного города», и их адаптация к конкретным внешним и внутренним условиям являются главными условиями их использования.

Системы «умного города» содержат компоненты параметрической оптимизации, что позволяет, варьируя параметрами, обеспечивать нахождение характеристик ИС в заданных разработчиком пределах. Структура самой системы при этом задается на этапе проектирования, и в дальнейшем ее изменение связано с большими затратами как времени, так и денежных средств. Выбранная в начале разработки структура к этапу внедрения системы может стать неоптимальной и содержать архитектурные ошибки. Поэтому на этапе проектирования систем «умного города» необходимо предоставить разработчику возможность не только параметрического синтеза, но и структурного. Это в значительной мере усложняет задачу оптимизации, так как нахождение оптимальной структуры не поддается полной автоматизации, отсутствуют алгоритмы нахождения решения за приемлемое время. Тем не менее вычислительные средства позволяют оценить и перебрать большее количество вариантов, что весьма упрощает для разработчика задачу структурного синтеза.

В отличие от задач параметрической оптимизации при структурном синтезе возникает ряд принципиальных трудностей математического характера: дискретный характер проектных решений и зависимостей, отсутствие отработанных методов формализации и решения этих задач.

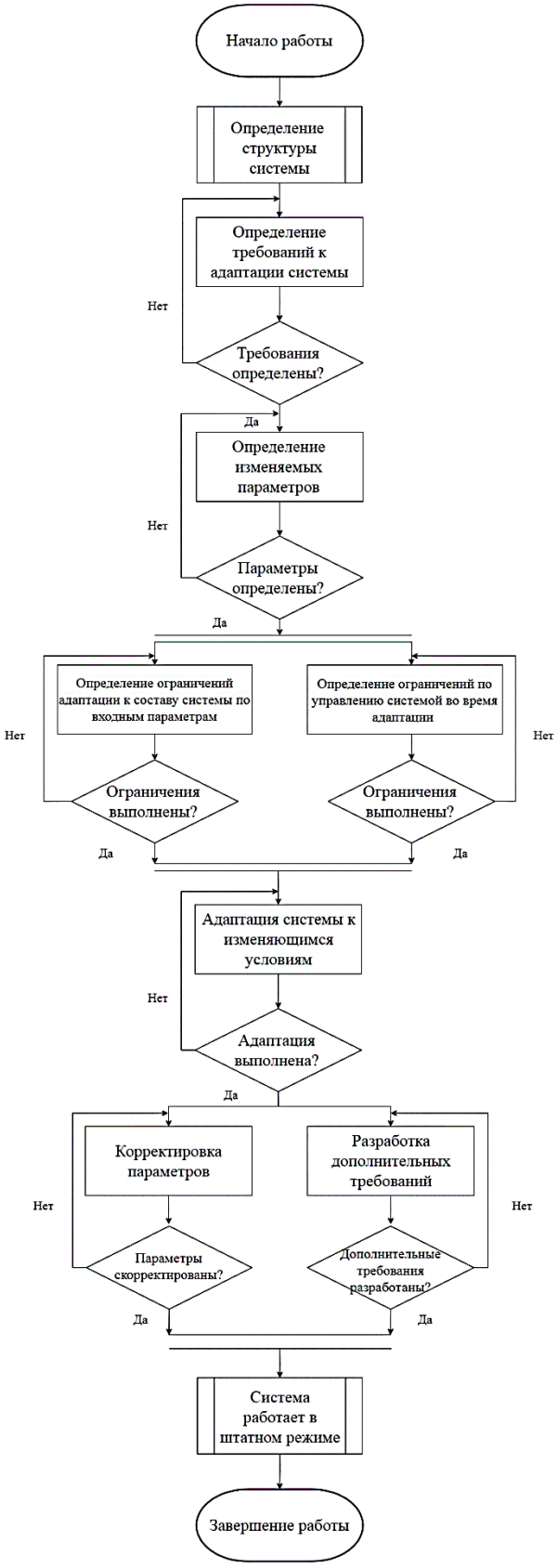


Рис. 1. Общий вид алгоритма структурно-параметрической адаптации системы

Fig. 1. General view of the algorithm for structural-parametric adaptation of the system

Отметим следующие особенности структурно-параметрического синтеза[1]:

– при структурно-параметрической оптимизации варьируются не только параметры, но и структура, что приводит к уникальной целевой функции для каждой структуры либо к добавлению ограничений на множество структур;

– математическая модель учитывает не только параметры объекта, но и структуру и ограничения на нее, т. е. должна быть достаточно универсальной, чтобы описывать не отдельно взятую систему «умного города», а некоторый их класс;

– для решения задачи структурно-параметрического синтеза необходимо разработать алгоритм, основанный либо на аналитическом подходе, либо на численных методах, первый отличается точностью полученного решения, а вторые – универсальностью.

Анализ структурных составляющих архитектуры «умного города» и процессов взаимодействия между ними, а также информационных потоков, сервисов и систем позволил определить необходимые компоненты математической модели системы «умного города», с помощью которой возможно решение задачи структурно-параметрического синтеза. Математическую модель системы «умного города» представим в виде кортежа

$$SCS = \{S, HW, SW, V_s, E_s\}, \quad (1)$$

где *SCS* – модель системы «умного города»;

S – структура системы «умного города»;

HW = {*ST*, *VST*, *QST*, *CST*} – множество аппаратных характеристик системы «умного города», в которые входят данные о множестве хранилищ информации *ST*, их стоимости *VST*, объеме накопителя *QST* и пропускной способности *CST*;

SW = {*SD*, *SM*, *SR*} – множество программных характеристик системы «умного города», включающих все возможные сочетания используемых средств разработки *SD*, структуры модулей системы «умного города» *SM* и конкретной программной реализации *SR*;

V_s – оценка годовых экономических затрат на системы «умного города»;

E_s – оценка производительности системы «умного города».

Под структурой системы «умного города» будем понимать формализованное представление перечисленных выше понятий в виде кортежа с множеством объектов, меняющих свои состояния в результате проведения операций множеством пользователей в некоторые моменты времени под влиянием множества внешних и внутренних воздействий:

$$S = \{U, P, E, O, T, G\}, \quad (2)$$

где $U = \{u_i | i = \overline{1..I}\}$ – множество объектов: цифровые и электронные сервисы, информационные системы и ресурсы, *I* – общее количество объектов;

$P = \{p_q | q = \overline{1..Q}\}$ – множество пользователей, *Q* – общее количество пользователей;

T – множество дискретных моментов времени, представленных натуральными числами;

E – множество воздействий на объекты, как внешних, так и внутренних;

$O = \{o_l | l = \overline{1..L}\}$ – множество операций, выполняемых над объектами, *L* – общее количество операций;

$G = \{G_i(C_i, O) | i = \overline{1..I}\}$ – множество структур информационных потоков, каждый из которых отражает жизненный цикл объекта u_i как последовательную смену состояний через осуществление множества операций *O*.

Структурные соображения играют первостепенную роль как при анализе, так и при синтезе систем самого разного типа. Действительно, наиболее важный этап процесса разработки модели как раз и состоит в выборе структуры модели интересующей нас системы. На основе модели, представленной выше, разработана структурная схема системы «умного города» (рис. 2).

[1]Обухов, А. Д. Структурно-параметрический синтез системы управления электронным документооборотом научно-образовательного учреждения : дис. … канд. техн. наук : 05.13.01 / А. Д. Обухов ; Тамбовский государственный технический университет. – Тамбов, 2016. – 216 с.

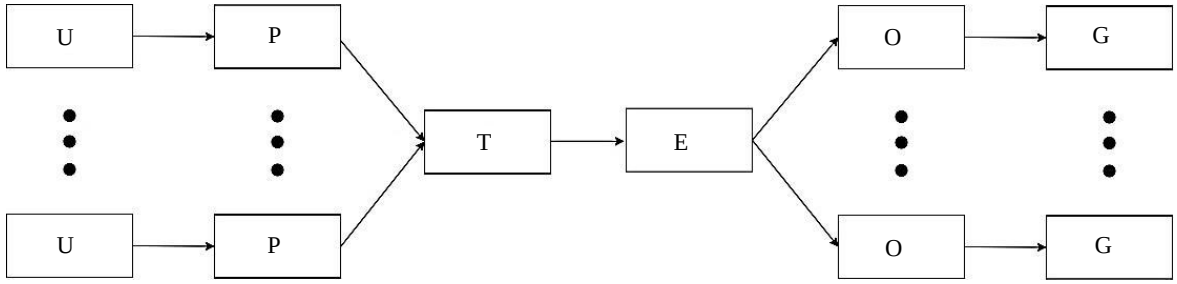


Рис. 2. Структурная схема «умного города»
Fig. 2. Structural diagram of a "smart city"

Рассмотрим элементы описанных выше множеств. Объекту u_i соответствует некоторое множество состояний, каждое из которых обусловливает содержимое объекта в определенный промежуток его жизненного цикла:

$$u_i \to C_i, \tag{3}$$

где $C_i = \{c_{ij} | j = \overline{1..J_i}\}$ – множество состояний объекта u_i, J_i – количество таких состояний. Каждое состояние определяется как кортеж из множества атрибутов объекта и их значений в заданный промежуток времени:

$$c_{ij} = (\{a_{in}, d_{ijn}) | a_{in} \in A_i, d_{ijn} \in D_{ijn}, n = \overline{1..N_i}\}, T_{c_{ij}}), \tag{4}$$

где $A_i = \{a_{in} | n = \overline{1..N_i}\}$ – множество атрибутов объекта u_i с соответствующими им элементами множества значений атрибутов $D_i = \{D_{ij} | j = \overline{1..J_i}\}$ для каждого состояния c_{ij}:

$$D_i = \{d_{ijn} | n = \overline{1..N_i}\}. \tag{5}$$

Здесь N_i – количество атрибутов объекта; $T_{c_{ij}}$ – множество дискретных моментов времени t, в которые существует состояние c_{ij}.

Для того чтобы получить текущее состояние объекта u_i в момент времени t, необходимо использовать функцию состояния

$$\Psi(u_i, t): u_i \to c_{ij}, c_{ij} \in C_i, t \in T_{c_{ij}}. \tag{6}$$

Элемент пользователь p_q в модели системы «умного города» представляет собой совокупность ряда атрибутов, характеризующую его роль:

$$p_q = \{(\{ap_{qm}\}, t) | t \in T_{p_q}\}, \tag{7}$$

где $\{ap_{qm}\}$ – множество атрибутов пользователя, определяющих доступные ему функции, уровень доступа, личные данные и т. д. Множество атрибутов объединено в кортеж с моментами времени, в которые значения атрибутов являются актуальными;

T_{p_q} – множество моментов времени существования пользователя p_q в системе «умного города».

Как было указано выше, воздействия разделяются на внешние EE и внутренние IE: $EE \cup IE$. Внешние воздействия включают новые нормативные правовые и технические акты и стандарты, регламентирующие построение ИС, а также заказы от сторонних организаций и прочие воздействия, осуществляемые извне на систему. Внутренние воздействия формируются на основе внешних прямым способом либо косвенно. Воздействие, вне зависимости от того, является оно внешним или внутренним, направлено на достижение конкретного результата – получе-

ние услуги или реализации сервиса либо некоторого их множества, которые удовлетворяют условиям, заданным воздействием. К таким условиям относятся ограничения на круг лиц, ответственных за получение результата; требования к содержанию сервисов / услуг / систем / ресурсов, затраченному времени и т. д.

Следующий компонент математической модели системы «умного города» – это годовые экономические затраты. Представим их в виде

$$V_s = V_d + V_{hw} + V_{st} + V_p + V_t, \quad (8)$$

где V_s – экономические затраты за год;

V_d – стоимость разработки системы;

V_{hw} – затраты на оборудование и терминалы пользователей;

V_{st} – стоимость хранения информации;

V_p – стоимость обработки информации;

V_t – стоимость доставки информации.

Производительность системы «умного города» напрямую зависит от количества запросов и оказываемых электронных и цифровых услуг, обрабатываемых в единицу времени, с учетом затрат на их обработку. Разработанная система «умного города» должна обеспечивать производительность сервисов / услуг / систем / ресурсов в заданных заказчиком пределах. Исходя из вышеперечисленного, представим производительность системы «умного города» следующим образом:

$$E_S = \frac{W_S}{T_S}, \quad (9)$$

где W_S – объем работы с запросами пользователей;

T_S – затраченное на эту работу время.

Заключение. Системы «умного города» являются сложными адаптивными системами. Они выделяются на фоне других информационных систем своим концептуальным отличием, необходимостью соответствовать множеству заданных параметров и учитывать воздействия внешней среды и изменяющихся условий. Применение метода структурно-параметрической адаптации позволяет адаптировать параметры модели системы и ее структуру для обеспечения оптимальной работы с учетом особенностей основных подсистем, запросов пользователей и требований заказчика. Определение альтернативных вариантов, критериев и требований, оптимизация состава и структуры системы «умного города» как раз и являются задачами структурно-параметрической адаптации. Анализируя совокупность достоинств и недостатков подходов построения систем «умного города», необходимо подчеркнуть, что выбор определенной структуры системы должен опираться на технические требования к системе и возможности используемой инфраструктуры, а применение метода структурно-параметрической адаптации является актуальным при разработке систем «умного города».

Вклад авторов. *К. А. Радкевич* предложила общий вид алгоритма структурно-параметрической адаптации системы. *С. В. Кругликов* сформулировал цель, задачи и методы исследования. Оба автора принимали участие в подготовке текста статьи, анализе и интерпретации результатов исследования.

Список использованных источников

1. Smart cities from the perspective of systems / U. Ammara [et al.] // Systems. – 2022. – Vol. 10, iss. 3. – P. 77. https://doi.org/10.3390/systems10030077

2. Postranecky, M. Smart city near to 4.0 – an adoption of industry 4.0 conceptual model / M. Postranecky, M. Svítek // 2017 Smart City Symposium Prague (SCSP), Prague, Czech Republic, 25–26 May 2017. – Prague, 2017. – P. 1–5. http://dx.doi.org/10.1109/SCSP.2017.7973870

3. A minimum set of common principles for enabling Smart City Interoperability / A. Frascella [et al.] // TECHNE – J. of Technology for Architecture and Environment. – 2018. – Vol. 1. – P. 56–61. https://doi.org/10.13128/Techne-22739

4. Edge-computing-enabled smart cities: A comprehensive survey / L. U. Khan [et al.] // IEEE Internet of Things J. – 2020. – Vol. 7, no. 10. – P. 10200–10232. https://doi.org/10.1109/JIOT.2020.2987070

5. Urban computing for sustainable smart cities: recent advances, taxonomy, and open research challenges / I. A. T. Hashem [et al.] // Sustainability. – 2023. – Vol. 15, iss. 5. – P. 3916. https://doi.org/10.3390/su15053916

6. Kumar, M. Probabilistic data structures in smart city: Survey, applications, challenges, and research directions / M. Kumar, A. Singh // J. of Ambient Intelligence and Smart Environments. – 2022. – Vol. 14, no. 4. – P. 229–284.

7. Do smart cities represent the key to urban resilience? Rethinking urban resilience / S. A. Apostu [et al.] // Intern. J. of Environmental Research and Public Health. – 2022. – Vol. 19, iss. 22. – P. 15410. https://doi.org/10.3390/ijerph192215410

8. Акимова, О. Е. Формирование адаптивной методологии регионального развития в контексте концепции «умный город» / О. Е. Акимова, С. К. Волков, И. М. Кузлаева // Вестник Томского гос. ун-та. Экономика. – 2020. – № 52. – С. 53–64. https://doi.org/10.17223/19988648/52/3

9. Bibri, S. E. The IoT for smart sustainable cities of the future: An analytical framework for sensor-based big data applications for environmental sustainability / S. E. Bibri // Sustainable Cities and Society. – 2018. – Vol. 38. – P. 230–253.

References

1. Ammara U., Rasheed K., Mansoor A., Al-Fuqaha A., Qadir J. Smart cities from the perspective of systems. *Systems*, 2022, vol. 10, iss. 3, p. 77. https://doi.org/10.3390/systems10030077

2. Postranecky M., Svítek M. Smart city near to 4.0 – an adoption of industry 4.0 conceptual model. *2017 Smart City Symposium Prague (SCSP), Prague, Czech Republic, 25–26 May 2017*. Prague, 2017, pp. 1–5. http://dx.doi.org/10.1109/SCSP.2017.7973870

3. Frascella A., Brutti A., Gessa N., De Sabbata P., Novelli C., …, He L. A minimum set of common principles for enabling Smart City Interoperability. *TECHNE – Journal of Technology for Architecture and Environment*, 2018. vol. 1, pp. 56–61. https://doi.org/10.13128/Techne-22739

4. Khan L. U., Yaqoob I., Tran N. H., Kazmi S. M. A., Dang T. N., Hong C. S. Edge-computing-enabled smart cities: A comprehensive survey. *IEEE Internet of Things Journal*, 2020, vol. 7, no. 10, pp. 10200–10232. https://doi.org/10.1109/JIOT.2020.2987070

5. Hashem I. A. T., Usmani R. S. A., Almutairi M. S., Ibrahim A. O., Zakari A., …, Chiroma H. Urban computing for sustainable smart cities: recent advances, taxonomy, and open research challenges. *Sustainability*, 2023, vol. 15, iss. 5, p. 3916. https://doi.org/10.3390/su15053916

6. Kumar M., Singh A. Probabilistic data structures in smart city: Survey, applications, challenges, and research directions. *Journal of Ambient Intelligence and Smart Environments*, 2022, vol. 14, no. 4, pp. 229–284.

7. Apostu S. A., Vasile V., Vasile R., Rosak-Szyrocka J. Do smart cities represent the key to urban resilience? Rethinking urban resilience. *International Journal of Environmental Research and Public Health*, 2022, vol. 19, iss. 22, p. 15410. https://doi.org/10.3390/ijerph192215410

8. Akimova O. E., Volkov S. K., Kuzlaeva I. M. *Formation of an adaptive methodology for regional development in the context of the "smart city" concept*. Vestnik Tomskogo gosudarstvennogo universiteta. Jekonomika [*Bulletin of Tomsk State University. Economics*], 2020, no. 52, pp. 53–64. (In Russ.). https://doi.org/10.17223/19988648/52/3

9. Bibri S. E. The IoT for smart sustainable cities of the future: An analytical framework for sensor-based big data applications for environmental sustainability. *Sustainable Cities and Society*, 2018, vol. 38, pp. 230–253.

Информация об авторах

Радкевич Ксения Анатольевна, исследователь в области технических наук, магистр управления, младший научный сотрудник, ОАО «Гипросвязь».
E-mail: radkevich@giprosvjaz.by

Кругликов Сергей Владимирович, доктор военных наук, кандидат технических наук, доцент, генеральный директор, Объединенный институт проблем информатики НАН Беларуси.
E-mail: kruglikov_s@newman.bas-net.by

Information about the authors

Kseniya A. Radkevich, Researcher in the Field of Technical Sciences, Master of Management, Junior Researcher, JSV "Giprosvjaz".
E-mail: radkevich@giprosvjaz.by

Sergey V. Kruglikov, D. Sc. (Milit.), Ph. D. (Eng.), Assoc. Prof., Director General, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: kruglikov_s@newman.bas-net.by

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ
MATHEMATICAL MODELING

УДК 519.872, 519.248, 519.21
https://doi.org/10.37661/1816-0301-2023-20-4-48-55

Оригинальная статья
Original Paper

Замкнутая сеть Гордона – Ньюэлла с однолинейными полюсами и экспоненциально ограниченным временем ожидания запросов

Ю. В. Малинковский, В. А. Немилостивая✉

Гомельский государственный университет
имени Франциска Скорины,
ул. Кирова, 119, Гомель, 246019, Беларусь
✉*E-mail: vetta.i.am@gmail.com*

Аннотация

Ц е л и . Рассмотрена экспоненциальная сеть массового обслуживания с однолинейными полюсами, отличающаяся от сети Гордона – Ньюэлла только тем, что время ожидания запросами обслуживания в полюсах сети является случайной величиной, условное распределение которой при фиксированном числе запросов в полюсе имеет экспоненциальное распределение. Запросы, обслуженные в полюсах, и запросы, не дождавшиеся обслуживания, движутся по сети в соответствии с разными матрицами маршрутизации. В работе поставлены цели исследовать сеть массового обслуживания, установить достаточные условия ее эргодичности, а также найти стационарное распределение данной сети.
М е т о д ы . Используются методы математического моделирования и аналитическое исследование сетей массового обслуживания.
Р е з у л ь т а т ы . Доказана теорема, обобщающая теорему Гордона – Ньюэлла.
З а к л ю ч е н и е . Возможность варьирования матрицами маршрутизации обслуженных и не дождавшихся обслуживания запросов позволяет учитывать самые разнообразные практические ситуации и снижать необходимым образом нагрузку в узких местах исследуемой сети, что очень важно при проектировании и модернизации информационно-вычислительных сетей.

Ключевые слова: сеть массового обслуживания, ограниченное время ожидания, матрица маршрутизации, уравнения равновесия, эргодичность, стационарное распределение

Для цитирования. Малинковский, Ю. В. Замкнутая сеть Гордона – Ньюэлла с однолинейными полюсами и экспоненциально ограниченным временем ожидания запросов / Ю. В. Малинковский, В. А. Немилостивая // Информатика. – 2023. − Т. 20, № 4. – С. 48–55. https://doi.org/10.37661/1816-0301-2023-20-4-48-55

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 18.09.2023
Подписана в печать | Accepted 17.10.2023
Опубликована | Published 29.12.2023

Closed Gordon – Newell network with single-line poles and exponentially limited request waiting time

Yuri V. Malinkovsky, Violetta A. Nemilostivaya✉

Francisk Skorina Gomel State University,
st. Kirova, 119, Gomel, 246019, Belarus
✉*E-mail: vetta.i.am@gmail.com*

Abstract

Objectives. An exponential queuing network with single-line poles is considered, which differs from the Gordon – Newell network only that the waiting time for service requests at the poles of the network is a random variable with conditional distribution for a fixed number of requests at the pole as an exponential distribution. Requests at poles and requests that did not get the service are moving through the network in accordance with different routing matrices. The objective was to investigate a queuing system and to establish sufficient conditions for its ergodicity, also to find stationary distribution of given network.

Methods. Methods of mathematical modeling and analytical research of queuing networks are used.

Results. A theorem generalizing the Gordon – Newell theorem is proved.

Conclusion. The possibility of varying the routing matrices of served and unserved requests makes it possible to take into account a wide variety of practical situations and reduce the load in the bottlenecks of the network under study. It is very important in the design and modernization of information and computer networks.

Keywords: queuing network, bounded waiting time, routing matrix, balance equations, ergodicity, stationary distribution

Keywords: queuing network, bounded waiting time, routing matrix, balance equations, ergodicity, stationary distribution

For citation. Malinkovsky Yu. V., Nemilostivaya V. A. *Closed Gordon – Newell network with single-line poles and exponentially limited request waiting time*. Informatika [*Informatics*], 2023, vol. 20, no. 4, pp. 48–55 (In Russ.). https://doi.org/10.37661/1816-0301-2023-20-4-48-55

Conflict of interest. The authors declare of no conflict of interest.

Введение. В связи с развитием информационной сферы в последнее время наметилась тенденция к проектированию информационно-вычислительных сетей самого различного назначения. Это выдвигает проблемы, от решения которых зависит эффективность их использования. Одной из них является проблема узких мест в сети, т. е. полюсов сети, в которых относительное использование полюса имеет максимальное значение. При большом числе запросов, циркулирующих в сети, в узком месте очередь неограниченно растет, в то время как в других узлах очереди незначительны либо вообще отсутствуют. Одним из способов преодоления этого недостатка является введение мгновенных обходов запросами полюсов [1, 2], что позволяет выравнивать нагрузку между полюсами. Другой способ уменьшения нагрузки в узких местах – размещение в соответствующих полюсах резервных каналов [3–5]. Еще одним способом выравнивания нагрузки является введенное Е. А. Ковалевым в статье [6] ограничение продолжительности ожидания обслуживания запросов в узлах случайными величинами, имеющими экспоненциальное распределение. Однако в этой работе предполагалось, что запросы, обслуженные в полюсах, и запросы, не дождавшиеся обслуживания, ведут себя одинаково, т. е. их движение по сети определяется одной и той же матрицей маршрутизации. В то же время в практических ситуациях, например, «недовольные клиенты», не дождавшиеся обслуживания, могут вообще покинуть сеть или как-то изменить свою маршрутизацию между полюсами. Если матрица маршрутизации одинакова для обслуженных запросов и запросов, не дождавшихся обслуживания, то уравнения Колмогорова и уравнения глобального равновесия оказываются точно такими же, как и соответствующие уравнения для сети Гордона – Ньюэлла, только с измененными интенсивностями обслуживания, к которым добавляется параметр экспоненциального распределения случайной величины, ограничивающей время ожидания. В этой связи математическое решение задачи определения стационарного распределения тривиально и может быть выполнено чисто автоматически. Случай ограничения на время пребывания и на время

ожидания рассматривался в статье [7]. К сожалению, в случае ограничения на время ожидания при различающихся матрицах маршрутизации была допущена досадная ошибка. По-видимому, приведенные результаты действительно будут выполняться, если заменить уравнение трафика на более сложное уравнение. Для одного случая это утверждение будет доказано в настоящей работе. Сети Геленбе с экспоненциальными ограничениями на время пребывания положительных заявок исследовались в работе [8].

В настоящей статье рассматривается случай различных матриц маршрутизации для обслуженных и не дождавшихся обслуживания запросов для замкнутой сети с однолинейными полюсами. Доказан аналог теоремы Гордона – Ньюэлла [9], в котором используется более сложное уравнение трафика. При этом существенно усложняется матрица маршрутизации по сравнению с матрицей маршрутизации для случая ограничения на время пребывания, которая теперь специальным образом зависит от состояний полюсов.

1. Постановка задачи. Маршрутизация запросов. Сеть массового обслуживания состоит из N однолинейных полюсов (систем) с бесконечным числом мест для ожидания. Время обслуживания запроса единственным прибором i-го полюса имеет экспоненциальное распределение с параметром μ_i $\left(i=\overline{1,N}\right)$. Время ожидания начала обслуживания запроса в очереди i-го полюса является случайной величиной, условное распределение которой, если в i-м полюсе находится n_i запросов, экспоненциальное с параметром $\dfrac{\nu_i}{n_i-1}$ $\left(i=\overline{1,N}\right)$. Другими словами, условная вероятность того, что длительность ожидания начала обслуживания каждого запроса в i-м полюсе закончится в промежутке времени $[t,t+h)$, если в момент t в полюсе находилось n_i запросов, равна $\dfrac{\nu_i}{n_i-1}h+o(h)$ при $h\to 0$, а условная вероятность завершения процесса ожидания хотя бы одного из этих запросов равна $\nu_i h+o(h)$. Если запрос поступает в полюс, свободный от запросов, он сразу начинает обслуживаться. Для определенности будем предполагать, что запросы обслуживаются в порядке поступления в полюсы (дисциплина FCFS). Предполагается, что промежутки времени между моментами поступления запросов, времена обслуживания запросов и времена ожидания запросов в узлах – независимые случайные величины. Запрос, обслуженный в i-м полюсе, мгновенно и независимо от других запросов с вероятностью p_{ij} переходит в j-й полюс $\left(i,j=\overline{1,N},\quad \sum\limits_{j=1}^{N}p_{ij}=1\right)$. Запрос, время ожидания которого в i-м полюсе закончилось, мгновенно и независимо от других запросов с вероятностью r_{ij} направляется в j-й полюс $\left(i,j=\overline{1,N},\sum\limits_{j=1}^{N}r_{ij}=1\right)$. Для упрощения выкладок предполагается, что для всех $i=\overline{1,N}$ выполняется $p_{ii}=r_{ii}=0$.

