ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 35 Выпуск 6

ISSN Online 2220-6426
Volume 35 Issue 6

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2023

ИСП РАН

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН
Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областях системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.

Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.

Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.

Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.

The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.

Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Сайт: http://www.ispras.ru/proceedings/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Web: http://www.ispras.ru/en/proceedings

Содержание

Table of Contents

DOI: 10.15514/ISPRAS-2023-35(6)-1

Валидация требований и ее влияние на качество при использовании программного обеспечения: тематическое исследование

[1] *Л. Канчари*, ORCID: 0000-0003-2382-7724 <*lcanchari@continental.edu.pe*>
[2] *П. Ангелери*, ORCID: 0000-0002-0719-7235 <*paula.angeleri@ub.edu.ar*>
[1,3] *А. Давила*, ORCID: 0000-0003-2455-9768 <*abraham.davila@pucp.edu.pe*>

[1] *Перу, Уанкайо, Континентальный университет.*
[2] *Аргентина, Буэнос-Айрес, Университет Бельграно.*
[3] *Перу, Лима, Папский католический университет Перу.*

Аннотация. Исследование посвящено изучению и анализу взаимосвязи между повышением качества требований к программному обеспечению и качеством используемого программного продукта. Анализ проводился при разработке реальных требований к программному продукту и вычислении метрик двух программных продуктов в соответствии со стандартом ISO/IEC 25010. Результаты показывают, что мероприятия по проверке качества, проводимые на этапе разработки требований к программному обеспечению, положительно влияют на качество реализуемых программных продуктов. В отношении изученного программного обеспечения можно сказать, что повышение качества требований способствовало повышению качества использования программных продуктов.

Ключевые слова: требования к программному обеспечению; проверка качества требований; качество при использовании программного обеспечения; стандарт ISO/IEC 25010.

Для цитирования: Канчари Л., Ангелери П., Давила А. Валидация требований и ее влияние на качество при использовании программного обеспечения: тематическое исследование. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 7–28. DOI: 10.15514/ISPRAS–2023–35(6)–1.

Благодарности: Представленное исследование поддерживалось Папским католическим университетом Перу, группой исследований и разработок в области программной инженерии GIDIS.

Requirements Validation and Its Impact in Software Quality in Use: A Case Study

[1] *L. Canchari*, ORCID: 0000-0003-2382-7724 <*lcanchari@continental.edu.pe*>
[2] *P. Angeleri*, ORCID: 0000-0002-0719-7235 <*paula.angeleri@ub.edu.ar*>
[1,3] *A. Dávila,* ORCID: 0000-0003-2455-9768 <*abraham.davila@pucp.edu.pe*>

[1] *Universidad Continental, Huancayo, Perú.*
[2] *Universidad de Belgrano, Buenos Aires, Argentina.*
[3] *Pontificia Universidad Católica del Perú, Lima, Perú.*

Abstract. In this study, the relationship between the improvement of software requirement quality and the software product quality in use was explored and analyzed. Analysis was based on the design of software product quality-in-use and the measure of metrics from ISO/IEC 25010 standard in two software products. The results show that the validation activities introduced in the software requirements stage have a positive relationship with the quality in use of the software products analyzed. In the software studied, it can be said that the improvement of the quality of the requirements has contributed to the improvement of the quality in use of software products.

Keywords: software requirements; requirement quality validation; software product quality-in-use; ISO/IEC 25010 standard.

For citation: Canchari L., Angeleri P., Dávila A. Requirements Validation and its impact in Software Quality in Use: A case study. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 7-28 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-1.

Acknowledgements. This research was supported by Software Engineering Research & Development Group (GIDIS) – Pontifical Catholic University of Peru.

1. Введение

Стандарт ISO/IEC 12207 был установлен в правовой базе Перу [1] в качестве обязательного. Подразделениями информационных технологий (Information Technology Unit, ITU) принятие стандарта ISO/IEC 29110-5-1-2 рассматривалось как возможный прогрессивный способ внедрения стандарта ISO/IEC 12207 в рамках общего развития технологий. Стандарт ISO/IEC 29110-5-1-2 – это руководство для небольших организаций по разработке программного обеспечения, которое в явном виде включает задачи верификации и валидации [2]. В частности, он определяет валидацию требований к программному обеспечению в качестве первичной задачи [2], которая положительным образом влияет на качество программного обеспечения [3].

Кроме того, некоторые исследования установили взаимосвязь между требованиями к программному обеспечению и успешности его разработки [4-7]. Другие авторы указывали, что анализ требований является одним из наиболее важных этапов [3, 5, 7-9]. В работе [10] указывается, что проблемы с требованиями вызвали кризис программного обеспечения. Таким образом, понятно, что действия, направленные на повышение качества требований, способствуют повышению продуктивности проекта и качества передаваемого пользователям разработанного программного продукта. Авторы также указывают, что существует заинтересованность исследователей в изучении качества при использовании программного продукта [4, 11-13].

В нашей предыдущей статье [14] характеристики, результаты и преимущества валидации требований к программному обеспечению в контексте общественного учреждения ITU были представлены в контексте некоторых характеристик качества и характеристик проекта. Настоящее исследование тоже базировалось на характеристиках и результатах работы [14]. Как видно из [14], нам удалось улучшить как процесс поддержания качества программного обеспечения, так и качество готового программного продукта. Чтобы это проверить, качество программного продукта измерялось с помощью метрик эффективности проекта и характеристик программного продукта.

В этой статье, беря за основу качество при использовании, авторы в качестве основной цели выбрали задачу определения преимуществ применения процедур валидации требований к программному обеспечению. По этой причине при оценке качества при использовании программного обеспечения, основанной на стандарте ISO/IEC 25010 и связанных с ним материалах, используется тот же контекст, что и в предыдущей работе [14], в которой исследовалось программное обеспечение информационных систем для общественных учреждений в Лиме. Остальная часть статьи структурирована следующим образом: в разделе 2 представлены принципы обеспечения “качества при использовании”, основанные на стандартах серии ISO/IEC 25000, в разделе 3 описан подход к проведению тематического исследования, в разделе 4 представлены результаты этого исследования, а в разделе 5 – итоговое обсуждение и будущие работы.

2. Основные концепции

В этом разделе представлены некоторые основные концепции и модель качества при использовании, введенные стандартом ISO/IEC 25000, а также кратко излагается содержание некоторых работ, тем или иным образом связанных с нашим исследованием.

2.1 Модель “качества при использовании” программного обеспечения в ISO

Серия стандартов ISO/IEC 25000 (Требования и оценка качества систем и программных средств) представляет собой набор технических документов, разработанных в отношении качества программного продукта в рамках проекта SQuaRE [15-18]. Эти документы объединяют все предыдущие стандарты и предлагают новые подходы. Стандарты ISO/IEC 25022 [15] и ISO/IEC 25010 [19] определяют качество при использовании как степень, в которой конкретные пользователи могут использовать продукт или систему для удовлетворения своих потребностей и достижения конкретных целей с эффективностью, результативностью, удовлетворением и свободой от риска в конкретных контекстах использования [15]; как это видно на рис. 1. В процессе определения модели качества программного продукта [20] набор характеристик, подхарактеристик и показателей выбирается из базовой модели (например, внутренней или внешней модели ISO/IEC 25010) и, в некоторых случаях, для всего программного обеспечения или его части устанавливаются их эталонные значения.

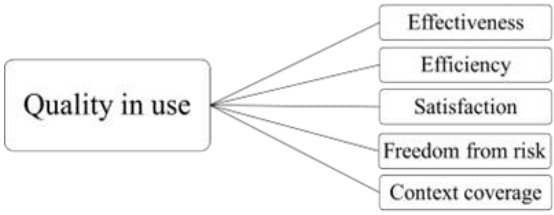


Рис 1. Модель качества при использовании ISO/IEC 25010, адаптированная из [19]
(Эффективность, Результативность, Удовлетворенность, Свобода от рисков, Контекстный охват)
Fig 1. Quality in use model ISO/IEC 25010 adapted from [19]

2.2 Показатели качества при использовании

Показатель – это определенный метод измерения, связанный со шкалой оценок [20-21]. Измерение – это набор операций, целью которых является определение некоторого численного значения [15]. Качество при использовании может быть измерено на основе предложений стандарта ISO 25010, которые создают основу для более комплексного подхода к спецификации и измерению качества [22-23]. В табл. 1 представлены показатели качества при использовании в соответствии с ISO/IEC 25022.

Согласно стандарту ISO/IEC 25022 [15], существует множество факторов, влияющих на качество при использовании программного продукта. Измерение или оценка этих факторов представляет собой сложный процесс [24]. В целом, программная инженерия предлагает и методы, и инструменты, помогающие снизить сложность разработок. Например, иногда применяются такие две методики: методика “Размышление вслух” и “Опрос удобства использования системы после изучения” (Post-Study System Usability Questionnaire, PSSUQ). По методике исследования “Размышление вслух” участники должны произносить вслух любые слова, которые приходят им на ум, пока они работают над заданием [25]. Согласно [26], участники озвучивают свои мысли по мере прохождения через программный интерфейс. Методика PSSUQ, согласно [27], обладает следующими характеристиками:

(i) это стандартизированный вопросник из 16 пунктов, который после выполнения задания используется для измерения удовлетворенности пользователей веб-сайтом, программным обеспечением, системой или продуктом,

(ii) пункты написаны в прошедшем времени для обозначения недавно выполненных заданий,

(iii) для присвоения баллов по каждому пункту используется 7-уровневая шкала, которая варьируется от 1, что означает “полностью согласен”, до 7, что означает “полностью не согласен”,

(iv) по трем группам – полезность системы (пункты 1-6), качество информации (пункты 7-12), качество интерфейса (пункты 13-15) – вычисляется среднее значение, отдельный 16-й пункт используется для общей оценки.

Табл. 1. Показатели качества при использовании в соответствии с ISO/IEC 25022 [15]
Table 1. Quality in use metrics according to ISO/IEC 25022 [15]

Показатель	**ID**	**Имя**
Показатели результативности	Ef-1-G	Выполненные задачи
	Ef-2-S	Достигнутые цели
	Ef-3-G	Ошибки в задаче
	Ef-4-G	Задачи с ошибками
	Ef-5-G	Интенсивность ошибок задачи
Показатели эффективности	Ey-1-G	Время выполнения задачи
	Ey-2-S	Эффективность затраченного времени
	Ey-3-S	Экономическая эффективность
	Ey-4-S	Коэффициент полезного времени
	Ey-5-S	Ненужные действия
	Ey-6-S	Последствия усталости
Показатели удовлетворенности	SUs-1-G	Общая удовлетворенность
	SUs-2-G	Удовлетворенность функциями
	SUs-3-G	Дискреционное использование
	SUs-4-G	Использование функций
	SUs-5-G	Доля жалующихся пользователей
	SUs-6-G	Доля жалоб пользователей на ту или иную функцию
Показатели доверия	STr-1-G	Доверие пользователей
Показатели удовольствия	SPl-1-G	Удовольствие пользователя
Показатели эргономического комфорта	SCo-1-G	Физический комфорт
Показатели свободы риска	Rec-1-G	Рентабельность инвестиций (ROI)
	Rec-2-G	Время окупаемости инвестиций
	Rec-3-G	Эффективность бизнеса
	Rec-4-G	Преимущества инвестиций в ИТ
	Rec-5-G	Обслуживание клиентов
	Rec-6-G	Количество посетителей сайта, ставших клиентами
	Rec-7-G	Выручка от каждого клиента
	Rec-8-G	Ошибки с экономическими последствиями
Показатели по снижению рисков для здоровья и безопасности труда	RHe-1-G	Частота возникновения проблем со здоровьем пользователей
	RHe-2-G	Воздействие на здоровье и безопасность пользователей
	RHe-3-G	Безопасность людей, пострадавших от использования системы
Показатели по снижению экологических рисков	REn-1-G	Экологическое воздействие
Показатели полноты контекста	CCm-1-G	Полнота контекста
Показатели гибкости	CFI-1-S	Использование гибких связей
	CFI-2-S	Гибкость продукта
	CFI-3-S	Независимость от квалификации

2.3 Обзор литературы

В научной литературе имеются описания исследований, которые помогают понять комплексность качества программного обеспечения, предлагая модели, атрибуты и показатели. Однако существует лишь несколько исследований, в которых анализируется влияние качества при использовании как следствия совершенствования процесса валидации требований. Эти тематически близкие работы рассматриваются далее.

В работе [28] авторы приходят к выводу, что валидация требований к программному обеспечению имела большое значение с точки зрения снижения стоимости проекта.

В работе [29] указаны:

(i) важность интеграции характеристик качества продукта в процессах их производства,

(ii) замечание о том, что организации проводят сертификацию своих процессов без учета влияния этих процессов на характеристики качества продукции, и,

(iii) подтверждение того, что качество продукта зависит от качества процесса его получения.

На основании проведенного исследования делается вывод о том, что качество взаимодействия влияет на решения пользователей о внедрении той или иной системы.

3. Разработка тематического исследования

В ходе проведенной нами работы было проведено тематическое исследование (case study, CS) с учетом рекомендаций, предложенных в работах [30-31]. В табл. 2 показана адаптация этапов CS. Необходимые теоретические обоснования (P1-3) были представлены в разделе 2.

Табл. 2. Фазы и шаги схемы тематического исследования, адаптированные из [30] и [31]
Table 2. Adaptation of the case study phases from [30] and [31]

Фаза	Функция
P1 Разработка тематического исследования	(P1-1) Цель исследования
	(P1-2) Что вы изучаете
	(P1-3) Требуемая теория
	(P1-4) Вопросы исследования
	(P1-5) Методы сбора
	(P1-6) Отбор данных
P2 Определение сбора данных	(P2-1) Определение метода сбора данных
E1 Сбор данных	(E1-1) Сбор данных
E2 Анализ данных	(E2-1) Анализ данных
	(E2-2) Интерпретация данных
R1 Отчетность	(R1-1) Представление результатов

3.1 Объекты исследования

Информационные системы, рассмотренные в данном тематическом исследовании, являются частью нашего предыдущего исследования [15] общественных учреждений ITU. В ходе ранее проведенного исследования было установлено, что качество требований к программному обеспечению для одной из систем превосходит качество требований к другой системе. Кроме того, было доказано, что качество требований напрямую связано со стоимостью, графиком работ и качеством процесса создания программного обеспечения информационной системы. В нашем новом исследовании мы стремимся определить взаимосвязь между повышением качества требований и качеством при использовании программного обеспечения, а также определить, как эти понятия преобразуются в преимущества для пользователя.

В настоящем исследовании из-за ограничений ITU использовались только две информационные системы из тех шести, что были представлены в [15]. Для обеспечения преемственности к предыдущему исследованию [15] были выбраны системы SI02 и SI03. Для

этих систем были определены участвовавшие пользователи, контекст для каждой из систем, протоколы оценки, методы получения данных и показатели для проведения измерений качества. Систему SI02 используют 40 должностных лиц ITU, которые отвечают за функционирование процесса государственных закупок. Эта система представляет собой настольное приложение, основанное на клиент-серверной архитектуре и реляционной базе данных. При разработке системы методы проверки требований не применялись. Система SI03 используется 20-ю должностными лицами ITU, отвечающими за набор и мониторинг сотрудников для консультационной службы. SI03 – это веб-приложение, и, как и SI02, оно было разработано в клиент-серверной архитектуре для доступа к реляционной базе данных. При разработке этого программного обеспечения были применены методы проверки требований.

3.2 Цели тематического исследования (P1-1)

Основная цель этого исследования (P1-1) состоит в том, чтобы выявить преимущества внедрения валидации требований к программному обеспечению с точки зрения качества при использовании, то есть качества программного обеспечения в специфическом контексте пользователя.

Контекст исследования имеет следующие характеристики:

(i) программные продукты эксплуатируются более 12 месяцев,

(ii) участники используют программное обеспечение в качестве основного инструмента для своей рабочей деятельности, и

(iii) системное окружение, то есть компьютеры, оборудование рабочих мест, графики работы и даже отвлекающие факторы, оставлены такими же, как в реальной жизни.

Для оценки качества при использовании была подготовлена копия приложений, связанная с копией базы данных, а на стороне пользователя были настроены ярлыки для этих приложений.

3.3 Что является объектом изучения в этом эмпирическом исследовании (P1-2)?

В ходе этого исследования определялись и анализировались показатели качества при использовании программного обеспечения информационных систем в ITU. Объектами анализа для данного исследования являлись два программных продукта. В одном случае проверка требований к программному обеспечению включалась в состав работ по улучшению процесса, а в другом – не включалась. Программы, выбранные для исследования (см. табл. 3), входили в состав проектов, анализировавшихся в предыдущем исследовании [14]. В проекте SI03 с самого начала проводилась проверка требований к программному обеспечению (улучшение процесса разработки программного обеспечения), в то время как в проекте SI02 использовались обычные практики, существовавшие ранее.

В ходе исследования измерялось качество при использовании программных систем, затем полученные результаты сравнивались между собой. Сравнение проводилось с учетом показателей эффективности, результативности и удовлетворенности атрибутом и без учета оценки безопасности и охвата контекстов использования. Такое решение обосновывалось следующими аргументами:

(i) показатели экономического риска не имеют отношения к делу, поскольку обе программные системы не ставили задач по достижению экономической выгоды,

(ii) одна из программ обрабатывает конфиденциальную информацию о людях, а другая – нет, сравнивать показатели риска в работе с людьми нецелесообразно,

(iii) считается, что улучшения, внесенные в процесс валидации требований, не влияют на окружающую среду, по этой причине экологический риск не оценивается, и

(iv) наличие пользователей было ограничением исследования, что вынудило нас проводить оценку в едином (реальном) контексте, поэтому охват контекста не оценивался.

Табл. 3. Список проектов исследования
Table 3. List of projects studies

Проект	Описание	Группа
SI02	Программная система управления процессом закупок	G-Con: Контрольная группа
SI03	Программная система мониторинга сотрудников консалтинговой службы	G-Exp: Экспериментальная группа

3.4 Вопросы исследования (P1-4)

В этом исследовании был задан следующий вопрос: "*Какие характеристики качества при использовании улучшились в результате принятия мер по проверке требований к программному обеспечению*"? Этот вопрос разбит на следующие гипотезы:

- Внедрение методов проверки требований к программному обеспечению позволяет получить программное обеспечение, которое работает более эффективно в специфическом контексте применения.
- Внедрение методов проверки требований к программному обеспечению позволяет получить программное обеспечение, которое работает более результативно в специфическом контексте использования.
- Внедрение методов проверки требований к программному обеспечению позволяет получить программное обеспечение, удовлетворяющее потребностям пользователя в специфическом контексте использования.

Показатели результативности оценивают точность и полноту, с которыми пользователи достигают конкретных целей. Эти показатели не учитывают, как достигаются цели, а только степень, в которой они достигнуты [15]. Показатели эффективности оценивают затраченные ресурсы в зависимости от точности и полноты, с которыми пользователи достигают поставленных целей. Для этого:

(i) в качестве основного ресурса для измерения эффективности используется время,

(ii) эффективность пользователя сравнивается с эффективностью эксперта, и

(iii) при выполнении задач устанавливается лимит времени [15].

Показатели удовлетворенности оценивают степень удовлетворения потребностей пользователей при использовании продукта или системы в специфическом контексте использования [15].

3.5 Методы сбора (P1-5)

Часть первичных данных была получена с помощью методики "Размышление вслух", для чего в протокол оценки программного обеспечения был включен специально разработанный набор задач. Применение этой методики было связано с выполнением следующих заданий:

(i) идентифицировать участников,

(ii) выдать участникам запланированные для них задания, и

(iii) ожидать, что, выполняя задания, пользователи будут вслух выражать свои мысли.

Для протоколирования времени выполнения каждой задачи, допущенных ошибок, ненужных действий и других данных использовались специальные программы, записывавшие состояние экрана. Каждая запись воспроизводилась, и из полученных записей извлекались

следующие данные: количество достигнутых целей, количество допущенных ошибок, количество ненужных действий, количество запросов о помощи, использованное время (минуты), время запроса о помощи (минуты), время устранения ошибок (минуты) и количество жалоб.

Для получения дополнительных данных был использован вопросник PSSUQ, причем вопросы участникам задавались сразу же после того, как они завершали выполнение протокольных заданий. При этом для получения нужных данных использовался следующий инструментарий: протокол оценки, разработанный для каждой программной системы (см. приложение A), и анкета PSSUQ, в которую для получения дополнительной информации для измерения удовлетворенности были добавлены два пункта (см. приложение B). В протокол включались сведения о цели задания, об основных индикаторах, наборы задач, действий и данных, предлагаемых для выполнения задач. Тестовые задачи, включенные в протокол, были предложены и заранее оценены опытным пользователем-экспертом, который устанавливал максимальное время, нужное для выполнения каждой задачи. Время, необходимое для выполнения всех задач в каждом протоколе, оценивалось в 40 минут.