Введем две стохастические квадратные матрицы $P=\left(p_{ij}\right)$ и $R=\left(r_{ij}\right)$ порядка N $\left(i,j=\overline{1,N}\right)$, которые назовем матрицами маршрутизации соответственно обслуженных и не дождавшихся обслуживания запросов. Очевидно, что матричная функция

$$S\left(\tilde{n}\right)=\left(s_{ij}\left(n_i\right),i,j=\overline{1,N}\right),$$

где $s_{ij}\left(n_i\right)=\dfrac{\mu_i p_{ij}+\nu_i r_{ij}\mathrm{I}_{\{n_i\neq 1\}}}{\mu_i+\nu_i\mathrm{I}_{\{n_i\neq 1\}}}$; I_A – индикатор события A, равный 1, если A происходит, и равный 0 в противном случае, также является стохастической матрицей и по смыслу управляет движением запросов между полюсами $1,2,\ldots,N$ без учета того, за счет чего (окончания об-

служивания или ожидания) запрос покидает полюс. Элементы i-й строки матрицы зависят от числа n_i запросов в i-м полюсе. Действительно,

$$\sum_{j=1}^{N} s_{ij}(n_i) = \frac{1}{\mu_i + \nu_i \mathrm{I}_{\{n_i \neq 1\}}} \left(\mu_i \sum_{j=1}^{N} p_{ij} + \nu_i \mathrm{I}_{\{n_i \neq 1\}} \sum_{j=1}^{N} r_{ij} \right) = \frac{1}{\mu_i + \nu_i \mathrm{I}_{\{n_i \neq 1\}}} \left(\mu_i + \nu_i \mathrm{I}_{\{n_i \neq 1\}} \right) = 1.$$

Будем называть $S(\tilde{n})$ матрицей маршрутизации. Заметим, что в отличие от сети Гордона – Ньюэлла, в которой матрица маршрутизации не зависит от $\mu_i, i = \overline{1,N}$, и в отличие от сети с ограничением на время пребывания, в которой матрица маршрутизации зависит от $\mu_i, \nu_i, i = \overline{1,N}$, и не зависит от чисел заявок в полюсах, матрица маршрутизации исследуемой сети зависит от $\mu_i, \nu_i, \quad i = \overline{1,N}$, а ее i-я строка достаточно простым образом зависит от числа запросов в i-м полюсе.

Обозначим через $\varepsilon_i(n_i)$ условную интенсивность потока запросов, выходящего из i-го полюса, который находится в состоянии $n_i \left(i = \overline{1,N} \right)$. Так как запросы не рождаются и не теряются при прохождении полюсов, в стационарном режиме выполняется следующий закон сохранения:

$$\varepsilon_j(n_j - 1) = \sum_{i=1}^{N} \varepsilon_i(n_i) s_{ij}(n_i - 1), \quad j = \overline{1,N}. \tag{1}$$

Далее предполагается, что матрица $S(\tilde{n})$ неприводима. Для ее неприводимости достаточно (но не необходимо), чтобы неприводимой была хотя бы одна из матриц P или R. Тогда уравнение (1), которое будем называть уравнением трафика, при фиксированных $\mu_i, \nu_i, n_1, n_2, \ldots, n_N$, $i = \overline{1,N}$, будет иметь единственное с точностью до постоянного множителя положительное решение. Если изменять эти параметры, то решения (1) будут функциями от них. Заметим, что в классической сети Гордона – Ньюэлла решение уравнения трафика определяется матрицей P и не зависит от параметров $\mu_i, i = \overline{1,N}$.

2. Уравнения глобального и локального равновесия. Состояния сети в момент t будем описывать марковским процессом или цепью Маркова с непрерывным временем $\tilde{n}(t) = (n_1(t), n_2(t), \ldots, n_N(t))$, где $n_i(t)$ – число заявок в i-м полюсе в момент времени t. Пространство состояний этого процесса представляет собой симплекс $\mathrm{X} = \{\tilde{n} = (n_1, n_2, \ldots, n_N) : n_1 + n_2 + \ldots + n_N = K, \quad n_1, n_2, \ldots, n_N \in Z_+ = \{0, 1, \ldots\}\}$. В силу неприводимости матрицы маршрутизации и положительности интенсивностей выхода из состояний в моменты скачков $\tilde{n}(t)$ – неприводимая цепь Маркова. Так как число ее состояний конечно, то по эргодической теореме Маркова цепь эргодична. Пусть $\{p(\tilde{n}), \tilde{n} \in \mathrm{X}\}$ – ее предельное эргодическое распределение, которое в этом случае будет единственным решением глобальных уравнений равновесия

$$p(\tilde{n}) \sum_{i=1}^{N} \left(\mu_i + \nu_i \mathrm{I}_{\{n_i \neq 1\}} \right) \mathrm{I}_{\{n_i \neq 0\}} = \sum_{i=1}^{N} \sum_{j=1}^{N} p(\tilde{n} + e_j - e_i) \left(\mu_j p_{ji} + \nu_j r_{ji} \mathrm{I}_{\{n_j \neq 0\}} \right), \quad \tilde{n} \in \mathrm{X}, \tag{2}$$

удовлетворяющим условию нормировки $\sum_{\tilde{n} \in \mathrm{X}} p(\tilde{n}) = 1$. Здесь e_i – единичный вектор i-го направления, причем предполагается, что $p(\tilde{n}) = 0$ при $\tilde{n} \notin \mathrm{X}$. Разобьем равенство (2) на урав-

нения локального равновесия, которые получаются, если для каждого полюса приравнять интенсивность потока вероятности из состояния $\tilde{n}$ за счет ухода запросов из этого полюса интенсивности потока вероятности в состояние $\tilde{n}$ за счет поступления запросов в данный полюс из других полюсов:

$$p(\tilde{n})\left(\mu_i + \nu_i \mathrm{I}_{\{n_i \neq 1\}}\right)\mathrm{I}_{\{n_i \neq 0\}} = \sum_{j=1}^{N} p\left(\tilde{n} + e_j - e_i\right)\left(\mu_j p_{ji} + \nu_j r_{ji} \mathrm{I}_{\{n_j \neq 0\}}\right),\ \tilde{n} \in \mathrm{X},\ i = \overline{1, N}. \tag{3}$$

Так как равенство (2) получается суммированием (3) по $i = \overline{1, N}$, то всякое решение локальных уравнений равновесия (3) будет являться решением глобального уравнения равновесия (2).

3. Изолированный полюс в искусственной случайной среде. Изолируем i-й полюс от сети, помещая его в фиктивную случайную среду, отличающуюся от условий функционирования полюса в сети только тем, что в него поступает поток моментов последовательных скачков процесса чистого размножения с интенсивностью рождения $\varepsilon_i(n_i)$, зависящей от числа запросов n_i в этом полюсе. Получается процесс размножения и гибели $\tilde{n}_i(t)$ с интенсивностями рождения $\varepsilon_i(n_i)$ $(n_i = 0,1,\ldots)$ и интенсивностями гибели $\mu_i + \nu_i \mathrm{I}_{\{n_i \neq 1\}}$ $(n_i = 1, 2,\ldots)$. При этом стационарные вероятности цепи Маркова $\tilde{n}_i(t)$, если они существуют, удовлетворяют следующим уравнениям равновесия для вертикальных сечений в графе переходов процесса размножения и гибели:

$$\varepsilon_i(n_i - 1)\, p_i(n_i - 1) = \left(\mu_i + \nu_i \mathrm{I}_{\{n_i \neq 1\}}\right) p_i(n_i) \quad (n_i = 1, 2,\ldots). \tag{4}$$

Система уравнений (4) может быть записана в виде

$$\varepsilon_i(0)\, p_i(0) = \mu_i p_i(1), \tag{5}$$

$$\varepsilon_i(n_i - 1)\, p_i(n_i - 1) = (\mu_i + \nu_i)\, p_i(n_i), \quad (n_i = 2, 3,\ldots). \tag{6}$$

Из уравнений (5) и (6) находим

$$p_i(n_i) = p_i(0) \frac{1}{\mu_i (\mu_i + \nu_i)^{n_i - 1}} \prod_{l=0}^{n_i - 1} \varepsilon_i(l). \tag{7}$$

Из условия нормировки следует равенство

$$p_i(0) = \left(1 + \frac{\mu_i + \nu_i}{\mu_i} \sum_{n_i = 1}^{\infty} \prod_{l=0}^{n_i - 1} \frac{\varepsilon_i(l)}{\mu_i + \nu_i}\right)^{-1} \tag{8}$$

и то, что условие

$$\sum_{n_i}^{\infty} \prod_{l=0}^{n_i - 1} \frac{\varepsilon_i(l)}{\mu_i + \nu_i} < \infty \tag{9}$$

является необходимым и достаточным для существования стационарного распределения, а значит, необходимым для эргодичности процесса размножения и гибели $\tilde{n}_i(t)$. Для доказательства достаточности этого условия для эргодичности остается показать, что при выполнении усло-

вия (9) процесс $\tilde{n}_i(t)$ является регулярным. Из сходимости ряда в условии (9) следует, что его общий член $\prod_{l=0}^{n_i-1}\frac{\varepsilon_i(l)}{\mu_i+\nu_i}$ стремится к нулю при $n_i\to\infty$. Тогда $\prod_{l=1}^{n_i}\frac{\mu_i+\nu_i}{\varepsilon_i(l)}\to\infty$, т. е. ряд

$$\sum_{n_i=1}^{\infty}\prod_{l=1}^{n_i}\frac{\mu_i+\nu_i}{\varepsilon_i(l)}$$

расходится к $+\infty$, что является достаточным условием регулярности процесса размножения и гибели. Итак, для эргодичности процесса $\tilde{n}_i(t)$, описывающего изолированный полюс, необходимо и достаточно выполнения условия (9). При этом эргодическое распределение определяется соотношениями (7), (8) и удовлетворяет равенствам (4).

4. Стационарное распределение процесса состояний сети. Докажем, что вероятности

$$p(n)=\left[C(N,K)\right]^{-1}p_1(n_1)\,p_2(n_2)\ldots p_N(n_N) \tag{10}$$

с множителями, определенными с помощью (7), удовлетворяют уравнениям локального равновесия (3), а значит, и уравнениям глобального равновесия (2). Здесь $C(N,K)$ является нормирующей постоянной. В силу однородности уравнений локального равновесия (3) при подстановке вероятностей (10) в (3) можно не учитывать нормирующий множитель $\left[C(N,K)\right]^{-1}$. Прежде всего отметим, что из уравнений (4) и (10) следует равенство

$$\frac{p\left(n+e_j-e_i\right)}{p(n)}=\frac{p_j\left(n_j+1\right)}{p_j\left(n_j\right)}\frac{p_i\left(n_i-1\right)}{p_i\left(n_i\right)}=\frac{\varepsilon_j\left(n_j\right)}{\varepsilon_i\left(n_i-1\right)}\frac{\mu_i+\nu_i\mathrm{I}_{\{n_i\neq1\}}}{\mu_j+\nu_j\mathrm{I}_{\{n_j\neq0\}}}, \tag{11}$$

если $n_i\neq0$.

Проверим, что произведение (10) удовлетворяет уравнениям (3). Если $n_i=0$, то (3) превращается в тождество $0=0$. Если же $n_i\neq0$, то, разделив (3) на $p(n)$, подставив в полученное равенство уравнения (10), (11) и использовав уравнение трафика (1), запишем

$$\mu_i+\nu_i\mathrm{I}_{\{n_i\neq1\}}=\sum_{j=1}^{N}\frac{p\left(n+e_j-e_i\right)}{p(n)}\left(\mu_j p_{ji}+\nu_j r_{ji}\mathrm{I}_{\{n_j\neq0\}}\right)=\sum_{j=1}^{N}\frac{\varepsilon_j\left(n_j\right)}{\varepsilon_i\left(n_i-1\right)}\frac{\mu_i+\nu_i\mathrm{I}_{\{n_i\neq1\}}}{\mu_j+\nu_j\mathrm{I}_{\{n_j\neq0\}}}\left(\mu_j p_{ji}+\nu_j r_{ji}\mathrm{I}_{\{n_j\neq0\}}\right)=$$

$$=\frac{\mu_i+\nu_i\mathrm{I}_{\{n_i\neq1\}}}{\varepsilon_i\left(n_i-1\right)}\sum_{j=1,\,j\neq1}^{N}\varepsilon_j\left(n_j\right)\frac{\mu_j p_{ji}+\nu_j r_{ji}\mathrm{I}_{\{n_j\neq0\}}}{\mu_j+\nu_j\mathrm{I}_{\{n_j\neq0\}}}=\frac{\mu_i+\nu_i\mathrm{I}_{\{n_i\neq1\}}}{\varepsilon_i\left(n_i-1\right)}\sum_{j=1}^{N}\varepsilon_j\left(n_j\right)s_{ji}\left(n_j-1\right)=\mu_i+\nu_i\mathrm{I}_{\{n_i\neq1\}},$$

т. е. уравнения (3) превращаются в тождество.

По эргодической теореме Маркова цепь Маркова с непрерывным временем и конечным числом состояний эргодична тогда и только тогда, когда она неприводима. Пространство состояний цепи Маркова $n(t)$ конечно в силу выполнения условия $n_1(t)+n_2(t)+\ldots+n_N(t)=K$, а ее неприводимость эквивалентна неприводимости матрицы маршрутизации при каждых фиксированных n_i.

Из условия нормировки $\sum_{n\in X} p(n)=1$ находим нормирующую постоянную

$$C(N,K)=\sum_{n\in Z_+^N:\, n_1+n_2+\ldots+n_N=K} p_1(n_1)\,p_2(n_2)\ldots p_N(n_N). \qquad (12)$$

Теорема. *Если матрица маршрутизации неприводима, то цепь Маркова* $n(t)$ *эргодична, а ее единственное стационарное распределение имеет форму произведения* (10) *с множителями* (7)*, где* $\{\varepsilon_i(n_i), i=\overline{1,N}\}$ *– решение уравнения трафика* (1)*,* $C(N,K)$ *– нормирующая постоянная, определенная с помощью равенства* (12)*.*

В силу однородности функции (10) при использовании теоремы можно не находить $p_i(0)$ по формуле (8), а считать, что $p_i(0)=1$. Таким образом, как и в обычной сети Гордона – Ньюэлла, в стационарном режиме в каждый фиксированный момент времени t состояния полюсов $n_1(t), n_2(t),\ldots,n_N(t)$ зависимы. Это является очевидным следствием линейной зависимости $n_1(t)+n_2(t)+\ldots n_N(t)=K$ координат случайного вектора $n(t)$.

Заключение. В настоящей работе исследовались только замкнутые сети с однолинейными полюсами, что ограничивает возможность применения полученных результатов. Само уравнение трафика, конечно, проще уравнений глобального равновесия для стационарных вероятностей состояний сети. Однако возникают определенные трудности, связанные с тем, что для каждого полюса i, полагая, например, $\varepsilon_i(0)=1$, придется решать систему $K-1$ линейных уравнений трафика. Таким образом, для применения теоремы предварительно необходимо решить $N(K-1)$ систем линейных уравнений. Хорошо, однако, что эти уравнения линейные.

Отметим, что возможность варьирования матрицами маршрутизации обслуженных и не дождавшихся обслуживания запросов позволяет учитывать самые разнообразные практические ситуации и снижать необходимым образом нагрузку в узких местах исследуемой сети, а это очень важно при проектировании и модернизации информационно-вычислительных сетей.

Вклад авторов. *Ю. В. Малинковский* внес существенный вклад в концепцию работы и утвердил окончательный вариант статьи для публикации, *В. А. Немилостивая* провела аналитическое исследование и изложила результаты в тексте статьи.

Список использованных источников

1. Малинковский, Ю. В. Сети массового обслуживания с обходами узлов заявками / Ю. В. Малинковский // Автоматика и телемеханика. – 1991. – № 2. – С. 102–110.

2. Копать, Д. Я. Анализ в нестационарном режиме экспоненциальной G-сети с обходами систем обслуживания положительными заявками / Д. Я. Копать, М. А. Маталыцкий // Вестник Томского гос. ун-та. Управление, вычислительная техника и информатика. – 2020. – № 52. – С. 66–72.

3. Малинковский, Ю. В. Сети массового обслуживания с симметричными резервными каналами / Ю. В. Малинковский // Изв. АН СССР. Техн. кибернетика. – 1986. – № 4 . – С. 69–77.

4. Ковалев, Е. А. Сети массового обслуживания с резервными приборами / Е. А. Ковалев, Ю. В. Малинковский // Автоматика и вычислительная техника. – 1987. – № 2. – С. 64–70.

5. Самочернова, Л. И. Оптимизация системы массового обслуживания с резервным прибором с управлением, зависящим от времени ожидания / Л. И. Самочернова // Известия Томского политехн. ун-та. – 2010. – Т. 316, № 5. – С. 94–97.

6. Ковалев, Е. А. Сети массового обслуживания с ограниченным временем ожидания в очередях / Е. А. Ковалев // Автоматика и вычислительная техника. – 1985. – № 2. – С. 50–55.

7. Malinkovskii, Yu. V. Jackson networks with single-line nodes and limited sojourn or waiting times / Yu. V. Malinkovsky // Automation and Remote Control. – 2015. – Vol. 76, no. 4. – P. 67–79.

8. Malinkovskii, Yu. V. Stationary probability distribution for states of G-networks with constrained sojourn waiting time / Yu. V. Malinkovsky // Automation and Remote Control. – 2017. – Vol. 564, no. 4. – P. 155–167.

9. Gordon, W. J. Closed queueing systems with exponential servers / W. J. Gordon, G. F. Newell // Operations Research. – 1967. – Vol. 15, no. 2. – P. 254–265.

10. Boyarovich, Yu. S Stationary distribution of the closed queuing network with batch transitions of customers / Yu. S. Boyarovich // Automation and Remote Control. – 2009. – Vol. 70, no. 11. – P. 1836–1842.

References

1. Malinkovskii Yu. V. *Queueing network with customer bypass*. Avtomatika i telemehanika [*Automation and Remote Control*], 1991, no. 2, pp. 102–110 (In Russ.).

2. Kopats D. Ya., Matalytski M. A. A*nalysis in non-stationary regime of exponential G-network with bypass of queueing systems positive customers.* Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitelnaja tehnika i informatika [*Tomsk State University Journal of Control and Computer Science*], 2020, no. 52. pp. 66–72 (In Russ.)

3. Malinkovskii Yu. V. *Queueing network with symmetrical backup channels*. Izvestija Akademii nauk Sojuza Sovetskih Socialisticheskih Respublik. Tehnicheskaja kibernetika [*News of the Academy of Sciences of the Union of Soviet Socialist Republics. Technical Cybernetics*], 1986, no. 4, pp. 69–77 (In Russ.).

4. Kovalev E. A., Malinkovskii Yu. V. *Queuing networks with backup devices*. Avtomatika i vychislitel'naja tehnika [*Automatic Control and Computer Sciences*], 1987, no. 2, pp. 64–70 (In Russ.).

5. Samochernova L. I. *Optimization of a queuing system with a backup device with control depending on the waiting time.* Izvestija Tomskogo politekhnicheskogo universiteta [*News of Tomsk Polytechnic University*], 2010, vol. 316, no. 5, pp. 94–97 (In Russ.).

6. Kovalev E. A. *Queuing networks with limited waiting time in queues*. Avtomatika i vychislitel'naja tehnika [*Automatic Control and Computer Sciences*], 1985, no. 2, pp. 50–55 (In Russ.).

7. Malinkovskii Yu. V. Jackson networks with single-line nodes and limited sojourn or waiting times. *Automation and Remote Control*, 2015, vol. 76, no. 4. pp. 67–79.

8. Malinkovskii Yu. V. Stationary probability distribution for states of G-networks with constrained sojourn waiting time. *Automation and Remote Control*, 2017, vol. 564, no. 4, pp. 155–167.

9. Gordon W. J., Newell G. F. Closed queueing systems with exponential servers. *Operations Research*, 1967, vol. 15, no. 2, pp. 254–265.

10. Boyarovich Yu. S. Stationary distribution of the closed queuing network with batch transitions of customers. *Automation and Remote Control*, 2009, vol. 70, no. 11, pp. 1836–1842.

Информация об авторах

Малинковский Юрий Владимирович, профессор, доктор физико-математических наук, профессор кафедры фундаментальной и прикладной математики, Гомельский государственный университет имени Франциска Скорины.
E-mail: malinkovsky@gsu.by

Немилостивая Виолетта Анатольевна, ассистент кафедры фундаментальной и прикладной математики, Гомельский государственный университет имени Франциска Скорины.
E-mail: vetta.i.am@gmail.com

Information about the authors

Yuri V. Malinkovsky, Prof., D. Sc. (Phys.-Math.), Prof. of the Department of Fundamental and Applied Mathematics, Francisk Skorina Gomel State University.
E-mail: malinkovsky@gsu.by

Violetta A. Nemilostivaya, Assistant at the Department of Fundamental and Applied Mathematics, Francisk Skorina Gomel State University.
E-mail: vetta.i.am@gmail.com

УДК 532.22:519.6
https://doi.org/10.37661/1816-0301-2023-20-4-56-68

Оригинальная статья
Original Paper

Вариационно-разностный метод численного моделирования равновесных капиллярных поверхностей

Ю. Н. Горбачёва✉, В. К. Полевиков

Белорусский государственный университет,
пр. Независимости, 4, Минск, 220030, Беларусь
✉*E-mail: gorbachevayun@gmail.com*

Аннотация

Цели. Предлагается вариационно-разностный метод численного моделирования равновесных капиллярных поверхностей, базирующийся на минимизации энергетического функционала. В качестве тестовой рассматривается известная осесимметричная задача о равновесных формах капли, находящейся на горизонтальной вращающейся плоскости в поле силы тяжести. Математическая модель задачи строится на основании вариационного принципа: капля принимает такую форму, при которой она обладает минимумом полной энергии при заданном объеме. С помощью метода конечных элементов задача минимизации функционала сводится к системе нелинейных уравнений, решение которой ищется с помощью итерационного метода Ньютона.

Методы. Используется вариационно-разностный подход (метод конечных элементов), в котором в качестве базисных функций выбираются финитные линейные функции.

Результаты. С помощью метода конечных элементов построены равновесные формы капли на вращающейся плоскости в широком диапазоне определяющих параметров: числа Бонда, вращательного числа Вебера и угла смачивания. Определено влияние этих параметров на форму капли. Численные результаты согласуются с результатами, полученными с помощью итерационно-разностного метода во всем диапазоне физической устойчивости относительно осесимметричных возмущений.

Заключение. Метод конечных элементов реагирует на потерю устойчивости капли относительно осесимметричных возмущений, поэтому может применяться для исследования устойчивости равновесия осесимметричных капиллярных поверхностей.

Ключевые слова: равновесные формы капли, вращающаяся плоскость, число Бонда, число Вебера, угол смачивания, метод конечных элементов

Для цитирования. Горбачёва, Ю. Н. Вариационно-разностный метод численного моделирования равновесных капиллярных поверхностей / Ю. Н. Горбачёва, В. К. Полевиков // Информатика. – 2023. − Т. 20, № 4. – С. 56–68. https://doi.org/10.37661/1816-0301-2023-20-4-56-68

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 08.08.2023
Подписана в печать | Accepted 10.10.2023
Опубликована | Published 29.12.2023

A variational-difference method for numerical simulation of equilibrium capillary surfaces

Yuliya N. Gorbacheva✉, Viktor K. Polevikov

Belarusian State University,
av. Nezavisimosti, 4, Minsk, 220030, Belarus
✉*E-mail: gorbachevayun@gmail.com*

Abstract
Objectives. A variational-difference method for numerical simulation of equilibrium capillary surfaces based on the minimization of the energy functional is proposed. As a test task a well-known axisymmetric hydrostatic problem on equilibrium shapes of a drop adjacent to a horizontal rotating plane under gravity is considered. The mathematical model of the problem is built on the basis of the variational principle: the shape of the drop satisfies the minimum total energy for a given volume. The problem of the functional minimization is reduced to a system of nonlinear equations using the finite element method. To solve the system a Newton's iterative method is applied.
Methods. The variational-difference approach (the finite element method) is used. The finite linear functions are chosen as basic functions.
Results. Equilibrium shapes of a drop on a rotating plane are constructed by the finite element method in a wide range of defining parameters: Bond number, rotational Weber number and wetting angle. The influence of these parameters on the shape of a drop is investigated. The numerical results are matched with the results obtained using the iterative-difference approach over the entire range of physical stability with respect to axisymmetric perturbations.
Conclusion. The finite element method responds to the loss of stability of a drop with respect to axisymmetric perturbations. Therefore it can be used to study the stability of the equilibrium of axisymmetric capillary surfaces.

Keywords: equilibrium shapes of a drop, rotating plane, Bond number, Weber number, wetting angle, finite element method

For citation. Gorbacheva Yu. N., Polevikov V. K. *A variational-difference method for numerical simulation of equilibrium capillary surfaces*. Informatika [*Informatics*], 2023, vol. 20, no. 4, pp. 56–68 (In Russ.).
https://doi.org/10.37661/1816-0301-2023-20-4-56-68

Conflict of interest. The authors declare of no conflict of interest.

Введение. Изучение капиллярных явлений имеет как теоретическое, так и практическое значение. Результаты исследований в этом направлении находят применение в микроэлектронике, газовой и нефтяной промышленности, медицине, биотехнологиях, космических технологиях, металлургии. Капиллярные поверхности являются непростым объектом исследования, так как в качестве математических моделей капиллярных поверхностей выступают нелинейные дифференциальные уравнения и системы [1].

В общем случае задача о равновесных формах капиллярной жидкости может быть решена лишь численными методами. Задачи гидростатики сводятся либо к минимизации функционала, описывающего полную энергию системы газ – жидкость – твердое тело [2–5], либо к решению соответствующей системы нелинейных дифференциальных уравнений Юнга – Лапласа, дополненных интегральным условием сохранения объема жидкости. Численное решение задачи во втором случае сводится либо к решению задачи Коши с применением метода стрельбы [6–8], либо к решению краевой задачи [9, 10].

Вариационные методы решения задачи о равновесии жидкости представляют интерес, так как основаны на непосредственной минимизации энергетического функционала, поэтому автоматически приводят к определению устойчивой формы равновесия, отвечающей минимуму полной энергии [11, 12].

В настоящей работе рассматривается осесимметричная задача о равновесных формах капли, расположенной на вращающейся горизонтальной плоскости в поле силы тяжести. Сформулирована вариационная постановка задачи. Для численного решения поставленной задачи приме-

няется метод конечных элементов. Такой подход использовался для численного решения задачи о капле, свисающей с кромки капилляра в поле силы тяжести [4]. Численные результаты, полученные с помощью метода конечных элементов, сравнивались с результатами, полученными с помощью итерационно-разностного метода второго порядка аппроксимации, который рассмотрен в работах [9, 10]. Проведенные исследования представляют интерес для нанесения тонких покрытий на твердых поверхностях методом центрифугирования [13].

Вариационная постановка задачи. Пусть на горизонтальной плоскости находится осесимметричная капля жидкости в поле силы тяжести с ускорением свободного падения g, направленным вниз, коэффициентом поверхностного натяжения σ, плотностью ρ, заданным объемом V и углом смачивания α. Принимая во внимание осевую симметрию капли, поместим начало координат в центр ее основания. Капля вместе с плоскостью вращается как одно твердое тело с постоянной угловой скоростью ω вокруг оси симметрии OZ. В осесимметричной постановке задача численно решалась в работах [8–10]. Новизна этой работы заключается в использовании вариационно-разностного подхода, сама идея которого предполагает адекватное реагирование на начало кризиса равновесных состояний.

Интегро-дифференциальное уравнение равновесия осесимметричной капли, вращающейся на горизонтальной плоскости, получим из вариационного принципа минимума полной энергии капли $\tilde{E}$, которая складывается из четырех составляющих: поверхностной энергии, потенциальной энергии в поле силы тяжести, потенциальной энергии в поле центробежных сил и энергии смачивания [12, 14]:

$$\tilde{E} = \sigma\left(\left|\tilde{\Gamma}\right| + \frac{\rho g}{\sigma}\int_{\tilde{\Omega}} Z d\tilde{\Omega} - \frac{\omega^2\rho}{2\sigma}\int_{\tilde{\Omega}}\left(X^2 + Y^2\right)d\tilde{\Omega} - \cos\alpha\left|\tilde{\Sigma}\right|\right), \tag{1}$$

где $\tilde{\Gamma}$ – свободная поверхность капли (граница между жидкостью и воздухом);

$\left|\tilde{\Gamma}\right|$ – площадь свободной поверхности;

$\tilde{\Omega}$ – объем, заполненный жидкостью, т. е. $\left|\tilde{\Omega}\right| = V$;

$\tilde{\Sigma}$ – смоченная область (граница между жидкостью и горизонтальной плоскостью);

$\left|\tilde{\Sigma}\right|$ – площадь смоченной области.

Условие сохранения объема жидкости

$$V = \int_{\tilde{\Omega}} 1 d\tilde{\Omega} \tag{2}$$

включается в энергетический функционал (1) на основе метода неопределенных множителей Лагранжа. Таким образом, задача условной минимизации (1), (2) сводится к задаче на безусловный экстремум функционала

$$\tilde{E}_\lambda = \tilde{E} + \lambda\left(\int_{\tilde{\Omega}} 1 d\tilde{\Omega} - V\right), \tag{3}$$

где λ – неопределенный множитель Лагранжа.

Перейдем к безразмерным переменным

$$E = \frac{\tilde{E}}{\sigma V^{2/3}},\ z = \frac{Z}{V^{1/3}},\ x = \frac{X}{V^{1/3}},\ y = \frac{Y}{V^{1/3}}, \tag{4}$$

где E – безразмерная полная энергия капли.

Функционал (3) в переменных (4) принимает вид

$$E_\lambda = \int_\Gamma 1 d\Gamma + \mathrm{Bo}\int_\Omega z d\Omega - \mathrm{P}\int_\Omega \left(x^2 + y^2\right) d\Omega - \cos\alpha \int_\Sigma 1 d\Sigma + \lambda\left(\int_\Omega 1 d\Omega - 1\right), \quad (5)$$

где Г – безразмерная свободная поверхность;

$\mathrm{Bo} = \rho g V^{2/3} / \sigma$ – число Бонда, характеризующее отношение гравитационных сил к капиллярным силам;

Ω – безразмерная область, заполненная жидкостью;

$\mathrm{P} = \rho\omega^2 V / (2\sigma)$ – вращательное число Вебера, характеризующее отношение между центробежными и капиллярными силами;

Σ – безразмерная смоченная область.

Учитывая осевую симметрию задачи и следуя подходу в [3, 4], представим безразмерную осесимметричную свободную поверхность капли Г в сферических координатах:

$$(x, y, z) = \left(u(\theta)\cos\varphi\sin\theta, u(\theta)\sin\varphi\sin\theta, u(\theta)\cos\theta\right), \quad \theta \in \left[0, \frac{\pi}{2}\right], \quad \varphi \in [0, 2\pi], \quad (6)$$

где $u(\theta)$ – расстояние от точки (x, y, z) на свободной поверхности Г до начала координат;

θ – угол между осью симметрии OZ и отрезком, соединяющим начало координат с точкой (x, y, z).

На рис. 1 показаны осевое сечение капли, связанная с ним система координат и принятые обозначения.

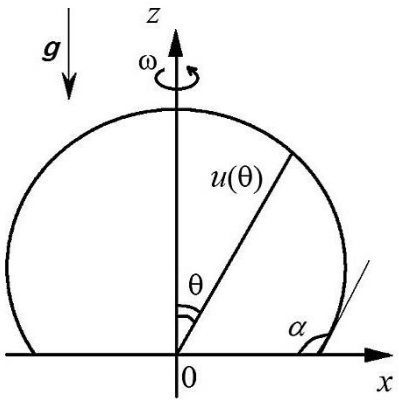


Рис. 1. Геометрия задачи

Fig. 1. Geometry of the problem

Запишем энергетический функционал (5) в переменных (6):

$$E_\lambda(u, \lambda) = 2\pi \int_0^{\pi/2} u\sqrt{(u')^2 + u^2}\sin\theta d\theta + \frac{\pi}{2}\mathrm{Bo}\int_0^{\pi/2} u^4 \sin\theta\cos\theta d\theta - \\ - \frac{2\pi}{5}\mathrm{P}\int_0^{\pi/2} u^5 \sin^3\theta\, d\theta - \pi\cos\alpha u^2\Big|_{\theta=\pi/2} + \lambda\left(\frac{2\pi}{3}\int_0^{\pi/2} u^3\sin\theta d\theta - 1\right), \quad (7)$$

где $' = d / d\theta$.

Будем искать в пространстве Соболева $W = H^1(0, \pi/2)$ функцию $u : [0, \pi/2] \to \mathbf{R}$, минимизирующую функционал (7). Под необходимым условием минимума энергетического функционала (7) понимается равенство нулю первой вариации функционала. Выпишем первую вариацию функционала:

$$\frac{d}{dt} E_\lambda(u + tv, \lambda + t\mu)\Big|_{t=0} = 2\pi \int_0^{\pi/2} \left\{ \frac{uu'v' + \left(2u^2 + (u')^2\right)v}{\sqrt{(u')^2 + u^2}} + \mathrm{Bo} u^3 v \cos\theta - \right.$$
$$\left. - \mathrm{P} u^4 v \sin^2\theta + \lambda u^2 v + \frac{\mu}{3} u^3 \right\} \sin\theta d\theta - 2\pi \cos\alpha uv\big|_{\theta=\pi/2} - \mu, \tag{8}$$

где $v = v(\theta)$ – произвольная непрерывно дифференцируемая функция;

t – параметр;

μ – константа.

Приравняв (8) к нулю и разделив на 2π, а затем подставив $\mu = 0$, получаем для всех $v \in W$ равенство

$$\int_0^{\pi/2} \left\{ \frac{uu'v' + \left(2u^2 + (u')^2\right)v}{\sqrt{(u')^2 + u^2}} + \mathrm{Bo} u^3 v \cos\theta - \mathrm{P} u^4 v \sin^2\theta + \lambda u^2 v \right\} \sin\theta d\theta - \cos\alpha uv\big|_{\theta=\pi/2} = 0. \tag{9}$$

Выражение (9) является уравнением равновесия жидкости, т. е. слабой формой нелинейного дифференциального уравнения Юнга – Лапласа с множителем Лагранжа λ. Отметим, что граничное условие с заданным углом смачивания α удовлетворяется в уравнении (9). Для задач с фиксированной линией контакта и неизвестным углом смачивания дополнительно необходимо учитывать граничное условие [4].

Условие сохранения объема

$$\frac{2\pi}{3} \int_0^{\pi/2} u^3 \sin\theta d\theta = 1 \tag{10}$$

получим путем приравнивания (8) нулю и подстановки $v = 0$, $\mu = 1$.

Таким образом, полную дифференциальную модель равновесного состояния жидкости составляют уравнение (9) и условие (10). Определяющими параметрами задачи (9), (10) являются число Бонда Bo, число Вебера P и угол смачивания α.

Метод конечных элементов. Решение задачи (9), (10) будем находить с помощью метода конечных элементов на равномерной сетке $\left\{\theta_i = ih \mid i = \overline{0, N};\ h = \pi/(2N)\right\}$. Пусть $I_i = (\theta_{i-1}, \theta_i)$ обозначает сеточный элемент, $\mathbf{P_1}$ – пространство многочленов степени 0 или 1. Рассмотрим конечномерное подпространство

$$W_h = \left\{ v_h \in H^1\left(0, \frac{\pi}{2}\right) :\ v_h\big|_{I_i} \in \mathbf{P_1} \right\}.$$

Приближенное решение задачи (9), (10) будем искать в виде

$$u_h(\theta) = \sum_{i=0}^{N} u_i \Phi_i(\theta),$$

где $\Phi_i(\theta) \in W_h$, $i = \overline{0, N}$: $\Phi_i(\theta_j) = \delta_{ij}$, $i, j = \overline{0, N}$ – финитные функции, задаваемые формулами

$$\Phi_i(\theta)=\begin{cases}\dfrac{\theta-\theta_{i-1}}{h}, & \theta\in[\theta_{i-1},\theta_i],\\ \dfrac{\theta_{i+1}-\theta}{h}, & \theta\in[\theta_i,\theta_{i+1}], \quad i=\overline{1,N-1},\\ 0, & \theta\notin[\theta_{i-1},\theta_{i+1}],\end{cases}$$

$$\Phi_0(\theta)=\begin{cases}\dfrac{\theta_1-\theta}{h}, & \theta\in[\theta_0,\theta_1],\\ 0, & \theta\notin[\theta_0,\theta_1],\end{cases}$$

$$\Phi_N(\theta)=\begin{cases}\dfrac{\theta-\theta_{N-1}}{h}, & \theta\in[\theta_{N-1},\theta_N],\\ 0, & \theta\notin[\theta_{N-1},\theta_N].\end{cases}$$

Интегралы, входящие в (9), (10), разбиваются на сумму интегралов по элементам и на каждом элементе аппроксимируются квадратурной формулой трапеций, т. е.

$$\int_0^{\pi/2}F(\theta)d\theta=\sum_{i=1}^{N}\int_{\theta_{i-1}}^{\theta_i}F(\theta)d\theta\approx\frac{h}{2}\sum_{i=1}^{N}\left(F(\theta_{i-1})+F(\theta_i)\right).$$

В результате вместо интегро-дифференциальной задачи (9), (10) получаем дискретную задачу. Подставляя в нее $v_i=\Phi_i$, $i=\overline{0,N}$, получаем систему нелинейных алгебраических уравнений

$$\mathbf{F}(\mathbf{u})=\begin{pmatrix}F_0(u_0,u_1)\\ F_1(u_0,u_1,u_2,\lambda_h,\mathrm{Bo},\mathrm{P})\\ \vdots\\ F_i(u_{i-1},u_i,u_{i+1},\lambda_h,\mathrm{Bo},\mathrm{P})\\ \vdots\\ F_{N-1}(u_{N-2},u_{N-1},u_N,\lambda_h,\mathrm{Bo},\mathrm{P})\\ F_N(u_{N-1},u_N,\lambda_h,\mathrm{P},\alpha)\\ F_{N+1}(u_1,\ldots,u_N)\end{pmatrix}=\mathbf{0}, \tag{11}$$

где $\mathbf{u}=(u_0,\ldots,u_N,\lambda_h)^T$ – вектор неизвестных;

$$F_0(u_0,u_1)=\frac{u_0u_1-u_1^2}{2\sqrt{(u_1-u_0)^2+u_1^2h^2}}\sin\theta_1;$$

$$F_i(u_{i-1},u_i,u_{i+1},\lambda_h,\mathrm{Bo},\mathrm{P}) = \frac{u_iu_{i-1}-u_{i-1}^2}{2\sqrt{(u_i-u_{i-1})^2+u_{i-1}^2h^2}}\sin\theta_{i-1} + \frac{2u_i^2-3u_iu_{i-1}+2u_i^2h^2+u_{i-1}^2}{2\sqrt{(u_i-u_{i-1})^2+u_i^2h^2}}\sin\theta_i +$$
$$+\frac{2u_i^2-3u_iu_{i+1}+2u_i^2h^2+u_{i+1}^2}{2\sqrt{(u_{i+1}-u_i)^2+u_i^2h^2}}\sin\theta_i + \frac{u_iu_{i+1}-u_{i+1}^2}{2\sqrt{(u_{i+1}-u_i)^2+u_{i+1}^2h^2}}\sin\theta_{i+1} +$$
$$+h\,\mathrm{Bo}u_i^3\sin\theta_i\cos\theta_i - h\mathrm{P}u_i^4\sin^3\theta_i + h\,\lambda_h u_i^2\sin\theta_i, \qquad i=\overline{1,N-1};$$

$$F_N(u_{N-1},u_N,\lambda_h,\mathrm{P},\alpha) = \frac{u_Nu_{N-1}-u_{N-1}^2}{2\sqrt{(u_N-u_{N-1})^2+u_{N-1}^2h^2}}\sin\theta_{N-1} + \frac{2u_N^2-3u_Nu_{N-1}+2u_N^2h^2+u_{N-1}^2}{2\sqrt{(u_N-u_{N-1})^2+u_N^2h^2}} -$$
$$-\frac{h}{2}\mathrm{P}u_N^4 + \frac{h}{2}\lambda_h u_N^2 - u_N\cos\alpha;$$

$$F_{N+1}(u_1,\ldots,u_N) = \frac{2\pi h}{3}\left(\sum_{i=1}^{N-1}u_i^3\sin\theta_i + \frac{u_N^3}{2}\right) - 1.$$

Таким образом, нахождение равновесной формы капли при заданном числе Бонда Bo, числе Вебера P и угле смачивания α сводится к решению нелинейной системы уравнений (11) с помощью итерационного метода Ньютона. Реализация метода Ньютона на *k*-й итерации ($k = 0, 1, \ldots$) сводится к нахождению вектора поправок $\Delta\mathbf{u}^k$ путем решения системы линейных алгебраических уравнений

$$\frac{\partial\mathbf{F}}{\partial\mathbf{u}}\left(\mathbf{u}^k\right)\Delta\mathbf{u}^k = -\mathbf{F}\left(\mathbf{u}^k\right),$$

где $\frac{\partial\mathbf{F}}{\partial\mathbf{u}}\left(\mathbf{u}^k\right)$ – матрица Якоби, которая является разреженной;

$\mathbf{u}^k = \left(u_0^k,\ldots,u_N^k,\lambda_h^k\right)^T$.

Затем с помощью найденного вектора поправок $\Delta\mathbf{u}^k$ вычисляется очередное приближение

$$\mathbf{u}^{k+1} = \Delta\mathbf{u}^k + \mathbf{u}^k.$$

Условием окончания итерационного процесса выбирали

$$\left\|\Delta\mathbf{u}^k\right\|_\infty \le \varepsilon,\ \varepsilon = 10^{-7}.$$

В качестве начального приближения $\mathbf{u}^0$ при заданном числе Бонда, числе Вебера и угле смачивания рассматривали сферический сегмент единичного объема с заданным углом смачивания α, т. е. функцию $u(\theta)$ вычисляли по формуле

$$u(\theta) = R\sqrt{1-\cos^2\alpha\sin^2\theta} - R\cos\alpha\cos\theta,$$

где $R = \left(3\Big/\left[\pi\left(2-3\cos\alpha+\cos^3\alpha\right)\right]\right)^{1/3}$ – безразмерный радиус сферы.

Дискретная модель строилась с помощью непосредственной аппроксимации энергетического функционала, что обеспечивает консервативность полученной схемы, так как при таком подходе в выражении дискретного энергетического функционала не возникают нефизические слагаемые.

Результаты и обсуждение. Расчеты выполнялись на равномерной сетке с числом разбиений $N = 500$ и проводились в широком диапазоне определяющих параметров. Для анализа полученных результатов использовали сеточные функции $x_i = u_i \sin\theta_i$, $z_i = u_i \cos\theta_i$, $i = \overline{0, N}$. При расчетах выяснилось, что при фиксированном числе Бонда Bo и угле смачивания α с увеличением числа Вебера P наступает момент, когда вычислительный процесс перестает сходиться. Чтобы стабилизировать вычислительный процесс, в качестве начального итерационного приближения для нового P выбиралось решение, полученное для предыдущего значения P, т. е. использовался метод продолжения по параметру. Считалось, что значение P превышает критическое P_{cr}, если при этом значении итерации расходились. Критические значения P_{cr} уточняли методом дихотомии, пока их погрешность не становилась менее 10^{-4}.

На рис. 2 показана эволюция равновесных форм вращающейся капли, полученных с помощью метода конечных элементов при увеличении P для угла смачивания $\alpha = \pi/2$ и некоторых чисел Бонда. Пунктирными линиями обозначены формы при $P = P_{cr}$. Видно, что равновесные формы капли сначала принимают выпуклые конфигурации, а с увеличением числа Вебера начинают прижиматься к центру горизонтальной плоскости. Это объясняется ростом центробежных сил. С увеличением числа Бонда потеря вычислительной устойчивости наступает при меньших значениях P_{cr}, при этом радиус области смачивания увеличивается, а высота поднятия капли уменьшается.

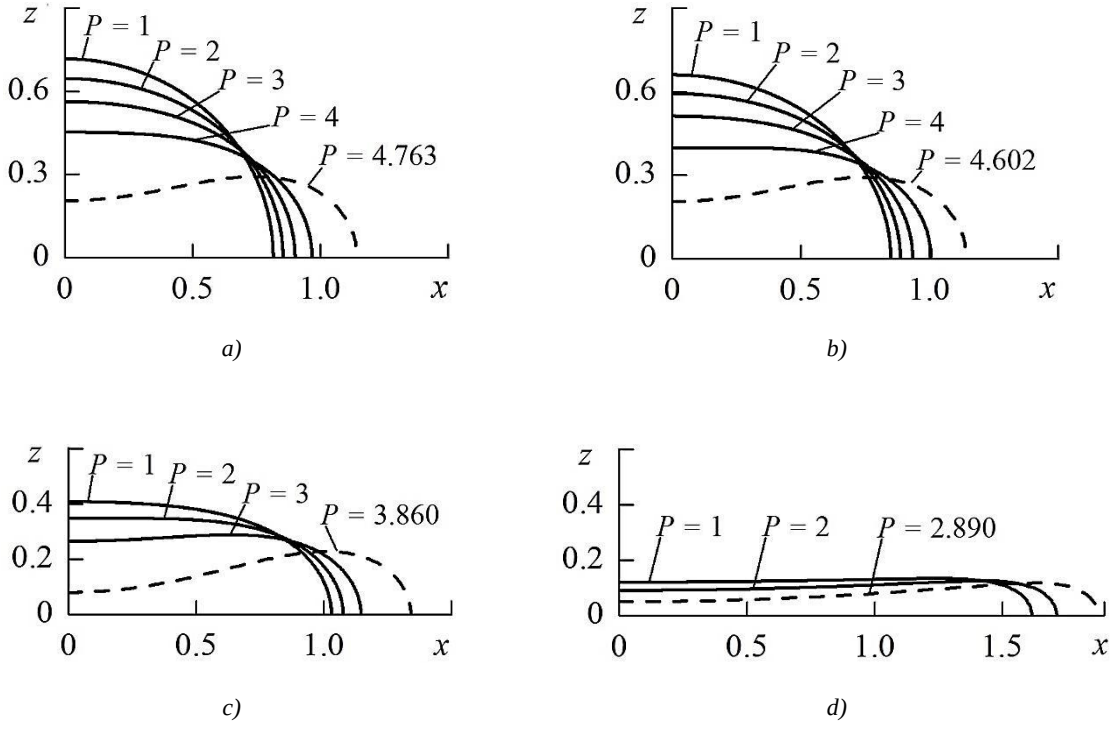
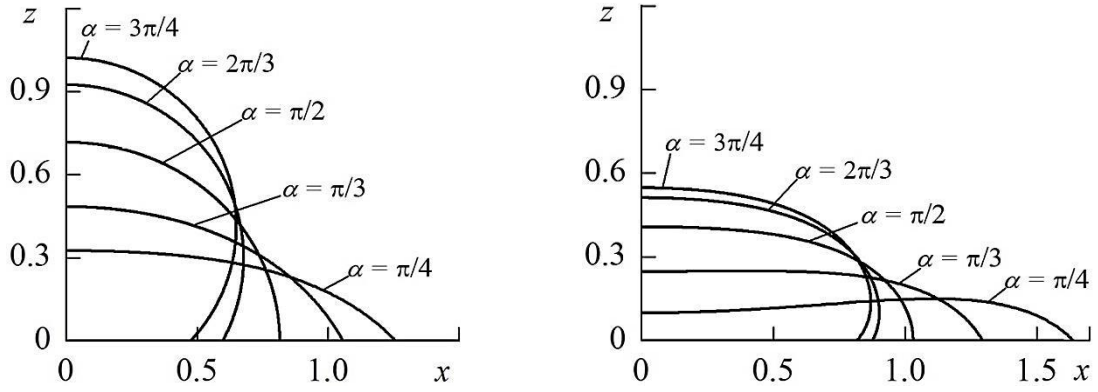


Рис. 2. Эволюция равновесных форм вращающейся капли с увеличением числа P при $\alpha = \pi/2$:
a) Bo = 0; *b)* Bo = 1; *c)* Bo = 10; *d)* Bo = 100

Fig. 2. Evolution of the equilibrium shapes of a rotating drop with increasing the number P *for* $\alpha = \pi/2$:
a) Bo = 0; *b)* Bo = 1; *c)* Bo = 10; *d)* Bo = 100

На рис. 3 продемонстрировано влияние изменения угла смачивания на равновесные формы капли. Видно, что с увеличением угла смачивания радиус области смачивания уменьшается, а высота поднятия капли увеличивается.



a) *b)*

Рис. 3. Эволюция равновесных форм вращающейся капли с увеличением угла смачивания α:
a) Bo = 0, P = 1; *b)* Bo = 10, P = 1

Fig. 3. Evolution of the equilibrium shapes of a rotating drop with increasing the wetting angle α:
a) Bo = 0, P = 1; *b)* Bo = 10, P = 1

В табл. 1 приведены критические значения числа Вебера P_{cr}, полученные для некоторых чисел Бонда Bo и углов смачивания α с помощью метода конечных элементов (метод 1) и итерационно-разностного метода второго порядка аппроксимации (метод 2). Итерационно-разностный метод численного решения задачи о капле, вращающейся на горизонтальной плоскости в гравитационном поле, рассматривается в статьях [9, 10]. Для итерационно-разностного метода расчеты осуществлялись на равномерной сетке с числом разбиений $N = 500$. Оба метода показали, что при фиксированном числе Бонда Bo и угле смачивания α с увеличением числа Вебера P наступает момент, когда P достигает значения $P \sim P_{cr}$, при котором возникает вычислительная неустойчивость методов. Полученные результаты согласуются.

Таблица 1
Критические значения числа Вебера

Table 1
Critical values of the Weber number

α	P_{cr} при Bo = 0 P_{cr} *for* Bo = 0		P_{cr} при Bo = 1 P_{cr} *for* Bo = 1		P_{cr} при Bo = 10 P_{cr} *for* Bo = 10		P_{cr} при Bo = 100 P_{cr} *for* Bo = 100	
	Метод 1 *Method 1*	Метод 2 *Method 2*	Метод 1 *Method 1*	Метод 2 *Method 2*	Метод 1 *Method 1*	Метод 2 *Method 2*	Метод 1 *Method 1*	Метод 2 *Method 2*
π / 6	0,632	0,630	0,598	0,593	0,481	0,487	0,400	0,412
π / 4	1,384	1,381	1,323	1,323	1,085	1,090	0,800	0,950
π / 3	2,367	2,367	2,276	2,276	1,891	1,893	1,387	1,628
π / 2	4,763	4,762	4,602	4,601	3,860	3,860	2,890	3,272
2π / 3	7,206	7,206	6,963	6,963	5,839	5,838	4,400	4,916
3π / 4	8,244	8,244	7,955	7,955	6,653	6,654	5,000	5,595
5π / 6	9,061	9,061	8,727	8,727	7,272	7,272	5,593	6,114

В работе [12] показано, что для рассматриваемой задачи наиболее опасными являются неосесимметричные возмущения, т. е. потеря физической устойчивости вращающейся капли относительно неосесимметричных возмущений наступает раньше, чем относительно осесимметричных. Так, при Bo = 0 и $\alpha = \pi/2$ потеря устойчивости относительно осесимметричных возмущений наступает при P = 4,763 [9, 10, 12]. Вычислительная неустойчивость для метода конечных элементов наступает при $P_{cr} \sim 4{,}763$, а для итерационно-разностного метода – при $P_{cr} \sim 4{,}762$. Следует предположить, что кризис вычислительного процесса для метода конечных элементов и итерационно-разностного метода возникает тогда, когда капля теряет физическую устойчивость относительно осесимметричных возмущений. Несмотря на то что кризис вычислительного процесса для метода конечных элементов и итерационно-разностного метода

наступает позже, чем потеря устойчивости капли относительно неосесимметричных возмущений, рассмотренные методы дают полезные оценки о влиянии параметров задачи на устойчивость равновесных форм относительно осесимметричных возмущений. Описанный в работе [8] метод также позволяет строить равновесные формы капли в широком диапазоне определяющих параметров, но не реагирует на кризис равновесного состояния.

Найдем экспериментальный порядок точности метода конечных элементов. Пусть равновесные формы определяются на равномерной сетке с шагом h. Зависимость погрешности $u_h - u$ от шага сетки при $N \to \infty$ рассмотрим в виде соотношения

$$\|u_h - u\|_\infty \sim a \cdot h^p = O\left(h^p\right),$$

где u – точное решение задачи;

u_h – численное решение задачи с шагом h;

p – порядок точности метода.

Для численного решения u_{2h}, полученного с шагом $2h$, справедливо приближение

$$\|u_{2h} - u\|_\infty \sim a \cdot (2h)^p .$$

В результате получим следующее соотношение:

$$\frac{\|u_{2h} - u\|_\infty}{\|u_h - u\|_\infty} \sim 2^p \quad \Rightarrow \quad p \sim \log_2 \left(\frac{\|u_{2h} - u\|_\infty}{\|u_h - u\|_\infty} \right).$$

Проведем вычислительный эксперимент на последовательности равномерных сеток с числом разбиений N = 32, 64, 128, 256, 512, 1024, 2048. В роли точного решения u рассмотрим численное решение, полученное на сетке при N = 4096. Результаты численного эксперимента представлены в табл. 2–5. По полученным результатам можно сделать вывод, что сходимость метода носит квадратичный характер.

Таблица 2
Скорость сходимости в равномерной норме при Bo = 1, P = 1, α = π/4

Table 2
Convergence rate in uniform norm for Bo = 1, P = 1, α = π/4

N	32	64	128	256	512	1024	2048
$\|u_h - u\|_\infty$	$6{,}39 \cdot 10^{-3}$	$1{,}55 \cdot 10^{-3}$	$3{,}85 \cdot 10^{-4}$	$9{,}57 \cdot 10^{-5}$	$2{,}36 \cdot 10^{-5}$	$5{,}63 \cdot 10^{-6}$	$1{,}13 \cdot 10^{-6}$
p	–	2,042	2,012	2,010	2,018	2,071	2,322

Таблица 3
Скорость сходимости в равномерной норме при Bo = 1, P = 1, α = π/2

Table 3
Convergence rate in uniform norm for Bo = 1, P = 1, α = π/2

N	32	64	128	256	512	1024	2048
$\|u_h - u\|_\infty$	$5{,}58 \cdot 10^{-4}$	$1{,}68 \cdot 10^{-4}$	$4{,}90 \cdot 10^{-5}$	$1{,}40 \cdot 10^{-5}$	$3{,}87 \cdot 10^{-6}$	$1{,}02 \cdot 10^{-6}$	$2{,}20 \cdot 10^{-7}$
p	–	1,733	1,776	1,811	1,850	1,929	2,211

Таблица 4
Скорость сходимости в равномерной норме при Bo = 1, P = 0, α = 3π/4

Table 4
Convergence rate in uniform norm for Bo = 1, P = 0, α = 3π/4

N	32	64	128	256	512	1024	2048
$\|u_h - u\|_\infty$	$5{,}61 \cdot 10^{-4}$	$1{,}72 \cdot 10^{-4}$	$5{,}07 \cdot 10^{-5}$	$1{,}46 \cdot 10^{-5}$	$4{,}07 \cdot 10^{-6}$	$1{,}07 \cdot 10^{-7}$	$2{,}32 \cdot 10^{-7}$
p	–	1,708	1,760	1,799	1,842	1,922	2,204

Таблица 5
Скорость сходимости в равномерной норме при Bo = 10, P = 2, α = π/2

Table 5
Convergence rate in uniform norm for Bo = 10, P = 2, α = π/2

N	32	64	128	256	512	1024	2048
$\|u_h - u\|_\infty$	$2{,}36\cdot10^{-3}$	$5{,}84\cdot10^{-4}$	$1{,}46\cdot10^{-4}$	$3{,}63\cdot10^{-5}$	$8{,}97\cdot10^{-6}$	$2{,}13\cdot10^{-6}$	$4{,}27\cdot10^{-7}$
p	–	2,01	2,00	2,00	2,02	2,07	2,32

Заключение. Построен алгоритм метода конечных элементов численного решения задачи о равновесных формах капли, вращающейся на горизонтальной плоскости в поле силы тяжести. Экспериментально исследован порядок точности метода. Построенный алгоритм реагирует на потерю устойчивости капли на вращающейся плоскости относительно осесимметричных возмущений. Полученные с его помощью численные результаты согласуются с результатами итерационно-разностного метода второго порядка аппроксимации во всем диапазоне физической устойчивости относительно осесимметричных возмущений. Таким образом, метод конечных элементов может применяться для исследования устойчивости равновесных форм капиллярных поверхностей относительно осесимметричных возмущений. Описанный метод может быть модифицирован и применен для нахождения равновесных форм капли на вращающемся диске с нерегулярными граничными условиями, т. е. когда известна линия контакта, а угол смачивания неизвестен.