Выполнение протокола было осуществлено после согласования с лицом, ответственным за прикладную область, при этом учитывались три аспекта: во-первых, обеспечение того, чтобы контекст использования был очень похож на реальный, во-вторых, предложение пользователям озвучивать свои мысли во время выполнения задач и, в-третьих, запись действий и комментариев пользователей во время выполнения протокола. Участники выполняли протокольные задания на своих рабочих местах и своих компьютерах в свое рабочее время по расписанию. Каждому участнику были предоставлены наушники, была активирована аудио- и видеозапись.

Число пользователей, выполнивших задачи протокола оценки программной системы SI02, составило 10 человек, то есть 83% от общего числа пользователей. Для системы SI03 также было 10 пользователей, что составляло 90% от общего числа пользователей. Собрать всех пользователей для исследования оказалось невозможно, но количество участников представляет собой значительную выборку. Опыт исполнителей работы [15] показал, что надежные результаты могут быть получены при выборке всего из восьми участников. Все участники пользовались программным обеспечением более пяти месяцев и могут быть квалифицированы как продвинутые пользователи, но не как эксперты. В данном исследовании пользователь – это человек, который работает с программным обеспечением как с основным инструментом своей рабочей деятельности, взаимодействует с ним каждый день и посвящает ему более половины своего рабочего времени.

Численные значения, собранные, обработанные и проанализированные в данном исследовании, представляют собой итоги измерения результативности, эффективности и удовлетворенности пользователей качеством при использовании информационных систем. Смысл показателей задавался на основе стандартов ISO/IEC 25022 [16]:

(i) показатели результативности оценивают точность и полноту, с которыми пользователи достигают своих специфических целей,

(ii) показатели эффективности оценивают затраченные ресурсы в зависимости от точности и полноты, с которыми пользователи достигают целей, и

(iii) показатели удовлетворенности оценивают степень удовлетворения потребностей пользователей при использовании программного продукта в конкретном контексте.

В исследовании была использована только часть показателей результативности, эффективности и удовлетворенности. Кроме того, в качестве метода наблюдения и сбора первичных данных для показателей, предложенных стандартом ISO/IEC 25022, использовалась методика “Размышления вслух”. После завершения исследования для оценки

удобства использования системы в качестве инструмента сбора данных, дополняющих показатели и характеристики удовлетворенности, был использован вопросник PSSUQ.

3.6 Отбор данных (P1-6)

Показатели, предложенные ISO/IEC 25022, требуют первичного набора данных, эти данные были извлечены из собранных видеороликов. Для каждого задания подсчитывались следующие первичные данные: достигнутые цели (OA), количество ошибок (NE), ненужные действия (UA), запросы на помощь (AR), использованное время (TU), время запроса помощи (HRT), время восстановления после ошибки (ERT), количество жалоб (NC). На основе этих данных были рассчитаны показатели результативности и эффективности. С другой стороны, показатели удовлетворенности были рассчитаны с использованием оценок факторов и пунктов PSSUQ.

Для измерения качества при использовании систем SI02 и SI03 применялись показатели ISO/IEC 25022, представленные в табл. 4. Результативность измеряли четырьмя показателями, эффективность – пятью, удовлетворенность – тоже пятью. Кроме того, были использованы три фактора из вопросника PSSUQ. Показатель “Sus-1-G Общая удовлетворенность” измерялся на основе интегральной оценки PSSUQ, а “Sus-2-G Удовлетворенность функциями”, “STr-1-G Доверие пользователей”, “SPl-1-G Удовольствие пользователей”, “Sco-1-G Физический комфорт” на основе баллов, полученных по пунктам 17, 18, 13 и 4 PSSUQ.

4. Результаты

В этом разделе представлены результаты и анализ валидности. Здесь же приведен анализ показателей результативности, эффективности и удовлетворенности. Валидность включает в себя внутреннюю, внешнюю и конструктивную.

4.1 Показатели результативности

В табл. 5 приведены данные по показателям результативности программного обеспечения системы SI02. Значения показаны с детализацией по всем участникам (P01, P02, ...), они были рассчитаны с использованием измерительных функций стандарта ISO/IEC 25022. Чтобы получить обобщенное значение, представляющее показатели на уровне всей системы, рассчитывалось среднее значение. Табл. 6 показывает данные, аналогичные данным табл. 5, но табл. 6 относится к системе SI03.

Значение показателя “Ef-1-G Выполненные задачи” для SI03 больше, чем значение для SI02, что указывает на то, что участники выполнили в системе SI03 больше заданий, чем в системе SI02. Значение показателя “Ef-2-S Достигнутые цели” для SI03 немного выше значения для SI02, это позволяет сделать вывод, что участники достигли того же количества целей с помощью системы SI03, что и с системой SI02. Значение показателя “Ef-3-G Ошибки в задаче” для SI03 меньше значения для SI02, это указывает на то, что участники допустили меньше ошибок в заданиях для SI03, чем в заданиях для SI02. Значение показателя “Ef-4-G Задачи с ошибками” для SI03 меньше значения для SI02, что указывает на то, что в системе SI03 без ошибок было выполнено больше задач, чем в системе SI02.

С использованием данных, приведенных в табл. 5 и табл. 6, были подготовлены диаграммы (см. рис. 2), которые показывают распределение значений показателей результативности для систем SI02 и SI03.

На рис. 2.a по показателю “Ef-1-G Выполненные задачи” видно, что участники, пользовавшиеся системой SI03, выполнили больше задач, чем участники, пользовавшиеся системой SI02. Этот вывод основан на положительном смещении, которое наблюдается для SI03, по сравнению с отрицательным смещением, наблюдаемым для SI02.

Табл. 4. Показатели, взятые из стандартов ISO/IEC 250XX и вопросника PSSUQ и выбранные для измерения качества при использовании
Table 4. Metrics selected to measure the quality in use, taken from ISO/IEC 250XX and PSSUQ

Id	Название	Функция	Источник
Ef-1-G	Выполненные задачи	X = A / B A = Количество выполненных уникальных задач B = Общее количество выполнявшихся уникальных задач	ISO/IEC
Ef-2-S	Достигнутые цели	{X = 1 ΣA_i | X≥0} A_i = Пропорциональное значение каждой пропущенной или неправильной цели в выходных данных задачи (максимальное значение = 1)	ISO/IEC
Ef-3-G	Ошибки в задаче	X = A A = Количество ошибок, допущенных пользователем во время выполнения задачи	ISO/IEC
Ef-4-G	Задачи с ошибками	X = A / B A = Количество задач с ошибками B = Общее количество задач	ISO/IEC
Ey-1-G	Время выполнения задачи	X = T T = Время выполнения задачи	ISO/IEC
Ey-2-S	Эффективность затраченного времени	X = A / T A = Количество достигнутых целей T = Время	ISO/IEC
Ey-3-S	Экономическая эффективность	X = A / B A = Общая стоимость выполнения задачи B = Количество достигнутых целей	ISO/IEC
Ey-4-S	Коэффициент полезного времени	X = T_a/T_b где T_a = Продуктивное время, то есть T_a = время, затраченное на выполнение задачи – время, затраченное на получение помощи – время, затраченное на восстановление после ошибок – время, затраченное на безрезультатный поиск T_b = Время, затраченное на выполнение задачи	ISO/IEC
Ey-5-S	Ненужные действия	X = A/B A = Количество действий, которые фактически не были необходимы для выполнения задачи B = Количество действий, выполненных пользователем	ISO/IEC
SUs-1-G	Общая удовлетворенность	X = Σ (A_i) A_i = Ответ на вопрос	ISO/IEC
SUs-2-G	Удовлетворенность функциями	X = Σ (A_i) A_i = Ответ на вопрос, связанный с определенной характеристикой	ISO/IEC
STr-1-G	Доверие пользователя	X = A A = Значение психометрической шкалы доверия	ISO/IEC
SPl-1-G	Удовольствие пользователя	X = A A = Значение психометрической шкалы удовольствия	ISO/IEC
SCo-1-G	Физический комфорт	X = A A = Значение психометрической шкалы комфорта	ISO/IEC
SysU	Полезность системы (SysUse)	X = A A = Значение психометрической шкалы полезности	PSSUQ
InfoQ	Качество информации (InfoQual)	X = A A = Значение психометрической шкалы качества информации	PSSUQ
InterQ	Качество интерфейса (InterQual)	X = A A = Значение психометрической шкалы качества интерфейса	PSSUQ

На рис. 2.b по показателю “Ef-2-S Достигнутые цели”, видно, что участники, которые использовали систему SI03, достигли в среднем такого же количества целей, что и участники, которые использовали систему SI02. Несмотря на то, что существует положительное смещение для SI03 и отрицательное смещение для SI02, разница в средних значениях очень мала, что не позволяет установить статистические различия.

На рис. 2.c по показателю “Ef-3-G ошибки в задаче”, замечено, что участники, которые использовали ПО SI03, допустили меньше ошибок, чем участники, которые использовали ПО SI02. Это верно, потому что SI03 имеет отрицательное смещение по отношению к асимметричному и концентрированному распределению SI02.

На рис. 2.d по показателю “Ef-4-G задачи с ошибками”, видно, что при использовании SI03 задач с ошибками меньше, чем при использовании SI02. Среднее значение, отрицательное

смещение SI03 по отношению к симметричному распределению SI02 и разброс данных позволяют нам заключить, что показатель SI03 “Ef-4-G задачи с ошибками” лучше, чем показатель SI02.

Табл. 5. Показатели результативности для SI02
Table 5. Software effectiveness metrics for SI02

PU	Ef-1-G	Ef-2-G	Ef-3-G	Ef-4-G
P01	0.81	0.81	7	0.50
P02	0.39	0.42	15	0.83
P03	0.48	0.52	5	0.50
P04	0.61	0.65	6	0.50
P05	0.81	0.84	6	0.50
P06	0.81	0.84	6	0.83
P07	0.81	0.81	7	0.67
P08	0.58	0.65	5	0.67
P09	0.52	0.58	7	0.83
P10	0.84	0.90	8	0.67
Среднее значение	0.66	0.70	7.20	0.65

Табл. 6. Показатели результативности для SI03
Table 6. Software effectiveness metrics for SI03

PU	Ef-1-G	Ef-2-G	Ef-3-G	Ef-4-G
P01	0.63	0.63	0	0.00
P02	0.75	0.75	7	0.57
P03	0.95	0.95	3	0.29
P04	0.90	0.90	2	0.14
P05	0.53	0.53	6	0.43
P06	1.00	1.00	0	0.00
P07	0.63	0.63	4	0.43
P08	0.45	0.45	6	0.43
P09	0.65	0.65	6	0.57
P10	0.65	0.65	9	0.57
Среднее значение	0.71	0.71	4.30	0.34

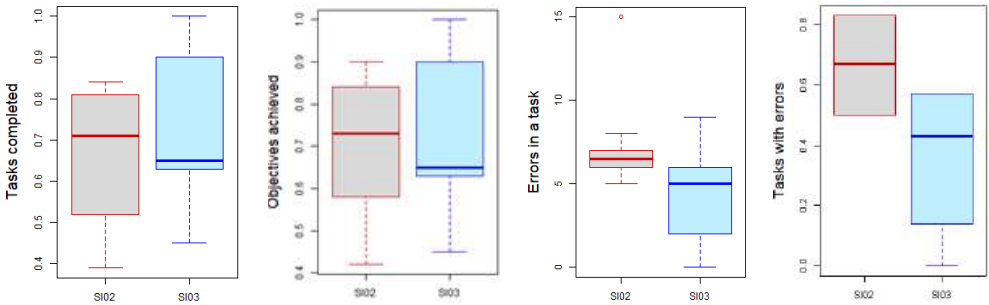


Рис 2. Сравнение показателей результативности для SI02 и SI03: (a) “Ef-1-G Выполненные задачи”, (b) “Ef-2-S Достигнутые цели”, (c) “Ef-3-G Ошибки в задаче”, (d) “Ef-4-G Задачи с ошибками”
Fig 2. Effectiveness metrics comparison for SI02 and SI03: (a) “Ef-1-G Tasks completed”, (b) “Ef-2-S Objectives achieved”, (c) “Ef-3-G Errors in a task”, (d) “Ef-4-G Tasks with errors”

Анализ данных в табл. 5 и табл. 6 и оценка диаграмм рис. 2 позволяют нам сделать вывод, что показатель “Ef-2-S Достигнутые цели” не показывает существенной разницы между SI03 и SI02; в то время как остальные три показателя для SI03 лучше, чем для SI02. Это позволяет сделать вывод, что программная система SI03 обладает большей результативностью при использовании, чем система SI02.

Процент улучшения был рассчитан путем вычисления по формуле

$$(avg02 - avg03) / avg02 \times 100$$

где *avg02* – среднее значение показателей результативности системы SI02, которое вычислялось до проведения работ по ее улучшению, а *avg03* – среднее значение показателей системы SI03, полученных после улучшений. Исходя из табл. 5 и 6, улучшение особенно хорошо заметно на примере показателей по количеству ошибок в одной задаче (на 40%) и по общему числу ошибок в задачах (на 47%).

4.2 Показатели эффективности

В табл. 7 и табл. 8 приведены данные показателей эффективности программных систем SI02 и SI03 соответственно. Данные представлены по каждому из участников, они были рассчитаны с использованием измерительных функций, рекомендованных стандартом ISO/IEC 25022. Чтобы получить значение, представляющее показатель на обобщенном уровне системы, также рассчитывалось среднее значение по всем участникам.

Табл. 7. Показатели эффективности для системы SI02
6Table 7. Efficiency metrics for software SI02

PU	Ey1G	Ey2S	Ey3S	Ey4S	Ey5S
P01	49	0.51	1.96	0.88	0.22
P02	41	0.30	3.42	0.83	0.33
P03	47	0.32	3.13	0.94	0.23
P04	47	0.40	2.47	0.94	0.28
P05	45	0.56	1.80	0.80	0.22
P06	46	0.54	1.84	1.00	0.20
P07	55	0.45	2.20	0.96	0.22
P08	49	0.36	2.72	0.92	0.35
P09	51	0.31	3.19	0.94	0.26
P10	53	0.49	2.04	0.89	0.25
Среднее значение	48	0.42	2.48	0.91	0.26

Табл. 8. Показатели эффективности для системы SI03
Table 8. Efficiency metrics for software SI03

PU	Ey1G	Ey2S	Ey3S	Ey4S	Ey5S
P01	48	0.52	1.92	0.96	0.04
P02	49	0.61	1.63	0.82	0.10
P03	64	0.59	1.68	0.94	0.07
P04	41	0.88	1.14	0.93	0.05
P05	53	0.40	2.52	0.83	0.22
P06	47	0.85	1.18	1.00	0.00
P07	52	0.48	2.08	0.96	0.14
P08	40	0.45	2.22	0.90	0.26
P09	40	0.65	1.54	0.93	0.15
P10	40	0.65	1.54	0.83	0.20
Среднее значение	47	0.61	1.75	0.91	0.12

Значение показателя “Ey-1-G Время выполнения задачи” для SI03 немного меньше значения для SI02, это указывает на то, что участники, которые использовали SI03, выполнили задания за более короткое время, чем участники, использовавшие SI02. Значение показателя “Ey-2-S Эффективность затраченного времени” для SI03 выше, чем значение для SI02, это указывает на то, что за время, затраченное на выполнение задач, пользователи программного обеспечения SI03 достигли большего количества целей. Значение показателя “Ey-3-S Экономическая эффективность” для SI02 имеет более высокое значение, чем SI03, это указывает на то, что участникам потребовалось больше времени для выполнения задач SI02. Значение показателя “Ey-4-S Коэффициент полезного времени” для SI02 и SI03 равны и имеют значение 0,91, это указывает на то, что большинство действий были эффективными и способствовали достижению целей задачи. Значение показателя “Ey-5-S Ненужные действия” для SI03 меньше значения для SI02, это указывает на то, что участники выполнили меньше ненужных действий с системой SI03, чем с системой SI02.

На основе данных, приведенных в табл. 7 и табл. 8 были подготовлены диаграммы (см. рис. 3), которые представляют распределение значений показателей эффективности SI02 и SI03.

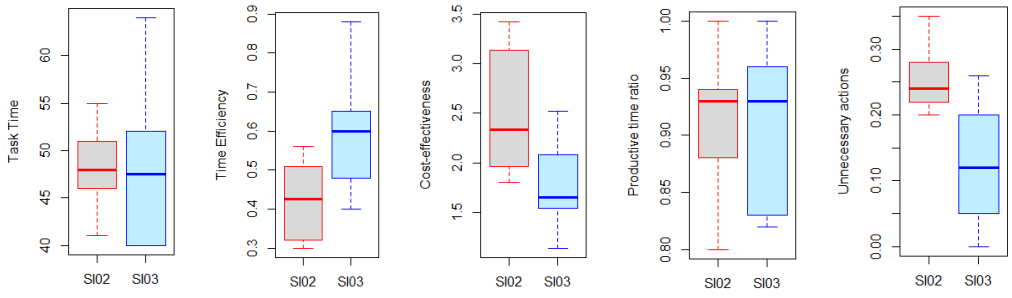


Рис 3. Сравнение показателей эффективности для SI02 и SI03.
(a) "Ey-1-G Время выполнения задачи", (b) "Ey-2-S Эффективность затраченного времени",
(c) "Ey-3-S Экономическая эффективность", (d) "Ey-4-S Коэффициент полезного времени",
(e) "Ey-5-E Ненужные действия"
Fig 3. Efficiency metrics comparison for SI02 and SI03. (a) "Ey-1-G Task time",
(b) "Ey-2-S Time efficiency", (c) "Ey-3-S Cost-effectiveness", (d) "Ey-4-S Productive time ratio",
(e) "Ey-5-S Unnecessary actions"

На рис. 3.a показатель "Ey-1-G Время реализации задачи" указывает на то, что система SI03 для выполнения участниками своих задач требует меньше времени, чем система SI02, это подтверждается отрицательным смещением SI03, симметричным распределением SI02 и средними значениями. Однако из-за высокой изменчивости показателей SI03 и небольшого отличия от медианного значения это утверждение не является решающим.

На рис. 3.b показатель "Ey-2-S Эффективность затраченного времени" показывает, что система SI03 имеет лучшую эффективность по времени по сравнению с системой SI02, низкая дисперсия распределения его данных и медиана, превышающая SI02, подтверждают это утверждение.

На рис. 3.c по показателю "Ey-3-S Экономическая эффективность", система SI03 имеет более хорошую оценку экономической эффективности, чем система SI02, более концентрированное распределение данных и более низкое среднее значение, что приводит к выводу, что SI03 имеет более высокую оценку эффективности, чем SI02.

Рис. 3.d иллюстрирует показатели "Ey-4-S Коэффициент полезного времени". Этот коэффициент для системы SI02 статистически эквивалентен аналогичному показателю для системы SI03, он смещен в отрицательную сторону по распределению данных систем, средние значения для двух систем тоже эквивалентны, что подтверждает сделанное утверждение.

Рис. 3.e демонстрирует показатели "Ey-5-S Ненужные действия". Из него видно, что доля ненужных действий в системе SI03 меньше, чем эта доля в системе SI02. Среднее значение, достигнутое в системе SI03, существенное меньше, чем среднее, полученное в системе SI02, симметричное распределение в системе SI03, положительное смещение в системе SI02 и концентрированная дисперсия в обоих случаях позволяют нам заключить, что показатель "Ey-5-S Ненужные действия" для системы SI03 лучше, чем тот же показатель для системы SI02.

Анализ данных в табл. 7 и табл. 8, а также оценка диаграмм рис. 3 подтверждают, что два показателя эквивалентны или не показывают существенных различий между системами SI03 и SI02. В то же время по трем показателям значения для SI03 лучше, чем для SI02. Это позволяет сделать вывод, что SI03 обладает большей эффективностью в использовании, чем SI02.

Процент улучшения, согласно табл. 7 и 8, значительно повышен для показателя "Эффективность затраченного времени" (45%) и снижен для показателей "Рентабельность" (29%) и "Ненужные действия" (53%).

4.3 Показатели удовлетворенности

В табл. 9 и 10 приведены значения показателей удовлетворенности (см. табл. 4) для программных продуктов SI02 и SI03. Значения были рассчитаны с использованием критериев PSSUQ, а для получения значения, представляющего совокупный показатель по всем участникам, было рассчитано среднее значение. Эти же данные показаны на рис. 4.