Вклад авторов. Все этапы работы по подготовке статьи для публикации осуществлялись авторами совместно.

Список использованных источников

1. Сокуров, А. А. Численно-аналитическое исследование математических моделей капиллярных менисков / А. А. Сокуров // Вестник КРАУНЦ. Физ.-мат. науки. – 2021. – Т. 36, № 3. – С. 80–93. https://doi.org/10.26117/2079-6641-2021-36-3-80-93

2. A finite element based algorithm for determining interfacial tension (γ) from pendant drop profiles / N. M. Dingle [et al.] // J. of Colloid and Interface Science. – 2005. – Vol. 286, no. 2. – P. 647–660. https://doi.org/10.1016/j.jcis.2005.01.052

3. Dingle, N. M. A robust algorithm for the simultaneous parameter estimation of interfacial tension and contact angle from sessile drop profiles / N. M. Dingle, M. T. Harris // J. of Colloid and Interface Science. – 2005. – Vol. 286, no. 2. – P. 670–680. https://doi.org/10.1016/j.jcis.2005.01.087

4. Simulation of a pending drop at a capillary tip / M. Gille [et al.] // Communications in Nonlinear Science and Numerical Simulation. – 2015. – Vol. 26. – P. 137–151. https://doi.org/10.1016/j.cnsns.2015.02.007

5. Basaran, O. A. Axisymmetric shapes and stability of pendant and sessile drops in an electric field / O. A. Basaran, L. E. Scriven // J. of Colloid and Interface Science. – 1990. – Vol. 140, no. 1. – P. 10–30. https://doi.org/10.1016/0021-9797(90)90316-G

6. Saad, S. M. I. Total Gaussian curvature, drop shapes and the range of applicability of drop shape techniques / S. M. I. Saad, A. W. Neumann // Advances in Colloid and Interface Science. – 2014. – Vol. 204. – P. 1–14. https://doi.org/10.1016/j.cis.2013.12.001

7. Shape analysis of a rotating axisymmetric drop in gravitational field: Comparison of numerical schemes for real-time data processing / K. D. Danov [et al.] // Colloids and Surfaces A: Physicochemical and Engineering Aspects. – 2016. – Vol. 489. – P. 75–85. https://doi.org/10.1016/j.colsurfa.2015.10.028

8. Авдейчик, Е. В. Численное исследование относительного равновесия капли с односвязной свободной поверхностью на вращающейся плоскости / Е. В. Авдейчик, П. Н. Конон // Журнал Белорусского государственного университета. Математика. Информатика. – 2022. – № 3. – С. 79–90. https://doi.org/10.33581/2520-6508-2022-3-79-90

9. Polevikov, V. K. Methods for numerical modeling of two-dimensional capillary surfaces / V. K. Polevikov // Computational Methods in Applied Mathematics. – 2004. – Vol. 4, no. 1. – P. 66–93. https://doi.org/10.2478/cmam-2004-0005

10. Полевиков, В. К. Численное исследование равновесных форм капли, вращающейся в гравитационном поле / В. К. Полевиков, В. М. Денисенко // Вестник Белорусского государственного университета им. В. И. Ленина. – 1985. – № 2. – С. 37–41.

11. Черноусько, Ф. Л. Задача о равновесии жидкости, подверженной действию сил тяжести и поверхностного натяжения / Ф. Л. Черноусько // Введение в динамику тела с жидкостью в условиях невесомости. – М. : ВЦ АН СССР, 1968. – С. 69–97.

12. Методы решения задач гидромеханики для условий невесомости / А. Д. Мышкис [и др.] ; под ред. А. Д. Мышкиса. – Киев : Наукова думка, 1992. – 592 с.

13. Investigation of the shape and stability of a liquid drop on a rotating substrate / P. V. Lebedev-Stepanov [et al.] // Acoustical Physics. – 2011. – Vol. 57, no. 3. – P. 320–325. https://doi.org/10.1134/S1063771011030122

14. Финн, Р. Равновесные капиллярные поверхности. Математическая теория : пер. с англ. / Р. Финн. – М. : Мир, 1989. – 312 с.

References

1. Sokurov A. A. *An analytical and numerical study of capillary menisci*. Vestnik KRAUNC. Fiziko-matematičeskie nauki [*Bulletin KRASEC. Physical and Mathematical Sciences*], 2021, vol. 36, no. 3, pp. 80–93 (In Russ.). https://doi.org/10.26117/2079-6641-2021-36-3-80-93

2. Dingle N. M., Tjiptowidjojo K., Basaran O. A., Harris M. T. A finite element based algorithm for determining interfacial tension (γ) from pendant drop profiles. *Journal of Colloid and Interface Science*, 2005, vol. 286, no. 2, pp. 647–660. https://doi.org/10.1016/j.jcis.2005.01.052

3. Dingle N. M., Harris M. T. A robust algorithm for the simultaneous parameter estimation of interfacial tension and contact angle from sessile drop profiles. *Journal of Colloid and Interface Science*, 2005, vol. 286, no. 2, pp. 670–680. https://doi.org/10.1016/j.jcis.2005.01.087

4. Gille M., Gorbacheva Yu., Hahn A., Polevikov V., Tobiska L. Simulation of a pending drop at a capillary tip. *Communications in Nonlinear Science and Numerical Simulation*, 2015, vol. 26, pp. 137–151. https://doi.org/10.1016/j.cnsns.2015.02.007

5. Basaran O. A., Scriven L. E. Axisymmetric shapes and stability of pendant and sessile drops in an electric field. *Journal of Colloid and Interface Science*, 1990, vol. 140, no. 1, pp. 10–30. https://doi.org/10.1016/0021-9797(90)90316-G

6. Saad S. M. I., Neumann A. W. Total Gaussian curvature, drop shapes and the range of applicability of drop shape techniques. *Advances in Colloid and Interface Science*, 2014, vol. 204, pp. 1–14. https://doi.org/10.1016/j.cis.2013.12.001

7. Danov K. D., Dimova S. N., Ivanov T. B., Novev J. K. Shape analysis of a rotating axisymmetric drop in gravitational field: Comparison of numerical schemes for real-time data processing. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 2016, vol. 489, pp. 75–85. https://doi.org/10.1016/j.colsurfa.2015.10.028

8. Audzeichyk Ya. V., Konon P. N. *Numerical study of the relative equilibrium of a droplet with a simply connected free surface on a rotating plane*. Zhurnal Belorusskogo gosudarstvennogo universiteta. Matematika. Informatika [*Journal of the Belarusian State University. Mathematics and Informatics*], 2022, no. 3, pp. 79–90 (In Russ.). https://doi.org/10.33581/2520-6508-2022-3-79-90

9. Polevikov V. K. Methods for numerical modeling of two-dimensional capillary surfaces. *Computational Methods in Applied Mathematics*, 2004, vol. 4, no. 1, pp. 66–93. https://doi.org/10.2478/cmam-2004-0005

10. Polevikov V. K., Denisenko V. M. *Numerical study of equilibrium shapes of a drop rotating in gravitational field*. Vestnik Belorusskogo gosudarstvennogo universiteta imeni V. I. Lenina [*Bulletin of the Belarusian State University named after V. I. Lenin*], 1985, no. 2, pp. 37–41 (In Russ.).

11. Chernous'ko F. L. *The problem of the equilibrium of a fluid subjected to the action of gravity and surface tension*. Vvedenie v dinamiku tela s zhidkost'ju v uslovijah nevesomosti [*Introduction to the Dynamics of a Body with Liquid in Weightlessness*]. Moscow, Vychislitel'nyj centr Akademii nauk Sojuza Sovetskih Socialisticheskih Respublik, 1968, p. 69–97 (In Russ.).

12. Myshkis A. D., Babskij V. G., Zhukov M. Ju., Kopachevskij N. D., Slobozhanin L. A., Tjupcov A. D. Metody reshenija zadach gidromehaniki dlja uslovij nevesomosti. *Methods for Solving Problems in Hydromechanics in Zero Gravity Conditions*. In A. D. Myshkis (ed.). Kiev, Naukova dumka, 1992, 592 p. (In Russ.).

13. Lebedev-Stepanov P. V., Karabut T. A., Chernyshov N. A., Rybak S. A. Investigation of the shape and stability of a liquid drop on a rotating substrate. *Acoustical Physics*, 2011, vol. 57, no. 3, pp. 320–325. https://doi.org/10.1134/S1063771011030122

14. Finn R. *Equilibrium Capillary Surfaces.* New York, Springer, 1986, 245 p.

Информация об авторах

Горбачёва Юлия Николаевна, старший преподаватель кафедры вычислительной математики факультета прикладной математики и информатики, Белорусский государственный университет.
E-mail: gorbachevayun@gmail.com

Полевиков Виктор Кузьмич, кандидат физико-математических наук, доцент, доцент кафедры вычислительной математики факультета прикладной математики и информатики, Белорусский государственный университет.
E-mail: polevikov@bsu.by
https://orcid.org/0000-0003-3846-7776

Information about the authors

Yuliya N. Gorbacheva, Senior Lecturer of the Department of Computational Mathematics of the Faculty of Applied Mathematics and Informatics, Belarusian State University.
E-mail: gorbachevayun@gmail.com

Viktor K. Polevikov, Ph. D. (Phys.-Math.), Assoc. Prof., Assoc. Prof. of the Department of Computational Mathematics of the Faculty of Applied Mathematics and Informatics, Belarusian State University.
E-mail: polevikov@bsu.by
https://orcid.org/0000-0003-3846-7776

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
INFORMATION TECHNOLOGIES

УДК 004.75
https://doi.org/10.37661/1816-0301-2023-20-4-69-86

Оригинальная статья
Original Paper

Онтологический анализ в задачах моделирования угроз системам на основе контейнерных приложений

А. И. Бражук✉, Е. В. Олизарович

Гродненский государственный университет имени Янки Купалы,
ул. Ожешко, 22, Гродно, 230023, Беларусь
✉*E-mail: brazhuk@grsu.by*

Аннотация

Цели. Основной целью работы является экспериментальная верификация методики автоматического моделирования угроз на основе онтологического подхода на примере многокомпонентных контейнерных приложений, представленных в виде диаграмм потоков данных.

Методы. В работе применены методы онтологического моделирования и управления знаниями. Для представления знаний использован язык веб-онтологий, для моделирования угроз – функции автоматического логического вывода на основе дескрипционных (описательных) логик.

Результаты. Разработан машинно-читаемый набор (датасет) из 200 диаграмм потоков данных, каждая диаграмма получена из конфигурации реального контейнерного приложения и представлена в виде онтологии и графа знаний. Сформирована онтологическая двухуровневая предметно-ориентированная модель угроз контейнерных приложений. Проведен эксперимент по сравнению величины покрытия угрозами посредством общепринятого подхода и посредством предметно-ориентированных угроз для разработанного датасета. Для 95 % диаграмм предметно-ориентированная модель угроз показала величину покрытия, аналогичную или бóльшую в сравнении с общепринятым подходом.

Заключение. Результаты эксперимента доказывают пригодность и эффективность онтологического подхода для автоматического моделирования угроз. Разработанный датасет может быть использован для различных исследований в области автоматизации моделирования угроз.

Ключевые слова: компьютерные системы, контейнерные приложения, системный анализ, информационная безопасность, моделирование угроз, онтологии, автоматический логический вывод, дескрипционные (описательные) логики

Для цитирования. Бражук, А. И. Онтологический анализ в задачах моделирования угроз системам на основе контейнерных приложений / А. И. Бражук, Е. В. Олизарович // Информатика. – 2023. − Т. 20, № 4. – С. 69–86. https://doi.org/10.37661/1816-0301-2023-20-4-69-86

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 12.07.2023
Подписана в печать | Accepted 11.09.2023
Опубликована | Published 29.12.2023

Ontological analysis in the problems of container applications threat modelling

Andrei I. Brazhuk✉, Evgeny V. Olizarovich

Yanka Kupala State University of Grodno,
st. Ozheshko, 22, Grodno, 230023, Belarus
✉*E-mail: brazhuk@grsu.by*

Abstract
Objectives. The main purpose of the work is the experimental verification of the method of automatic threat modelling based on the ontological approach using the example of multicomponent container applications presented in the form of data flow diagrams.
Methods. Methods of ontological modelling and knowledge management are used in the work. The Web Ontology Language is used to represent knowledge; automatic reasoning based on description logics is used for threat modelling.
Results. A machine-readable set (dataset) of 200 data flow diagrams is developed; each diagram is obtained from the configuration of a real container application and is presented as an ontology and a knowledge graph. An ontological two-level domain-specific threat model of container applications is formed. An experiment is conducted to compare the coverage by threats using the common approach and using domain-specific threats for created dataset. For 95 % of the diagrams, the domain-specific threat model showed the coverage similar or greater than the common approach.
Conclusion. The results of the experiment prove the suitability and effectiveness of the ontological approach for automatic threat modelling. The created dataset can be used for various research in the field of automation of threat modelling.

Keywords: computer systems, container applications, system analysis, information security, threat modelling, ontologies, automatic reasoning, description logics

For citation. Brazhuk A. I., Olizarovich E. V. *Ontological analysis in the problems of container applications threat modelling*. Informatika [*Informatics*], 2023, vol. 20, no. 4, pp. 69–86 (In Russ.).
https://doi.org/10.37661/1816-0301-2023-20-4-69-86

Conflict of interest. The authors declare of no conflict of interest.

Введение. Моделирование угроз – это дисциплина, обеспечивающая анализ компьютерных систем на ранних стадиях их жизненного цикла (формирование требований, архитектурное проектирование), а также при аудите существующих систем с целью построения списков релевантных угроз, которые могут быть использованы для выбора мер и средств защиты [1, 2]. В течение долгого времени моделирование угроз рассматривалось как ручной, сложный, итеративный процесс, основанный на знаниях и взаимодействии экспертов [3]. В настоящее время ручной подход все чаще неприменим, поскольку широкое распространение получили быстрые методики разработки программного обеспечения, а также технологии и средства автоматического развертывания приложений [4], что требует существенного сокращения времени анализа защищенности, внедрения методик, работающих в условиях неполной формальной документации приложений, и автоматизации данного процесса [5].

Теоретическая постановка задачи автоматизации моделирования угроз указывает на недостатки существующих экспертных методов и предполагает широкий спектр возможных подходов, в частности машинное обучение, нейронные сети, логико-лингвистическое моделирование [6], системы нечеткого вывода [7], элементы науки о данных [8]. В этом направлении также активно исследуются технологии обработки естественных языков, трансформеры [9], цифровые двойники [10], а также интеллектуальный анализ текста и когнитивное моделирование [11].

Следует отметить, что практические вопросы моделирования угроз исследованы недостаточно [11]. Для представления различных артефактов при прикладном моделировании угроз используются различные нотации и графические форматы, такие как контрольные списки, диа-

граммы процессов, а также диаграммы потоков данных (Data Flow Diagram, DFD). Неформальные методики на основе популярных нотаций имеют целый спектр проблем, связанных с взаимодействием экспертов и разработчиков в процессе проектирования систем [12], а также определением оптимальной структуры системы защиты [13].

Настоящая работа посвящена практическим аспектам автоматизации моделирования угроз на основе диаграмм потоков данных посредством онтологий с использованием предметно-ориентированных моделей угроз. Переход к автоматизации требует исследований ее эффективности в сравнении с существующими подходами, а также тестовых наборов данных для подобных оценок. Для решения этих задач в работе исследуется авторская методика автоматического моделирования угроз на примере многокомпонентных контейнерных приложений.

1. Онтологическое моделирование угроз компьютерных систем. Диаграммы потоков данных являются наиболее известным способом представления структуры системы как набора процессов, хранилищ данных и внешних сущностей, связанных различными направленными потоками передачи информации [14, 15]. Этот подход хорошо отражает основную тенденцию в информационной безопасности, согласно которой большинство угроз реализуются посредством каналов взаимодействия, а также показывает важность защиты передаваемых данных.

Диаграмма потоков данных представляет собой веб-приложение, состоящее из двух программных контейнеров Docker (рис. 1). В данном варианте нотации окружности соответствуют процессам (process0, process1), прямоугольник – внешней сущности (user), а фигуры, состоящие из двух параллельных линий, – хранилищам данным (hostStorage, storage0).

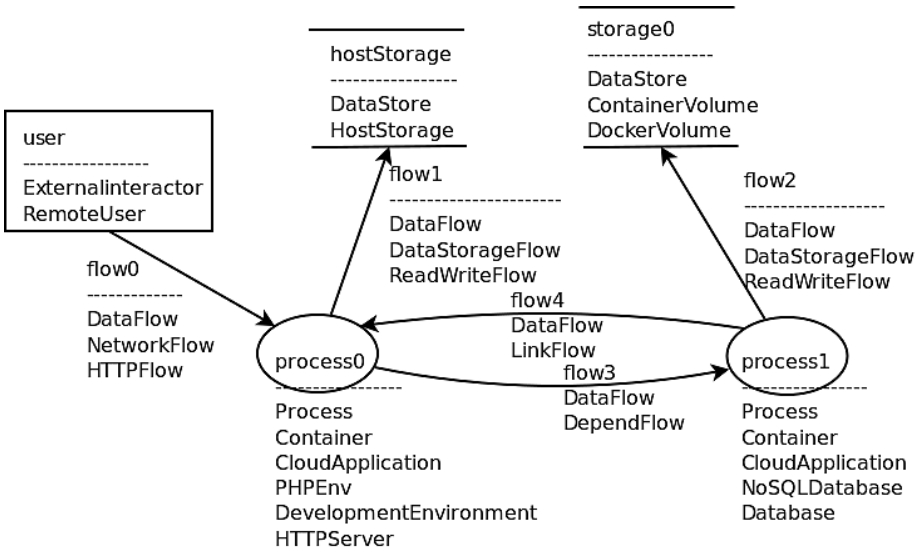


Рис. 1. Пример диаграммы потоков данных

Fig. 1. An example of Data Flow Diagram

Особенностью диаграммы, изображенной на рис. 1, является то, что она содержит предметно-ориентированные знания, т. е. информацию о функциях и реализациях компонентов и потоков. Так, flow0 представляет собой сетевой поток (NetworkFlow), в частности поток HTTP (HTTPFlow). Процесс process0 является средой исполнения (DevelopmentEnvironment) и включает интерпретатор языка программирования PHP (PHPEnv). В связи с тем что process0 является назначением потока HTTP, он может быть распознан как реализация сервера HTTP (HTTPServer).

С точки зрения моделирования угроз предметно-ориентированные знания позволяют рассматривать более конкретные угрозы компонентам или потокам. Так, с сетевым потоком можно ассоциировать некий высокоуровневый набор сетевых угроз, но уточнение о том, что поток ис-

пользует протокол HTTP, позволяет рассматривать угрозы, характерные для данного прикладного протокола, а также учитывать особенности организации обработки сообщений этого протокола клиентом (user) и сервером HTTP (process0).

Методика онтологического моделирования угроз. Для автоматизации моделирования угроз использована авторская методика, основанная на онтологическом подходе [16]. Методика позволяет, во-первых, внедрять технологии объектно-ориентированного проектирования в процесс моделирования угроз на основе концептов, объектных свойств и экземпляров, а во-вторых, автоматизировать анализ защищенности компьютерных систем на основе онтологий.

Онтологическое моделирование и графы знаний [17, 18] как средства построения интеллектуальных компьютерных систем [19, 20] активно используются для решения различных задач [21]. В частности, рассматриваемая методика реализует подход к прикладным онтологиям, наиболее близкий к сценарию семантического анализа [22], который подразумевает применение к знаниям функций автоматического логического вывода и семантической обработки, что позволяет создавать и использовать «новые» знания [23]. Отличительным признаком подхода онтологического моделирования, реализованного в методике, является использование экземпляров наравне с концептами не только в конечных моделях, но и в метамоделях.

Схема методики онтологического моделирования угроз показана на рис. 2. Основная идея методики заключается в том, что разработчик (архитектор) компьютерной системы описывает ее в виде диаграммы потоков данных. Программная система моделирования угроз способна семантически интерпретировать диаграмму (т. е. представить ее) в виде онтологии, а затем применить к данной семантической интерпретации процедуры автоматического логического вывода для нахождения списка релевантных угроз, соответствующих описанию системы.

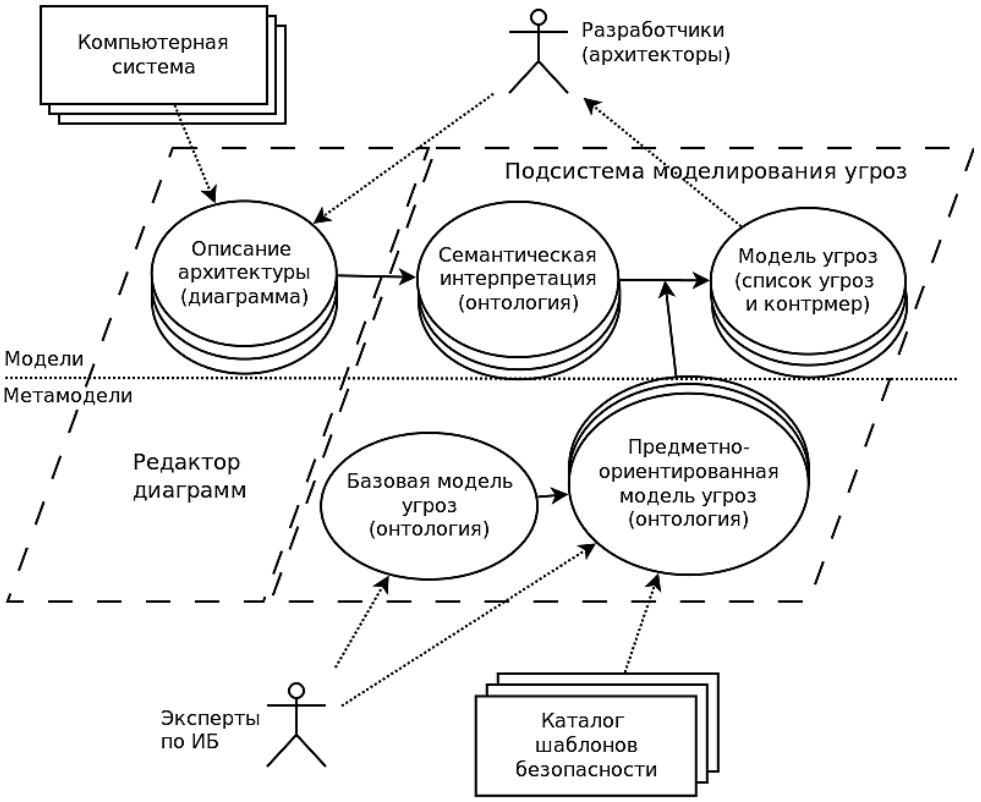


Рис. 2. Онтологическое моделирование угроз

Fig. 2. Ontological threat modelling

Для обеспечения автоматического логического вывода угроз в методике использованы два типа метамоделей (онтологий) в комбинации их с семантической интерпретацией диаграммы [16]:

– базовая онтологическая модель угроз, которая содержит концепты, объектные свойства и экземпляры, представляющие компоненты диаграмм, информационные потоки и угрозы;

– предметно-ориентированные модели угроз, каждая из которых описывает аспекты безопасности определенного типа компьютерных систем (облачных систем, систем интернета вещей, контейнерных систем и т. п.). Для построения предметно-ориентированной модели угроз необходимо, используя знания экспертов и различные источники информации, расширить базовую модель путем наследования ее концептов и создания необходимых подклассов архитектурных компонентов и экземпляров угроз, относящихся к данной предметной области.

Семантические интерпретации диаграмм (рис. 2), представленные как онтологии, а также описания систем и модели (перечни) угроз являются конечными моделями.

Метамодели и модели реализованы на языке веб-онтологий OWL (Web Ontology Language) [24, 25]. Язык OWL основан на дескрипционных логиках DL (Description Logics) [26, 27], используемых для управления знаниями; для OWL созданы редакторы онтологий (Protege), системы автоматического логического вывода (Fact++, Hermit, Pellet) и средства разработки (Java OWL API). Формат RDF (Resource Description Framework) может быть использован для представления знаний в виде графа знаний, состоящего из триплетов «объект – свойство – субъект».

Основная информация о методике и исходные файлы онтологий опубликованы в репозитории Github (URL: https://github.com/nets4geeks/OdTM).

Семантическая интерпретация диаграмм. Опишем компоненты диаграммы следующим образом: процессы как концепты Process, внешние сущности – ExternalInteractor, хранилища данных – DataStore. Взаимодействия компонентов описываются потоками данных (DataFlow) между компонентами. Потоки данных описываются через их источники и назначения, т. е. для некоторого потока можно утверждать, что он имеет источником (hasSource) и назначением (hasTarget) некоторый компонент. На рис. 3 показан пример простой диаграммы. Для формализации подобного примера может быть использован редактор онтологий Protege, при этом далее в тексте применяется смешанная терминология языка OWL и редактора Protege.

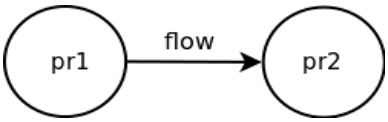


Рис. 3. Простая диаграмма потоков данных
Fig. 3. A simple data flow diagram

Используя концепты и объектные свойства, упомянутые выше, эту диаграмму можно описать на языке OWL набором утверждений, показанным в листинге 1.

Листинг 1.

```
Process (pr1)
Process (pr2)
DataFlow (flow)
hasSource (flow, pr1)
hasTarget (flow, pr2)
```

В листинге 1 отражается то, что считается исходными, явными (explicit) знаниями. Однако свойства базовой модели угроз и функции автоматического логического вывода позволяют получить дополнительные знания, называемые неявными (implicit). Например, модель содержит свойства «является источником» (isSourceOf) и «является назначением» (isTargetOf), которые описаны как обратные к свойствам hasSource и hasTarget соответственно. Это позволяет прави-

лам логического вывода автоматически установить, что процесс pr1 служит источником потока flow, а pr2 – его назначением (листинг 2).

Листинг 2.
isSourceOf (pr1, flow)
isTargetOf (pr2, flow)

Автоматический логический вывод угроз. Разработанная методика моделирования угроз предполагает использование автоматического логического вывода с включением предметно-ориентированных знаний для нахождения в онтологическом представлении диаграмм некоторых структурных шаблонов, которые могут быть ассоциированы с набором потенциальных угроз. В данной работе такие формальные конструкции называются семантическими шаблонами.

Например, пусть сервисы, которые обслуживают клиентов по протоколу HTTP, считаются небезопасными (так как HTTP передает данные в открытом виде). Семантический шаблон для этой угрозы неформально будет звучать как «процесс, который является назначением HTTP-потока». Формальное определение такого шаблона посредством утверждения OWL (эквивалентность – Equivalent to) будет иметь вид, показанный в листинге 3 (в листингах 3 и 4 используется нотация, применяемая для описания утверждений в Protege).

Листинг 3.
Process and (isTargetOf some HTTPFlow)

Если вернуться к диаграмме на рис. 3 и семантической интерпретации в листинге 1, то для соответствия процесса pr2 данному шаблону необходимо, чтобы поток flow был экземпляром концепта HTTPFlow. Это знание может быть передано системе явно или же следовать из соответствующей предметно-ориентированной модели угроз.

В листинге 4 показано выражение OWL (подкласс Subclass), которое позволяет сопоставить шаблон, приведенный в листинге 3, соответствующему экземпляру угрозы (insecureProcessThreat).

Листинг 4.
isAffectedBy value insecureProcessThreat

Описание определяемого класса (Defined Class), включающего два последних утверждения (листинги 3, 4), позволит посредством автоматического логического вывода обнаружить все процессы, которые небезопасно используют протокол HTTP, и сопоставить им экземпляр угрозы insecureProcessThreat.

Путем применения определяемых классов и необходимых экземпляров в разработанной базовой модели угроз реализован общепринятый подход к моделированию угроз STRIDE [1], который заключается в анализе каждого элемента диаграммы на возможность осуществления злоумышленником следующего набора действий: спуфинг (Spoofing), незаконное изменение (Tampering), отказуемость (Repudiation), раскрытие информации (Information Disclosure), отказ в обслуживании (Denial of Service), повышение привилегий (Elevation of Privilege). При этом считается [1], что процессы могут быть подвержены всем вышеперечисленным угрозам: хранилища данных – незаконному изменению, отказуемости, раскрытию информации и отказу в обслуживании; внешние сущности – спуфингу и отказу в обслуживании; потоки данных – раскрытию информации, незаконному изменению и отказу в обслуживании.

В табл. 1 показано, как подход STRIDE автоматизирован в базовой онтологической модели угроз. Столбцы ID и «Наименование» содержат идентификаторы и названия соответствующих угроз. В столбце «Семантический шаблон» перечислены элементы диаграммы, к которым могут быть применены угрозы.

Таблица 1
Подход STRIDE в базовой онтологической модели угроз
Table 1
The STRIDE approach in the Basic ontological threat model

ID	Наименование угрозы *Threat name*	Семантический шаблон *Semantic template*
Spoofing	Подмена (спуфинг)	Process, ExternalInteractor
Tampering	Незаконное изменение	Process, DataStore, DataFlow
Repudiation	Отказуемость	Process, DataStore
InformationDisclosure	Раскрытие информации	Process, DataStore, DataFlow
Denial of Service	Отказ в обслуживании	Process, DataStore, ExternalInteractor, DataFlow
Elevation of Privilege	Повышение привилегий	Process

Следует отметить, что для процессов, хранилищ данных и внешних сущностей ассоциация угрозы возможна при участии в каком-либо потоке. Семантические шаблоны подхода STRIDE реализованы в виде соответствующих определяемых классов.

2. Набор (датасет) семантических диаграмм контейнерных приложений. Одним из факторов, ограничивающих исследования по автоматизации моделирования угроз, является недостаток наборов данных (формальных описаний систем в виде диаграмм потоков данных), которые могли бы использоваться для оценки эффективности и корректности различных методик моделирования угроз [28]. Современные исследования оперируют всего лишь десятками диаграмм [28, 29].

В рамках настоящей работы для количественной оценки характеристик онтологического моделирования угроз был создан открытый набор (датасет) из 200 семантических диаграмм потоков данных на основе конфигураций реальных многоконтейнерных приложений. Диаграммы являются семантическими, потому что каждая из них представлена в машинно-читаемом виде как онтология OWL и граф знаний RDF.

В работе были использованы автоматические конфигурации Docker compose, которые являются декларативными описаниями в одном файле (docker-compose.yml) контейнерных приложений [30, 31], состоящих из нескольких сервисов (контейнеров). Соответствующие файлы были получены из публичных репозиториев (Github.com, Gitlab.com), а также из ряда корпоративных репозиториев. Для обеспечения приватности исходных данных созданный набор включает только «деперсонифицированные» артефакты – описания диаграмм потоков данных в виде онтологий и графов знаний. Также диаграммы представлены в формате YAML (YAML Ain't Markup Language),что может быть использовано для создания их графических представлений. Процесс создания диаграммы из файла docker-compose.yml показан на рис. 4.

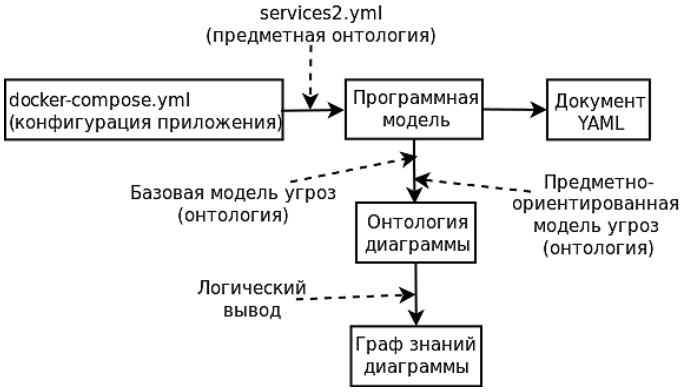


Рис. 4. Создание семантической диаграммы
Fig. 4. Creating a semantic diagram

Могут быть выделены следующие стадии данного процесса:

1. *Создание программной модели из файла docker-compose.yml*. Включает чтение исходного файла; создание экземпляров программных классов для сущностей, описанных в этом файле, а также классификацию сущностей в соответствии с предметной онтологией. Ниже показан пример файла docker-compose.yml.

Листинг 5.

```
services:
  web:
    image: php:8.0
    volumes:
      - ./app:/var/www/html
    depends_on:
      - mongodb
    ports:
      - 80:80
  mongodb:
    image: mongo:latest
    volumes:
      - dbdata:/data/db
    links:
      - web
```

В листинге 5 описаны два сервиса (web и mongodb), каждый из которых реализован как контейнер. Свойство image указывает на базовый образ контейнера (набор ПО, необходимый для работы сервиса): для первого контейнера это среда разработки PHP, для второго – нереляционная СУБД MongoDB. Описание каждого контейнера также содержит перечни внешних хранилищ (раздел volumes), опубликованных сетевых портов (ports) и взаимосвязи с другими контейнерами (depends_on и links). В системе Docker внешние данные могут быть сохранены как папки на хосте или в томе, обеспечиваемом контейнерной системой.

Программная модель подразумевает следующие трансформации сущностей для создания диаграмм:

– контейнеры считаются процессами (Process);

– внешние данные моделируются как хранилища данных (DataStore), причем папки на хосте принадлежат классу HostStorage, а тома – DockerVolume;

– для моделирования сетевых взаимодействий приложения с внешней средой создается сущность внешнего пользователя (RemoteUser).

Между сущностями создаются следующие потоки (DataFlow):

между контейнерами и хранилищами – потоки хранилищ данных (DataStorageFlow);

между удаленным пользователем и контейнерами (при наличии открытых портов) – сетевые потоки (NetworFlow);

между контейнерами (при наличии свойств depends_on и links) – потоки зависимости (DependFlow) и связи (LinkFlow).

На данном этапе ключевое значение имеет предметная онтология, которая используется для ассоциации сущностей с дополнительными классами и содержит следующие данные:

– категории используемых сервисов (image); например, mongodb – это нереляционная СУБД (NoSQLDatabase) и СУБД в общем (Database);

– типы используемых хранилищ; например, для mongodb данные хранятся в папке /data/db;

– типы опубликованных сервисов; например, если контейнер публикует порт 80, то он может быть распознан как сервер HTTP (HTTPServer), а поток к нему – как HTTP-поток (HTTPFlow).

Создание предметной онтологии является итеративным ручным процессом по мере добавления новых конфигураций. Предметная онтология для данного набора сохранена в формате YAML.

В листинге 6 представлен фрагмент предметной онтологии: поле images позволяет отнести контейнер к данной категории по базовому образу, а поля name и labels являются дополнительными классами, ассоциированными с процессом, представляющим контейнер на диаграмме.

Листинг 6.

```
 - name: NoSQLDatabase
   images:
     - mongodb
     - mongo
     - influxdb
     - zookeeper
     - couchdb
   labels:
     - Database
```

Концепты, описанные в предметной онтологии, могут быть использованы для формирования семантических шаблонов различных предметно-ориентированных моделей угроз.

Показанное ранее на рис. 1 графическое представление диаграммы получено из конфигурации Docker compose, приведенной в листинге 5, на основе предметной онтологии.

2. *Создание онтологии диаграммы*. Для этого необходимо сформировать новую онтологию и импортировать базовую онтологическую модель, затем добавить необходимые утверждения OWL относительно диаграммы, в частности:

– экземпляры процессов, хранилищ, а также экземпляр удаленного пользователя;

– ассоциации экземпляров и их концептов с использованием предметной онтологии;

– определения потоков через объектные свойства hasSource и hasTarget;

– ряд ассоциаций с концептами базовой модели угроз и предметно-ориентированных моделей угроз.

3. *Создание графа знаний диаграммы*. Онтология диаграммы, созданная на предыдущем этапе, содержит неполные (явные) знания о диаграмме. Два шага должны быть сделаны, чтобы расширить знания: во-первых, импортирована предметно-ориентированная модель угроз (см. разд. 3), во-вторых, выполнена процедура автоматического вывода для заполнения базы знаний дополнительными (неявными) знаниями. Заполненная онтология может быть сохранена в формате RDF, что позволяет использовать язык запросов SPARQL для получения необходимых фактов.

В рамках исследования был разработан набор утилит на языке программирования Java с помощью библиотеки OWL API, который использовался для обработки 200 файлов docker-compose.yml. В результате был получен датасет семантических диаграмм потоков данных, который опубликован в открытом доступе как репозиторий Github (URL: https://github.com/nets4geeks/DockerComposeDataset) в папке clear2, предметная онтология опубликована в файле services2.yml.

3. Построение предметно-ориентированных моделей для анализа защищенности. Для целей эксперимента была сформирована предметно-ориентированная модель угроз, которая включает две подмодели: общие угрозы облачных приложений и угрозы, специфичные для контейнерных приложений.

3.1. Модель угроз облачных компьютерных систем. В качестве перечня общих угроз облачным системам был использован авторский каталог шаблонов угроз облачных систем ACCTP (Academic Cloud Computing Threat Patterns), разработанный ранее [32]. Каталог агрегирует угрозы облачным системам, перечисленные в некоторых известных каталогах безопасности и научной литературе. Он реализован как онтология OWL и может быть автоматически конвертирован в соответствующую предметно-ориентированную модель угроз, в которой семантические шаблоны представлены в виде определяемых классов. Текстовая версия каталога доступна по ссылке URL: https://nets4geeks.github.io/acctp/catalog/.

Каталог ACCTP включает несколько профилей (архитектурный, операционный, инфраструктура как услуга) и ряд сценариев в рамках этих профилей [32]. Для эксперимента в настоящей работе использованы сценарии «Простое облачное приложение» и «Взаимодействие облачных приложений» архитектурного профиля ACCTP. В первом сценарии рассматривается взаимодействие удаленных пользователей с облачным приложением: в разрезе защищенности как пользователи могут влиять на облачное приложение, так и облачное приложение может влиять на пользователей. Во втором сценарии рассматривается взаимодействие облачных приложений друг с другом: в случае многокомпонентных приложений каждый компонент считается облачным приложением (например, веб-сервер взаимодействует с СУБД). Табл. 2 содержит перечень угроз, сформированный согласно этим сценариям. Столбцы ID и «Наименование угрозы» описывают идентификаторы и имена шаблонов угроз. Столбец «Семантический шаблон» содержит условное описание семантических шаблонов в виде потоков между концептами. В данном случае потоки являются направленными: первый концепт – клиент (источник), второй концепт – сервер (назначение), столбец «Цель» определяет цель угрозы (клиент или сервер).

Таблица 2
Шаблоны угроз облачным приложениям

Table 2
Threat patterns of cloud applications

ID	Наименование угрозы *Threat name*	Семантический шаблон *Semantic template*	Цель *Target*
AB01	Сбой облачного приложения	RemoteUser > CloudApplication CloudApplication > CloudApplication	Клиент
AB02	Потеря соединения с облачным приложением		
AC01	Вредоносный контент от облачного приложения	RemoteUser > CloudApplication	Клиент
AC02	Доступ к облачному приложению через незащищенную сеть		
AC04	Социальная инженерия против пользователя		
AD01	Ошибки аутентификации	RemoteUser > CloudApplication CloudApplication > CloudApplication	Сервер
AD02	Ошибки контроля доступа		
AD03	Утечка данных облачного приложения		
AD04	Потеря данных облачным приложением		
AD05	Потеря резервной копии облачного приложения		
AD06	Утечка резервной копии облачного приложения		
AD07	Потеря журнала событий облачного приложения		
AD08	Утечка журнала событий облачного приложения		
AE01	Отказ в обслуживании (DoS/DDoS)	RemoteUser > CloudApplication	Сервер
AE02	Экономический отказ в обслуживании (EDoS)		
AE03	Сетевые атаки на облачное приложение		
AE04	Неправомерное использование облачного приложения		
AE05	Блокировка разделяемых ресурсов облачного приложения		
AE06	Утечка учетных данных интерфейса администрирования		

Применение указанных шаблонов в эксперименте с контейнерными приложениями предполагает использование концепта удаленного пользователя (RemoteUser), а также трактовку всех процессов как облачных приложений (CloudApplication). Так как предметно-ориентированная модель угроз АССТР позволяет определять угрозы посредством автоматического логического вывода, то для ее использования достаточно при создании онтологии диаграммы ассоциировать экземпляры с соответствующими концептами (RemoteUser, CloudApplication) и импортировать онтологию модели угроз в онтологию диаграммы.

Ниже приведен пример реализации шаблона, который в онтологии OWL является определяемым классом, что позволяет автоматически классифицировать соответствующие экземпляры (листинг 7, выражение эквивалентности).

Листинг 7.

```
(CloudApplication) and (isTargetOf some (hasSource some RemoteUser))
```

Экземпляры угроз ассоциируются с компонентами диаграмм посредством специальных выражений, описывающих подклассы. Пример такой конструкции на языке OWL показан в листинге 8.

Листинг 8.

```
SubClassOf(:CloudApplicationAndIsTargetOfHasSourceRemoteUser ObjectHasValue
(<http://www.grsu.by/net/OdTMBaseThreatModel#isAffectedBy>
<http://www.grsu.by/net/ACCTP#threatAD01_BrokenAuthentication>))
```

3.2. Модель угроз контейнерных приложений. В настоящее время существует ряд неформальных моделей угроз и руководств по безопасности контейнеров, например методический документ ФСТЭК (URL: https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/trebovaniya-po-bezopasnosti-informatsii-utverzhdeny-prikazom-fstek-rossii-ot-4-iyulya-2022-g-n-118) и соответствующая база данных угроз (URL: https://bdu.fstec.ru), ряд корпоративных и общественных матриц угроз, например Microsoft (URL: https://www.microsoft.com/en-us/security/blog/2021/03/23/secure-containerized-environments-with-updated-threat-matrix-for-kubernetes/) и ATT&CK (URL: https://attack.mitre.org/matrices/enterprise/containers/). На основе этих источников в рамках настоящей работы был создан дополнительный профиль шаблонов угроз многокомпонентных контейнерных приложений для каталога АССТР. Профиль ориентирован, во-первых, на угрозы, возникающие при взаимодействии компонентов, во-вторых, на угрозы, которые не перекрывают угрозы архитектурного профиля.

Для описания соответствующих онтологических шаблонов в модель были добавлены следующие концепты:

контейнер (Container) – каждый процесс диаграммы;

хранилище хоста (HostStorage) – способ хранения данных контейнера на файловой системе хоста (устаревший подход);

том контейнера (ContainerVolume) – способ хранения данных контейнера посредством абстракций, создаваемых системой управления контейнерами;

сокет контейнера (ContainerSocket) – канал для чтения (записи) команд для контейнерной системы.

Кроме того, в профиле применяется концепт удаленного пользователя (RemoteUser).

Табл. 3 содержит созданный перечень шаблонов угроз контейнерных приложений. Все цели в табл. 3 имеют роль сервера, шаблоны EC05-EC07 содержат угрозы для потоков. В связи с тем что профиль является частью АССТР, подход к реализации и использованию шаблонов такой же, как описан в разд. 3.1.

Таблица 3
Шаблоны угроз контейнерных приложений
Table 3
Threat patterns of container applications

ID	Наименование угрозы *Threat name*	Семантический шаблон *Semantic template*
EA01	Использование хранилища хоста	Container > HostStorage
EA02	Использование сокета системы управления	Container > ContainerSocket
EB01	Доступ к тому в режиме записи	Container > HostStorage Container > ContainerVolume
EB02	Утечка данных с тома	
EB03	Повреждение данных, хранимых томом	
EC01	Использование средств управления контейнерами (CLI, API)	Container > Container RemoteUser > Container
EC02	Выход за границы контейнера (escape)	
EC03	Повышение привилегий контейнера	
EC04	Скомпрометированные токены доступа	
EC05	Утечка данных при передаче	Container > Container (flow) RemoteUser > Container (flow)
EC06	Повреждение данных при передаче	
EC07	Недоступность канала взаимодействия	
ED01	Доступ к интерфейсу управления контейнера (WEB)	RemoteUser > Container
ED02	Уязвимости опубликованных приложений	
ED03	Развертывание контейнера с вредоносным ПО	
ED04	Обнаружение контейнеров	

4. Экспериментальная верификация методики онтологического моделирования угроз. Важной характеристикой результатов моделирования угроз является количество потенциальных угроз, сопоставленных с элементами диаграмм [28]. В рамках проведенного эксперимента было выполнено сравнение количества угроз, полученных посредством общепринятого подхода STRIDE (табл. 1), с количеством угроз, полученных посредством предметно-ориентированных моделей шаблонов угроз (табл. 2 и 3), для семантических диаграмм из созданного датасета.

В эксперименте использованы базовая модель угроз, которая содержит шаблоны угроз STRIDE (табл. 1), и предметно-ориентированная модель угроз, которая включает архитектурную подмодель (профиль) шаблонов угроз облачных систем (табл. 2) и профиль шаблонов угроз контейнерных приложений (табл. 3). Для каждой семантической диаграммы датасета с помощью автоматического логического вывода и запросов SPARQL сформированы списки релевантных угроз, затем скриптами командного интерпретатора подсчитано количество предметно-ориентированных угроз (ПОУ) и количество угроз STRIDE. Далее для каждой диаграммы рассчитано значение показателя эффективности путем деления количества полученных ПОУ на количество угроз STRIDE.

Полученная в результате выборка показателей эффективности для 200 объектов датасета (рис. 5) подтверждает, что в 95 % случаев набор угроз, сформированный с использованием рассматриваемой методики автоматического моделирования угроз на основе онтологического подхода, равен или больше количества угроз, полученных на основе модели STRIDE. Среднее зна-

чение показателя эффективности равно 159 %. Результаты эксперимента показывают пригодность и эффективность предложенной методики для использования в задачах автоматизации моделирования угроз.

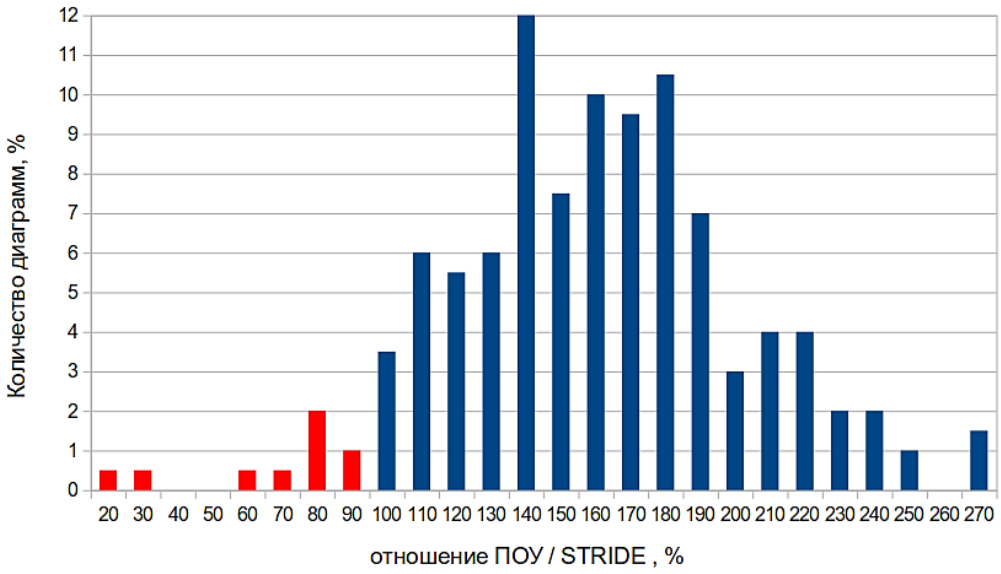


Рис. 5. Результаты эксперимента по анализу автоматического покрытия угрозами

Fig. 5. Experiment results of automatic threat covering

На рис. 5 изображена гистограмма распределения результатов эксперимента после их обработки. Ось X отображает отношения ПОУ к STRIDE в процентах (значения отношений предварительно округлены до одного знака после запятой), ось Y – количество диаграмм от общего числа диаграмм в процентах, соответствующих данным значениям отношения ПОУ/STRIDE.

Как преимущество предлагаемой методики следует рассматривать также то, что ПОУ, подобные использованным в данной работе, сразу ориентированы на анализ конкретных проблем защищенности, а подход STRIDE дает общие угрозы, которые сначала должны быть конкретизированы для данного приложения (т. е. для каждой категории STRIDE синтезированы новые частные угрозы) и только потом оценивается их применимость. Таким образом, использование ПОУ сокращает время анализа защищенности. Однако для получения численных оценок данного преимущества требуется проведение дополнительных исследований на массированных наборах данных с привлечением экспертного сообщества, в частности качественной оценки ПОУ в сравнении с угрозами STRIDE, что выходит за рамки данной работы.

Наличие 5 % случаев, в которых ПОУ меньше, чем угроз STRIDE, не может рассматриваться как недостаток методики, поскольку ПОУ могут применяться и детализироваться в соответствии с целями моделирования. Например, в данном эксперименте фактически были использованы два уровня абстракции: общие модели угроз облачных систем (архитектурный профиль ACCTP) и угрозы многокомпонентных приложений Docker (профиль контейнеров ACCTP). Очевидно, что добавление новых профилей (например, операционного профиля или профиля персональных данных) увеличило бы долю предметно-ориентированных моделей относительно STRIDE. В то же время, если цель моделирования угроз точно сформулирована, например защита персональных данных, то избыточные профили могут быть исключены, чтобы уменьшить количество угроз для рассмотрения.

Заключение. Данная работа посвящена экспериментальной верификации разработанной методики онтологического моделирования угроз на основе предметно-ориентированных моделей на примере контейнерных приложений. Для обеспечения эксперимента был разработан датасет из 200 семантических диаграмм на основе реальных облачных приложений, который может быть использован для различных исследований в области автоматизации моделирования угроз.

Каждая диаграмма была описана в терминах предметной онтологии и представлена в виде онтологии и графа знаний. Исходными данными набора являются реальные облачные многокомпонентные приложения (конфигурационные файлы Docker compose). Набор отличается от аналогов большим количеством элементов (сотни против десятков) и машинно-читаемым представлением.

Сформирована онтологическая предметно-ориентированная модель угроз, включающая соответствующие шаблоны угроз, что обеспечивает автоматический логический вывод угроз по онтологическим представлениям диаграмм. Предложенная модель угроз состоит из двух уровней: общие угрозы облачных приложений и угрозы, специфичные для контейнерных приложений.

Для разработанного датасета выполнен расчет показателей эффективности предложенной методики автоматического моделирования угроз и экспериментально показана ее адекватность, в частности то, что применение онтологической предметно-ориентированной модели угроз позволяет успешно выявлять угрозы, которые основаны на взаимодействии компонентов приложения. Для использованного датасета количество ПОУ для 95 % диаграмм в среднем на 59 % больше количества угроз, полученных посредством общепринятого подхода STRIDE.

Выявление новых критериев и методик оценки эффективности автоматизации анализа угроз является перспективным направлением научных исследований, требующих привлечения экспертного сообщества и представителей индустрии. Также научно-практический интерес представляет создание классификаций датасета, его расширение и сегментирование; разработка новых наборов данных для других типов приложений и их использование для исследований в области информационной безопасности, в том числе с применением методов машинного обучения.

Практическая значимость полученных результатов заключается в возможности их применения для автоматизации анализа защищенности компьютерных систем. В частности, предметно-ориентированные модели угроз могут быть научной основой для разработки средств автоматизации анализа состава, структуры и информационных потоков информационных систем в рамках процесса создания и аттестации систем защиты информационных систем.

Вклад авторов. *А. И. Бражук* сформировал датасет и необходимые онтологические модели, разработал программное обеспечение и выполнил расчет результатов эксперимента. *Е. В. Ализарович* участвовал в анализе и интерпретации эксперимента, осуществлял научное редактирование статьи.

Список использованных источников

1. Shostack, A. Experiences threat modeling at Microsoft / A. Shostack // MODSEC@ MoDELS. – 2008. – Vol. 2008. – 35 p.

2. A survey on threat-modeling techniques: protected objects and classification of threats / A. Konev [et al.] // Symmetry. – 2022. – Vol. 14, no. 3. – P. 549.

3. Макаревич, В. А. Анализ и моделирование угроз информационной безопасности предприятия на основе универсального шаблона / В. А. Макаревич, Е. А. Минюкович, К. С. Мулярчик // Журнал Белорусского государственного университета. Экономика. – 2021. – № 1. – С. 57–68.

4. Кочин, В. П. Проектирование и обеспечение безопасности интегрированных образовательных информационно-коммуникационных систем / В. П. Кочин, Ю. И. Воротницкий. – Минск : БГУ, 2022. – 168 с.

5. Verreydt, S. Expressive and systematic risk assessments with instance-centric threat models / S. Verreydt, D. Van Landuyt, W. Joosen // SAC'23: Proceedings of the 38th ACM/SIGAPP Symp. on Applied Computing, Tallinn, Estonia, 27–31 Mar. 2023. – Tallinn, 2023. – P. 1450–1457.

6. Язов, Ю. К. Логико-лингвистическое моделирование угроз безопасности информации в информационных системах / Ю. К. Язов, С. В. Соловьев, М. А. Тарелкин // Вопросы кибербезопасности. – 2022. – № 4(50). – С. 13–25.

7. Большаков, А. С. Управление информационной безопасностью персональных данных с использованием нечеткой логики / А. С. Большаков, А. И. Жила, А. В. Осин // Наукоемкие технологии в космических исследованиях Земли. – 2021. – Т. 13, № 4. – С. 37–47.

8. Миняев, А. А. Моделирование угроз безопасности информации в территориально-распределенных информационных системах / А. А. Миняев // Наукоемкие технологии в космических исследованиях Земли. – 2021. – Т. 13, № 2. – С. 52–65.

9. Васильев, В. И. Оценка актуальных угроз безопасности информации с помощью технологии трансформеров / В. И. Васильев, А. М. Вульфин, Н. В. Кучкарова // Вопросы кибербезопасности. – 2022. – № 2(48). – С. 27–38.