Табл. 9. Показатели удовлетворенности для системы SI02
Table 9. Satisfaction metrics for software SI02

PU.	SUs-1-G	SUs-2-G	STr-1-G	SPI-1-G	SCo-1-G	SysU	InfoQ	InterQ
P01	3.25	6.00	2.00	4.00	3.00	2.83	3.50	4.00
P02	5.06	5.00	6.00	7.00	5.00	5.17	4.67	5.67
P03	5.25	4.00	5.00	5.00	5.00	5.00	4.83	6.00
P04	4.94	4.00	7.00	5.00	4.00	5.50	4.00	5.33
P05	3.75	3.00	3.00	5.00	2.00	2.50	5.17	3.67
P06	3.00	2.00	3.00	5.00	3.00	2.17	3.17	4.33
P07	3.19	5.00	2.00	3.00	3.00	2.50	3.17	4.33
P08	3.31	3.00	4.00	1.00	2.00	3.33	3.50	2.33
P09	3.81	5.00	5.00	4.00	4.00	4.00	3.83	3.67
P10	2.63	6.00	1.00	3.00	1.00	2.17	2.67	3.67
Среднее значение	3.82	4.30	3.80	4.20	3.20	3.52	3.85	4.30

Табл. 10. Показатели удовлетворенности для системы SI03
Table 10. Satisfaction metrics for software SI03

PU.	SUs-1-G	SUs-2-G	STr-1-G	SPI-1-G	SCo-1-G	SysU	InfoQ	InterQ
P01	3.13	4.00	4.00	4.00	3.00	3.17	3.00	3.33
P02	3.50	4.00	4.00	2.00	4.00	3.83	3.67	2.33
P03	2.25	5.00	2.00	2.00	2.00	1.67	2.50	3.00
P04	3.00	6.00	2.00	2.00	3.00	2.50	3.50	2.67
P05	3.25	3.00	3.00	3.00	3.00	3.17	3.50	3.00
P06	2.88	5.00	4.00	3.00	2.00	2.50	3.00	3.33
P07	1.63	2.00	1.00	1.00	1.00	1.50	1.83	1.33
P08	2.31	3.00	2.00	2.00	2.00	1.83	2.50	2.67
P09	4.25	5.00	5.00	4.00	3.00	3.00	5.33	4.67
P10	2.56	3.00	2.00	3.00	2.00	2.33	2.50	3.00
Среднее значение	2.88	4.00	2.90	2.60	2.50	2.55	3.13	2.93

На рис. 4 для сравнения показателей удовлетворенности использованы гистограммы. Оцениваемые системы обозначены полосами разных цветов, красный цвет выбран для системы SI02, синий для SI03. На рис. 4 показано, что показатели удовлетворенности SI03 лучше, чем эти показатели дляSI02, так как согласно критериям PSSUQ, более низкие баллы показывают более полное удовлетворение программной системой.

Рис. 5.a показывает данные по показателю “SUs-1-G Общая удовлетворенность”, по нему видно, что система SI03 позволяет пользователям достичь большего удовлетворения от работы по сравнению с системой SI02. Более низкое среднее значение, отрицательное смещение данных и меньшая дисперсия позволяет сделать заключение, что система SI03 полнее удовлетворяет пользователей, чем система SI02. Более того, можно сделать вывод, что разница между системами статистически значима, поскольку на рис. 5.a видно, что продолжение одной из медиан не пересекает прямоугольник другой.

Рис. 5.b построен на основании данных по показателю “SUs-2-G Удовлетворенность функциями”, этот показатель измеряется с помощью пункта 17 вопросника и связан с восприятием пользователем полноты функциональных возможностей. Рис. 5.b показывает, что пользователи считают функциональные возможности обеих систем SI02 и SI03 неполными, это видно из того, что полученные оценки в основном сосредоточены между значениями 3 (“частично согласен”) и 5 (“частично не согласен”). Близость значений дисперсии, разброс оценок и их среднее значение позволяют нам считать, что по показателю “SUs-2-G Удовлетворенность функциями” системы SI02 и SI03 эквивалентны.

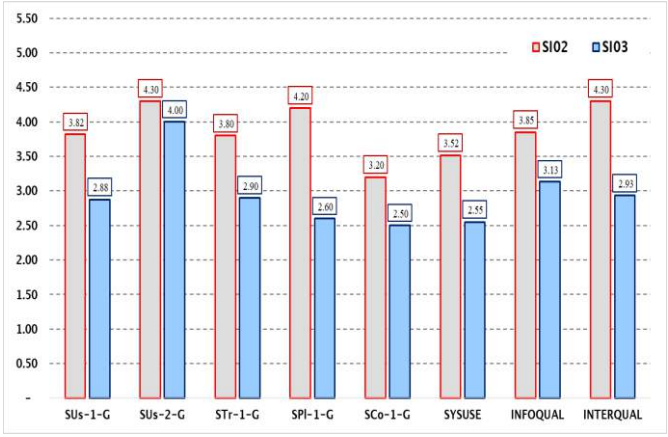
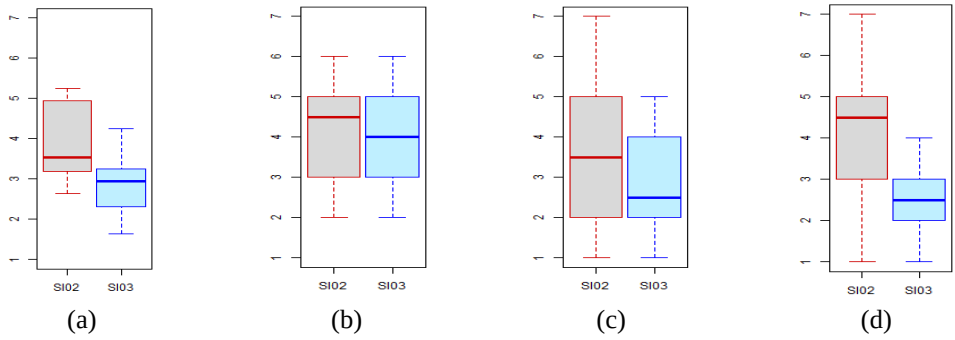


Рис 4. Сравнение показателей удовлетворенности, гистограмма средних значений
Fig 4. Satisfaction metrics comparison, bar chart of averages



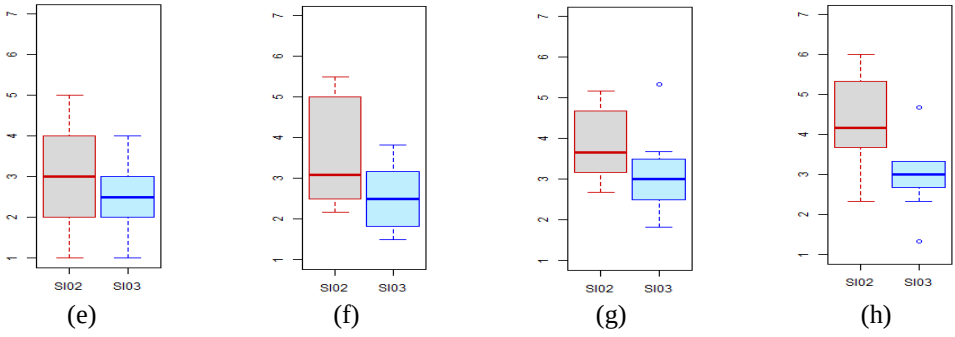


Рис 5.Сравнение показателей удовлетворенности для SI02 и SI03,
(a) "SUs-1-G Общая удовлетворенность", (b) "SUs-2-G Удовлетворенность функциями",
(c) "STr-1-G Доверие пользователей", (d) "SPl-1-G Удовольствие пользователей",
(e) "SCo-1-G Физический комфорт", (f) "SysUse Полезность системы",
(g) "InfoQual Качество информации", (h) "InterQual Качество интерфейса"
Fig 5. Satisfaction metrics comparison for SI02 and SI03,
(a) "SUs-1-G Overall satisfaction", (b) "SUs-2-G Satisfaction with features", (c) "STr-1-G User trust",
(d) "SPl-1-G User pleasure", (e) "SCo-1-G Physical comfort", (f) "SysUse System Usefulness",
(g) "InfoQual Information Quality", (h) "InterQual Interface Quality"

Рис. 5.c демонстрирует показатель “STr-1-G Доверие пользователей”, этот показатель измеряется с помощью пункта 18 вопросника, здесь видно, что пользователи больше доверяют системе SI03, чем системе SI02. Подтверждение этого утверждения находим в том, что дисперсия и разброс оценок при более низких абсолютных значениях тоже низки.

Рис. 5.d иллюстрирует показатель “SPl-1-G Удовольствие пользователя”, этот показатель измеряется с помощью пункта 13 вопросника. Видно, что пользователи воспринимают систему SI03 как более приятную, чем систему SI02. Подтверждение этого утверждения также находим в том, что дисперсия и разброс оценок при более низких абсолютных значениях низки. Кроме того, продолжение медианы значений для одной из систем не пересекает прямоугольник значений для другой системы, что делает разницу оценок статистически значимой, а вывод убедительным.

Рис. 5.e составлен по данным для показателя “SCo-1-G Физический комфорт”, этот показатель измеряется с помощью пункта 4 вопросника, из него видно, что пользователи считают, что система SI03 обеспечивает более высокий уровень комфорта, чем система SI02. Подтверждается это утверждение тем, оценки по этому показателю имеют меньший разброс, а также меньшую дисперсию при более низких абсолютных значениях.

Рис. 5.f составлен по данным для показателя “SysUse Полезность системы”. Этот рисунок показывает, что пользователи считают систему SI03 более полезной, чем систему SI02. Полученные оценки имеют меньший разброс и меньшую дисперсию при меньших абсолютных значениях, что подтверждает это утверждение.

На рисунке 5.g показан показатель “InfoQual Качество информации”. Легко видеть, что по мнению пользователей система SI03 предлагает информацию более высокого качества, чем система SI02. Можно видеть не очень большой разброс значений, меньшую дисперсию при небольших абсолютных значениях, симметричное распределение оценок системы SI03 по сравнению с положительно смещенным распределением оценок SI02. Продолжение линии медианы одной из систем не пересекает поле значений другой системы, что делает разницу показателей статистически значимой и вывод убедительным.

На рис. 5.h демонстрируется диаграмма для показателя “InterQual Качество интерфейса”. По ней видно, что пользователи считают, что у системы SI03 более хороший пользовательский интерфейс, чем у системы SI02. Здесь видны небольшой разброс значений, небольшое значение дисперсии при небольших абсолютных значениях показателя, симметричное распределение значений системы SI03 по сравнению с положительным смещением распределения значений для системы SI02. Продолжение линии медианы одной из систем не пересекает поле значений другой системы, что делает разницу показателей статистически значимой и вывод убедительным.

Анализ содержания табл. 9 и табл. 10, а также рис. 4 и рис. 5 показывают, что из восьми показателей, измеряющих удовлетворенность использованием, семь благоприятны для SI03, а по одному показателю система SI03 эквивалентна системе SI02. Это позволяет сделать вывод, что качество при использовании системы SI03 выше, чем качество при использовании системы SI02.

4.4 Угрозы валидности

Чтобы гарантировать валидность проведенного исследования, была проведена валидация требований к качеству программ и собранной информации, использовавшейся для расчета показателей. Валидация проводилась путем получения экспертных оценок с использованием критериев ясности, объективности, организованности, возможности, актуальности, достаточности, согласованности и непротиворечивости. Была применена методология, предложенная Эрнандесом-Ньето [32], где в качестве измерения использовался коэффициент валидности контента (Content Validity Coefficient, CVC). Коэффициент CVC оценивает степень согласия экспертов относительно каждого из пунктов и инструмента в целом. Был

получен коэффициент 0,8719, который квалифицирует приборы и конструкцию как валидные, сама процедура валидации была описана в [33].

Такие понятия, как качество при использовании, результативность при использовании и удовлетворенность при использовании, широко изучаются разработчиками программного обеспечения и специалистами по использованию программ. Эти концепции были пересмотрены, согласованы и стандартизированы в рамках серии стандартов ISO/IEC 25000.

- Внутренняя валидность. Это исследование сосредоточено на измерении качества программного обеспечения с использованием модели, предложенной ISO/IEC 25010. Для обеспечения валидности измерений были использованы: показатели и рекомендации из ISO/IEC 25022, стандартизированный вопросник PSSUQ и методика тестирования "Размышления вслух".

  Чтобы обеспечить валидность исследования, контекст, в котором проводится оценка программного обеспечения, в точности совпадает с реальным: физическая среда, компьютерное оборудование, графики работы, пользователи и даже мешающие элементы были такими же, как в реальной обстановке.

- Внешняя валидность. Исследование ограничено сравнением двух программных продуктов – информационных систем, поддерживающих жизненный цикл разработки программ в общественном учреждении, и результаты не поддаются обобщению. Он представлен научному сообществу разработчиков программного обеспечения в качестве эмпирического исследования, в котором устанавливается взаимосвязь между качеством требований и качеством при использовании программного продукта. Утверждения являются результатами описательного статистического анализа.

5. Завершающее обсуждение и будущая работа

В этом исследовании оценивалось и сравнивалось качество при использовании двух программных продуктов, которые участвовали в другом, ранее проведенном исследовании. Анализ показателей продемонстрировал, что программное обеспечение системы SI03 обладает лучшим качеством при использовании, чем SI02. В исследовании [14] было определено качество требований к программному обеспечению (SRQI), и система SI02 получила оценку SRQI, равную 0,5054, а система SI03 получила значение SRQI, равное 0,8495. Разница была объяснена улучшением процесса проверки требований в SI03. Мы считаем, что на разницу в качестве при использовании также повлияло внедрение методов проверки на этапе формирования требований к программам, аналогичных тем, что описаны в работе [34], подтверждающих, что качество процесса влияет на качество продукта, как указано в работе [29].

Анализ результатов показывает, что внедрение методов валидации на этапе разработки требований оказывает положительное влияние на качество в использовании. Результаты выражаются в терминах:

(i) точной и добросовестной работы с ними для достижения их целей,

(ii) оптимизации используемых ресурсов или повышения их производительности, и

(iii) удовлетворенности работой пользователя в отношении используемого программного обеспечения.

Все это вместе повлияло на то, что пользователи оценили систему SI03 лучше, чем систему SI02, что соответствует результатам исследования [35]. Подводя итог, можно утверждать, что валидация требований позволяет получить программное обеспечение с лучшим качеством при использовании.

Результативность, эффективность и удовлетворенность моделью качества при использовании ISO/IEC 25010 – это показатели, важные для сравнения качества при

использовании двух программных продуктов, это утверждение согласуется с выводами работ [36-37, 22].

Применение методологий "Размышления вслух" и PSSUQ подтверждает правильность исследования, и, в частности, PSSUQ позволяет избежать повторного возникновения вопросов и беспорядка в их постановке, решая одну из проблем измерения удовлетворенности, поднятых в работе [38].

При выборе показателей удовлетворенности на данном этапе была исключена группа показателей из стандарта ISO/IEC 25022, которые измеряют полезность; в качестве будущей работы предлагается измерять эти показатели путем автоматического сбора данных и статистического моделирования для анализа, предложенного в работе [39].

Наконец, в этом исследовании доступ к информации о конкретных примерах был ограничен. Было невозможно разработать экспериментальную модель, которая объясняла бы с помощью проверки гипотез взаимосвязь между валидацией требований к программному обеспечению и качеством при использовании.

Приложения A и B на испанском языке оригинала доступны по ссылке: https://drive.google.com/open?id=169Z4QhsO4nVxoRaDj_kEndoGS7JdyBey.

Список литературы / References

[1]. PCM: RM. N° 041-2017-PCM. Uso Obligatorio de la Norma Técnica Peruana NTP-ISO/IEC 12207:2016. , Peru (2017).

[2]. ISO/IEC: ISO/IEC TR 29110-1:2016 Systems and software engineering — Lifecycle profiles for Very Small Entities (VSEs) — Part 1: Overview, Geneva (2016).

[3]. Demirel, S.T., Das, R.: Software requirement analysis: Research challenges and technical approaches. In: 6th International Symposium on Digital Forensic and Security, ISDFS 2018 - Proceeding. pp. 1–6. Institute of Electrical and Electronics Engineers Inc. (2018). https://doi.org/10.1109/ISDFS.2018.8355322.

[4]. Atoum, I., Baklizi, M.K., Alsmadi, I., Otoom, A.A., Alhersh, T., Ababneh, J., Almalki, J., Alshahrani, S.M.: Challenges of Software Requirements Quality Assurance and Validation: A Systematic Literature Review, (2021). https://doi.org/10.1109/ACCESS.2021.3117989.

[5]. Anuar, U., Ahmad, S., Emran, N.A.: A simplified systematic literature review: Improving Software Requirements Specification quality with boilerplates. In: 2015 9th Malaysian Software Engineering Conference, MySEC 2015. pp. 99–105. Institute of Electrical and Electronics Engineers Inc. (2015). https://doi.org/10.1109/MySEC.2015.7475203.

[6]. García-Mireles, G.A.: Addressing product quality characteristics using the ISO/IEC 29110. In: Mejia J., Munoz M., Rocha Á., C.-M.J. (ed.) Trends and Applications in Software Engineering. Advances in Intelligent Systems and Computing. pp. 25–34. Springer, Cham, Sinaloa (2016). https://doi.org/https://doi.org/10.1007/978-3-319-26285-7_3.

[7]. Hussain, A., Mkpojiogu, E.O.C., Kamal, F.M.: The Role of Requirements in the Success or Failure of Software Projects. In: International Soft Science Conference (ISSC) in International Review of Management and Marketing. pp. 306–311. EconjournalsLimonluk Mah. 24117 sokSahinRezidance, A6 33100 Yenisehir, Langkawi Island, Kedah, Malaysia (2016).

[8]. Niazi, M., Mahmood, S., Alshayeb, M., Qureshi, A.M., Faisal, K., Cerpa, N.: Toward Successful Project Management in Global Software Development. International Journal of Project Management. 34, 1553–1567 (2016). https://doi.org/https://doi.org/10.1016/j.ijproman.2016.08.008.

[9]. Bhardwaj, M., Rana, A.: Key Software Metrics and its Impact on each other for Software Development Projects. International Journal of Electrical and Computer Engineering (IJECE). 6, 242–248 (2016). https://doi.org/https://doi.org/10.11591/ijece.v6i1.8247.

[10]. Duran Toro, A.: Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información, https://idus.us.es/handle/11441/15365, (2000).

[11]. Mishra, D., Abdalhamid, S.: Software Quality Issues in SCRUM: A Systematic Mapping. Journal of Universal Computer Science. 24, 1690–1716 (2018).

[12]. Atoum, I.: A novel framework for measuring software quality-in-use based on semantic similarity and sentiment analysis of software reviews. Journal of King Saud University - Computer and Information Sciences. 32, 113–125 (2020). https://doi.org/10.1016/j.jksuci.2018.04.012.

[13]. Salomón, S., Duque, R., Montaña, J.L., Tenés, L.: Towards automatic evaluation of the Quality-in-Use in context-aware software systems. Journal of Ambient Intelligence and Humanized Computing. (2022). https://doi.org/10.1007/s12652-021-03693-w.
[14]. Canchari, L., Dávila, A.: Requirements Validation in the Information Systems Software Development: An Empirical Evaluation of Its Benefits for a Public Institution in Lima. In: Mejia J., Muñoz M., Rocha Á., A.C.-M.J. (ed.) Trends and Applications in Software Engineering. CIMPS 2019. Advances in Intelligent Systems and Computing. pp. 23–35. Springer Nature Switzerland AG 2020, Guanajuato, Mexico (2020). https://doi.org/https://doi.org/10.1007/978-3-030-33547-2_3.
[15]. ISO/IEC: ISO/IEC 25022:2016 Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Measurement of quality in use. Springer US, Geneva (2016).
[16]. Kim, S.-H., Kim, W.-J.: Evaluation of Software Quality-in-use Attributes Based on Analysis Network Process. Cluster Computing. 22, 2101–2114 (2019). https://doi.org/https://doi.org/10.1007/s10586-018-2309-6.
[17]. ISO/IEC: ISO/IEC 25001:2014 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Planning and management, Geneva (2014).
[18]. Marín, B., Condori-Fernández, N., Pastor, O.: Calidad en Modelos Conceptuales: Un Análisis Multidimensional de Modelos Cuantitativos Basados en la ISO 9126. In: VIII Conferencia Anual de la Asociación Española de Métricas de Sistemas Informáticos. In Revista de Procesos y Métricas (RPM) - AEMES. pp. 153–167 (2007).
[19]. ISO/IEC: ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, Geneva (2011).
[20]. Fenton, N., Nell, M.: Software Metrics: Roadmap. In: Conference on the Future of Software Engineering - In 22nd International Conference on Software Engineering - ICSE. pp. 357–370, Limerick Ireland (2000). https://doi.org/https://doi.org/10.1145/336512.336588.
[21]. Estayno, M., Dapozo, G., Cuenca Pletsch, L., Greiner, C.: Modelos y Métricas para Evaluar Calidad de Software. In: XI Workshop de Investigadores en Ciencias de la Computación. pp. 382–388, San Juan, Argentina (2009).
[22]. Bevan, N.: Los Nuevos Modelos de ISO para la Calidad y la Calidad en Uso del Software. In: Calero, C., Moraga, A., and Piattini, M. (eds.) Calidad del Producto y Proceso Software. pp. 55–78. RA-MA, Madrid - España (2012).
[23]. Kurosu, M.: Usability, Quality in Use and the Model of Quality Characteristics. In: Human-Computer Interaction: Design and Evaluation - HCI. In Lecture Notes in Computer Science. pp. 227–237. Springer International Publishing Switzerland 2015, Los Angeles (2015). https://doi.org/10.1007/978-3-319-20901-2_21.
[24]. Atoum, I., Bong, C.H., Kulathuramaiyer, N.: Towards Resolving Software Quality-in-Use Measurement Challenges. Journal of Emerging Trends in Computing and Information Sciences. 5, 877–885 (2014).
[25]. Charters, E.: The Use of Think-aloud Methods in Qualitative Research An Introduction to Think-aloud Methods. Brock Education Journal. 12, 68–82 (2003). https://doi.org/https://doi.org/10.26522/brocked.V12I2.38.
[26]. Nielsen, J.: Thinking Aloud: The #1 Usability Tool, https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/, last accessed 2021/05/30.
[27]. Sauro, J.: 10 Things to Know About the Post Study System Usability Questionnaire, https://measuringu.com/pssuq/, last accessed 2019/03/03.
[28]. Allasi, D., Dávila, A.: Financial impact on the adoption of software validation tasks in the analysis phase: A business case. In: Mejia J., Muñoz M., Rocha Á., Quiñonez Y., C.-M.J. (ed.) Trends and Applications in Software Engineering. CIMPS 2017. Advances in Intelligent Systems and Computing. pp. 106–116. Springer International Publishing AG 2018, Zacatecas, Mexico (2018). https://doi.org/https://doi.org/10.1007/978-3-319-69341-5_10.
[29]. Pardo, C., García, F., Pino, F., Piattini, M.: Producto y Proceso: Una Relación Compleja en la Ingeniería de Software. El Hombre y la Máquina. 67–72 (2013).
[30]. Runeson, P., Höst, M.: Guidelines for Conducting and Reporting Case Study Research in Software Engineering. Empirical Software Engineering. 14, 131–164 (2009). https://doi.org/https://doi.org/10.1007/s10664-008-9102-8.
[31]. Genero Bocco, M., Cruz-Lemus, J.A., PiattiniVelthuis, M.G.: Métodos de Investigación en Ingeniería de Software. Ra-Ma, Madrid, España (2013).

[32]. Pedrosa, I., Suárez-Álvarez, J., García-Cueto, E.: Evidencias sobre la Validez de Contenido: Avances Teóricos y Métodos para su Estimación. Acción Psicológica. 10, 3–18 (2014). https://doi.org/https://doi.org/10.5944/ap.10.2.11820.
[33]. Canchari, L.: La validación de requisitos de software como base del éxito de los proyectos de sistemas informáticos desarrollados e implementados en la Comisión Nacional para el Desarrollo y Vida sin Drogas-DEVIDA. (2018).
[34]. Pérez-Verdejo, J.M., Sánchez-García, J., Ocharán-Hernández, J.O., Mezura-Montes, E., Cortés-Verdín, K.: Requirements and GitHub Issues: An Automated Approach for Quality Requirements Classification. Programming and Computer Software. 47, (2021). https://doi.org/10.1134/S0361768821080193.
[35]. González-Sánchez, J.-L., Montero-Simarro, F., Gutiérrez-Vela, F.-L.: Evolución del Concepto de Usabilidad como Indicador de Calidad del Software. El Profesional de la Informacion. 21, 529–536 (2012). https://doi.org/https://doi.org/10.3145/epi.2012.sep.13.
[36]. Covella, G., Olsina, L.: Medición y Evaluación de Calidad en Uso: Un Caso de Estudio para una Aplicación E-Learning. In: Castro, J., Cernuzzi, L., and Gordillo, S.E. (eds.) IX Conferencia Iberoamericana de Software Engineering - CIbSE. pp. 317–330. , La Plata, Argentina (2006).
[37]. Sierra González, J.C.: Métodos de Evaluación de Usabilidad para Sistemas de Información Web: Una Revisión. In: Conferencia Colombiana en Gestión de Sistemas de Información y de TIC - GSTIC. pp. 1–13. Universidad Nacional de Colombia, Manizalez (2013).
[38]. Hornbæk, K.: Current Practice in Measuring Usability: Challenges to Usability Studies and Research. International Journal of Human Computer Studies. 64, 79–102 (2006). https://doi.org/10.1016/j.ijhcs.2005.06.002.
[39]. Dávila Nicanor, L., MejiaAlvarez, P.: Evaluación de la Calidad de Software en Sistemas de Información en Internet. In: Congreso de IngenieríaElectrica, CInvestAv-IPN. pp. 1–11. , Zacatenco (2003).

Информация об авторах / Information about authors

Luis CANCHARI, Universidad Continental, Huancayo, Perú - Master's degree in Systems Engineering, Computer Engineer, Specialist in Government and Digital Transformation, Software Project Manager. Computer Engineer with a master's degree in Systems Engineering. Experienced in planning, directing, and executing information technology projects in the public sector. Responsible for implementing technological tools to optimize management of hospital centers in ESSALUD (Junín Network and INCOR). Part of a group of IT professionals who monitor servers and services of the SUNAT systems. Academic and researcher in software quality and the application of ICTs in educational innovation. Education degree holder, with experience in both basic and university teaching.

Луис КАНЧАРИ, Универсидад Континенталь, Хуанкайо, Перу - Магистр системной инженерии, инженер информатики, специалист по управлению и цифровому преобразованию, управляющий проектами программного обеспечения. Инженер-информатик с магистерской степенью в системной инженерии. Опыт в планировании, руководстве и выполнении проектов в области информационных технологий в государственном секторе. Ответственный за внедрение технологических инструментов для оптимизации управления больничными центрами в ESSALUD (Red Junín e INCOR). Был частью команды информатиков, отслеживающих серверы и службы систем SUNAT. Академический исследователь в области качества программного обеспечения и применения ИКТ в инновационном образовании. Преподавал в основной и высшей школе.

Paula Maria ANGELERI, Universidad de Belgrano, Buenos Aires, Argentina. Paula Maria Angeleri currently works at the Information Technology, University of Belgrano - Universidad de Belgrano, Argentina (UB). Paula does research in Teaching Methods, Software Engineering and Information Systems (Business Informatics). She is the research director of MyFEPS project (2010-current) MyFEPS': Methodologies and Framework for software product evaluation (Metodologías y Framework para la Evaluación de Productos Software). Angeleri is an expert in IT standardization, and she participates actively in ISO/IEC JTC1/SC7 Software and Systems Engineering Subcommittee. She is an expert in the field of Software Quality Standards (SW processes,

management systems, software products, IT services and information security, among others standards)

Паула Мария АНГЕЛЕРИ Университет Бельграно, Буэнос-Айрес, Аргентина. Паула Мария Ангелери настоящее время работает в Университете Бельграно, Буэнос-Айрес, Аргентина. Паула занимается исследованиями в области методов преподавания, программной инженерии и информационных систем (бизнес-информатики). Она является руководителем исследовательского проекта "МайФЭПС" (с 2010 года и по настоящее время) "Методологии и Фреймворк для оценки программного обеспечения". Ангелери является экспертом в стандартизации информационных технологий и активно участвует в ИСО/МЭК JTC1/SC7 Подкомитете по программной и системной инженерии. Она является экспертом в области стандартов качества программного обеспечения (процессы ПО, системы управления, программные продукты, ИТ-услуги и информационная безопасность, среди прочих стандартов).

Abraham Eliseo DÁVILA RAMÓN, Pontificia, Universidad Católica del Perú, Lima, Perú. Abraham Dávila is a researcher and professor at the Pontifical Catholic University of Peru (PUCP) since 2000. He is a principal research of the ProCalProSer Project (2013-2016 Phase I and 2017-2019 Phase II) and he is a founding member of the research group in software engineering (GIDIS-PUCP). Master's degree in Computer Science from PUCP and bachelor's degree in science with a major in Mechanical Engineering. He is member of the ISO/IEC working group of ISO/IEC 29110 standards. Their main areas of interest are computer quality (at the level of software process, products and service management) and education in software engineering.

Авраам Элисео ДАВИЛА РАМОН, Папский католический университет Перу, Лима, Перу. Авраам Давила — исследователь и профессор Папского католического университета Перу (PUCP) с 2000 года. Он является главным научным сотрудником проекта ProCalProSer (Фаза I 2013–2016 гг. и Фаза II 2017–2019 гг.), а также одним из основателей исследовательской группы в области разработки программного обеспечения (GIDIS-PUCP). Степень магистра компьютерных наук PUCP и степень бакалавра наук по специальности «Инженер-механик». Является членом рабочей группы ISO/IEC по стандартам ISO/IEC 29110. Основные области интересов — качество компьютеров (на уровне управления программными процессами, продуктами и услугами) и образование в области разработки программного обеспечения.

DOI: 10.15514/ISPRAS-2023-35(6)-2

Усовершенствование модели оценки нефункциональных требований, классифицирующей диапазоны одинакового размера с помощью алгоритма k-ближайших соседей

[1] *Ф. Вальдес-Соуто,* ORCID: 0000-0001-6736-0666 <*fvaldes@ciencias.unam.mx*>
[2] *Х. Валериано-Ассем,* ORCID: 0009-0008-6473-1271 <*jorge.valeriano@spingere.com.mx*>
[1] *Д. Торрес-Робледо,* ORCID: 0009-0002-7168-9709 <*dtorres@ciencias.unam.mx*>

[1] *Национальный автономный университет Мексики, факультет Науки, Мехико, Мексика.*
[2] *SPINGERE, Мехико, Мексика.*

Аннотация. Любой проект разработки программного обеспечения должен оценивать нефункциональные требования. Обычно для таких оценок руководители разработок программного обеспечения вынуждены обращаться к экспертам. Сегодня не существует стандартизированных программных инструментов для оценки нефункциональных требований, поэтому большинство моделей оценки сосредоточены на изучении функциональных требований пользователя и не учитывают нефункциональных, хотя оба эти термина часто являются субъективными. Целью этой статьи было показать, как, применяя нечеткую логику и алгоритм k-ближайших соседей, в мексиканской компании для решения некоторой конкретной проблемы была построена модель оценки нефункциональных требований, учитывающая указанную субъективность терминологии. Предложенная модель использовала базы данных реальных проектов этой частной мексиканской компании.

Ключевые слова: стандарт COSMIC, функциональные точки COSMIC, нефункциональные требования, функциональные требования, машинное обучение, KNN-классификация, оценка затрат, модель оценки состояния проектов в условиях неопределенности EPCU.

Для цитирования: Вальдес-Соуто Ф., Валериано-Ассем Х., Торрес-Робледо Д. Усовершенствование модели оценки нефункциональных требований, классифицирующей диапазоны одинакового размера с помощью алгоритма k-ближайших соседей. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 29–42. DOI: 10.15514/ISPRAS–2023–35(6)–2.

Improving a Model for NFR Estimation Classifying Equal Size Bands with KNN

[1] *F. Valdés-Souto,* ORCID: 0000-0001-6736-0666 <*fvaldes@ciencias.unam.mx*>
[2] *J. Valeriano-Assem,* ORCID: 0009-0008-6473-1271 <*jorge.valeriano@spingere.com.mx*>
[1] *D. Torres-Robledo,* ORCID: 0009-0002-7168-9709 <*dtorres@ciencias.unam.mx*>

[1] National Autonomous University of Mexico Science Faculty, CDMX, México.
[2] SPINGERE, CDMX, México.

Abstract. Any software development project needs to estimate Non-Functional Requirements (NFR). Typically, software managers are forced to use expert judgment to estimate the NFR. Today, NFRs cannot be measured, as there is no standardized unit of measurement for them. Consequently, most estimation models focus on the Functional User Requirements (FUR) and do not consider the NFR in the estimation process because these terms are often subjective. The objective of this paper was to show how an NFR estimation model was created using fuzzy logic, and K-Nearest Neighbors classifier algorithm, aiming to consider the subjectivity embedded in NFR terms to solve a specific problem in a Mexican company. The proposed model was developed using a database with real projects from a Mexican company in the private sector.

Keywords: COSMIC, CFP, NFR, FUR, ML, KNN classification, Effort estimation, EPCU.

For citation: Valdés-Souto F., Valeriano-Assem J., Torres-Robledo D. Improving a model for NFR estimation classifying equal size bands with KNN. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 29-42 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-2.

1. Введение

Оценка программного обеспечения привлекает внимание многочисленных исследователей с момента его появления в 1950-х годах и в течение более чем 70-летнего существования [1]. Эти исследователи обнаружили среди прочего, что правильная оценка важна для успеха разработки и оказывает значительное влияние на бюджетирование и планирование проектов в отрасли [2-4].

В литературе по оценке программного обеспечения за более чем шесть десятилетий был разработан широкий спектр методов оценки [5-6], множество классифицирующих методов оценки [7, 2, 4, 8-9] и топологии процесса оценки [10-11], однако, до сих пор не достигнуто соглашение по единой модели, которая устойчиво дает точные результаты для всех производственных проектов. Модели оценки тесно связаны с процессом измерения входных переменных, используемых для генерации оценки [12]. В результате рекомендуется использовать стандартизированные метрики [13].

Однако даже когда функциональный размер (количество функциональных возможностей) является центральным элементом оценки, полный объем программного проекта определяется, по крайней мере, функциональными (Functional User Requirements, FUR) и нефункциональными (Nonfunctional Requirements, NFR) требованиями пользователя. Вот почему некоторые авторы указывают, что с учетом NFR результат оценки улучшается, хотя эти элементы рассматриваются в очень немногих исследованиях [14].

При использовании NFR наблюдаются и некоторые проблемы, в частности, отсутствие консенсусной терминологии, что приводит к значительному разнообразию терминов NFR [15-16], а также к их субъективному пониманию [17].

В этой статье представлен практический подход, разработанный и примененный в мексиканской организации, которая использует для измерения функционального размера стандартную методологию консорциума COSMIC (COmmon Software Measurement International Consortium), хотя полученные оценки имеют высокую дисперсию. Проанализировав основную причину разброса, разработчики пришли к выводу, что она

заключается в разнообразии нефункциональных требований. Следовательно, компания должна количественно оценить NFR, чтобы оценить необходимые усилия по проекту.

Настоящая работа представлена по следующей схеме. В разделе 2 приведен краткий обзор моделей оценки состояния проектов в условиях неопределенности (Estimation of Projects in Contexts of Uncertainty, EPCU), классификатора для машинного обучения, использующего алгоритм k-ближайших соседей (K-Nearest Neighbors, KNN), методов классификации диапазонов равного размера и показывается необходимость измерения или оценки NFR. В разделе 3 показаны этапы разработки модели оценки усилий на реализацию NFR с описанием используемых методов анализа и алгоритмов. В разделе 4 представлены выводы, сделанные на основе проведенных исследований. В разделе 5 описываются направления будущих исследований и выявленные ограничения предлагаемой модели оценки.

2. Обзор литературы

2.1 Назначение программного обеспечения

Требования к проекту программной системы, названные Бульони “Требованиями пользователя” (User Requirements, UR), делятся на две группы [18]. Первая группа требований состоит из функциональных (связанных с реализуемым проектом) и нефункциональных (связанных с реализуемым продуктом) требований, вторая группа – требований относится к некоторым иным потенциальным результатам, связанным с проектом. Руководством по управлению проектами (Project Management Body of Knowledge, PMBOK), даются два определения: рамки или границы продукта и рамки проекта [19]. В этих двух подходах можно наблюдать прямую связь между требованиями к проекту и рамками продукта, а также между требованиями к продукту и рамками проекта.

Особый подход предложен стандартом COSMIC, точнее ISO/IEC 19761:2011 [16], где требования к проекту программной системы разделены на функциональные, нефункциональные, а также на требования и ограничения проекта (Project Requirements and Constraints, PRC).

2.2 Нефункциональные требования (NFR)

“Стандарт” для “измерения” NFR был опубликован в 2019 году под названием IEEE 2430™ (https://standards.ieee.org/ieee/2430/7045/) на основе рекомендаций Международной группы по изучению функциональных точек IFPUG (International Function Point Users Group), которая предложила свой подход к измерению нефункциональных требований к программному обеспечению [20], получивший наименование “процесс нефункциональной оценки программ” (Software Non-functional Assessment Process, SNAP). Такие функциональные точки получили наименование функциональных точек COSMIC (COSMIC functional points, CFP). Стандарт 2019 года является примером ошибочной метрологической практики, что делает его бесполезным: он не позволяет сравнивать разные проекты [21].

Терминология нефункциональных требований столь обширна, что, согласно [16], “мы далеки от полного, общепринятого списка” терминов NFR, большинство из них субъективны. Это затрудняет их оценку и придание им численных значений [17].

В работе [16] был представлен глоссарий NFR, включающий 60 терминов, разделенных на три основных класса: требования к качеству (систем программного обеспечения), требования к системной среде и технические требования. Этот глоссарий основан на нескольких документах, таких как [22-24, 14-15]. Важно учитывать, что некоторая часть нефункциональных требований в течение жизненного цикла проекта может быть проявлена в виде требований функциональных, а этот цикл можно измерять с помощью какого-либо стандартного метода, например, COSMIC.

2.3 Оценка программного обеспечения с учетом нефункциональных требований

Хотя оценки необходимых на разработку программ усилий существенно влияют на планирование проектов и составление их бюджетов, методы получения таких оценок все еще остаются предметом дискуссий [4]. Принято считать, что любая модель оценки тесно связана с процессом измерения входных переменных, используемых для генерации оценки [1]. Причина в том, что функциональный размер – единственная характеристика, которую можно последовательно оценить количественно [13].

Однако для создания моделей оценки, кроме функционального размера, используются и другие переменные. Функциональный размер – не единственная характеристика стоимости или трудозатрат, необходимых для реализации программного проекта [6]. Сравнительное исследование, проведенное в работе [4], показало, что “большинство исследований (71,67%) используют несколько факторов затрат, а не отдают приоритет конкретному”. Кроме того, Джонс в работе [25] рассматривает четыре важные характеристики, влияющие на методологию оценки программного обеспечения:

- опыт персонала,
- используемые технологии (языки программирования, инструменты поддержки и так далее),
- процесс разработки и
- среду программирования, в которой работает разработчик.

Хотя всеми признано, что для улучшения оценки следует учитывать нефункциональные требования, согласно [14], только 33% изученных моделей оценки использовали нефункциональные требования в систематическом обзоре. Авторы при этом утверждают, что принятие во внимание таких требований снижает ошибку оценки на 30%.

Авторы работы [38] отмечают, что существуют отдельные нефункциональные требования, которые, пусть и в разной степени, прямо влияют на каждое функциональное требование. В работе [39] упоминается, что программное обеспечение со сложной архитектурой и связанные с ним системные нефункциональные требования не могут быть оценены с использованием единого коэффициента производительности для всех архитектурных компонентов.

Одной из основных проблем при использовании нефункциональных требований является отсутствие единого определения терминов [15-16], а ограничение, заключающееся в том, что большинство терминов нефункциональных требований являются субъективными, затрудняет последовательную оценку конкретного термина. Предложение рассматривать влияние конкретного нефункционального требования на каждое функциональное требование, которое рассматривается в работах [38-39], является более сложно реализуемым, чем рассмотрение его влияния на уровне проекта в целом. В настоящем исследовании мы использовали подход на уровне проекта.

Проведенное нами исследование было посвящено поискам способа включать нефункциональные требования в модель оценки таким образом, чтобы более явно проявлялось их влияние на оценки трудозатрат или бюджета. Оно должно быть полезно руководителям корпоративных проектов по разработке программного обеспечения.

2.4 Модель нечеткой логики для оценки субъективных переменных, EPCU

Модель оценки проектов в условиях неопределенности (EPCU), разработанная Валдисом и другими [26] в 2007 году, позже была использована не для оценки трудозатрат, а для

приближенного определения функционального размера [27-29]. В модели представлены шесть шагов:

1. идентификация входных переменных,
2. определение выходных переменных,
3. создание правил влияния,
4. фаззификация,
5. оценка правил влияния и
6. дефаззификация.

Первые три шага называются “контекстом EPCU”, который определяется как “набор переменных (входных и выходных) и отношений, которые влияют на конкретный проект или набор сходных проектов” [30].

В данной статье показывается, как использование в конкретном проекте трех основных классов NFR, определенных в работе [16], позволяет продемонстрировать достоинства подхода EPCU [26, 30-31] для характеристики нефункциональных требований.

2.5 Метод k-ближайших соседей

Алгоритм классификации, основанный на k-ближайших соседях (KNN), представляет собой контролируемую модель машинного обучения, целью которой является характеристика объекта в соответствии с его признаками. Он также стремится маркировать экземпляры в соответствии с расстояниями между значениями их признаков [32].