10. Массель, Л. В. Семантическое моделирование при построении цифровых двойников энергетических объектов и систем / Л. В. Массель, А. Г. Массель // Онтология проектирования. – 2023. – Т. 13, № 1(47). – С. 44–54.

11. Методика оценки актуальных угроз и уязвимостей на основе технологий когнитивного моделирования и Text Mining / В. И. Васильев [и др.] // Системы управления, связи и безопасности. – 2021. – № 3. – С. 110–134.

12. Kourbatski, A. Semantic aspects of the experts' communication problem in relation to the conceptual design of complex systems / A. Kourbatski, K. Mulyarchik // Open Semantic Technologies for Intelligent Systems: 11th Intern. Conf., OSTIS 2021. Communications in Computer and Information Science. – Cham : Springer, 2022. – Vol. 1625. – P. 77–88.

13. Касумов, В. А. Модель и метод определения оптимальной структуры системы обеспечения безопасности для критической информационной инфраструктуры / В. А. Касумов, Д. И. Мамедов // Доклады БГУИР. – 2023. – Т. 21, № 2. – С. 95–103.

14. Никитина, И. С. Использование диаграмм потоков данных для представления предметной области / И. С. Никитина // Вестник современных исследований. – 2018. – № 7.1. – С. 324–328.

15. Давлетшина, Л. А. Моделирование информационных потоков IT-компании на основе методологии диаграммы потоков данных / Л. А. Давлетшина, И. К. Будникова // Информационные технологии в строительных, социальных и экономических системах. – 2021. – № 1. – С. 87–91.

16. Brazhuk, A. Framework for ontology-driven threat modelling of modern computer system / A. Brazhuk, E. Olizarovich // Intern. J. of Open Information Technologies. – 2020. – Vol. 8, no. 2. – P. 14–20.

17. Гаврилова, Т. А. Онтологический инжиниринг от истории к практическому формированию / Т. А. Гаврилова // Когнитивные исследования ; под ред. В. Д. Соловьева. – 2022. – № 2. – С. 293–307.

18. Грибова, В. В. Онтологические инфраструктуры для решения интеллектуальных задач / В. В. Грибова, Е. А. Шалфеева // Интегрированные модели и мягкие вычисления в искусственном интеллекте (ИММВ-2021) : сб. науч. тр. X Междунар. науч.-техн. конф., Коломна, 17–20 мая 2021 г. – Смоленск : Универсум, 2021. – Т. 1. – С. 68–77.

19. Городецкий, В. И. Искусственный интеллект: метафора, наука и информационная технология / В. И. Городецкий, Р. М. Юсупов // Мехатроника, автоматизация, управление. – 2020. – Т. 21, № 5. – С. 282–294.

20. Голенков, В. В. Основные направления развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии / В. В. Голенков, Н. А. Гулякина, Д. В. Шункевич // Science and innovation. – 2023. – Т. 2, special iss. 3. – С. 267–280.

21. Интеллектуальный анализ данных и облачные вычисления / М. М. Татур [и др.] // Доклады БГУИР. – 2019. – № 6. – С. 62–71.

22. Гаврилова, Т. А. Инженерия знаний. Модели и методы / Т. А. Гаврилова, Д. В. Кудрявцев, Д. И. Муромцев. – СПб. : Лань, 2016. – 324 с.

23. Милько, Д. С. База знаний экспертной системы оценки угроз безопасности информации / Д. С. Милько, А. В. Данеев, А. Л. Горбылев // Докл. Томского гос. ун-та систем управления и радиоэлектроники. – 2022. – Т. 25, № 1. – С. 61–69.

24. Разин, В. В. Метод принятия решений на основе анализа ситуаций и семантических технологий / В. В. Разин, А. Ф. Тузовский // Изв. Томского политехн. ун-та. Инжиниринг георесурсов. – 2012. – Т. 321, № 5. – С. 188–193.

25. Буракова, Е. Е. Языки описания онтологий для технических предметных областей / Е. Е. Буракова, Н. М. Боргест, М. Д. Коровин // Вестник Самарского гос. аэрокосмического ун-та им. академика С. П. Королёва (Национального исследовательского ун-та). – 2014. – № 3(45). – С. 144–158.

26. Осипов, Г. Методы искусственного интеллекта / Г. Осипов. – М. : ФИЗМАТЛИТ, 2011. – 296 с.

27. Маторин, С. И. Системно-объектный детерминантный анализ. Построение таксономии предметной области / С. И. Маторин, В. В. Михелев // Искусственный интеллект и принятие решений. – 2021. – № 1. – С. 15–24.

28. Automating the early detection of security design flaws / K. Tuma [et al.] // Proceedings of the 23rd ACM/IEEE Intern. Conf. on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 16–23 Oct. 2020. – Canada, 2020. – P. 332–342.

29. Adopting threat modelling in agile software development projects / K. Bernsmed [et al.] // J. of Systems and Software. – 2022. – Vol. 183. – P. 111090.

30. Architecture of cloud telecommunication network monitoring platform based on knowledge graphs / K. Krinkin [et al.] // 2021 30th Conf. of Open Innovations Association FRUCT 2021, Oulu, Finland, 27–29 Oct. 2021. – Oulu, 2021. – P. 107–114.

31. Забавский, В. В. Уязвимости в технологии контейнеризации docker / В. В. Забавский, Т. В. Борботько // Управление информационными ресурсами : материалы XVII Междунар. науч.-практ. конф., Минск, 12 марта 2021 г. – Минск : Академия управления при Президенте Республики Беларусь, 2021. – С. 208–209.

32. Brazhuk, A. Threat modeling of cloud systems with ontological security pattern catalog / A. Brazhuk // Intern. J. of Open Information Technologies. – 2021. – Vol. 9, no. 5. – P. 36–41.

References

1. Shostack A. Experiences threat modeling at Microsoft. *MODSEC@ MoDELS*, 2008, vol. 2008, 35 p.

2. Konev A., Shelupanov A., Kataev M., Ageeva V., Nabieva A. A survey on threat-modeling techniques: protected objects and classification of threats. *Symmetry*, 2022, vol. 14, no. 3, p. 549.

3. Makarevich V. A., Miniukovich K. A., Mulyarchik K. S. *Organisation's information security threat analysis and modelling based on a universal canvas.* Zhurnal Belorusskogo gosudarstvennogo universiteta. Ekonomika [*Journal of the Belarusian State University. Economics*], 2021, no. 1, pp. 57–68 (In Russ.).

4. Kochin V. P., Vorotnitsky U. I. Proektirovanie i obespechenie bezopasnosti integrirovannyh obrazovatel'nyh informacionno-kommunikacionnyh system. *Design and Security of Integrated Educational Information and Communication Systems*. Minsk, Belarusian State University, 2022, 168 p. (In Russ.).

5. Verreydt S., Van Landuyt D., Joosen W. Expressive and systematic risk assessments with instance-centric threat models. *SAC'23: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, Tallinn, Estonia, 27–31 March 2023*. Tallinn, 2023, pp. 1450–1457.

6. Yazov Yu. K., Soloviev S. V., Tarelkin M. A. *Logical-linguistic modeling of security threats information in information systems*. Voprosy kiberbezopasnosti [*Cybersecurity Issues*], 2022, no. 4(50), pp. 13–25 (In Russ.).

7. Bolshakov A. S., Zhila A. I., Osin A. V. *Fuzzy logic data protection management*. Naukoemkie tekhnologii v kosmicheskih issledovaniyah Zemli [*High Tech in Earth Space Research*], 2021, vol. 13, no. 4, pp. 37–47 (In Russ.).

8. Minyaev A. A. *Modeling information security threats in territorial-distributed information systems.* Naukoemkie tekhnologii v kosmicheskih issledovaniyah Zemli [*High Tech in Earth Space Research*], 2021, vol. 13, no. 2, pp. 52–65 (In Russ.).

9. Vasilyev V. I., Vulfin A. M., Kuchkarova N. V. *Assessment of current threats to information security using transformer technology*. Voprosy kiberbezopasnosti [*Cybersecurity Issues*], 2022, no. 2(48), pp. 27–38 (In Russ.).

10. Massel L. V., Massel A. G. *Semantic modeling in the construction of digital twins of energy objects and systems*. Ontologia proektirovania [*Ontology of Design*], 2023, vol. 13, no. 1(47), pp. 44–54 (In Russ.).

11. Vasilyev V. I., Vulfin A. M., Kirillova A. D., Kuchkarova N. V. *Methodology for assessing current threats and vulnerabilities based on cognitive modeling technologies and text minin*g. Sistemy upravlenia, sviazi i bezopasnosti [*Systems of Control, Communication and Security*], 2021, no. 3, pp. 110–134 (In Russ.).

12. Kourbatski A., Mulyarchik K. Semantic aspects of the experts' communication problem in relation to the conceptual design of complex systems. *Open Semantic Technologies for Intelligent Systems: 11th International Conference, OSTIS 2021. Communications in Computer and Information Science*. Cham, Springer, 2022, vol. 1625, pp. 77–88.

13. Kasumov V. A., Mamedov D. I. *Model and method for determining the optimal structure of the security system for critical information infrastructure*. Doklady Belorusskogo gosudarstvennogo universiteta informatiki i radioèlektroniki [*Reports of the Belarusian State University of Informatics and Radioelectronics*], 2023, no. 21(2), pp. 95–103 (In Russ.).

14. Nikitina I. S. *The use of data flow diagrams for representation of subject area*. Vestnik sovremennyx issledovanii [*Bulletin of Modern Research*], 2018, no 7.1, pp. 324–328 (In Russ.).

15. Davletshina L. A., Budnikova I. K. *Modeling of IT company information flows based on data flow diagrams methodology*. Informacionnye texnologii v stroitelnych, socialnych i ekonomicheskich sistemah [*Information Technologies in Construction, Social and Economic Systems*], 2021, no. 1, pp. 87–91 (In Russ.).

16. Brazhuk A., Olizarovich E. Framework for ontology-driven threat modelling of modern computer system. *International Journal of Open Information Technologies*, 2020, vol. 8, no. 2, pp. 14–20.

17. Gavrilova T. A. *Ontological engineering from history to practical use*. Kognitivnye issledovania [*Cognitive Research*]. In V. D. Soloviev (ed.), 2022, no. 2, pp. 293–307 (In Russ.).

18. Gribova V. V., Shalfeeva E. A. *Ontological infrastructures for solving intellectual tasks*. Integrirovannye modeli i mjagkie vychislenija v iskusstvennom intellekte (IMMV-2021) : sbornik nauchnyh trudov X Mezhdunarodnoj nauchno-tehnicheskoj konferencii, Kolomna, 17–20 maja 2021 g. [*Integrated Models and Soft Computing in Artificial Intelligence (IMMV-2021) : Collection of Scientific Papers of the X International Scientific and Technical Conference, Kolomna, 17–20 May 2021*]. Smolensk, Universum, 2021, vol. 1, pp. 68–77 (In Russ.).

19. Gorodetsky V. I., Yusupov R. M. *Artificial intelligence: metaphor, science and information technology*. Mekhatronika, avtomatizatsiya, upravlenie [*Mechatronics, Automation, Control*], 2020, no. 21(5), pp. 282–294 (In Russ.).

20. Golenkov V. V., Guliakina N. A., Shunkevich D. V. Main directions of development of intelligent computer systems of new generation and appropriate technology. *Science and Innovation*, 2023, vol. 2, special iss. 3, pp. 267–280 (In Russ.).

21. Tatur M. M., Lukashevich M. M., Pertsev D. Y., Iskra N. A. *Intelligent data analysis and cloud computing.* Doklady Belorusskogo gosudarstvennogo universiteta informatiki i radioèlektroniki [*Reports of the Belarusian State University of Informatics and Radioelectronics*], 2019, no. 6, pp. 62–71 (In Russ.).

22. Gavrilova T. A., Kudrjavcev D. V., Muromcev D. I. Inzhenerija znanij. Modeli i metody. *Knowledge Engineering. Models and Methods*, Saint Petersburg, Lan', 2016, 324 p. (In Russ.).

23. Milko D. S., Daneev A. V., Gorbylev A. L. *Knowledge base of the expert system for cyber security threat modeling*. Doklady Tomskogo gosudarstvennogo universiteta sistem upravlenija i radiojelektroniki [*Reports of Tomsk State University of Control Systems and Radioelectronics*], 2022, vol. 25, no. 1, pp. 61–69 (In Russ).

24. Razin V. V., Tuzovsky A. F. *Decision-making method based on situation analysis and semantic technologies*. Izvestija Tomskogo politehnicheskogo universiteta. Inzhiniring georesursov [*News of Tomsk Polytechnic University. Georesources Engineering*], 2012, vol. 321, no. 5, pp. 188–193 (In Russ.).

25. Burakova E. E., Borgest N. M., Korovin M. D. *Ontology description languages for high-tech fields of applied engineering.* Vestnik Samarskogo gosudarstvennogo ajerokosmicheskogo universiteta im. akademika S. P. Koroljova (Nacional'nogo issledovatel'skogo universiteta) [*Bulletin of Samara State Aerospace University named after Academician S. P. Korolev (National Research University)*], 2014, no. 3(45), pp. 144–158 (In Russ.).

26. Osipov G. Metody iskusstvennogo intellekta. *Artificial Intelligence Methods*. Moscow, FIZMATLIT, 2011, 296 p. (In Russ.).

27. Matorin S. I., Mikhelev V. V. *System-object determinant analysis. Partitive classification using the formal-semantic normative system*. Iskusstvennyj intellekt i prinjatie reshenij [*Artificial Intelligence and Decision Making*], 2021, no. 1, pp. 15–24 (In Russ).

28. Tuma K., Sion L., Scandariato R., Yskout K. Automating the early detection of security design flaws. *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 16–23 October 2020*. Canada, 2020, pp. 332–342.

29. Bernsmed K., Cruzes D. S., Jaatun M. G., Iovan M. Adopting threat modelling in agile software development projects. *Journal of Systems and Software*, 2022, vol. 183, p. 111090.

30. Krinkin K., Kulikov I., Vodyaho A., Zhukova N. Architecture of cloud telecommunication network monitoring platform based on knowledge graphs. *2021 30th Conference of Open Innovations Association FRUCT 2021, Oulu, Finland, 27–29 October 2021*. Oulu, 2021, pp. 107–114.

31. Zabavskii V. V., Borbotko T. V. *Vulnerabilities in docker container technology*. Upravlenie informacionnymi resursami : materialy XVII Mezhdunarodnoj nauchno-prakticheskoj konferencii, Minsk, 12 marta 2021 g. [*Information Resource Management : Materials of the XVII International Scientific and Practical Conference, Minsk, 12 March 2021*]. Minsk, Akademija upravlenija pri Prezidente Respubliki Belarus', 2021, pp. 208–209 (In. Russ.).

32. Brazhuk A. Threat modeling of cloud systems with ontological security pattern catalog. *International Journal of Open Information Technologies*, 2021, vol. 9, no. 5, pp. 36–41.

Информация об авторах

Бражук Андрей Иосифович, магистр естественных наук, ведущий инженер-программист Информационно-аналитического центра Гродненского государственного университета имени Янки Купалы.
E-mail: brazhuk@grsu.by

Олизарович Евгений Владимирович, кандидат технических наук, доцент, начальник Информационно-аналитического центра Гродненского государственного университета имени Янки Купалы.
E-mail: e.olizarovich@grsu.by

Information about the authors

Andrei I. Brazhuk, M. Sc., Lead Software Engineer at the Information and Analytical Center, Yanka Kupala State University of Grodno.
E-mail: brazhuk@grsu.by

Evgeny V. Olizarovich, Ph. D. (Eng.), Assoc. Prof., Head of the Information and Analytical Center, Yanka Kupala State University of Grodno.
E-mail: e.olizarovich@grsu.by

УДК 811.161.3'373.423.1'322
https://doi.org/10.37661/1816-0301-2023-20-4-87-100

Арыгінальны артыкул
Original Paper

Мадэль аўтаматызаванай ідэнтыфікацыі амографаў для беларускай мовы

Ю. С. Гецэвіч, Я. С. Зяноўка✉, Д. І. Латышэвіч, А. А. Бакуновіч, А. Я. Драгун, М. А. Казлова

Аб'яднаны інстытут праблем інфарматыкі
Нацыянальнай акадэміі навук Беларусі,
вул. Сурганава, 6, Мінск, 220012, Беларусь
✉*E-mail: evgeniakacan@gmail.com*

Анатацыя
Мэты. Мэтай працы з'яўляецца апісанне прататыпнай сістэмы для аўтаматызаванага здымання аманіміі ў электронных тэкстах на беларускай і рускай мовах. Гэта звязана з актуальнай праблемай аўтаматычнай апрацоўкі тэкстаў на марфалагічным узроўні, працэс якой ускладняецца флектыўнасцю беларускай мовы з разнастайнай і багатай сістэмай марфалагічных характарыстык часцін мовы.
Метады. У працы выкарыстоўваюцца правілавыя метады ідэнтыфікацыі амаграфіі і метады, заснаваныя на ведах.
Вынікі. Прапанаваны метады і падыходы для праектавання сістэм аўтаматычнага вызначэння амографаў. Падрабязна прадстаўлены метад, заснаваны на ведах, на аснове якога распрацаваны пакрокавы алгарытм ідэнтыфікацыі амографаў і рэалізаваны эфектыўны і хуткадзейны прататып для іх здымання на рускай і беларускай мовах.
Заключэнне. Прадстаўлены працоўны прататып пошуку амографаў, які з'яўляецца першым рэсурсам па здыманні шматзначнасці для беларускай мовы ў адкрытым доступе.

Ключавыя словы: аманімія, здыманне аманіміі, шматзначнасць, аўтаматычная апрацоўка электронных тэкстаў, беларуская мова, слоўнік

Для цытавання. Мадэль аўтаматызаванай ідэнтыфікацыі амографаў для беларускай мовы / Ю. С. Гецэвіч [і інш.] // Информатика. – 2023. − Т. 20, № 4. – С. 87–100.
https://doi.org/10.37661/1816-0301-2023-20-4-87-100

Канфлікт інтарэсаў. Аўтары заяўляюць аб адсутнасці канфлікту інтарэсаў.

Паступіла ў рэдакцыю | Received 22.08.2023
Падпісана ў друк | Accepted 01.11.2023
Апублікавана | Published 29.12.2023

A model of homographs automatic identification for the Belarusian language

Yuras S. Hetsevich, Yauheniya S. Zianouka✉, David I. Latyshevich, Andrey A. Bakunovich, Anastasia Ya. Drahun, Margarita A. Kazlova

The United Institute of Informatics Problems
of the National Academy of Sciences of Belarus,
st. Surganova, 6, Minsk, 220012, Belarus
✉*E-mail: evgeniakacan@gmail.com*

Abstract
Objectives. A prototype system for automated removal of homonyms in Belarusian and Russian electronic texts is described. This is due to the urgent problem of automatic text processing at the morphological level, the process of which is complicated by the inflection of the Belarusian language with a diverse and rich system of morphological characteristics of parts of speech.
Methods. The work uses regular homographs identification methods and knowledge-based methods.
Results. Methods and approaches for designing systems for automatic detection of homographs are proposed. An algorithm for identifying homographs on the basis of knowledge-based method has been developed. An effective and fast-acting prototype for their removal in Russian and Belarusian has been implemented.
Conclusion. A working prototype of the homograph search is presented, which is the first resource for removing ambiguity for the Belarusian language in open access.

Keywords: homonymy, removal of homonyms, ambiguity, automatic processing of electronic texts, the Belarusian language, dictionary

For citation. Hetsevich Yu. S., Zianouka Ya. S., Latyshevich D. I., Bakunovich A. A., Drahun A. Ya., Kazlova M. A. *A model of homographs automatic identification for the Belarusian language*. Informatika [*Informatics*], 2023, vol. 20, no. 4, pp. 87−100 (In Russ.). https://doi.org/10.37661/1816-0301-2023-20-4-87-100

Conflict of interest. The authors declare of no conflict of interest.

Уводзіны. Натуральная мова ўяўляе сабой вялікую адкрытую шматузроўневую сістэму знакаў, якая ўзнікла для абмену інфармацыяй у працэсе практычнай дзейнасці чалавека. Складанасць яе мадэлявання абумоўлена бесперапыннымі зменамі ў сувязі з гэтай дзейнасцю. Тэкст на натуральнай мове складаецца з асобных элементаў, якія ў сваім адзінстве аб'яднаны рознымі тыпамі лексічнай, граматычнай і лагічнай сувязяў. Існуюць разнастайныя спосабы члянення тэксту на адзінкі. Як вынік, немагчыма распрацаваць адзіную фармальную мадэль канкрэтнай натуральнай мовы і пабудаваць адпаведны лінгвістычны працэсар. Патрабуюцца пастаяннае папаўненне ведаў на ўсіх яе ўзроўнях і карэкцыя існуючых мадэляў. Адной з самых вялікіх складанасцяў пры апрацоўцы тэкстаў на натуральнай мове з'яўляецца двухзначнасць яе адзінак, што выяўляецца ў з'явах шматзначнасці, сінаніміі і асабліва аманіміі. Праблема аманіміі яскрава праяўляецца ў прыкладных навуках. Яна закранае такія практычныя напрамкі, як аўтаматычны пераклад, аўтаматычнае рэферыраванне тэкстаў, стварэнне рознага роду інтэлектуальных сістэм, складанне слоўнікаў [1].

Амонімы (ад грэч. *homos* – аднолькавы, *onyma* – імя) – словы, якія аднолькава гучаць і пішуцца, але маюць розныя, не звязаныя паміж сабой значэнні (напрыклад, «тур» – першабытны дзікі бык, «тур» – этап адбору і «тур» – паездка па розных мясцінах) [2]. Амонімы складаюць значны пласт лексікі (толькі амографаў у беларускай мове налічваецца каля 4000) і падзяляюцца на наступныя групы. Лексічныя – словы, якія аднолькава пішуцца і гучаць, аднак не маюць агульных элементаў сэнсу. У марфалагічных амонімах супадаюць формы аднаго і таго ж слова (лексемы), а лексіка-марфалагічныя амонімы (найбольш часты варыянт) узнікаюць пры аднолькавых словаформах дзвюх розных лексем. Сінтаксічныя амонімы апісваюць неадназначнасць сінтаксічнай структуры, што прыводзіць да розных інтэрпрэтацый. У галіне аўтаматычнай апрацоўкі мовы адной з найскладаных праблем выступае вызначэнне амонімаў для ліквідацыі неадназначнасці ў сэнсе, выпраўлення памылак і разумення адценняў у значэнні слоў.

Амографы (ад грэч. *γμός* – аднолькавы, *γράφω* – пішу) – словы, якія супадаюць у напісанні, але адрозніваюцца ў вымаўленні (у беларускай мове часцей за ўсё з-за адрозненняў у націску (напрыклад, *па+ра* і *пара+*, *му+зыка* і *музы+ка*). Да амографаў могуць адносіцца словы, якія маюць рознае значэнне, а таксама розныя формы аднаго і таго ж слова. Іх здыманне дазваляе павысіць дакладнасць і якасць апрацоўкі тэкставых даных. Для аўтаматызаванага вырашэння неадназначнасці слоў, у прыватнасці амографаў, выкарыстоўваюцца такія падыходы, як дэтэрмінаваныя правілы, якія працуюць на аснове лексічных і граматычных даных; базы ведаў аб навакольным свеце і анталогіі, якія даюць магчымасць улічваць экстралінгвістычныя даныя; імавернасныя аналізатары, якія ўлічваюць статыстычныя даныя мовы [3–6]. Кожны з прыведзеных падыходаў мае свае абмежаванні па эфектыўнасці і не дае вынікаў на самым высокім узроўні. Статыстычныя алгарытмы здымаюць аманімію на этапе марфалагічнага аналізу тэксту з выкарыстаннем статыстыкі ўжывання граматычных прыкмет слоў з карпусоў тэкстаў, размечаных уручную [7]. Дадзеныя метады часцей за ўсё прымяняюцца для аналітычных моў (напрыклад, англійскай), у якіх граматычныя адносіны маюць тэндэнцыю да перадачы ў асноўным праз сінтаксіс, асобныя службовыя словы (прыназоўнікі, мадальныя дзеясловы і г. д.), фіксаваны парадак слоў, кантэкст і (або) інтанацыйныя варыяцыі, а не праз словазмяненне з дапамогай залежных марфем (канчаткаў, суфіксаў, прыставак і г. д.). Разнастайнасць аманіміі ў беларускай і рускай мовах тлумачыцца наяўнасцю флексіі – фармантаў, якія спалучаюць адразу некалькі значэнняў, што і выклікае шматзначнасць і неаднароднасць слоў. Менавіта таму, што беларуская і руская мовы характарызуюцца адвольным парадкам слоў у сказе, узнікае складанасць прымянення матэматычных мадэлей вызначэння аманіміі. У сувязі з гэтым для дадзеных моў часцей за ўсё выкарыстоўваюць метады, заснаваныя на правілах.

У сучасных даследаваннях апісваюцца чатыры асноўныя метады здымання шматзначнасці:

1) метады, заснаваныя на ведах (dictionary- і knowledge-based methods), пераважна выкарыстоўваюць слоўнікі, тэзаўрус, лексікаграфічныя базы даных [8, с. 110]. Яны грунтуюцца на гіпотэзе, што словы, якія знаходзяцца побач у тэксце, звязаны адно з адным і гэтую сувязь можна назіраць у азначэннях слоў і іх значэннях. Два (ці больш) словы могуць аказацца блізкімі, калі ў абодвух з іх будзе выяўлена пара значэнняў з найбольшым перасячэннем слоў у іх азначэннях у слоўніку [9–12];

2) метады навучання з настаўнікам (supervised methods) выкарыстоўваюць размечаныя карпусы тэкстаў для трэніроўкі класіфікатараў [1]. Яны грунтуюцца на наступным меркаванні: кантэкст слова, што разглядаецца, дае дастатковую інфармацыю для таго, каб вылічыць, у якім значэнні ў дадзеным выпадку яно ўжываецца (а значыць, веды, атрыманыя з слоўнікаў і тэзаўрусаў, выдаляюцца як лішнія). Усе мадэлі навучання з настаўнікам прымяняліся да праблемы WSD (бел. вырашэнне лексічнай мнагазначнасці), у тым ліку звязаныя з імі тэхнікі, такія як выбар пераменных, аптымізацыя параметраў і змешаныя мадэлі (англ. ensemble learning);

3) метады частковага навучання з настаўнікам (minimally-supervised methods) базіруюцца на другасных ведах, такіх як вызначэнне тэрмінаў у тлумачэннях слоў ці выраўнаваны двухмоўны корпус [4];

4) метады навучання без настаўніка (unsupervised methods) не прадугледжваюць выкарыстанне якіх-небудзь знешніх даных і працуюць толькі з неанатаванымі карпусамі (raw unannotated corpora). Таксама яны вядомы пад тэрмінам кластарызацыі і распазнавання сэнсу слоў [13, 14]. Асноўная ідэя гэтага метаду заключаецца ў тым, што «падобныя значэнні сустракаюцца ў падобных кантэкстах» і такім чынам яны могуць быць выняты з тэксту з дапамогай кластарызацыі з ужываннем некаторай меры падабенства кантэкстаў. Таму новыя кантэксты могуць быць аднесены да аднаго з бліжэйшых кластараў.

Існуюць іншыя метады, якія адрозніваюцца ад вышэйпералічаных прынцыпаў: вызначэнне дамінантнага значэння слова (Determining Word Sense Dominance); вырашэнне, заснаванае на тэмах корпуса (Domain-Driven Disambiguation); WSD, якое выкарыстоўвае крос-моўныя даныя (Cross-Lingual Evidence) [1].