Важно отметить, что этот алгоритм лучше всего работает, если признаков у объектов достаточно много, поскольку их совокупность помогает лучше характеризовать объекты [33].

Выбор оптимального k зависит от данных, используемых для обучения. Выбор большого значения k уменьшает влияние шума в данных, однако, одновременно увеличивает вероятность перекрытия выявляемых классов [34].

2.6 Классификация групп равного размера

Метод определения функционального размера COSMIC (ISO/IEC 19761) вводит методы аппроксимации и используется, когда требования известны лишь частично, или определены на ранних стадиях проекта. Среди этих методов был введен метод диапазонов равного размера (Equal Size Bands, ESB).

В подходе выделения диапазонов равного размера [35] значения анализируемой переменной (функционального размера) делятся на несколько диапазонов. Все они имеют одинаковый размер в единицах, характерных для анализируемой переменной [35-36].

3. Практический пример

Проблема, с которой столкнулась компания, в которой мы проводили исследование, очень часто выявляется на различных предприятиях, разрабатывающих программное обеспечение. Наша работа выполнялась на основе исследования, описанного в работе [37], но с использованием наших собственных данных.

Компания уже пользовалась методологией COSMIC в течение последних семи лет для измерения функционального размера и применяет формальные модели оценки функциональных требований. Тем не менее полученные оценки характеризуются высокой дисперсией. Проанализировав основную причину разброса, специалисты пришли к выводу, что это связано с воздействием нефункциональных требований. Обычно производители

программного обеспечения вынуждены принимать во внимание экспертные заключения, основанные на субъективности NFR. Однако при таком подходе нефункциональные требования не могут выражаться количественно, и эксперты полагают, что результат оценивающей процедуры повторить невозможно. Отсюда следует, что компании нуждаются в методах количественных оценок нефункциональных требований и затрат на их реализацию.

Наша компания регистрировала затраты на реализацию нефункциональных требований по реализуемым проектам, но не делала этого в разрезе каждого отдельного функционального требования. Таким образом, в данном тематическом исследовании нами использовался подход к оценке нефункциональных требований на уровне проекта, а не на уровне требований функциональных.

Вопрос, на поиски ответа на который направлено проведенное исследование, таков: *как оценить усилия, необходимые для разработки нефункциональных требований в проектах разработки программного обеспечения?*

Чтобы получить ответ на поставленный вопрос авторы использовали базу данных проектов компании. Релевантной информацией для анализа являлись идентификатор проекта и затраты на реализацию нефункциональных требований, выраженные в рабочих часах. В базе данных была обнаружена информация о 80 проектах, ранее выполненных компанией. Для разработки модели были использованы 57 проектов, остальные 23 проекта использовались для тестирования построенной модели.

3.1 Построение модели оценки нефункциональных требований (NFR)

Построение предлагаемой модели оценки нефункциональных требований выполняется в три этапа (рис. 1).

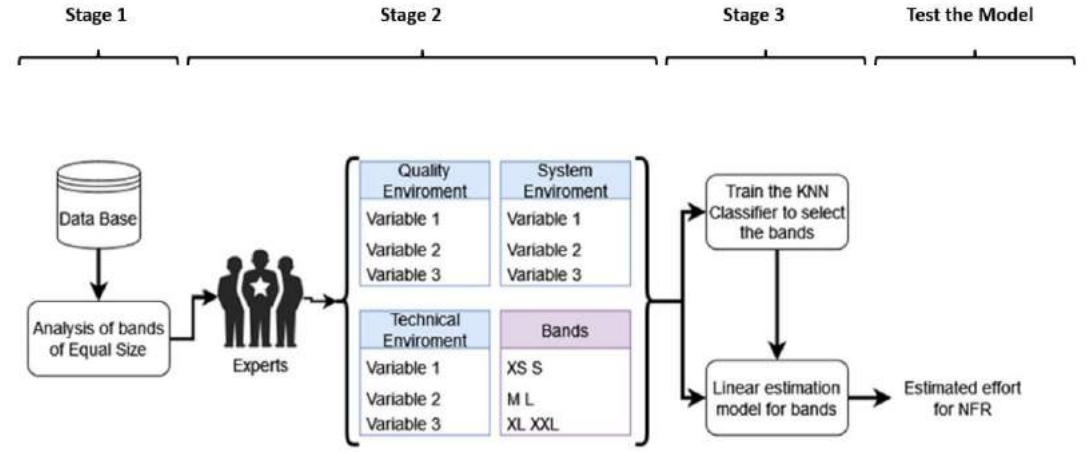


Рис. 1. Этапы разработки модели оценки затрат на нефункциональные требования
Fig. 1. Stages to develop the NFR effort estimation model

Первый этап заключается в отборе проектов, для которых реально выставлялись известные нефункциональные усилия. Анализ диапазонов равного размера (ESB) проводился на основе сведений о трудоемкости реализации требований, которая определялась в рабочих часах (WH). Этими классифицирующими диапазонами были выбраны следующие: XS (очень мало), S (мало), M (средне), L (много), XL (очень много), XXL (чрезвычайно много).

На втором этапе исследователи определили три контекста оценки проектов в условиях неопределенности EPCU. Группа экспертов заранее прошла обучение по оценке контекста EPCU и, приступив к работе, запросила входные переменные, связанные с оценкой индексов

NFR с использованием этих контекстов. При этом были получены результирующие количественные значения, соответствующие каждому контексту NFR и индексам NFR, то есть индексу требований к качеству (Quality Requirements Index, QRI), индексу требований к системной среде (System Environment Requirements Index, SERI) и индексу технических требований (Technical Requirements Index, TRI). Кроме того, экспертной группе было предложено приписать каждый проект к одному из диапазонов, ранее определенных с использованием экспертного заключения (табл. 1).

На третьем этапе выполняются два шага: во-первых, следует обучить классификатор KNN, во-вторых, разработать модель полилинейной регрессии с использованием индексов и с учетом выбранного диапазона в качестве входных переменных, при этом выходной переменной будут затраты, необходимые для реализации нефункциональных требований.

a. **Классификатор KNN**: для обучения модели этот блок в качестве признаков использует индексы требований SERI, TRI и QRI, результатом классификации является выбираемый диапазон.

b. **Линейный оценщик диапазона**: этот блок в качестве входных данных получает индексы SERI, TRI и QRI, а также диапазон, приписанный данному проекту классификатором KNN. Затем выбираются параметры, соответствующие выбранному диапазону: среднее значение, стандартное отклонение и сигма-фактор. Наконец, на основе трех влияющих индексов и параметров диапазона применяется многофакторная линейная модель, которая выдает значение оценки трудозатрат в рабочих часах.

Табл. 1. Второй этап оценки NFR
Table 1. Second stage for the NFR estimation

Проекты	Затраты на NFR (рабочие часы)	SERI	TRI	QRI	Диапазон, определённый экспертами
P1	1800	0.3	0.5	0.4	XL
P2	900	0.5	0.3	0.2	L
….	….	….	….	….	….
Pn	440	0.6	0.4	0.3	M

3.2 Определение диапазонов равного размера

На первом этапе, чтобы определить количество диапазонов по исходной базе данных, выполнялась классификация групп одинакового размера в соответствии с трудозатратами на реализацию нефункциональных требований. В табл. 2 показаны результаты анализа с распределением проектов по шести диапазонам трудозатрат на реализацию нефункциональных требований. В табл. 3 показаны значения статистических параметров, которые использовались моделью оценки для каждого отдельного диапазона.

Табл. 2. Анализ групп одинакового размера по данным усилий NFR. μ = среднее значение, σ = стандартное отклонение, NA = неприменимо
Table 2. Equal size band analysis on the NFR effort data. μ = mean, σ = standard deviation, NA = Not Applicable

№ п/п	Размер (час)	Диапазон XS		Диапазон S		Диапазон M		Диапазон L		Диапазон XL		Диапазон XXL	
		μ	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**
1	9094.7	159.6	203.8	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	4547.4	91.9	74.2	574.2	256.7	NA	NA	NA	NA	NA	NA	NA	NA
3	3031.6	70.4	55.4	277.5	90.9	771.8	208.5	NA	NA	NA	NA	NA	NA
4	2273.7	58.8	44.3	206.0	26.2	417.8	114.5	834.7	203.7	NA	NA	NA	NA
5	1818.9	51.0	36.4	178.0	22.9	279.9	61.0	556.0	62.1	952.1	17.1	NA	NA
6	1515.8	46.8	33.1	160.0	27.0	236.6	32.2	402.0	72.6	591.5	12.0	952.1	17.1

3.3 Определение контекстов для определения показателей влияния

Как ранее упоминалось, были определены три контекста EPCU, с помощью которых рассчитывалось влияние индексов каждой категории нефункциональных требований, определенных в работе [16]. Данные для определения контекстов приведены в табл. 4.

Согласно концептуальной модели оценки NFR (рис. 1), выходные переменные – это числа в диапазоне от 0 до 1, отражающие степень влияния той или иной категории требований.

Табл. 3. Параметры, используемые моделью оценки
Table 3. Parameters used by the estimation model

Диапазон	Среднее значение (**μ**)	Стандартное отклонение (**σ**)	Используемый сигма-фактор
XS	46.8	33.1	1
S	160.0	27.0	3
M	236.6	32.2	3
L	402.0	72.6	3
XL	591.5	12.0	3
XXL	952.1	17.1	3

Табл. 4. Спецификация входных переменных контекста в соответствии с работой [16]
Table 4. Context's input variables specification [16]

Выходная переменная	Диапазон выходной переменной	Входная переменная	Диапазон входной переменной	Лингвистические группы		
Индекс качества среды (QRI)	[0, 1]	Сложность в зависимости от типа приложения (домена)	[0, 5]	Low	Mid	High
	[0, 1]	Сложность настройки сред разработки, тестирования и выполнения	[0, 5]	Low	Mid	High
	[0, 1]	Сложность от количества видов пользователей	[0, 5]	Low	Mid	High
Индекс требований к системной среде (SERI)	[0, 1]	Сложность системной производительности	[0, 5]	Low	Mid	High
	[0, 1]	Сложность данных, поддерживаемых системой	[0, 5]	Low	Mid	High
	[0, 1]	Сложность из-за совместимости с другими системами	[0, 5]	Low	Mid	High
	[0, 1]	Сложность из-за требуемого уровня легкости использования	[0, 5]	Low	Mid	High
	[0, 1]	Сложность из-за требуемого уровня надежности	[0, 5]	Low	Mid	High
	[0, 1]	Сложность из-за требуемого уровня управления доступом	[0, 5]	Low	Mid	High
	[0, 1]	Сложность из-за требуемого уровня управления доступом	[0, 5]	Low	Mid	High
Индекс технических требований (TRI)	[0, 1]	Сложность операционной системы	[0, 5]	Low	Mid	High
	[0, 1]	Сложность базы данных	[0, 5]	Low	Mid	High
	[0, 1]	Сложность из-за ограничений операционной платформы	[0, 5]	Low	Mid	High
	[0, 1]	Сложность из-за требований разработки	[0, 5]	Low	Mid	High
	[0, 1]	Сложность проектирования интерфейсов	[0, 5]	Low	Mid	High
	[0, 1]	Сложность из-за выбранной архитектуры системы	[0, 5]	Low	Mid	High

3.4 Обучение классификатора KNN

Для обучения модели классификатора KNN мы будем использовать в качестве классифицирующих признаков три индекса, рассчитанные по рекомендациям EPCU (SERI, TRI и QRI), а метки для классификации будут задаваться диапазонами XS, S, M, L, XL, и XLL, которые были предложены группой экспертов.

После обучения классификатора мы получим модель, которая позволит классифицировать новые экземпляры по заданным параметрам (QRI, SERI, TRI).

3.5 Модель линейной оценки с использованием групп, выбранных моделью KNN

Согласно концептуальной модели оценки, показанной на рис. 1, модуль под названием "Линейная оценка в диапазонах" выполняет оценку трудозатрат на основе параметров:

a. Влияющие индексы: *QRI, SERI, TRI*

b. Диапазон, присвоенный проекту классификатором KNN

c. Статистические параметры выбранного диапазона по результатам табл. 3

Вычисление оценки трудозатрат и длительности выполняется этим модулем решением линейного уравнения, основанного на индексах влияния (QRI, SERI, TRI) и параметрах выбранного диапазона. Получающееся значение может варьироваться от низкого предела до верхнего, что определяется интервалом $[\mu-\alpha\sigma, \mu+\alpha\sigma]$, где μ — среднее значение затрат в выбранном диапазоне, σ – стандартное отклонение, α – сигма-фактор. Все эти параметры показаны в табл. 3. При вычислениях используется следующее уравнение:

$$Затраты = f(QRI, SERI, TRI) = 2/3\ (\alpha\sigma)(QRI + SERI + TRI) + \mu - (\alpha\sigma) \quad (1)$$

3.6 Результаты применения модели для оценки NFR

В итоге для проверки результатов модели оценки на соответствие реальным усилиям, требуемым для реализации NFR, мы выбрали 23 проекта. В табл. 5 приведены критерии качества, а результаты обработки набора тестовых данных приведены в табл. 6.

Видно, что более точные результаты получаются при использовании полной модели оценки NFR, которая использует индексы и автоматический выбор диапазонов (рис. 1). Результаты модели, которая использует индексы влияния только с использованием модели множественной линейной регрессии, несколько хуже.

Табл. 5. Критерии качества
Table 5. Quality criteria

	Предложенная полная модель оценки NFR (рис. 1)	Модель с использованием множественной линейной регрессией
Средняя величина относительной ошибки	**107.8%**	**139.5%**
Стандартное отклонение	**136.5%**	**162.2%**
Доля проектов с относительной ошибкой, меньшей 25%	**26.1%**	**21.7%**
Медианная величина относительной ошибки	**47.4%**	**71.2%**

Табл. 6. Набор тестовых данных
Table 6. Test data set

Проект	QRI	SERI	TRI	Прогноз диапазона	Фактический диапазон	Прогноз затрат	Фактические затраты
P01	0.18	0.13	0.3	M	M	179.04	153
P02	0	0.08	0.05	XS	XS	16.55	27
P03	0.2	0.16	0.21	M	M	176.47	185
P04	0.31	0.12	0.15	M	M	176.87	120
P05	0.34	0.22	0.37	L	M	200.02	65
P06	0.38	0.15	0.33	L	L	307.92	48
P07	0.15	0.11	0.15	S	XS	22.69	27
P08	0.12	0.2	0.07	M	S	100.06	27
P09	0.15	0.16	0.09	L	S	100.33	180

P10	0.2	0.26	0.08	L	M	174.78	36
P11	0.19	0.23	0.14	L	M	175.46	665
P12	0.12	0.07	0.06	XS	S	92.31	180
P13	0.39	0.36	0.33	M	L	340.03	200
P14	0.08	0.23	0.11	M	L	245.42	180
P15	0.14	0.4	0.21	M	L	293.39	540
P16	0.14	0.37	0.26	M	XXL	927.23	440
P17	0.62	0.79	0.43	M	M	258.78	320
P18	0	0.13	0	M	M	148.40	135
P19	0.18	0.32	0.18	L	L	282.65	500
P20	0.07	0.07	0.04	XS	S	88.58	27
P21	0.21	0.13	0.1	L	M	168.04	63
P22	0.05	0	0	S	S	81.72	55
P23	0.11	0.02	0.06	XS	S	89.52	85

4. Заключение

Модель оценки NFR, концепция которой проиллюстрирована на рис. 1, решает проблему оценки нефункциональных требований для организации, в которой проводилось исследование. Она допускает постоянное совершенствование, позволяющее повышать точность выполняемых оценок, поскольку она официально используется в организации и периодически перенастраивается.

Модель, разработанная для оценки NFR, является инновационной, поскольку она сочетает в себе широко известный в литературе набор нефункциональных требований с математическими элементами, которые позволяют эти требования оценить численно. В литературе нет описаний способов измерения подобных требований.

Используя информацию и модель, представленные в этой работе, можно предлагать локальные расширения метода измерения функционального размера COSMIC, вычисляя общий размер затрат на создание программных систем, включающий как функциональные, так и нефункциональные требования. При этом можно использовать следующее уравнение:

$$\textit{Размер программной системы} = CFP + SERI + TRI + QRI + \textit{Диапазон}$$

Эта информация помогает менеджерам определить влияние различных нефункциональных требований на конкретный проект и позволяет им генерировать количественную информацию для анализа этого влияния.

5. Будущие исследования

Чтобы подтвердить правильность предложенной модели оценки затрат на реализацию нефункциональных требований, мы намерены применить ее в других учреждениях.

Модель постоянно совершенствуется, это можно видеть на примере критериев качества, меняющихся по мере получения большего количества данных.

Использование различных найденных в литературе методов классификации, основанных на машинном обучении, позволит сравнить результаты и получить лучшее соответствие группам, описанным в этом тематическом исследовании.

Усовершенствовать модель можно, рассмотрев неоднородные архитектуры [39] и различные виды влияния нефункциональных требований на каждое из функциональных требований [38], сравнивая результаты с подходом на уровне проекта, предложенным в этой статье.

Список литературы / References

O. Fedotova, L. Teixeira, A.H. Alvelos, Software effort estimation with multiple linear regression: Review and practical application, J. Inf. Sci. Eng. 29 (2013) 925–945.

T.K. Lee, K.T. Wei, A.A.A. Ghani, Systematic literature review on effort estimation for Open Sources (OSS) web application development, in: FTC 2016 - Proc. Futur. Technol. Conf., IEEE, San Francisco, California, USA, 2016: pp. 1158–1167. https://doi.org/10.1109/FTC.2016.7821748.

P. Sharma, J. Singh, Systematic literature review on software effort estimation using machine learning approaches, in: Proc. - 2017 Int. Conf. Next Gener. Comput. Inf. Syst. ICNGCIS 2017, IEEE, Jammu, India, 2018: pp. 54–57. https://doi.org/10.1109/ICNGCIS.2017.33.

C.E. Carbonera, K. Farias, V. Bischoff, Software development effort estimation: A systematic mapping study, IET Res. Journals. 14 (2020) 1–14. https://doi.org/10.1049/iet-sen.2018.5334.

R. Silhavy, Z. Prokopova, P. Silhavy, Algorithmic optimization method for effort estimation, Program. Comput. Softw. 42 (2016) 161–166. https://doi.org/10.1134/S0361768816030087.

M. Durán, R. Juárez-Ramírez, S. Jiménez, C. Tona, User Story Estimation Based on the Complexity Decomposition Using Bayesian Networks, Program. Comput. Softw. 46 (2020) 569–583. https://doi.org/10.1134/S0361768820080095.

M. Jørgensen, M. Shepperd, A systematic review of software development cost estimation studies, IEEE Trans. Softw. Eng. 33 (2007) 33–53. https://doi.org/10.1109/TSE.2007.256943.

A. Abran, Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers, 1st ed., John Wiley & Sons, 2015.

S. Bilgaiyan, S. Sagnika, S. Mishra, M. Das, A systematic review on software cost estimation in Agile Software Development, J. Eng. Sci. Technol. Rev. 10 (2017) 51–64. https://doi.org/10.25103/jestr.104.08.

R. Britto, V. Freitas, E. Mendes, M. Usman, Effort estimation in global software development: A systematic literature review, Proc. - 2014 IEEE 9th Int. Conf. Glob. Softw. Eng. ICGSE 2014. (2014) 135–144. https://doi.org/10.1109/ICGSE.2014.11.

F. Valdés-Souto, Validation of supplier estimates using cosmic method, CEURInternational Work. Softw. Meas. Int. Conf. Softw. Process Prod. Meas. (IWSM Mensura 2019). 2476 (2019) 15–30.

F. Valdés-Souto, L. Naranjo-Albarrán, Improving the Software Estimation Models Based on Functional Size through Validation of the Assumptions behind the Linear Regression and the Use of the Confidence Intervals When the Reference Database Presents a Wedge-Shape Form, Program. Comput. Softw. 47 (2021) 673–693. https://doi.org/10.1134/S0361768821080259.

ISO/IEC, ISO/IEC 14143-1:2007 Information technology — Software measurement — Functional size measurement, (2007) 6. https://www.iso.org/standard/38931.html.

S. Silva, M. Cortes, Use of Non-functional Requirements in Software Effort Estimation: Systematic Review and Experimental Results, Proc. - 2017 5th Int. Conf. Softw. Eng. Res. Innov. CONISOFT 2017. 2018-January (2018) 1–9. https://doi.org/10.1109/CONISOFT.2017.00008.

European Cooperation for Space Standardization, Space Engineering: Software- Part 1 Principles and Requirements, (2005).

Common Software Measurement International Consortium, Guideline on Non-Functional & Project Requirements, (2015).

F. Valdés-Souto, A.S. Núñez-varela, H.G. Pérez-gonzález, Evaluating the software quality non-functional requirement through a fuzzy logic- based model based on the ISO / IEC 25000 (SQuaRE) standard, in: 2019 7th Int. Conf. Softw. Eng. Res. Innov., Conference Publishing Services (CPS), México, CDMX, 2019: pp. 16–25. https://doi.org/10.1109/CONISOFT.2019.00014.

L. Buglione, The Next Frontier: Measuring and Evaluating Non-Functional Productivity, Metr. Views, IFPUG Newsl. 6 (2012) 11–14. http://www.ifpug.org/Metric Views/MVBuglione.pdf.

Project Management Institute, A Guide to the Project Management Body of Knowledge, PMBOK, 5th ed., Project Management Institute, 2013.

C. Tichenor, A new software metric to complement function points: The software non-functional assessment process (SNAP), CrossTalk. 26 (2013) 21–26.
A. Abran, IEEE 2430 Non-Functional Sizing Measurements: A Numerical Placebo, IEEE Softw. 38 (2020) 113–120. https://doi.org/10.1109/MS.2020.3028061.
P. Lago, P. Avgeriou, R. Hilliard, guest editors' introduction Software Architecture: IEEE Softw. (2010) 20–24.
Y. Saito, A. Monden, K. Matsumoto, Evaluation of non-functional requirements in a request for proposal (RFP), in: Proc. 2012 Jt. Conf. 22nd Int. Work. Softw. Meas. 2012 7th Int. Conf. Softw. Process Prod. Meas. IWSM-MENSURA 2012, IEEE, 2012: pp. 106–111. https://doi.org/10.1109/IWSM-MENSURA.2012.23.
L. Chung, B. Nixon, E. Yu, J. Mylopoulos, Non-functional Requirements in Software Engineering, Kluwer Academic Publishing, 2000.
C. Jones, Estimating Software Costs: Bringing Realism to Estimating, Second, McGraw-Hill Companies, Inc., New York, N.Y., 2007.
F. Valdés-Souto, A. Abran, Industry Case Studies of Estimation Models Using Fuzzy Sets, in: Reiner Dumke (Ed.), Softw. Process Prod. Meas. Int. Conf. IWSM-Mensura 2007, UIB-Universitat de les Illes Baleares, Illes Baleares, Spain, 2007: pp. 87–101.
F. Valdés-Souto, A. Abran, Case Study: COSMIC Approximate Sizing Approach Without Using Historical Data, in: Jt. Conf. 22nd Int. Work. Softw. Meas. 2012 Seventh Int. Conf. Softw. Process Prod. Meas., IEEE, Assisi, Italy, 2012: pp. 178–189. https://doi.org/10.1109/IWSM-MENSURA.2012.34.
F. Valdés-Souto, A. Abran, COSMIC Approximate Sizing Using a Fuzzy Logic Approach: A Quantitative Case Study with Industry Data, in: F. Vogelezang, M. Daneva (Eds.), 2014 Jt. Conf. Int. Work. Softw. Meas. Int. Conf. Softw. Process Prod. Meas., Conference Publishing Services (CPS), Rotterdam (Netherlands), 2014: pp. 282–292. https://doi.org/10.1109/IWSM.Mensura.2014.44.
F.V. Souto, A. Abran, Improving the COSMIC approximate sizing using the fuzzy logic EPCU model, 2015. https://doi.org/10.1007/978-3-319-24285-9_13.
F. Valdés-Souto, A. Abran, Comparing the Estimation Performance of the EPCU Model with the Expert Judgment Estimation Approach Using Data from Industry, in: R. Lee (Ed.), Softw. Eng. Res. Manag. Appl. 2010, Springer-Verlag, Berlin, 2010: pp. 227–240.
F. Valdés-Souto, Design of a Fuzzy Logic Software Estimation Process, École De Technologie Supérieure, Université Du Québec, 2011.
Jacob Goldberger, Sam Roweis, Geoff Hinton, Ruslan Salakhutdinov, Neighbourhood components analysis, Adv. Neural Inf. Process. Syst. 17 (2005) 513–520.
T. Seidl, Nearest Neighbor Classification, Encycl. Database Syst. (2009).
https://doi.org/https://doi.org/10.1007/978-0-387-39940-9_561.
Scikit-Learn, 1.6. Nearest Neighbors, 2023. (n.d.). https://scikit-learn.org/stable/modules/neighbors.html#classification.
A. Abran, A. Lestherhuis, B. Reynolds, A. Sellami, H. Soubra, S. Trudel, F. Valdés-Souto, F. Vogelezang, Early Software Sizing with COSMIC: Experts Guide, 2020 (2020) 1–67.
https://doi.org/10.13140/RG.2.1.4195.0567.
L. Lavazza, S. Morasca, Empirical evaluation and proposals for bands-based COSMIC early estimation methods, Inf. Softw. Technol. 109 (2019) 108–125. https://doi.org/10.1016/j.infsof.2019.02.002.
B.R. Per Runeson, Martin Host, Austen Rainer, Case Study Research in Software Engineering: Guidelines and Examples, John Wiley & Sons, Inc., 2012. https://doi.org/10.1002/9781118181034.
Fellir, F., Nafil, K., & Touahni, R. (2015). Analyzing the non-functional requirements to improve accuracy of software effort estimation through case-based reasoning. 2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA). doi:10.1109/sita.2015.7358402.
van der Vliet, Eric & Nijland, René & Mols, Harry & Vries, Jelle & Poort, Eltjo & Vogelezang, Frank. (2017). A Shortcut to Estimating Non-Functional Requirements? Architecture Driven Estimation as the Key to Good Cost Predictions. 10.1145/3143434.3143440.

Информация об авторах / Information about authors

Франсиско ВАЛЬДЕС-СОУТО - имеет докторскую степень в области инженерии программного обеспечения по специальности "Измерение и оценка программного обеспечения" в Высшей технологической школе (ETS) в Канаде, две магистерские степени в

Мексике и Франции. Президент COSMIC. Доцент факультета наук Национального автономного университета Мексики. Основатель Мексиканской ассоциации метрик программного обеспечения (AMMS). Более 25 лет опыта в разработке критически важного программного обеспечения. К настоящему времени опубликовал более 50 научных работ, включая статьи в индексированных журналах, трудах научных конференций, книгах и главах книг. Является главным промоутером проекта изучения формальных метрик программного обеспечения в Мексике, продвигая COSMIC (ISO/IEC 19761) в качестве национального стандарта. Член Национальной системы исследователей (SNI). Область научных интересов: измерение и оценка программного обеспечения, применяемого для управления проектами программного обеспечения, управление тематикой, производительностью и экономикой разработок программного обеспечения.

Francisco VALDÉS-SOUTO – Doctor in Software Engineering with a specialty in Software Measurement and Estimation at the École de Technologie Supérieure (ETS) in Canada, two master's degrees in Mexico and France. President of COSMIC. Associate Professor of the Faculty of Sciences of the National Autonomous University of Mexico (UNAM). Founder of the Mexican Association of Software Metrics (AMMS). More than 25 years of experience in critical software development. He currently has more than 50 publications including articles in Indexed Journals, Proceedings, books and book chapters. He is the main promoter of the topic of formal software metrics in Mexico, promoting COSMIC (ISO/IEC 19761) as a National Standard. Member of the National System of Researchers (SNI). Research interests: software measurement and estimation applied to software project management, scope management, productivity and economics in software projects.

Хорхе ВАЛЕРИАНО-АССЕМ – Магистр компьютерных наук и инженерии в Национальном автономном университете Мексики, специалист-консультант по формальному измерению и оценке программного обеспечения с 2016 года. Сфера научных интересов: метрики программного обеспечения (COSMIC), модели оценки программного обеспечения, модели валидации программного обеспечения, оценка функциональных и нефункциональных требований, оценка эффективности проектов разработки программного обеспечения на основе метрик программного обеспечения, оценка качества программных продуктов.

Jorge VALERIANO-ASSEM - Master in Computer Science and Engineering from the National Autonomous University of Mexico, specialist consultant in formal software measurement and estimation since 2016. Areas of interest: Software metrics (COSMIC), Software estimation models, Software Validation Models, Estimation of Functional and Non-Functional Requirements, Evaluation of the Performance of Software Development Projects aligned to Software Metrics, Evaluation of the Quality of the Software Development Product.

Даниэль ТОРРЕС-РОБЛЕДО – магистрант Исследовательского института в области прикладной математики и систем, имеет ученую степень по программированию от научного факультета Национального автономного университета Мексики.

Daniel TORRES-ROBLEDO – Master student at Research Institute in Applied Mathematics and Systems, degree in Computer Science from Science Faculty of the UNAM.

DOI: 10.15514/ISPRAS-2023-35(6)-3

Оценки сложности программного обеспечения на основе косвенных связей

[1] *Х. Навас-Су*, ORCID: 0000-0003-3431-0122 <*jnavas@tec.ac.cr*>
[2] *А. Гонсалес-Торрес*, ORCID: 0000-0001-5427-0637 <*antonio.gonzalez@tec.ac.cr*>

[1] *Коста-Риканский технологический институт, компьютерная школа, Коста-Рика.*
[2] *Коста-Риканский технологический институт, факультет вычислительной техники, Коста-Рика.*

Аннотация. Разработка программного обеспечения может быть длительным и дорогостоящим процессом, требующим значительных усилий. Перед разработчиками часто ставят задачи по программированию или внесению изменений в существующие программы так, чтобы общая сложность не увеличилась. Понятно, что прежде, чем вносить какие-либо изменения, важно понять зависимости между компонентами программы. Однако по мере роста размеров программ менеджерам проектов становится все сложнее обнаруживать косвенные связи между компонентами. Эти скрытые связи могут усложнять систему, приводить к неточной оценке необходимых затрат и ставить под угрозу качество получающихся программ. Чтобы решить эти проблемы, данное исследование направлено на выработку набора мер, которые дополняют теорию измерений и выявляют скрытые связи между компонентами программного обеспечения, расширяя сферу применения, увеличивая эффективность и полезность общепризнанных метрик программного обеспечения. Исследование велось в двух главных направлениях: (1) как измерения косвенных зависимостей могут помочь разработчикам при сопровождении программ и (2) как метрики косвенных связей могут количественно оценивать сложность и размер программного обеспечения, используя взвешенные различия между методами. В исследовании представлен комплекс мер, призванных помочь менеджерам проектов и разработчикам в управлении проектами и их сопровождении. Используя возможности измерений косвенных связей, эти меры могут повысить качество и эффективность процессов разработки и поддержки программного обеспечения.

Ключевые слова: Косвенное связывание; сопровождение программного обеспечения; удобство сопровождения; метрики.

Для цитирования: Навас-Су Х., Гонсалес-Торрес А. Оценки сложности программного обеспечения на основе косвенных связей. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 43–74. DOI: 10.15514/ISPRAS–2023–35(6)–3.

Measuring Software Complexity using Indirect Coupling

[1] *J. Navas-Su*, ORCID: 0000-0003-3431-0122 <*jnavas@tec.ac.cr*>
[2] *A. Gonzalez-Torres*, ORCID: 0000-0001-5427-0637 <*antonio.gonzalez@tec.ac.cr*>

[1] *Computing School, Costa Rica Institute of Technology, Costa Rica*.
[2] *Computer Engineering Department, Costa Rica Institute of Technology, Costa Rica.*

Abstract. Software development can be a time-consuming and costly process that requires a significant amount of effort. Developers are often tasked with completing programming tasks or making modifications to existing code without increasing overall complexity. It is essential for them to understand the dependencies between the program components before implementing any changes. However, as code evolves, it becomes increasingly

challenging for project managers to detect indirect coupling links between components. These hidden links can complicate the system, cause inaccurate effort estimates, and compromise the quality of the code. To address these challenges, this study aims to provide a set of measures that leverage measurement theory and hidden links between software components to expand the scope, effectiveness, and utility of accepted software metrics. The research focuses on two primary topics: (1) how indirect coupling measurements can aid developers with maintenance tasks and (2) how indirect coupling metrics can quantify software complexity and size, leveraging weighted differences across techniques. The study presents a comprehensive set of measures designed to assist developers and project managers with project management and maintenance activities. Using the power of indirect coupling measurements, these measures can enhance the quality and efficiency of software development and maintenance processes.

Keywords: Indirect coupling**;** maintainability**;** software maintenance**;** metrics**.**

For citation: Navas-Su J., Gonzalez-Torres A. Measuring Software Complexity using Indirect Coupling. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 43-74 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-3.

1. Введение

Сопровождение программного обеспечения включает в себя внесение в него адаптивных, совершенствующих, корректирующих и профилактических изменений. По мере развития программного обеспечения его сложность и качество могут улучшаться или ухудшаться [1]. Таким образом, основными целями сопровождения программного обеспечения являются продление срока его службы и сохранение его целостности с течением времени [2].

Управление развитием системы – это задача, которую должны решить менеджеры и программисты [3]. Для изучения модификаций и их влияния на программное обеспечение требуются эффективные методы, такие как идентификация косвенных связей и измерение.

Отслеживать изменения и измерять их влияние позволяют графы косвенных связей [4]. Для отслеживания изменений в зависимостях программных компонентов друг от друга программистам необходим структурный подход, каким, например, является использование ориентированных графов. Для численной оценки вклада одного программного компонента в другие требуется иметь представление о всей системе в целом, которое поможет приписать элементам графа весовые значения. Кроме того, чтобы лучше понять влияние изменений, вносимых в программы, на ее характеристики, разработчики должны иметь возможность сравнивать метрические показатели до и после внесения изменений.

Построенный граф позволяет отслеживать эффекты, вызванные изменениями одного компонента, но проявляющиеся в других компонентах, а также измерять потенциальное влияние планируемой модификации. Этот метод отслеживает связи между компонентами, которые выстраиваются в цепочки, и анализирует эти связи. Таким образом, в результате можно получить представление о внутренних зависимостях между применяемыми в программах методами, а также о том, как эти методы совместно выполняют те или иные функции системы.

Помощь при построении вышеупомянутой структуры может оказать обнаружение явных (прямых, непосредственных) связей (Direct Coupling, DC) [5-10], которые могут существовать между сущностями программ, такими как модули, классы, объекты, методы или элементы данных. Явные связи обычно классифицируются по той силе, с которой элементы соединены между собой [11, 12]. Комбинирование формаций явных связей создает конфигурацию сети косвенных связей (Indirect Coupling, IC), упрощая анализ и способствуя пониманию влияния изменений на другие компоненты системы [13-16]. Прямые связи транзитивно замыкаются на косвенных связях.

Следующие разделы настоящей статьи построены таким образом: в разделе 2 представлена постановка исследовательской задачи, в разделе 3 объясняются некоторые основы теории измерений. Далее в разделе 4 представлены расчетные критерии метрик, в разделе 5 описаны методы сбора данных, в разделе 6 представлены результаты, а в разделе 7 обсуждаются основные выводы.

2. Постановка задачи

Изучение косвенных связей (IC) включает в себя определение концепций [17, 18], теоретических основ [8], методов графов зависимостей [19–21], метрик [22–27], методов извлечения и обнаружения графов неявных зависимостей (косвенных связей) [12, 28] и создание аналитических инструментов [29–31]. Однако многие исследовательские подходы сосредоточены исключительно на идентификации графов косвенных связей и расчете метрических показателей на основе подсчета входящих и исходящих парных связей, передачи параметров и ссылок на элементы данных. Эти подходы не учитывают различия в степени вклада компонентов в исследуемый элемент.

Для восполнения этого пробела, необходимо учитывать такие понятия, как хрупкость и жесткость (иначе – рискованность), которые напрямую связаны с косвенными связями [32]. Хрупкость означает зависимость объекта от множества других компонентов, что делает его уязвимым для изменений, вызванных артефактами, от которых он зависит. Напротив, жесткость (или рискованность) отражает влияние модификаций объекта на зависимые от него артефакты. Таким образом, хрупкость элементов растет с увеличением количества влияющих на них [32], а их жесткость растет с увеличением количества зависимых от них артефактов.

В этой статье предлагается метод обнаружения графов косвенных связей (indirect coupling graph, ICG), а также набор метрик для измерения вклада в функциональные возможности программного обеспечения. Графы косвенных связей – это направленные ациклические графы (directed acyclic graph, DAG), которые фиксируют зависимости между методами, используемыми в программе в конкретной ситуации. Вес графа косвенных связей равен весу его корневого узла, вычисляемому рекурсивно путем суммирования весов всех узлов, ведущих от корня к листьям графа. Аналогичным образом, вес некоторого узла в этом графе рекурсивно рассчитывается как сумма весов достигаемых из него элементов. Следовательно, сложность метода прямо пропорциональна количеству факторов, как прямых, так и косвенных, от которых он зависит, а метод с многочисленными уровнями зависимостей, вероятно, будет более сложным, чем метод с меньшим количеством зависимостей.

Итак, это исследование предлагает подход к построению графов косвенной связи на уровне методов и определяет набор метрик для измерения сложности и размера элементов, основанных на концепциях жесткости и хрупкости [32]. Подход позволяет анализировать распространение изменений и ошибок, учитывать влияние компонентов на их партнеров по графу косвенной связи. В этом подходе для выявления и измерения косвенной связи выбирается степень детализации на уровне методов классов, поскольку в объектно-ориентированном программировании именно методы классов являются базовыми функциональными единицами. Методы – это сущности, на основе которых программисты, организуя их правильное взаимодействие, создают сложную функциональность.

Метрики программного обеспечения играют решающую роль в разработке программного обеспечения; их разработка требует высокого уровня теоретической и математической строгости. Достоверность результатов, описанных в данной статье, подтверждается той теоретической основой, которая положена в основу предлагаемого набора метрик, а также использованием общепризнанных стандартов [7, 33]. Формальная теория измерений применяется для определения набора мер на уровне детализации метода, основанного на концепциях устойчивости и неустойчивости косвенной связи [34].

В исследовании с некоторыми модификациями используются теоретический подход и обозначения, предложенные в работе [7], а также задействованы критерии проверки метрик для оценки и объяснения требуемых качеств (см. раздел 4). Все предлагаемые метрики вводятся в разделе 3, где уделяется особое внимание их формальным определениям, теоретическим основам и аналитическим оценкам.

Для демонстрации свойств и приложений метрик в этой статье использованы экспериментальные данные, полученные из коммерческих проектов и проектов с открытым исходным кодом. В разделе 5 описана методика сбора эмпирических данных, алгоритм и программный инструментарий, разработанный специально для этой цели, там же представлен набор данных используемых программных систем. Кроме того, в разделе 7 по каждой метрике даны основанные на собранных статистических данных рекомендации, предназначенные для руководителей проектов и разработчиков.

В статье тщательно и строго представлен подход к разработке метрик программного обеспечения с упором на теоретические основы, критерии проверки и экспериментальные данные. Предлагаемые метрики могут помочь оценить качество программного обеспечения, выявить потенциальные риски и уязвимости и принять обоснованные решения в проектах разработки программного обеспечения. Таким образом, данное исследование затрагивает следующие исследовательские вопросы.

RQ1. Как измерить сложность и размер программного обеспечения с помощью косвенных связей, чтобы воспользоваться преимуществами взвешенных различий между методами?

RQ2. Как метрики косвенной связи могут помочь программистам выполнять задачи по сопровождению программного обеспечения?

3. Теоретическая основа метрики косвенных связей

Пусть $S \in \mathbb{S}$ – множество всех программных систем, и пусть $S \in \mathbb{S}$ – некоторая программная система. Система S имеет ациклический граф потока управления $G = (V, A)$. Этот граф описывает поток управления между каждой парой методов в S, причем поток управления здесь означает передачу сообщений или обращения к методу. Следовательно, $V = \{v_1, v_2, v_3, \ldots\}$ – это конечное множество, содержащее все методы, объявленные в исходном коде S, а $A = \{a_1, a_2, a_3, \ldots\}$ – это конечное множество, содержащее информацию о потоке управления S. Каждая дуга $a_k \in A$ представляет собой обращение одного из методов исходного кода S к другому, например $a_k = (v_i, v_j)$, где $v_i, v_j \in V$, $v_i \neq v_j$ и $v_i \rightarrow v_j$.

Каждую систему S можно обобщенно представлять как эмпирическую систему отношений E [7]. Формальное определение этой эмпирической реляционной системы представляет собой:

$$E \equiv (V, T, ER, EO) \quad (1)$$

Первый компонент $V = \{v_1, v_2, v_3, \ldots\}$ из E в (1) – это то же самое множество V всех методов системы S.