Электронныя лінгвістычныя рэсурсы для аўтаматызаванага здымання беларускамоўнай амаграфіі. Агляд наяўных навукова-папулярных крыніц і разнастайных даследаванняў сведчыць аб вялікай зацікаўленасці навукоўцаў праблемай аўтаматычнага здымання

аманіміі. Гэта прыведзена ў працах [1–14]. Аднак у адкрытым доступе адсутнічае інфармацыя аб прыкладной рэалізацыі праграмных прадуктаў здымання аманіміі для беларускай мовы. На падставе аналізу праблемнага поля супрацоўнікамі лабараторыі распазнавання і сінтэзу маўлення АІПІ НАН Беларусі (https://ssrlab.by/) была распрацавана мадэль аўтаматызаванай сістэмы па вызначэнні амографаў, двухсэнсоўных слоў у тэксце беларускай літаратурнай мовы [15]. Асноўны кірунак дзейнасці лабараторыі – высакаякасны сінтэз маўлення па тэксце, які заключаецца ў шматэтапным пераўтварэнні тэксту ў маўленне. Немалаважнай задачай з'яўляецца перапрацоўка электроннага тэксту, якая, акрамя этапаў ачысткі тэксту, дэшыфроўкі лікаў, абрэвіятур, замежных слоў і карэкціроўкі «ё», уключае вызначэнне амонімаў [16]. Для вырашэння праблемы здымання шматзначнасці слоў быў распрацаваны самастойны прататып у форме вэб-сэрвіса *«Ідэнтыфікатар амографаў»*, які даступны для вольнага выкарыстання ў Інтэрнэце па адрасе https://corpus.by/HomographIdentifier/?lang=be [17]. Для яго стварэння аўтарамі даследавання быў выкарыстаны метад, заснаваны на ведах (dictionary-based method). Дадзены падыход заснаваны на вялікім наборы слоўнікаў, згодна з якімі памылковыя значэнні шматзначных слоў выключаюцца з улікам іх значэнняў, што параўноўваюцца паміж сабой.

На ўваход сэрвісу падаецца электронны тэкст. Па выніках апрацоўкі карыстальнік атрымлівае спіс знойдзеных у тэксце амографаў з іх падрабязнымі данымі і, згодна сэнсу апрацаванага тэксту, сам выбірае правільны варыянт. Бо́льшая ўвага надаецца амографам, таму што сістэма апрацоўвае электронны тэкст, гэта значыць аналізуе графічныя знакі. Такім чынам, здаецца магчымым апрацоўваць толькі графічную форму слова з рознымі значэннямі, у адрозненне ад амафонаў, якія маюць адзіную форму вымаўлення і розную графічную сістэму.

У аснову прататыпа пакладзены наступныя электронныя слоўнікі, згодна якім адбываецца вызначэнне амографаў:

SBM1987 (паводле публікацыі «Слоўнік беларускай мовы. Арфаграфія. Арфаэпія. Акцэнтуацыя. Словазмяненне / пад рэд. М. В. Бірылы. – Мінск, 1987»);

SBM2012initial (пачатковыя формы паводле публікацыі «Слоўнік беларускай мовы / навук. рэд. А. А. Лукашанец, В. П. Русак. – Мінск : Беларус. навука, 2012»);

NOUN2013 (паводле публікацыі «Граматычны слоўнік назоўніка / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»);

ADJECTIVE2013 (паводле публікацыі «Граматычны слоўнік прыметніка, займенніка, лічэбніка, прыслоўя / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»);

NUMERAL2013 (паводле публікацыі «Граматычны слоўнік прыметніка, займенніка, лічэбніка, прыслоўя / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»);

PRONOUN2013 (паводле публікацыі «Граматычны слоўнік прыметніка, займенніка, лічэбніка, прыслоўя / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»);

VERB2013 (паводле публікацыі «Граматычны слоўнік дзеяслова / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»);

ADVERB2013 (паводле публікацыі «Граматычны слоўнік прыметніка, займенніка, лічэбніка, прыслоўя / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»);

ZALIZNIAK (паводле публікацыі «Грамматический словарь русского языка: Словоизменение / А. А. Зализняк. – М. : Русский язык, 1980. – 880 с.»);

CMU (паводле «Carnegie Mellon University Pronouncing Dictionary»).

Акрамя вышэй апісаных выданняў, распрацаваны дадатковыя слоўнікі *UWP_BE, UWP_RU,* якія папаўняюцца неалагізмамі і словамі, адсутнымі ў папярэдніх слоўніках, за кошт апрацоўкі электронных тэкстаў сэрвісамі платформы для апрацоўкі тэкставай і гукавой інфармацыі для розных тэматычных даменаў corpus.by (https://corpus.by/). Карыстальнік можа абраць з дадзенага спісу тыя слоўнікі, якія яму патрэбны для аналізу тэксту. Напрыклад, для пошуку амаграфіі ў тэксце на беларускай мове мэтазгодна абраць слоўнікі *SBM1987, SBM2012initial* і іншыя, на рускай мове – толькі слоўнік *ZALIZNIAK*, на англійскай – толькі слоўнік *CMU*.

Апрацоўка тэксту сэрвісам на дадзены момант мае наступныя асаблівасці:

1. Пошук ажыццяўляецца асобна па кожным абраным карыстальнікам слоўніку. Гэта значыць, што выпадкі міжмоўнай амаграфіі, напрыклад: бел. «выго́да» 1) усё, што задавальняе максімальныя запатрабаванні, чым зручна карыстацца; 2) прыволле) – рус. «вы́года» (прыбытак, даход, які атрымліваецца з чаго-небудзь) ці бел. «свая́» (прыналежны займеннік) – рус. «сва́я» (брус, які забіваецца ў грунт для апоры збудавання), выяўлены не будуць. Калі міжмоўная амаграфія прысутнічае, але для слова могуць быць знойдзены амаграфічныя варыянты ў межах адной мовы, напрыклад, як у выпадках «го́дны» і «годны́» для рускай мовы (адна з форм прысутнічае таксама ў беларускай мове) ці «ра́са» і «раса́» для беларускай мовы (адна з форм прысутнічае таксама ў рускай мове), сэрвісам будзе прадэманстравана толькі амаграфія ў межах адной мовы.

2. Для ажыццяўлення карэктнага пошуку ў слоўніках словы прыводзяцца да аднаго рэгістру – верхняга ці ніжняга, у залежнасці ад слоўніка. Таксама на дадзены момант слоўнікі, якія можна падключыць, не змяшчаюць уласныя імёны. Выпадкі кшталту рус. «Ко́ли» (мужчынскае імя ў родным склоне) і «коли́» (загадны лад 2-й асобы адзіночнага ліку дзеяслова «колоть») ці «Ма́ши» (жаночае імя ў родным склоне) і «маши́» (загадны лад 2-й асобы адзіночнага ліку дзеяслова «махать») не будуць разглядацца сэрвісам як амаграфічныя.

3. Літары рускага алфавіта «е» і «ё» лічацца сэрвісам рознымі. Падобныя прыклады: «же́ны» («жёны») і «жены́», не будуць пазначаны як амаграфічныя.

4. Паводле Оксфардскага слоўніка, у англійскай мове слова «homograph» азначае *«a word that is spelt like another word but has a different meaning from it, and may have a different pronunciation, for example bow /baʊ/, bow /bəʊ/»* (бел. «слова, якое пішацца так, як іншае слова, але мае рознае з ім значэнне і можа мець рознае вымаўленне, напрыклад, *bow /baʊ/, bow /bəʊ/*»). Англійскія амографы вызначаюцца сэрвісам па іншых правілах. Напрыклад, амографам будзе з'яўляцца слова «awesome», паколькі яно мае розныя варыянты вымаўлення пачатковага галоснага, а таксама «record», паколькі мае адно напісанне для слоў з рознымі значэннямі (дзеяслова і назоўніка), да таго ж вымаўленне дадзеных слоў адрозніваецца націскамі.

Алгарытм працы сэрвіса «Ідэнтыфікатар амографаў». Алгарытм дае магчымасць атрымаць спіс амографаў згодна ўбудаваным слоўнікам падчас апрацоўкі электроннага тэксту на беларускай, рускай і англійскай мовах. Больш якасныя вынікі сістэма прапануе для беларускай мовы, што звязана з колькасцю ўбудаваных слоўнікаў для яе. Аднак, згодна ўніверсальнаму характару алгарытму, пры больш багатым слоўнікавым напаўненні сістэмы ёсць магчымасць апрацоўваць амографы іншых моў.

Уваходныя даныя алгарытму ўяўляюць сабой наступны набор інструментаў:

– карыстальніцкі тэкставы ўвод, *UText*;
– мноства слоўнікаў, падключаных карыстальнікам, *Dictionaries*;
– мноства кірылічных сімвалаў у верхнім і ніжнім рэгістрах, *LettersCyr*;
– мноства лацінскіх сімвалаў у верхнім і ніжнім рэгістрах, *LettersLat*;
– мноства дадатковых сімвалаў (апострафы, сімвалы націскаў, злучок), *AdditionalCharacters*.

Алгарытм складаецца з чатырох асноўных каманд, якія падзяляюцца на больш падрабязныя крокі:

Крок 1. Апрацоўка ўваходнай інфармацыі.

1.1. Стварэнне мноства асноўных сімвалаў *MainCharacters* шляхам зліцця мностваў *LettersCyr* і *LettersLat*.

1.2. Раздзяленне *UText* на мноства *ParagrapsArr* паводле сімвала пераводу радка. Стварэнне пустога асацыятыўнага масіву U*niqueWordsArr* для наступнага запісу ўнікальных слоў, знойдзеных у *ParagrapsArr*.

1.3. Для кожнага элемента *Paragraph* мноства *ParagrapsArr* выканаць 1.3.1–1.3.4.

1.3.1. Выдаліць з *Paragraph* усе пачатковыя і канцавыя сімвалы водступаў.

1.3.2. Калі *Paragraph* = ∅, перайсці да наступнага элемента *Paragraph* і выканаць крок 1.3.1.

1.3.3. Стварэнне масіву *WordsArr* і запаўненне яго элементамі, складзенымі са зместу *Paragraph*. Элементам лічыцца спецыяльны набор сімвалаў ад пачатку радка да прабелу, ад

прабелу да прабелу і ад прабелу да канца (або сімвала пераводу) радка, які адпавядае шаблону <[1 сімвал MainCharacters]+[0 або больш сімвалаў MainCharacters і AdditionalCharacters]> альбо шаблону <[0 або больш сімвалаў MainCharacters]>. Стварэнне пераменнай *WordsCnt* і запіс у яе колькасці элементаў *WordsArr*.

1.3.4. Для I = 0; I < WordsCnt, I++ выканаць 1.3.4.1 і 1.3.4.2.

1.3.4.1. Стварэнне пераменнай *Word = WordsArr[I]*.

1.3.4.2. Калі *Word* ≠ *Ø*, прывесці ўсе сімвалы Word да ніжняга рэгістра і замяніць усе пачатковыя сімвалы «ў» на «у» (неабходна для наступнага карэктнага пошуку ў слоўніках). Паспрабаваць запісаць Word у асацыятыўны масіў *UniqueWordsArr*, выкарыстоўваючы значэнне Word у якасці ключа і прысвоіўшы яму значэнне 1. Калі слова паводле ключа Word ужо прысутнічае ў *UniqueWordsArr*, інкрэментаваць адпаведнае яму значэнне.

Крок 2. Пошук амографаў.

2.1. Стварэнне асацыятыўнага масіву *UniqueHomographsArr* для наступнага запісу ўнікальных амографаў і спадарожнай інфармацыі.

2.2. Для кожнага *UniqueWord* у *UniqueWordsArr* і кожнага *Dictionary* у *Dictionaries* выканаць 2.2.1–2.2.3.

2.2.1. Стварыць асацыятыўныя масівы *AccentArr* і *ResultArr*. Калі *Dictionary=Homographs_*Be (гэта значыць, што на дадзеным кроку цыкла праглядаецца слоўнік з такою назвай), праверыць, ці прысутнічае ў *Dictionary* элемент, ідэнтычны бягучаму элементу *UniqueWord*. Калі прысутнічае, запісаць у масіў *AccentArr* значэнне элемента слоўнікавага артыкула *Category*, які адпавядае бягучаму *UniqueWord*, а ў масіў *ResultArr* – значэнне адпаведных элементаў слоўнікавага артыкула *AccentedWord*, *Category* і *LexemeId*, пасля чаго перайсці да 2.2.3. Інакш – перайсці да наступнага кроку.

2.2.2. Калі *Dictionary≠Homographs_Be*, выканаць 2.2.2.1–2.2.2.3.

2.2.2.1. Стварыць пераменную *UniqueWordWithSlashes* і запісаць у яе значэнне бягучага *UniqueWord*, акружанае сімваламі слэша. Калі пры гэтым *UniqueWordWithSlashes* складаецца толькі з сімвалаў, якія належаць да мноства *LettersLat*, акружаных сімваламі слэша, прывесці ўсе сімвалы *UniqueWordWithSlashes* да верхняга рэгістра. Стварыць пераменную *Query* для захоўвання запыту да слоўнікавай базы і сфарміраваць яе значэнне згодна з шаблонам «SELECT * FROM %s WHERE word='%s», дзе %s – значэнне *UniqueWordWithSlashes*. Дадзены крок неабходны для ажыццяўлення карэктнага пошуку па слоўнікавых базах.

2.2.2.2. Калі паводле запыту, запісанага ў пераменнай *Query*, у бягучым *Dictionary* удалося знайсці якія-небудзь даныя, то да таго моманту, пакуль стандартная функцыя звароту да базы даных не дойдзе да апошняга радка Row у табліцы вынікаў, выконваць 2.2.2.2.1–2.2.2.2.3.

2.2.2.2.1. Стварыць пераменную *AccentedWord*. Калі ў бягучым *Row* зададзены элемент слоўнікавага артыкула *Accent*, запісаць у *AccentedWord* яго значэнне, прыведзенае да ніжняга рэгістра; калі пры гэтым у *AccentedWord* адсутнічае сімвал «+», завяршыць дадзены крок цыкла, перайсці да наступнага *Row* і выканаць 2.2.2.2.1 яшчэ раз. Калі элемент *Accent* не зададзены, запісаць у *AccentedWord* значэнне элемента *Transcription*, прыведзенае да ніжняга рэгістра.

2.2.2.2.2. Калі ў бягучым Row зададзены і элемент *LexemeId*, і элемент *Tag*, рэініцыялізаваць *Query* значэннем «SELECT * FROM tags WHERE tag='%s' LIMIT 1», дзе %s – значэнне элемента *Tag*, і паспрабаваць знайсці якія-небудзь даныя паводле гэтага запыту. Калі даныя ўдалося знайсці, то для кожнага радка табліцы вынікаў запісаць у створаную пераменную *CategoryVar* значэнне элемента слоўнікавага артыкула *Category* (калі ён зададзены), а ў пераменную *LexemeIdVar* – значэнне элемента слоўнікавага артыкула *LexemeId*, пасля чаго вызваліць сістэмныя рэсурсы ад вынікаў запыту.

2.2.2.2.3. Запісаць у масіў *AccentArr* значэнне *CategoryVar*, у масіў *ResultArr* – значэнні *AccentedWord*, *CategoryVar* і *LexemeIdVar*.

2.2.2.3. Ачысціць сістэмныя рэсурсы ад вынікаў запытаў да слоўнікавых баз даных.

2.2.3. Калі ў *AccentArr* прысутнічае больш за адзін элемент, выканаць 2.2.3.1–2.2.3.8.

2.2.3.1. Калі ў *UniqueHomographsArr* не зададзены элемент, роўны бягучаму *UniqueWord*, задаць такі элемент у выглядзе пустога масіву. Стварыць пераменную *AccentList* і ініцыялізаваць яе пустым значэннем.

2.2.3.2. Для кожнага элемента *AccentedWord* у *AccentArr* (дадзенаму элементу адпавядае свой масіў *CategoryArr*) здзейсніць наступныя дзеянні ў прамым парадку:

– дадаць да *AccentList* значэнне бягучага *AccentedWord*;

– калі *CategoryArr≠Ø,* дадаць да *AccentList* радок « <small>(», значэнне ўсіх унікальных элементаў *CategoryArr,* аб'яднаных у радок з дапамогай радка «, », і радок «)</small>»;

– дадаць да *AccentList* радок «
\n».

2.2.3.3. Замяніць у *AccentList* усе камбінацыі [любы сімвал + сімвал «=»] на той жа сімвал, але забяспечаны сімвалам пабочнага націску і акружаны тэгамі HTML для надання яму сіняга колеру. Замяніць у *AccentList* усе камбінацыі [любы сімвал + сімвал «+»] на той жа сімвал, але забяспечаны сімвалам асноўнага націску і акружаны тэгамі HTML для надання яму чырвонага колеру.

(2.2.3.2 і 2.2.3.3 – неабходны для фарміравання карэктнай выдачы.)

2.2.3.4. Выдаліць з *ResultArr* значэнні, якія паўтараюцца. Стварыць асацыятыўныя масівы *LexemeIdArr* і *CategoryArr*.

2.2.3.5. Калі элемент *HomographArray*, які адпавядае бягучаму *UniqueWord*, які, у сваю чаргу, адпавядае бягучаму *Dictionary*, не зададзены (тут і далей гэты элемент будзе названы як X), то для кожнага элемента *WordInfo* у *ResultArr* праверыць, ці зададзены ў ім элемент слоўнікавага артыкула *Category* і элемент *LexemeId*. Калі хаця б адзін з дадзеных элементаў не зададзены, запісаць у X значэнне *Accents*, роўнае *AccentList*, і значэнне *Type*, роўнае «-» (гэта значыць тып амаграфіі не вызначаны).

2.2.3.6. Калі пасля папярэдняга кроку X не зададзены, то для кожнага элемента *WordInfo* у *ResultArr* праверыць, ці зададзены ў *LexemeIdArr* элемент *WordInfo*, які ў сваю чаргу адпавядае элементу слоўнікавага артыкула *LexemeId*. Калі не зададзены, зрабіць яго роўным адзінцы, інакш запісаць у X значэнне *Accents*, роўнае *AccentList*, і значэнне *Type*, роўнае «one paradigm» (гэта значыць амографы належаць да адной парадыгмы скланення або спражэння).

2.2.3.7. Калі пасля папярэдняга кроку X не зададзены, то для кожнага элемента *WordInfo* у *ResultArr* праверыць, ці зададзены ў *CategoryArr* элемент *WordInfo*, які ў сваю чаргу адпавядае элементу слоўнікавага артыкула *Category*. Калі не зададзены, зрабіць яго роўным адзінцы, інакш запісаць у X значэнне *Accents*, роўнае *AccentList*, і значэнне *Type*, роўнае «one part of speech» (гэта значыць амографы належаць да адной часціны мовы).

2.2.3.8. Калі пасля папярэдняга кроку X не зададзены, то запісаць у X значэнне *Accents*, роўнае *AccentList*, і значэнне *Type*, роўнае «different parts of speech» (гэта значыць амографы належаць да розных часцін мовы).

Крок 3. Фарміраванне кантэкстаў. Для кожнага элемента *Paragraph* мноства *ParagrapsArr* выканаць 3.1–3.4.

3.1. Выдаліць з *Paragraph* усе пачатковыя і канцавыя сімвалы водступаў.

3.2. Калі *Paragraph=Ø*, перайсці да наступнага элемента *Paragraph* і выканаць 3.1.

3.3. Стварэнне масіву *WordsArr* і запаўненне яго элементамі, складзенымі са зместу *Paragraph*. «Элементам» лічыцца спецыяльны набор сімвалаў ад пачатку радка да прабелу, ад прабелу да прабелу і ад прабелу да канца (або сімвала пераводу) радка, які адпавядае шаблону <[1 сімвал MainCharacters]+[0 або больш сімвалаў MainCharacters і AdditionalCharacters]> альбо шаблону <[0 або больш сімвалаў MainCharacters]>. Стварэнне пераменнай *WordsCnt* і запіс у яе колькасці элементаў *WordsArr*.

3.4. Для *I = 0; I < WordsCnt, I++* выканаць 3.4.1–3.4.3.

3.4.1. Стварэнне пераменнай *Word = WordsArr[I]*.

3.4.2. Калі *Word≠Ø,* прывесці ўсе сімвалы Word да ніжняга рэгістра і замяніць усе пачатковыя сімвалы «ў» на «у».

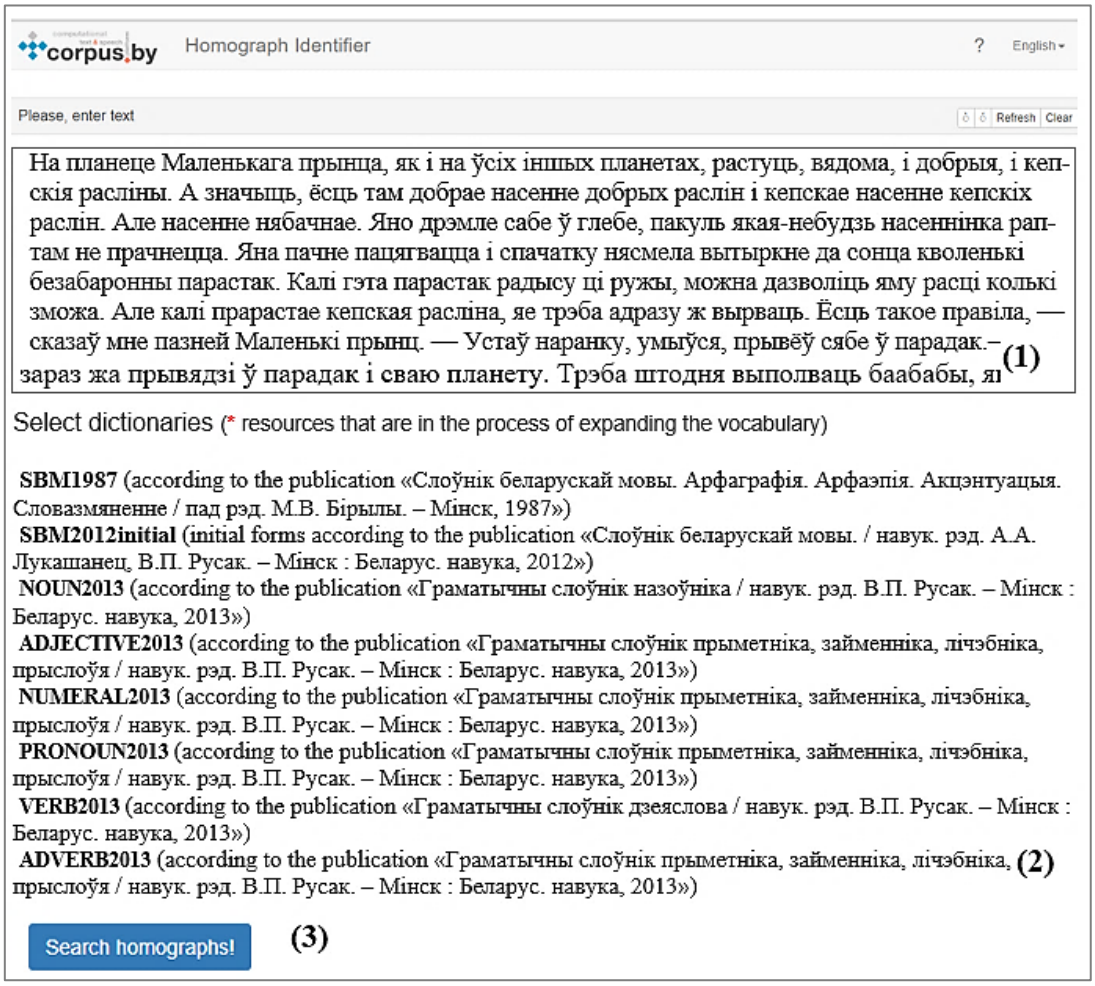
3.4.3. Для элемента масіву *UniqueHomographsArr,* роўнага значэнню *Word,* запісаць у *UniqueHomographsArr Y* элементаў, якія ідуць у *WordsArr* да элемента Word, сам элемент

Word і Y элементаў, якія ідуць у *WordsArr* пасля элемента *Word*, $0 \leqslant Y \leqslant 3$, $Y \in N$, дзе N – мноства натуральных лікаў.

Крок 4. Фарміраванне выдачы. Для кожнага знойдзенага амографа прадставіць варыянты націску, тып амографа, колькасць разоў, якія дадзены амограф сустракае ў зыходным тэксце, кантэксты ўжывання і назву слоўніка, у якім дадзены амограф быў знойдзены.

Канец алгарытму.

Інтэрфейс прататыпа і яго выкарыстанне. Сістэма ў форме вэб-сэрвіса знаходзіцца ў вольным доступе, што зручна для апрабацыі, тэсціравання і нагляднасці працаздольнасці апісанага вышэй алгарытму па спасылцы https://corpus.by/HomographIdentifier/?lang=be. Карыстальніцкі інтэрфейс сэрвіса прадстаўлены на мал. 1, які змяшчае наступныя вобласці: поле ўводу электроннага тэксту (1); поле выбару слоўнікаў (2); кнопка «Шукаць амографы!», якая запускае апрацоўку (3); поле выводу вынікаў, якое з’яўляецца пасля апрацоўкі і мае выгляд табліцы.



Мал. 1. Графічны інтэрфейс сэрвіса «Ідэнтыфікатар амографаў»

Fig. 1. Graphical interface of the "Omograph ID" service

Поле выводу вынікаў адлюстроўвае наступныя даныя: слова-амограф; варыянты націску ў словах-амографах; тып амографа (вызначэнне часціны мовы); колькасць разоў ужывання амографа ў тэксце; кантэксты ўжывання; слоўнік(і), па якім (якіх) быў праведзены пошук амографаў (мал. 2). Таксама сэрвіс прапануе карыстальніку атрымаць амографы ў форме звычайнага спіса. Каб атрымаць такі спіс, патрэбна націснуць на адпаведную спасылку ў полі выводу. Магчымы вынік працы сэрвіса прадстаўлены на мал. 2.

Homograph	Accent variant	Homograph type	Amount	Contexts	Dictionary
значыць	зна́чыць значы́ць	–	1	А значыць, ёсць там	SBM2012INIT
зараз	за́раз зара́з	–	1	– зараз жа прывядзі ў …	SBM2012INIT
яму	я́му (назоўнік) ямў (займеннік)	one part of speech	1	можна дазволіць яму расці колькі зможа. …	SBM1987
распазнаеш	распазнае́ш (дзеясл) распазна́еш (дзеясл)	one part of speech	1	як толькі распазнаеш. Дык вось, на …	SBM1987
сказаў	ска́заў (назоўнік) сказа́ў (дзеяслоў)	different parts of speech	1	… Ёсць такое правіла, - сказаў мне пазней Ма- ленькі …	SBM1987
зараз	за́раз (прыслоўе) зара́з (прыслоўе, назоўнік)	one part of speech	1	— зараз жа прывядзі ў …	SBM1987
часам	ча́сам (прыслоўе, назоўнік) часа́м (назоўнік)	one paradigm	1	… дык гэта спатрэбіц- ца. Часам работу можна адкласці …	SBM1987
павучаць	павуча́ць (дзеясл) павў́чаць (дзеясл)	one part of speech	1	… страшэнна не люб- лю павучаць людзей. Але небяспека …	SBM1987
адкідаю	адкі́даю (дзеясл) адкіда́ю (дзеясл)	one part of speech	1	… што сёння я адкідаю сваю стрыма- насць і …	SBM1987

Мал. 2. Вынік працы сэрвіса «Ідэнтыфікатар амографаў»

Fig. 2. The result of the "Omograph ID" service

Сэрвіс прапаноўвае два сцэнарыя працы: пошук па слоўніках, абраных па змоўчанні, ці пошук па слоўніках, абраных карыстальнікам. У першым сцэнары выкарыстоўваюцца ўсе слоўнікі, убудаваныя ў сістэму, у другім – вызначэнне амографаў адбываецца па асобных слоўніках, абраных карыстальнікам. Далей карыстальнік выбірае той амограф, які падыходзіць па кантэксце згодна сэнсу выказвання.