Второй элемент $T = \{t_1, t_2, t_3, \ldots\}$ представляет собой множество транзитивных замыканий [35, p. 353], каждое t_i есть транзитивное замыкание подграфа косвенных связей ICG_i для соответствующего метода v_i, где $1 \leq i \leq |V|$. Мощность этого множества равна мощности V, то есть $|V| = |T|$. Транзитивные замыкания в T могут быть *прямыми транзитивными замыканиями* или *обратными транзитивными замыканиями*, в зависимости от направления, в котором рассматривается транзитивность. В первом случае прямой подграф $ICG_i^f = \left(V_i^f, A_i^f\right)$ содержит каждый простой путь из G, начинающийся с v_i. Согласно этому определению, $t_i^f = ICG_i^{f*} = \left(V_i^f, A_i^{f*}\right)$ это такой граф, что $V_i^f \subseteq V$, и A_i^{f*} содержит ребро (v_j, v_k), если и только если в ICG_i^f существует простой путь от v_j к v_k. В другом случае обратный подграф $ICG_i^r = (V_i^r, A_i^r)$ содержит все простые пути из G, заканчивающиеся в v_i. По этому определению, $t_i^r = ICG_i^{r*} = (V_i^r, A_i^{r*})$ это такой граф, что $V_i^r \subseteq V$, и A_i^{r*} содержит ребро (v_j, v_k), если и только если в ICG_i^r существует простой путь от v_j к v_k.

В парадигме объектно-ориентированного программирования мы управляем сложностью программных систем, используя модульность, инкапсуляцию, сокрытие информации и механизмы повторного использования кода. Строительными блоками этой парадигмы

являются абстракции реальных объектов, состояние и поведение которых моделируются с помощью атрибутов и методов. Объекты в этой парадигме взаимодействуют посредством передачи сообщений, так что, когда метод v_i отправляет сообщение другому методу v_j, он запускает процесс обмена сообщениями, который следует несколькими путями, достигая некоторых или всех методов в ICG_j.

Третий элемент тождества (1) $ER = \{er_1, er_2, er_3, \ldots\}$ представляет собой конечное множество действительных эмпирических отношений между каждой парой методов из V. К таким эмпирическим отношениям, сравнивающим размер или сложность двух методов относятся следующие:

- *больше,*
- *меньше,*
- *такой же большой, как,*
- *сложнее,*
- *менее сложный,*
- *такой же сложный, как.*

Например, если количество строк кода метода v_i меньше количества строк кода метода v_j, тогда v_i меньше, чем v_j в пересчете на строки кода, аналогично, если число цикломатической сложности метода v_j больше, чем число цикломатической сложности метода v_j, тогда v_j сложнее, чем v_j с точки зрения цикломатической сложности.

Наконец, последний компонент тождества (1) $EO = \{eo_1, eo_2, eo_3, \ldots\}$ представляет собой набор допустимых эмпирических бинарных операций над любой парой методов из V. Над парой методов v_k и v_l разрешены такие операции, как операции *добавить вызов* и *удалить вызов* от v_k к v_l. Предположим, мы выбираем методы $v_k, v_l \in t_i$ и выполняем операцию *удалить вызов* от v_k к v_l. С одной стороны, если удаленное ребро не отсоединяет ни один подграф от ICG_i, тогда t_i остается неизменным, но с другой стороны, если удаленное ребро изолирует подграф от подграфа зависимостей, то t_i меняется, поскольку теперь у элемента меньше методов и ребер. Аналогичный эффект возникает при добавлении вызова, но на этот раз либо t_i не меняется, либо на графе появляются новые методы и новые ребра.

Введем над множеством $\mathbb{S}$ бинарное отношение $\triangleright$, истинное для произвольных, но разных систем $S_1, S_2, S_3 \in \mathbb{S}$, обладающее такими свойствами:

Полнота: $(S_1 \triangleright S_2) \vee (S_2 \triangleright S_1)$

Транзитивность: $(S_1 \triangleright S_2) \wedge (S_2 \triangleright S_3) \Rightarrow (S_1 \triangleright S_3)$

Например, предположим, что операция $\triangleright$ относится к сложности, тогда свойство полноты позволяет утверждать, что всегда либо S_1 сложнее S_2, либо S_2 сложнее S_1. Точно также, если S_1 сложнее S_2 и S_2 сложнее S_3, тогда с учетом свойства транзитивности S_1 по определению сложнее S_3. Свойства полноты и транзитивности также должны соблюдаться, когда аргументами введенной операции являются не системы, а методы. Эти свойства также требуют, чтобы сложность или размер систем или методов не были одинаковыми.

Ранее введенная нами *эмпирическая система отношений* E из (1) транслируется в формальную систему отношений F [7], что и показано в следующем формальном определении:

$$F \equiv (M, IC, FR, FO) \tag{2}$$

Здесь первый аргумент $M = \{m_1, m_2, m_3, \ldots\}$ – это набор значений метрик, рассчитанных с помощью методов в V_i где для каждого метода выполняется $v_i \in V$. Формула этой метрики такова:

$$m_i = \sum_{\forall w \in V_i} \mu(w) \tag{3}$$

Множество V_i представляет собой множество методов в транзитивном замыкании t_i, построенном над m_i, а μ – это базовая метрика, например, количество строк кода или численное значение цикломатической сложности. Каждая метрика m_i является значением метрики для соответствующего метода v_i, поэтому $|M| = |V|$, то есть оба множества имеют одинаковую мощность.

Второе множество из (2) $IC = \{ic_1, ic_2, ic_3, \ldots\}$ содержит множества всех косвенно связанных методов для каждого $v_i \in V$, включая метод v_i. Каждое множество $ic_i \in IC$ совпадает с множеством V_i соответствующего транзитивного замыкания (имея в виду, что $t_i = (V_i, A_i^*)$). Множество ic_i совпадает с множеством V_i^f *прямого транзитивного замыкания* t_i^f, в этом случае используется обозначение ic_i^f, или множеству V_i^r *обратного транзитивного замыкания* t_i^r, при используемом обозначении ic_i^r.

Третий элемент формулы (2) $FR = \{fr_1, fr_2, fr_3, \ldots\}$ это конечное множество правильных *формальных отношений* между каждой парой метрик из M. Формальными отношениями, которые сравнивают размер или сложность двух методов, являются отношения <, =, и >. Например, если M включает значения метрики цикломатической сложности, и значение цикломатической сложности для метода v_i меньше, чем число цикломатической сложности другого метода v_j, тогда отношение $m_i < m_j$ истинно. Аналогично, если M – это размер кода, и число строк для v_i, больше, чем число строк v_j, тогда истинно отношение $m_i > m_j$.

В нашей работе мы рассчитываем метрики косвенных связей, детализируя их на уровне метода, и элементом, используемым в этом исчислении, является связность объектов, то есть прямое или обратное транзитивное замыкание. Наши метрики измеряют размер и сложность скрытых или многоуровневых функций. Эта скрытая сложность является результатом соблюдения таких принципов проектирования программного обеспечения, как инкапсуляция и повторное использование кода.

Например, размер или сложность любого конкретного метода можно измерить путем извлечения показателей из статической структуры его исходного кода (например, таких как параметры шаблонов FAN-IN и FAN-OUT, количество методов NOM, количество строк кода LOC и значение цикломатической сложности Маккейба CYC [36]). Набор вычисляемых метрик можно расширить, определив пользовательские функции вычисления метрик и подключив их к системе.

Следовательно, для любого метода $v_i \in V$, мы могли бы использовать конечный набор метрик, связанных с размером или сложностью метода. Чтобы представить конкретное значение индивидуальной метрики для метода v_i, мы используем выражения FAN-IN_i, FAN-OUT_i, NOM_i, LOC_i, и CYC_i. Эти отдельные показатели комбинируются для получения показателей косвенной связи. Стоит напомнить, что метод v_i косвенно связан с другим методом v_j, если существует хотя бы один путь $p = \langle v_1 \cdots v_l \rangle$, который содержит l таких методов, как $v_1 = v_i$, $v_l = v_j$, $v_k \rightarrow v_{k+1}$ для $k \in \{1 \cdots l - 1\}$, и $\{(v_1, v_2), (v_2, v_3), \cdots, (v_{l-1}, v_l)\} \subseteq t_i^f$, то есть v_j является одним из методов прямого транзитивного замыкания для v_i).

Наконец, $FO = \{fo_1, fo_2, fo_3, \ldots\}$ это конечное множество допустимых формальных бинарных операций над любой парой значений метрики из M. Формальными бинарными операциями, разрешенными для двух значений метрики $m_i, m_j \in M$ являются операции + и -. Предположим, мы выполняем операцию + над значениями метрики $m_i, m_j \in M$, которая является следствием добавления к A вызова метода (v_i, v_j). Здесь мы предполагаем, что цикл при этом не возникаеи. С одной стороны, если $ic_i \cap ic_j = \emptyset$ (при $ic_i = V_i$ и $t_i = (V_i, A_i^*)$), то есть добавленные значения метрики принадлежат несвязанным методам $v_i, v_j \in V$, то новое значение метрики для v_i равно $m_i' = m_i + m_j$. С другой стороны, если $ic_i \cap ic_j = ic_j$, то есть все методы из ic_j входят ic_i, новые методы не добавляются к ic_i и $m_i' = m_i$. Следовательно, значение m_i' находится в диапазоне $[m_i \ldots m_i + m_j]$, а значение $ic_i \cap ic_j \neq \emptyset \wedge ic_i \cap ic_i \neq ic_j$ истинно, то есть только подмножество ic_j

находится в ic_i. Аналогичные рассуждения приводят к выводу, что $m_i' \in [m_j \dots m_i + m_j]$, когда к A вместо ребра (v_i, v_j) добавляется ребро (v_j, v_i).

Эмпирическая система E преобразуется в формальную систему F путем реализации метрики μ. Каждому элементу v_i из E предназначен соответствующий элемент m_i в F, то есть, $m_i = \mu_i = \mu(v_i)$. Выбранный метод трансляции гарантирует, что понятия, лежащие в основе эмпирических отношений, остаются неизменными. Подобное преобразование позволяет автоматизировать поиск и сравнение самых сложных или крупных методов, которые, вероятно, имеют больше уровней зависимостей. Кроме того, становится возможным автоматически находить те методы, изменение которых рискованно из-за высокого уровня их повторного использования.

4. Расчетные критерии метрик

Вейкер в работе [34] предлагает систему оценок, обладающую девятью свойствами. Предложенный нами подход призван создать основу для измерения и сравнения одних предложенных метрик с другими. Таким образом, данное исследование представляет собой адаптацию свойств системы оценки Вейкер. Полученные данные перечислены ниже.

Свойство 1. Для любого метода $v_i \in V$ и метрики μ, должен существовать другой метод $v_j \in V$ такой, что $\mu(v_i) \neq \mu(v_j)$. Следовательно, значение метрики не будет одинаковым для всех методов системы. Метрика, которая считает все методы одинаковыми по размеру или сложности, бесполезна.

Свойство 2. Требуется, чтобы существовало только конечное число методов, имеющих одно и то же значение метрики. Поскольку каждая программная система всегда имеет лишь конечное число методов, мы решили проблему с циклами в графе G, превратив сильно связные компоненты (strongly connected components, SCC) в одиночные метаузлы SCC. Значения метрик размера и сложности SCC становятся суммой значений метрик их компонентов. Следовательно, каждый метод будет иметь только одно значение для каждой метрики, и этому свойству будет удовлетворять любой измеряемый метод.

Свойство 3. Два разных метода $v_i, v_j \in V$ могут иметь $\mu(v_i) = \mu(v_j)$. Другими словами, два разных метода из одной и той же системы могут иметь одно и то же значение метрики (то есть оба метода имеют одинаковый размер или сложность, в зависимости от метрики μ).

Свойство 4. В случае, если два разных метода $v_i, v_j \in V$ реализуют одну и ту же функциональность, то не обязательно истинно, что $\mu(v_i) = \mu(v_j)$. Значения метрики зависят от деталей реализации соответствующих методов. Для расчета значений метрик не имеет значения то факт, что оба метода генерируют один и тот же набор выходных данных и при одном и том же наборе входных данных имеют в системе одинаковые побочные эффекты.

Свойство 5. Для всех различных методов $v_i, v_j \in V$ должно выполняться: $\mu(v_i) \leq \mu(v_i \oplus v_j) \wedge \mu(v_j) \leq \mu(v_i \oplus v_j)$. Здесь, $v_i \oplus v_j$ представляет собой перестроение графов косвенной связи v_i и v_j (то есть ICG_i и ICG_j), которое получается добавлением связи для двух методов, в каждом из двух подграфов. Это означает, что ложны оба отношения $m_i > m_i'$ и $m_j > m_j'$, то есть новое значение метрики для объединения двух подграфов косвенных связей всегда должно быть больше или равно метрике любого из подграфов.

Свойство 6. Пусть $v_i, v_j, v_k \in V$ – три разных метода в системе, тогда $\mu(v_i) = \mu(v_j) \nRightarrow \mu(v_i \oplus v_k) = \mu(v_j \oplus v_k)$. Может случиться, что значение метрики объединения графов ICG_i и ICG_k будет отличаться от значения метрики графа, полученного путем объединения ICG_j и ICG_k.

Свойство 7. Размер или сложность метода v_i должны влиять на расположение или порядок путей вызова ICG_i. Пусть $(v_i, v_j), (v_j, v_k) \in A_i$ – два разных вызова метода в ICG_i, и предположим, что другого пути от v_i к v_j не существует. Новый граф ICG_i' для v_i' получается в результате изменения порядка этих вызовов на $(v_j', v_i'), (v_i', v_k)$. Тогда должно быть

истинно, что $\mu(v_i) \neq \mu(v_i')$, учитывая, что теперь $v_j' \notin V_i'$. Это свойство обретает смысл в контексте метрик, предназначенных для измерения сложности данных или потока управления.

Свойство 8. Если метод v_i' это переименованный метод $v_i \in V$, то $\mu(v_i) = \mu(v_i')$. В данном случае переименование – это последовательная систематическая замена всех вхождений идентификатора в одной области определения на другой, ранее не использовавшийся идентификатор. Это свойство выполняется всякий раз, когда мы применяем замену к подграфу косвенных связей любого метода.

Свойство 9. Могут существовать разные методы $v_i, v_j \in V$, такие, что $\mu(v_i) + \mu(v_j) < \mu(v_i \oplus v_j)$. В некоторых случаях объединенный граф может быть больше или сложнее, чем сумма их размеров или сложностей. Это свойство пытается отразить тот факт, что между комбинированными зависимостями может возникать некоторое взаимодействие.

Кроме того, мы высказываем некоторые предположения, которые необходимы для аналитической оценки предложенных метрик. Первое предположение касается статистического распределения значений метрик, а остальные два – мощности объединенных методов посредством реструктуризации.

Предположение 1. Пусть $v_i \in V$ – произвольный метод из системы S, и

$FAN\text{-}IN_i$ = Количество методов, вызывающих v_i,
$FAN\text{-}OUT_i$ = Количество методов, вызываемых из v_i,
LOC_i = Количество строк кода в v_i,
CYC_i = Цикломатическая сложность v_i.

Значения $FAN\text{-}IN_i$, $FAN\text{-}OUT_i$, LOC_i, и CYC_i, являются дискретными случайными переменными. Каждая из этих переменных имеет свою функцию распределения. Для каждого $v_i \in V$, все значения $FAN\text{-}IN_i$ независимы и одинаково распределены (independent and identically distributed, i.i.d.), это же относится ко всем значениям $FAN\text{-}OUT_i$, LOC_i, и CYC_i. В этом плане количество вызовов методов, число строк кода и цикломатическая сложность подчиняются неочевидному статистическому распределению. Более того, невозможно предсказать количество вызовов методов, число строк кода и цикломатическую сложность одного метода, основываясь на знании количества вызовов методов, числа строк кода и цикломатической сложности другого метода в той же системе.

Предположение 2. Любые два метода v_i и v_j могут иметь конечное число общих методов в ICG_i^f и ICG_j^f, в том плане, что объединение обоих методов в один приведет к тому, что какой-нибудь из общих методов может перестать быть общим (то есть сумма мощностей обоих подграфов будет больше или равна мощности объединенных подграфов хрупкости)

$$\left|ICG_i^f \cup ICG_j^f\right| = \left|ICG_i^f\right| + \left|ICG_j^f\right| - \left|ICG_i^f \cap ICG_j^f\right|.$$

Предположение 3. Аналогично, любые два метода v_i и v_j могут иметь конечное число общих методов в ICG_i^r и ICG_j^r, в том плане, что объединение обоих методов в один приведет к тому, что какой-нибудь из общих методов может перестать быть общим (то есть сумма мощностей обоих подграфов будет больше или равна мощности объединенных подграфов жесткости)

$$\left|ICG_i^r \cup ICG_j^r\right| = |ICG_i^r| + \left|ICG_j^r\right| - \left|ICG_i^r \cap ICG_j^r\right|.$$

5. Сбор эмпирических данных

В этом разделе объясняется метод, который реализован в нашем программном инструменте для извлечения данных из исходного кода программных систем и расчета количественных характеристик косвенных связей. В разделе также представлен и описывается набор данных программных систем, используемых при проверке метрик.

5.1 Алгоритм

Разработанный нами инструмент строит графы косвенных связей (ICG) для каждой функции, обнаруженной в программной системе. Программа написана на языке C, использует платформу с открытым исходным кодом и для компиляции исходного кода программ и построения его *абстрактного синтаксического дерева* (Abstact Syntax Tree, AST) с семантическими привязками, выполнения статического анализа AST и создания графа зависимости программы (Program Dependence Graph, PDG) использует технологии Roslyn и .NET Compiler Platform (см. алгоритм 1 и рис. 1). Узлы PDG – это методы из AST, а его ребра – это полиморфные вызовы между этими методами. Полиморфный вызов – это любой прямой вызов между двумя методами, любой вызов, который может быть выполнен через реализацию интерфейса, или любой вызов, уточненный схемой наследования.

PDG может иметь циклы, то есть циклические пути вызова между двумя или более методами. Другое название подграфов, образованных этими циклами, – сильно связанные компоненты (Strongly Connected Components, SCC). Методы, определенные вне цикла, могут вызывать один или несколько методов цикла. Аналогично, методы в цикле могут вызывать методы вне цикла. Тем самым, трудно гарантировать, что циклы будут иметь не более одной точки входа и одной точки выхода. Наш инструмент реализует алгоритм Габоу [37] для преобразования PDG в ориентированный ациклический граф путем выявления сильно связанных компонентов и замены каждого из них одним свернутым метаузлом. Весовое значение этих свернутых компонентов SCC представляет собой сумму весовых значений его методов. Каждый метод имеет несколько различных весовых характеристик: FAN-IN, FAN-OUT, NOM, LOC и CYC.

```
1: Input C – множество файлов исходного кода
2: Output M – множество извлеченных метрик
3: Output IC – множества косвенно связанных методов
4:
5: DC ← ∅                          ▷ DC: Граф анализа вызовов методов, иерархии классов и реализации интерфейсов
6:
7: for all S ∈ C do                                                          ▷ S: Файл исходного кода
8:         ASTs ← Generate AST from S                          ▷ AST_S: Абстрактное синтаксическое дерево из S
9:         for all references (i, j) ∈ AST_S do
10:            Add Reference (i, j) to Graph DC    ▷ Преобразование вызовов полиморфных методов в фактические
11:        end for
12: end for
13:
14: SCC ← Запуск алгоритма Габоу (поиск SCC в DC)          ▷ SCC: множество сильно связанных компонентов
15: DAG ← Свертка SCC на основе DC и SCC                        ▷ DAG: Направленный ациклический граф
16: IC ← Поиск в ширину на множестве методов IC из DAG       ▷ IC: Множество косвенно связанных методов
17: M ← Извлечение метрик с использованием DAG и IC                    ▷ M: Множество метрик
18:
19: return ⟨ M, IC⟩
```

Алгоритм 1. Алгоритм извлечения компонентов M и IC для формальной системы отношений F
Algorithm 1. Algorithm used by our tool to extract components M and IC for the formal relational system F

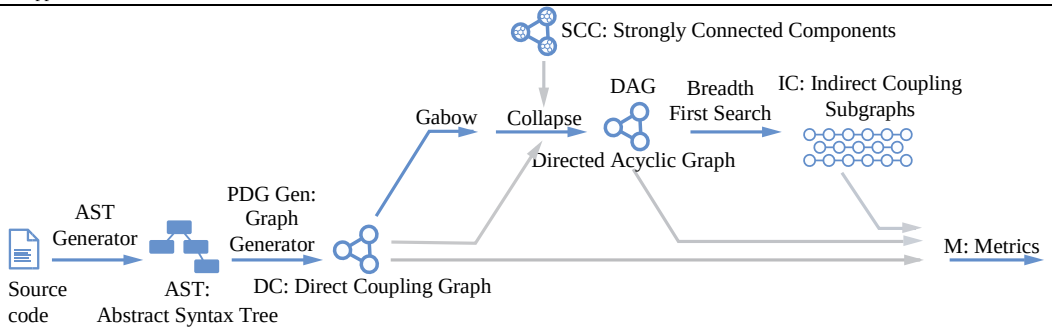


Рис. 1. Алгоритм извлечения метрик и подграфов косвенных связей из исходного кода.
Fig. 1. Algorithm used to extract metrics and indirect coupling subgraphs from source code.