«Ідэнтыфікатар амографаў» з'яўляецца адным з асноўных сэрвісаў, які выкарыстоўваецца для вычыткі электроннага тэксту праз праграмнае забеспячэнне, распрацаванае супрацоўнікамі лабараторыі распазнавання і сінтэзу маўлення АІПІ НАН Беларусі. Прапанаваным праграмным забеспячэннем выступаюць сэрвісы апрацоўкі электроннай тэкставай інфармацыі, якія размешчаны на інтэрнэт-платформе www.corpus.by. Выкарыстанне спецыялізаванай методыкі вычыткі электроннага тэксту, якая прадстаўлена на афіцыйным сайце лабараторыі праз спасылку https://ssrlab.by/5406, дазваляе атрымаць арфаграфічна правільны тэкст на беларускай мове. У межах задання «Мадэлі, метады, алгарытмы і праграмныя сродкі інтэлектуальнай апрацоўкі, аналізу і распазнавання медыка-біялагічных даных, малюнкаў, маўленчай і тэкставай інфармацыі і распрацоўка на іх аснове інфармацыйных тэхналогій і сістэм медыцынскага і сацыяльнага прызначэння» сэрвіс «Ідэнтыфікатар амографаў» выкарыстоўваецца для стварэння і кампіляцыі

мадэляў паралельных электронных карпусоў маўлення і тэкстаў медыцынскай, сацыяльнай і прававой тэматыкі на рускай, беларускай і англійскай мовах для распрацоўкі. Акрамя таго, сэрвіс прайшоў апрабацыю падчас складання тэкстаў рэлігійнай тэматыкі [18].

Доступ да сэрвіса праз API. Для доступу да сэрвіса «Ідэнтыфікатар амографаў» праз API неабходна адправіць *AJAX-запыт* тыпу *POST* на адрас https://corpus.by/HomographIdentifier/api.php. Праз масіў *data* перадаюцца наступныя параметры:

text – адвольны ўваходны тэкст;

маркеры выкарыстання слоўнікаў:

sbm1987 – «Слоўнік беларускай мовы. Арфаграфія. Арфаэпія. Акцэнтуацыя. Словазмяненне / пад рэд. М. В. Бірылы. – Мінск, 1987»;

sbm2012initial – «Слоўнік беларускай мовы / навук. рэд. А. А. Лукашанец, В. П. Русак. – Мінск : Беларус. навука, 2012»;

noun2013 – назоўнікі паводле кнігі «Граматычны слоўнік назоўніка / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»;

adjective2013 – прыметнікі паводле кнігі «Граматычны слоўнік прыметніка, займенніка, лічэбніка, прыслоўя / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»;

numeral2013 – лічэбнікі паводле кнігі «Граматычны слоўнік прыметніка, займенніка, лічэбніка, прыслоўя / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»;

pronoun2013 – займеннікі паводле кнігі «Граматычны слоўнік прыметніка, займенніка, лічэбніка, прыслоўя / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»;

verb2013 – дзеясловы паводле кнігі «Граматычны слоўнік дзеяслова / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»;

adverb2013 – прыслоўі паводле кнігі «Граматычны слоўнік прыметніка, займенніка, лічэбніка, прыслоўя / навук. рэд. В. П. Русак. – Мінск : Беларус. навука, 2013»;

zalizniak – «Грамматический словарь русского языка: Словоизменение / А. А. Зализняк. – М. : Русский язык, 1980. – 880 с.»;

cmu – «Carnegie Mellon University Pronouncing Dictionary»;

uwp_be – беларускія словы, сабраныя сістэмай «Апрацоўка невядомых слоў»;

uwp_ru – рускія словы, сабраныя сістэмай «Апрацоўка невядомых слоў».

Ніжэй прадстаўлены прыклад AJAX-запыту:

```
$.ajax({
type: “POST”,
url: “https://corpus.by/HomographIdentifier/api.php”,
data:{
“text”: “– Маё жыццё такое аднастайнае. Я палюю на курэй, людзі палююць на мяне. Усе куры падобны адна на адну, і ўсе людзі падобны адзін на аднаго. З гэтай прычыны мне і сумнавата. Але, калі ты прыручыш мяне, жыццё маё нібы сонцам азарыцца. Я навучуся распазнаваць твае крокі сярод тысячы іншых. Калі я чую людскія крокі, я ўцякаю і хаваюся. Твае ж паклічуць мяне з нары як музыка. І потым – паглядзі! Бачыш, там, удалечыні, жытняе поле? Я не ем хлеба. Жыта мне ні да чаго. Збажына нічога не напамінае мне. І гэта так сумна! А ў цябе залатыя валасы. І як цудоўна было б, калі б ты прыручыў мяне! Залатое жыта заўсёды было б мне напамінкам пра цябе… Я палюбіў бы песню ветру ў калоссі…”,
“sbm1987”: 1,
“sbm2012initial”: 1
},
success: function(msg){ },
error: function() { }
});
```

Сервер верне JSON-масіў з уваходным тэкстам (параметр text), спісам знойдзеных у тэксце амографаў (параметр result), масівам з дэталямі выніку (параметр resultArr), колькасцю выяўле-

ных амографаў (параметр resultCnt) і спасылкай на спіс выяўленых амографаў (параметр resultUrl). Напрыклад, па вышэй прыведзеным AJAX-запыце будзе сфарміраваны наступны адказ:


```
[
{
“text”: “Груша цвіла апошні год.”,
“result”: “куры
людскія
нары
музыка
музыка”,
“resultArr”: {
“SBM1987”: {
“куры”: {
“accents”: “ку́ры куры́”,
“type”: “адна часціна мовы”,
“count”: 1,
“contexts”: “… на мяне. Усе куры падобны адна на …”
},
“людскія”: {
“accents”: “лю́дскія людскі́я”,
“type”: “адна часціна мовы”,
“count”: 1,
“contexts”: “… Калі я чую людскія крокі, я ўцякаю …”
},
“нары”: {
“accents”: “нары́ на́ры”,“type”: “адна часціна мовы”,
“count”: 1,
“contexts”: “… паклічуць мяне з нары як музыка. І …”
},
“музыка”: {
“accents”: “музы́ка му́зыка”,
“type”: “адна часціна мовы”,
“count”: 1,
“contexts”: “… з нары як музыка. І потым – паглядзі! …”
}
},
“SBM2012initial”: {
“музыка”: {
“accents”: “музы́ка му́зыка”,
“type”: “–”,
“count”: 1,
“contexts”: “… з нары як музыка. І потым – паглядзі! …”
}
}
},
“resultCnt”: “5”,
“resultUrl”: “https://corpus.by/<…>”,
}
]
```


Заключэнне. Прыклад працы прататыпа сістэмы «Ідэнтыфікатар амографаў» дэманструе працаздольнасць распрацаваных алгарытмаў. Карэктнасць працы алгарытму можа быць адлюстравана ў працэсе выкарыстання сістэмы для пошуку амографаў і аналізу вынікаў апрацаванага электроннага тэксту. Сэрвіс будзе карысным для вырашэння ўсіх прыкладных задач, рэалізацыя якіх ускладняецца наяўнасцю амографаў. Такія патрэбы могуць узнікнуць у наступных выпадках: пры правядзенні даследавання пэўнага тэксту на наяўнасць і функцыянаванне ў ім амографаў, што запатрабавана ў корпуснай лінгвістыцы; пры падрыхтоўцы тэксту да апрацоўкі сінтэзатарам маўлення з мэтай атрымання агучанага тэксту. Своечасовае выяўленне амографаў і правільная расстаноўка націскаў дазволяць значна павысіць якасць працы беларускамоўных сістэм сінтэзу маўлення. Немалаважна выкарыстанне праграмы падчас вычыткі тэкстаў вялікага памеру. Сэрвіс дапамагае выявіць амографы для далейшай расстаноўкі націскаў у тых выпадках амаграфіі, ад якіх залежыць сэнс тэксту.

Рэалізацыя метаду, заснаванага на ведах у дадзенай сістэме, дазваляе з упэўненасцю сцвярджаць, што гэта эфектыўны рэсурс для аўтаматычнага пошуку амографаў беларускай, рускай і англійскай моў і іх вызначэння для далейшай апрацоўкі натуральнай мовы камп’ютарнымі сродкамі. Прататып з’яўляецца першым рэсурсам па здыманні шматзначнасці для беларускай мовы ў адкрытым доступе, што дазваляе кожнаму зацікаўленаму выкарыстаць яго для асабістых мэт. У сваю чаргу, для распрацоўшчыкаў пастаяннае выкарыстанне сэрвіса дае магчымасць працягваць бесперапыннае тэсціраванне працаздольнасці алгарытмаў. Паляпшэнне якасці прыкладання можа быць дасягнута дабаўленнем іншых слоўнікаў. Акрамя таго, адкрыты

код робіць магчымым выкарыстоўваць сэрвіс для іншых славянскіх моў шляхам яго адаптацыі да канкрэтнай мовы і дадання новых слоўнікавых рэсурсаў. На далейшым этапе распрацоўкі плануецца дапрацаваць сістэму ідэнтыфікацыі амографаў дабаўленнем дадатковай опцыі пошуку амографаў у тэкстах з пазнакай адрозных часцін мовы пры аднолькавых націсках слоў.

Уклад аўтараў. *Ю. С. Гецэвіч* прапанаваў канцэпцыю аўтаматызаванай экстракцыі амонімаў і выканаў адбор метадаў і алгарытмаў для распрацоўкі прататыпа сэрвіса «Ідэнтыфікатар амографаў». *А. А. Бакуновіч* распрацаваў пакрокавы алгарытм працы сэрвіса і ажыццявіў яго рэалізацыю на API, унёс карэкціроўкі пасля тэсціравання прататыпа. *А. Я. Драгун* падрыхтавала тэкставы корпус літаратурнага дамену з дастатковым аб'ёмам даных, якія змяшчаюць усе тыпы амографаў, для тэсціравання сэрвіса. *Я. С. Зяноўка* і *Д. І. Латышэвіч* правялі дэтальнае тэсціраванне прататыпа з вызначэннем тэхнічных і лінгвістычных памылак апрацоўкі тэкстаў прататыпам. *М. А. Казлова* прадставіла падрабязную інструкцыю выкарыстання «Ідэнтыфікатара амографаў».

Спіс выкарыстаных крыніц

1. Word Sense Disambiguation: Algorithms and Applications / eds.: E. Agirre, P. Edmonds. – Springer, 2007. – Series: Text, Speech and Language Technology. – Vol. 33. – 377 p.

2. Ширшикова, А. А. О проблемах омонимии / А. А. Ширшикова // Альманах современной науки и образования. – Тамбов : Грамота, 2012. – № 2(57). – С. 190–192.

3. Tian, T. Improving web search results for homonyms by suggesting completions from an ontology / T. Tian, J. Geller, S. A. Chun // Current Trends in Web Engineering – 10th Intern. Conf. on Web Engineering, ICWE 2010 Workshops, Vienna, Austria, July 2010. – Vienna, Austria, 2010. – P. 41–44.

4. Van den Beukel, S. Homonym detection for humor recognition in short text / S. van den Beukel, L. Aroyo // Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Brussels, Belgium, 31 October 2018. – Brussels, Belgium, 2018. – P. 286–291.

5. Pozdniakov, K Regular homophones: a tool for semantic typology and for linguistic reconstruction / K. Pozdniakov, G. Segerer // Africana Linguistica. – 2019. – Vol. 25. – P. 231–279.

6. Roll, U. Using machine learning to disentangle homonyms in large text corpora / U. Roll, R. A. Correia, O. Berger-Tal // Conservation Biology. – June 2018. – Vol. 32, iss. 3. – P. 716–724.

7. Рысаков, С. В. Статистические методы снятия омонимии / С. В. Рысаков, Э. С. Клышинский // Новые информационные технологии в автоматизированных системах. – 2015. – № 18. – С. 555–563.

8. Navigli, R. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation / R. Navigli, P. Velardi // IEEE Transactions on Pattern Analysis and Machine Intelligence. – July 2005. – Vol. 27, iss. 7. – P. 1075–1086.

9. Гатауллин, Р. Р. Аналитический обзор методов разрешения морфологической многозначности / Р. Р. Гатауллин // Электронные библиотеки. – 2016. – Т. 19, № 2. – С. 98–114.

10. Зеленков, Ю. Г. Вероятностная модель снятия морфологической омонимии на основе нормализующих подстановок и позиции соседних слов / Ю. Г. Зеленков, И. В. Сегайлович, В. А. Титов // Компьютерная лингвистика и интеллектуальные технологии : тр. Междунар. конф. «Диалог-2005», Звенигород, 1–6 июня 2005 г. – М. : Наука, 2005. – С. 616–638.

11. Мухамедшин, Д. Р. Модуль разрешения морфологической неоднозначности: архитектура и организация базы данных / Д. Р. Мухамедшин, Д. Ш. Сулейманов // Программные продукты и системы. – 2020. – Т. 33, № 1. – С. 38–46.

12. Порохнин, А. А. Анализ статистических методов снятия омонимии в текстах на русском языке / А. А. Порохнин // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. – 2013. – № 2. – С. 168–174.

13. Лесько, О. Н. Использование онтологии предметной области для снятия омонимии в естественно-языковых текстах / О. Н. Лесько, Ю. В. Рогушина // Проблеми програмування : науковий журнал. – 2017. – № 2. – С. 61–71.

14. Зинькина, Ю. В. Разрешение функциональной омонимии в русском языке на основе контекстных правил / Ю. В. Зинькина, Н. В. Пяткин, О. А. Невзорова // Компьютерная лингвистика и интеллектуальные технологии : тр. Междунар. конф. «Диалог-2005», Звенигород, 1–6 июня 2005 г. – М. : Наука, 2005. – С. 198–202.

15. Okrut, T. Context-sensitive homograph disambiguation with NooJ in Belarusian and Russian electronic texts / T. Okrut, B. Lobanov, Y. Yakubovich // Intern. Scientific Conf. on the Automatic Processing of Natural-Language Electronic Texts “NooJ’2015”, Minsk, Belarus, 11–13 June 2015. – Minsk : UIIP NASB, 2015. – P. 48.

16. Камп’ютарна-лінгвістычныя сэрвісы www.corpus.by для аўтаматычнай апрацоўкі тэкстаў / Я. С. Качан [і інш.] // Нацыянальна-культурны кампанент у літаратурнай і дыялектнай мове : зб. навук. арт. – Брэст : БрДУ імя А. С. Пушкіна, 2016. – С. 93–104.

17. The problem of automatic search and determonation of homonyms for the Belarusian and Russian languages / Ya. Zianouka [et al.] // Информационные технологии в промышленности, логистике и социальной сфере. – Минск : Объединенный институт проблем информатики НАН Беларуси, 2021. – С. 182–184.

18. Новы Запавет – Кніга Прыповесьцяў : пер. А. Бокуна. – Мінск : Пазітыў-цэнтр, 2016. – 511 с.

References

1. Agirre E., Edmonds P. (eds.). *Word Sense Disambiguation: Algorithms and Applications.* Springer, 2007, Series: Text, Speech and Language Technology, vol. 33, 377 p.

2. Shirshikova A. *On the problems of homonymy*. Al'manakh sovremennoy nauki i obrazovaniya [*Almanac of Modern Science and Education*], Tambov, Gramota, 2012, no. 2(57), pp. 190–192 (In Russ.).

3. Tian T., Geller J., Chun S. A. Improving web search results for homonyms by suggesting completions from an ontology. *Current Trends in Web Engineering: 10th International Conference on Web Engineering, ICWE 2010 Workshops, July 2010, Vienna, Austria, July 2010*. Vienna, Austria, 2010, pp. 41–44.

4. Van den Beukel S., Aroyo L. Homonym detection for humor recognition in short text. *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Brussels, Belgium, 31 October 2018.* Brussels, Belgium, 2018, pp. 286–291.

5. Pozdniakov K., Segerer G. Regular homophones: a tool for semantic typology and for linguistic reconstruction. Africana Linguistica, 2019, vol. 25, pp. 231–279.

6. Roll U., Correia R. A., Berger-Tal O. Using machine learning to disentangle homonyms in large text corpora. *Conservation Biology*, June 2018, vol. 32, iss. 3, pp. 716–724.

7. Rysakov S. V., Klyshinsky E. S. *Statistical methods of homonymy removal.* Novye informacionnye tehnologii v avtomatizirovannyh sistemah [*New Information Technologies in Automated Systems*], 2015, no. 18, pp. 555–563 (In Russ.).

8. Navigli R., Velardi P. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, July 2005, vol. 27, no. 7, pp. 1075–1086.

9. Gataullin R. R. *Analytical review of methods for resolving morphological polysemy*. Elektronnyye biblioteki [*Electronic Libraries*], 2016, vol. 19, no. 2, pp. 98–114 (In Russ).

10. Zelenkov Yu. G., Segalovich I. V., Titov V. A. *Probabilistic model for removing morphological homonymy based on normalizing substitutions and positions of neighboring words*. Komp'juternaja lingvistika i intellektual'nye tehnologii : trudy Mezhdunarodnoj konferencii «Dialog-2005», Zvenigorod, 1–6 ijunja 2005 g. [*Computer linguistics and intellectual technologies: proceedings of the international conference "Dialogue-2005", Zvenigorod, 1–6 June 2005*], Moscow, Nauka, 2005, pp. 616–638 (In Russ.).

11. Mukhamedshin D. R., Suleymanov D. Sh. *Module of morphological ambiguity resolution: database architecture and organization*. Programmnyye produkty i sistemy [*Software Products and Systems*], 2020, vol. 33, no. 1, pp. 38–46 (In Russ.).

12. Porokhnin A. A. *Analysis of statistical methods for removing homonymy in Russian texts*. Vestnik Astrakhanskogo gosudarstvennogo tekhnicheskogo universiteta. Serija: Upravlenie, vychislitel'naja tehnika i informatika [*Bulletin of the Astrakhan State Technical University. Series: Management, Computing and Information Science*], 2013, no. 2, pp. 168–174.

13. Les'ko O. N., Rogushina Yu. V. *Using the domain ontology for removing homonymy in natural language texts*. Problemi programuvannya [*Programming Problems*], 2017, no. 2, pp. 61–71 (In Russ.).

14. Zin'kina Yu. V., Pyatkin N. V., Nevzorova O. A. *Resolution of functional homonymy in Russian based on contextual rules*. Komp'juternaja lingvistika i intellektual'nye tehnologii : trudy Mezhdunarodnoj konferencii «Dialog-2005», Zvenigorod, 1–6 ijunja 2005 g. [*Computer linguistics and intellectual technologies: proceedings of the international conference "Dialogue-2005", Zvenigorod, 1–6 June 2005*], Moscow, Nauka, 2005, pp. 198–202 (In Russ.).

15. Okrut T., Lobanov B., Yakubovich Y. Context-sensitive homograph disambiguation with NooJ in Belarusian and Russian electronic texts. *International Scientific Conference on the Automatic Processing of Natural-Language Electronic Texts "NooJ'2015", Minsk, Belarus, 11–13 June 2015*. UIIP NASB, 2015, p. 48.

16. Hiecevic Ju., Kacan Ya, Lysy S., Stanislavienka H., Hiuntar A. *Computer-linguistic services www.corpus.by for automatic text processing*. Nacyjanalna-kulturny kampanient u litaraturnaj i dyjaliektnaj movie : zbornik navukovyh artykulaŭ [*National-cultural Component in Literary and Dialect Language : Collection of Scientific Articles*], Brest, Brjescki dzjarzhaŭny ŭniversitjet imja A. S. Pushkina, 2016, pp. 93–104 (In Bel).

17. Zianouka Ya., Hetsevish Yu., Majeŭski S., Dzienisiuk Dz. *The problem of automatic search and determonation of homonyms for the Belarusian and Russian languages*. Informacionnye tehnologii v promyshlennosti, logistike i socialnoj sfere [*Information Technologies in Industry, Logistics and Social Sphere*], Minsk, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, 2021, pp. 182–184.

18. Novy Zapavet – Kniga Prypoves'cjaŭ. *The New Testament – The Book of Proverbs*. Transl. A. Bokuna. Minsk, Pazityŭ-cjentr, 2016, 511 p. (In Bel).

Інфармацыя пра аўтараў

Гецэвіч Юрась Станіслававіч, кандыдат тэхнічных навук, дацэнт, загадчык лабараторыі распазнавання і сінтэзу маўлення, Аб'яднаны інстытут праблем інфарматыкі НАН Беларусі.
E-mail: yuras.hetsevich@gmail.com

Зяноўка Яўгенія Сяргееўна, малодшы навуковы супрацоўнік, Аб'яднаны інстытут праблем інфарматыкі НАН Беларусі.
E-mail: evgeniakacan@gmail.com

Латышэвіч Давід Іосіфавіч, стажор малодшага навуковага супрацоўніка, Аб'яднаны інстытут праблем інфарматыкі НАН Беларусі.
E-mail: david.latyshevich@gmail.com

Бакуновіч Андрэй Аляксеевіч, малодшы навуковы супрацоўнік, Аб'яднаны інстытут праблем інфарматыкі НАН Беларусі.
E-mail: bakunovich.andrei.work@gmail.com

Драгун Анастасія Яўгеньеўна, малодшы навуковы супрацоўнік, Аб'яднаны інстытут праблем інфарматыкі НАН Беларусі.
E-mail: ndrahun@gmail.com

Казлова Маргарыта Аляксандраўна, стажор малодшага навуковага супрацоўніка, Аб'яднаны інстытут праблем інфарматыкі НАН Беларусі.
E-mail: margaryta.kazlova@gmail.com

Information about the authors

Yuras S. Hetsevich, Ph. D. (Eng.), Assoc. Prof., Head of the Speech Synthesis and Recognition Laboratory, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: yuras.hetsevich@gmail.com

Yauheniya S. Zianouka, Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: evgeniakacan@gmail.com

David I. Latyshevich, Trainee of Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: david.latyshevich@gmail.com

Andrey A. Bakunovich, Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: bakunovich.andrei.work@gmail.com

Anastasia Ya. Drahun, Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: ndrahun@gmail.com

Margarita A. Kazlova, Trainee of Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: margaryta.kazlova@gmail.com

Правила для авторов

Редакция журнала «Информатика» просит авторов руководствоваться приведенными ниже правилами.

I. Статьи принимаются в редакцию через электронную систему подачи по адресу http://inf.grid.by в формате файлов текстовых редакторов Microsoft Word. Объем оригинальной статьи – от 8 до 16 стр., включая рисунки, таблицы и достаточное количество наиболее актуальных ссылок; объем обзорной статьи – от 16 до 32 стр., включая все основные ссылки. Текст набирается с переносами, шрифт Times New Roman 11 пт, интервал между строками одинарный, абзацный отступ 0,5 см, поля по 2,5 см со всех сторон.

Материал статьи должен быть четко структурированным: Введение; основные разделы, в которых изложены цели и задачи, методы, результаты; Заключение (выводы).

II. Статьи о результатах работ, проведенных в научных учреждениях, должны иметь разрешение на публикацию (сопроводительное письмо за подписью руководителя или выписку из заседания ученого совета, отдела или кафедры, акт экспертизы).

III. Статьи в обязательном порядке должны включать аннотацию, ключевые слова, список литературы, информацию об авторах на русском и английском языках.

На титульной странице располагаются следующие метаданные:

1. Индекс по универсальной десятичной классификации (УДК); *на русском и английском языках* тип статьи (оригинальная или обзорная), название статьи, инициалы и фамилии всех авторов, полное наименование учреждений, где работают авторы, с указанием почтового адреса, при наличии указывается ученая степень и ORCID, e-mail ответственного лица.

2. Аннотация (Abstract) объемом 150–250 слов в оригинальной статье должна быть структурирована отдельными подразделами: Цели, Методы, Результаты, Заключение, а также максимально характеризовать содержательную часть рукописи. Сюда не следует включать впервые введенные термины, аббревиатуры (за исключением общеизвестных), ссылки на литературу.

3. Ключевые слова (Keywords) – наиболее значимые слова или словосочетания по теме работы, отражающие специфику темы, объекты и результаты исследования; перечень ключевых слов должен содержать 5–10 слов.

4. В разделе Благодарности (Acknowledgements) указываются все источники финансирования исследования, а также благодарности людям, которые участвовали в работе над статьей.

5. Автор обязан уведомить редакцию о реальном или потенциальном конфликте интересов, включив информацию в раздел Конфликт интересов (Conflict of interest).

6. Формулы, рисунки, таблицы в статье нумеруются в соответствии с порядком их упоминания в тексте. Ссылки на рисунки и таблицы в тексте обязательны. Рисунки должны быть выполнены с хорошим разрешением в масштабе, позволяющем четко различать надписи и обозначения. Цветные иллюстрации печатаются только в том случае, когда это необходимо для понимания излагаемого материала. Подрисуночные подписи с расшифровкой всех позиций, представленных на рисунке, и названия таблиц набираются шрифтом гарнитуры основного текста размером 9 пт. Перевод подрисуночной подписи и пояснений к рисунку, а также перевод названия таблицы, заголовки строк или столбцов располагаются курсивом после русскоязычной версии.

7. Набор формул выполняется в формульном редакторе Microsoft Equation или Math Type. Прямым шрифтом набираются: греческие и русские буквы; математические символы (sin, lg, ∞); символы химических элементов (C, Cl, $CHCl_3$); цифры (римские и арабские); индексы (верхние и нижние), являющиеся сокращениями слов. Курсивом набираются латинские буквы, символы физических величин (в том числе и в индексе).

8. Список использованной литературы оформляется в соответствии с требованиями Высшей аттестационной комиссии Республики Беларусь (ГОСТ 7.5–2008). Номер литературной ссылки в тексте дается порядковым номером в квадратных скобках. Ссылаться на неопубликованные работы не допускается.

10. Отдельно оформляется References со следующей структурой: авторы (транслитерация), транслитерированное название монографии, *Перевод названия монографии на английский язык*. Выходные данные с обозначениями на английском языке. От транслитераций названий статей можно отказаться.

Ссылки на учебно-методическую литературу, ГОСТы, авторефераты, статистические отчеты в список не включаются, а оформляются в виде сносок (с подробными рекомендациями можно ознакомиться на сайте журнала в разделе Правила для авторов).

11. В разделе Информация об авторах (Information about the authors) приводятся ФИО авторов полностью, ученая степень, звание, должность, название организации, ORCID (при наличии).

IV. Все поступающие в редакцию рукописи проходят предварительную проверку на соответствие Правилам для авторов. Статья может быть возвращена автору на доработку с просьбой устранить недостатки или дополнить информацию. После проверки на соответствие правилам статья направляется рецензенту с указанием сроков рецензирования.

V. При наличии замечаний рецензента автору предоставляется определенное время на доработку рукописи. Статьи, направляемые на доработку, должны быть возвращены в исправленном виде с ответами на все замечания. Окончательное решение о публикации или отклонении рукописи принимается редколлегией журнала. При положительном заключении рецензента статья передается редактору для подготовки к печати. Редакция оставляет за собой право на редакционные изменения, не искажающие основное содержание статьи.

VI. Редакция журнала предоставляет возможность первоочередного опубликования статей, представленных лицами, которые осуществляют послевузовское обучение (аспирантура, докторантура, соискательство) в год завершения обучения.

VII. Авторы несут ответственность за направление в редакцию статей, уже опубликованных ранее или принятых к публикации другими изданиями.

ИНДЕКСЫ

00827
для индивидуальных
подписчиков

008272
для предприятий
и организаций

ИНФОРМАТИКА ▪ INFORMATICS ТОМ ▪ VOL. 20 4 | 2023

ISSN 1816-0301 (Print)