Наш инструментарий рассматривает ациклический граф DAG со свернутыми SCC и применяет поиск в ширину, составляя для каждого метода в системе множество транзитивных замыканий *T* и множество методов с косвенными связями *IC*.

Набор транзитивных замыканий *T* содержит прямые транзитивные замыкания $\boldsymbol{t_i^f}$ для каждого метода $\boldsymbol{v_i} \in \boldsymbol{V}$ для расчета хрупкости или обратные транзитивные замыкания $\boldsymbol{t_i^r}$ для расчета жесткости. Множество $\boldsymbol{IC}$ методов с косвенными связями содержит множества косвенных связей $\boldsymbol{ic_i}$ для каждого $\boldsymbol{v_i}$. Заметьте, что $\boldsymbol{ic_i}$ равно $\boldsymbol{ic_i^f}$ в метриках хрупкости и равно $\boldsymbol{ic_i^r}$ в метриках жесткости, и что $\boldsymbol{ic_i^f} = \boldsymbol{V_i^f}$ – это множество узлов $\boldsymbol{t_i^f} = \left(\boldsymbol{V_i^f}, \boldsymbol{A_i^{f*}}\right)$, а $\boldsymbol{ic_i^r} = \boldsymbol{V_i^r}$ – это множество узлов $\boldsymbol{t_i^r} = (\boldsymbol{V_i^r}, \boldsymbol{A_i^{r*}})$. Наш инструмент вычисляет метрики $\boldsymbol{m_i} \in \boldsymbol{M}$, используя весовые значения элементов в $\boldsymbol{ic_i}$.

5.2 Набор данных

Программы, которые использовались для проверки вычисленных метрик, представляют собой 11 систем, написанных на языке C#. Шесть из них относятся к категории промышленных, а остальные пять – это программы с открытым исходным кодом. В табл. 1 для этих систем приведены сведения о количестве методов или узлов в графе *G* (NOM), общее количество строк кода (LOC), общая комбинированная цикломатическая сложность (CYC), число классов (Classes) и вызовов методов или ребер в графе *G* (Calls). В этой таблице также указано общее количество косвенных связей в *G* (Paths), максимальная и средняя длина таких связей, а также среднее количество строк кода всех методов в каждой системе. Системы отсортированы в порядке убывания общего количества косвенных связей в каждой системе (Paths). Значения в столбцах NOM, LOC, CYC, Classes и Calls – это наши подсчеты, они не учитывают косвенную связь между методами.

Промышленные системы созданы двумя компаниями-разработчиками программного обеспечения, названными здесь Компания 1 и Компания 2, чтобы соблюсти соглашения о неразглашении информации. Исследуются системы Система 1, Система 4, Система 5 и Система 6, разработанные Компанией 1, а также Система 2 и Система 3 Компании 2. Размер этих программ колеблется от 4606 до 188463 строк текста (LOC).

Нами также анализировались приложения с открытым исходным кодом, к которым относятся MongoDB C# Driver 2.9 [38], Json.NET 12.0.1 [39], .NET Micro Framework 4.4 [40], NodeJS Tools [41] и Neo4j .NET Driver 2.0 [42], размер которых находится в диапазоне от 3403 до 67633 строк текста (LOC). По мере увеличения количества строк кода в системах также увеличивается количество классов и количество методов, за исключением приложения O_2, у которого меньше классов, чем у приложения O_5, но в среднем более крупные методы.

Табл. 1. Численные характеристики промышленных систем и систем с открытым исходным кодом
Table 1. Size properties of industrial and open source systems

ID	Системное имя	NOM	LOC	CYC	Classes	Calls	Paths	Длина (max)	Длина (средняя)	$\frac{LOC}{Methods}$
I_1	System 1	2,832	17,020	6,062	842	3,290	73,667	15	7	6.01
I_2	System 2	20,362	188,463	55,693	2,633	23,351	51,202	14	2	9.26
I_3	System 3	18,525	177,050	42,598	3,507	23,893	42,030	9	3	9.56
I_4	System 4	3,786	29,077	10,293	1,563	347	3,760	8	1	7.68
I_5	System 5	1,065	6,062	2,102	336	349	1,037	8	1	5.69
I_6	System 6	825	4,603	1,542	251	386	739	8	1	5.58
O_1	MongoDB C# Driver	8,034	37,181	16,128	1,415	16,073	20,751,439	35	14	4.63
O_2	Json.NET	1,574	10,185	5,401	167	3,104	11,013,586	29	16	6.47
O_3	.NET Micro Framework	9,038	67,633	25,217	1,824	10,823	57,428	22	8	7.48
O_4	NodeJS Tools	3,288	18,507	8,731	596	3,185	22,893	18	8	5.63
O_5	Neo4j .NET Driver	1,018	3,403	1,523	210	1,150	8,324	22	10	3.34

6. Расчетные критерии метрик

В этом разделе описание каждой предлагаемой метрики выделено в отдельный подраздел. На вопрос (RQ1), поставленный в начале статьи, отвечают подразделы “Определение”, “Теоретические основы” и “Аналитические оценки”. На вопрос (RQ2) отвечают подразделы с заголовками: “Важные соображения” и “Экспериментальные данные”.

6.1 Вычисление показателя хрупкости методов (FNOM)

Метрика FNOM для данного метода показывает количество методов в подграфе хрупкости.

6.1.1 Определение

Пусть $\boldsymbol{v_i} \in \boldsymbol{V}$ – это метод, $\boldsymbol{\mu} = \text{FNOM}$, и $\boldsymbol{ic_i} = \{\boldsymbol{v_1} \cdots \boldsymbol{v_n}\}$ – множество всех методов в $\boldsymbol{ic_i^f}$ (то есть все методы в графе хрупкости косвенных связей метода $\boldsymbol{v_i}$, включая сам этот метод). Тогда:

$$m_i = \mu_i = \text{FNOM}_i = \sum_{k=1}^{n} \text{NOM}_k \tag{4}$$

6.1.2 Теоретические основы

В отношении конкретного метода метрика FNOM соответствует расширению метрики сложности FAN-OUT. FNOM – это FAN-OUT, рассчитываемая для подграфа хрупкости косвенных связей этого метода (таким образом, получаемое значение соответствует количеству методов, косвенно вызываемых из данного, включая сам этот метод). FNOM включает в себя значения метрики FAN-OUT всех методов в подграфе хрупкости метода. Эта метрика учитывает скрытые зависимости на этапе, возникшие при проектировании, и применяется при управлении размером и сложностью, а также для следования принципам инкапсуляции.

6.1.3 Аналитические оценки

Пусть $\boldsymbol{v_i}, \boldsymbol{v_j} \in \boldsymbol{V}$ два разных произвольных метода из системы $\boldsymbol{E}$. Предполагается, что переменные FAN-OUT_i и FAN-OUT_j являются i.i.d. (см. Предположение 1). Аналогичные

предположения делаются для переменных $\boldsymbol{m_i} = \text{FNOM}_i$ и $\boldsymbol{m_j} = \text{FNOM}_j$. Соответственно с ненулевой вероятностью $\boldsymbol{\mu_i \neq \mu_j}$, тем самым удовлетворяется Свойство 1. Свойство 2 также выполняется (см. свойства в разделе 4). Существует ненулевая вероятность того, что $\exists\, \boldsymbol{v_k} \in \boldsymbol{V} \mid \boldsymbol{\mu_i = \mu_k}$, следовательно, Свойство 3 выполняется. Функциональность, выполняемая методом, не определяет количество вызовов методов в его подграфе хрупкости, поскольку это число в большей степени зависит от проектных решений, не зависящих от этой функциональности, поэтому Свойство 4 выполняется. Пусть $\boldsymbol{\mu_i = m_i}$ и $\boldsymbol{\mu_j = m_j}$, тогда значение метрики объединения обоих методов равно $\boldsymbol{\mu(v_i \oplus v_j) = m_i + m_j - \delta}$, где $\boldsymbol{\delta}$ количество общих методов в графах $\boldsymbol{ic_i^f}$ и $\boldsymbol{ic_j^f}$. Максимальное значение $\boldsymbol{\delta}$ равно $\boldsymbol{\min(m_i, m_j)}$. Следовательно, $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu_i}$ и $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu_j}$, что гарантирует выполнение Свойства 5. Далее, пусть $\boldsymbol{\mu_i = \mu_j}$, и $\exists\, \boldsymbol{v_k} \in \boldsymbol{V}$ такой, что $\boldsymbol{ic_k^f}$ имеет $\boldsymbol{\beta}$ методов, общих с $\boldsymbol{ic_i^f}$ (см. Предположение 2) и $\boldsymbol{\delta}$ методов, общих с $\boldsymbol{ic_j^f}$, где $\boldsymbol{\beta \neq \delta}$. Тогда

$$\boldsymbol{\mu(v_i \oplus v_k) = \mu_i + \mu_k - \beta}$$
$$\boldsymbol{\mu(v_j \oplus v_k) = \mu_j + \mu_k - \delta}$$

Следовательно, $\boldsymbol{\mu(v_i \oplus v_k) \neq \mu(v_j \oplus v_k)}$, то есть Свойство 6 выполняется. Свойства 7 и 8 выполняются (см. свойства в разделе 4). Пусть для любых двух методов $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$ число $\boldsymbol{\gamma}$ – это количество общих методов в графах $\boldsymbol{ic_i^f}$ и $\boldsymbol{ic_j^f}$, а $\boldsymbol{\delta}$ – это количество новых методов, возникших в результате реструктуризации, необходимой для объединения графов $\boldsymbol{ic_i^f}$ и $\boldsymbol{ic_j^f}$ и получения графа $\boldsymbol{ic_{i'}^f}$. Тогда если $\boldsymbol{\delta > \gamma}$, то истинно отношения $\boldsymbol{m_i + m_j - \gamma + \delta > m_i + m_j}$ (то есть $\boldsymbol{\mu(v_i \oplus v_j) > \mu(v_i) + \mu(v_j)}$ при заданных $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$). Следовательно, Свойство 9 выполняется.

6.1.4 Важные соображения

Ниже приводятся некоторые соображения, которые полезно учитывать в отношении FNOM:

- FNOM – это более полная метрика сложности метода, чем FAN-OUT. FNOM – своего рода рекурсивный вариант метрики FAN-OUT.
- Количество методов, решающих сформулированные в программе задачи, можно использовать для предсказания того, сколько времени и усилий потребуется для разработки, понимания (включая прослеживание методов) и сопровождения функциональности, выполняемой методом (entry point method) [7].
- Чем больше значение метрики FNOM метода, тем больше вероятность того, что любое изменение повлияет на этот метод через его зависимости (то есть, что возникнет волновой эффект ошибок, изменений, тестирования, реструктуризации).
- Методы с многочисленными зависимостями, считаются менее пригодными для повторного использования, они более специфичны для приложения.

У корневого метода метрика FAN-IN равна нулю; у листового метода равна нулю метрика FAN-OUT. Средние значения метрик для всех корневых и конечных методов всех систем приведены в табл. 2. Средние значения метрик хрупкости рассчитаны только для корневых методов, а средние значения метрик жесткости представлены только для листовых методов.

6.1.5 Экспериментальные данные

Сводная статистика по метрике FNOM во всех системах представлена в табл. 3. Помимо минимума, 1-го квартиля, медианы, 3-го квартиля и максимума для FNOM, в ней показано наиболее частое расстояние (MFD) между значениями метрик FAN-OUT и FNOM (то есть для каждого метода v_i в системе расстояние равно $D_i = \text{FNOM}_i - \text{FAN-OUT}_i$ так что MFD

будет наиболее частым D_i в системе), а также процентная доля методов со значением D_i = MFD от их общего числа. В таблице также представлен коэффициент корреляции Кендалла между значениями метрик FAN-OUT и FNOM, а также его *p-значение*.

Наиболее широко используемые статистические коэффициенты корреляции – Пирсона, Спирмена и Кендалла. Мы не могли использовать корреляцию Пирсона, поскольку она требует, чтобы данные соответствовали нормальному распределению. Мы также не могли использовать корреляцию Спирсмена, потому что она неверно ведет учет зависимостей. Мы выбрали τ-статистику Кендалла [43], используемую для оценки ранговой меры связи между двумя метриками.

Табл. 2. Средние значения метрик из методов root/leaf†*

Место для уравнения.*Table 2. Average metric values from root*/leaf† methods*

Системное имя	FNOM	FLOC	FCYC	RNOM	RLOC	RCYC
I_1	16	122	50	14	110	31
I_2	3	29	10	3	83	19
I_3	6	66	27	6	182	28
I_4	1	8	3	1	9	3
I_5	1	8	3	2	10	3
I_6	2	10	3	2	13	4
O_1	74	544	224	76	522	147
O_2	49	335	166	56	588	324
O_3	8	53	20	9	131	49
O_4	5	29	13	6	67	31
O_5	11	40	15	10	46	21

**Корневые методы для FNOM, FLOC и FCYC (Root methods for FNOM, FLOC and FCYC)*
†Листовые методы для RNOM, RLOC и RCYC (Leaf methods for RNOM, RLOC and RCYC)

Табл. 3. Сводная статистика для метрики FNOM
Table 3. Summary Statistics for the FNOM metric

Sys	MFD*	%MDF	$\tau^{\dagger}$	*p-val*	Min	1st Q	Med	3rd Q	Max
I_1	1	64	0.82	0	1	1	2	12	119
I_2	1	85	0.95	0	1	1	1	3	229
I_3	1	63	0.88	0	1	1	2	4	100
I_4	1	98	1.00	0	1	1	1	1	47
I_5	1	95	0.99	0	1	1	1	1	47
I_6	1	92	0.98	0	1	1	1	2	47
O_1	1	58	0.80	0	1	1	2	31	560
O_2	1	49	0.73	0	1	1	3	23	689
O_3	1	65	0.85	0	1	1	2	6	300
O_4	1	75	0.90	0	1	1	1	3	234
O_5	1	70	0.88	0	1	1	1	4	153

**MFD: наиболее частое расстояние между FNOM и FAN-OUT (MFD: most frequent distance between FNOM and FAN-OUT)*

†τ: коэффициент корреляции Кендалла между FNOM и FAN-OUT (τ: Kendall correlation coefficient between FNOM and FAN-OUT)

Функции плотности [44] метрики FNOM для всех систем представлены на рис. 2, а линейные графики со значениями метрики для FNOM и FAN-OUT для всех методов [45] в каждой системе, ранжированной по FAN-OUT, показаны на рис. 3 (каждое значение по ординате линейного графика представляет значения метрик FNOM и FAN-OUT, а значения по оси абсцисс представляют ранг методов, упорядоченных по FAN-OUT, а затем по FNOM, те значения, для которых разница между FNOM и FAN-OUT составляет ровно MFD, не отображаются).

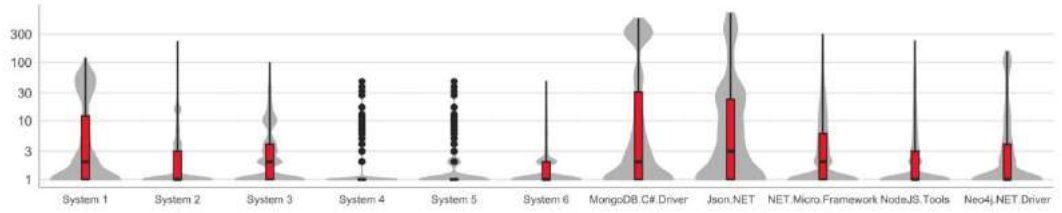


Рис. 2. Функция плотности для FNOM для всех систем
Fig. 2. Density function for FNOM from all systems

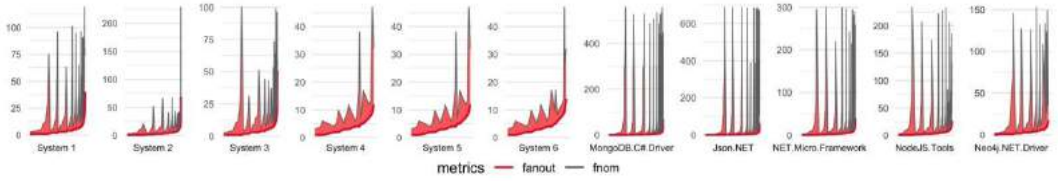


Рис. 3. Расстояние между значениями FNOM и FAN-OUT для ранжированных методов всех систем
Fig. 3. Distance between FNOM and FAN-OUT values for the ranked methods from all systems

6.2 Вычисление показателя хрупкости числа строк текста (FLOC)

FLOC — это совокупное количество строк кода всех методов в подграфе хрупкости данного метода.

6.2.1 Определение

Пусть $\boldsymbol{v_i} \in \boldsymbol{V}$ является методом, $\boldsymbol{\mu} = \text{FLOC}$, и $\boldsymbol{ic_i} = \{\boldsymbol{v_1} \cdots \boldsymbol{v_n}\}$ множество всех методов в $\boldsymbol{ic_i^f}$, тогда:

$$m_i = \mu_i = \text{FLOC}_i = \sum_{k=1}^{n} \text{LOC}_k \quad (5)$$

6.2.2 Теоретические основы

Метрика FLOC – это, по сути, расширение метрики LOC (число строк текста), вычисленное для некоторого метода. FLOC – это метрика LOC, применяемая к подграфу хрупкости косвенных связей метода (она соответствует совокупному количеству строк текста во всех методах, которые данный метод может прямо или косвенно вызывать, а также в самом этом методе). Эта метрика также демонстрирует зависимости, не проявившиеся на этапе проектирования, это помогает управлять сложностью и размером программ, а также следовать принципам инкапсуляции.

6.2.3 Аналитические оценки

Пусть $\boldsymbol{v_i, v_j} \in \boldsymbol{V}$ – два разных произвольных метода из системы $\boldsymbol{E}$. Предполагается, что переменные $\mathbf{LOC_i}$ и $\mathbf{LOC_j}$ обладают свойством i.i.d. (см. Предположение 1), то же самое предполагается для переменных $\boldsymbol{m_i} = \text{FLO}\mathbf{C_i}$ и $\boldsymbol{m_j} = \text{FLO}\mathbf{C_j}$; следовательно $\boldsymbol{\mu_i \neq \mu_j}$, с ненулевой вероятностью, и выполняется Свойство 1. Свойство 2 также выполняется (см. свойства в разделе 4). Существует ненулевая вероятность того, что $\exists\, \mathbf{v_k} \in \boldsymbol{V} \mid \boldsymbol{\mu_i = \mu_k}$, следовательно Свойство 3 выполняется. Функциональность, выполняемая методом, не определяет общее количество строк кода во всех методах в его подграфе хрупкости, поскольку размер программы – это проектное решение, не зависящее от реализуемой функциональности, следовательно, Свойство 4 выполняется. Пусть $\boldsymbol{\mu_i = m_i}$ и $\boldsymbol{\mu_j = m_j}$, тогда $\boldsymbol{\mu(v_i \oplus v_j) = m_i + m_j - \delta}$, где $\boldsymbol{\delta}$ – общее количество строк кода в общих методах между $\boldsymbol{ic_i^f}$ и $\boldsymbol{ic_j^f}$. Максимальное значение $\boldsymbol{\delta}$ равно $\boldsymbol{\min(m_i, m_j)}$. Следовательно, $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu_i}$ и $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu_j}$, поэтому Свойство 5 выполняется. Далее, пусть $\boldsymbol{\mu_i = \mu_j}$, и $\exists\, \boldsymbol{v_k} \in \boldsymbol{V}$ такой, что $\boldsymbol{ic_k^f}$ имеет $\boldsymbol{\beta}$ строк текста в общих с $\boldsymbol{ic_i^f}$ методах (см. Условие 2) и $\boldsymbol{\delta}$ строк текста в общих методах с $\boldsymbol{ic_j^f}$, где $\boldsymbol{\beta \neq \delta}$. Тогда

$$\boldsymbol{\mu(v_i \oplus v_k) = \mu_i + \mu_k - \beta}$$
$$\boldsymbol{\mu(v_j \oplus v_k) = \mu_j + \mu_k - \delta}$$

Следовательно, $\boldsymbol{\mu(v_i \oplus v_k) \neq \mu(v_j \oplus v_k)}$ и Свойство 6 выполняется. Свойства 7 и 8 выполняются. Для любых двух методов $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$, пусть $\boldsymbol{\gamma}$ будет значением совокупного числа строк общих методов в графах $\boldsymbol{ic_i^f}$ и $\boldsymbol{ic_j^f}$, и пусть $\boldsymbol{\delta}$ будет совокупным значением числа строк новых методов, добавленных в результате реструктуризации при объединении графов $\boldsymbol{ic_i^f}$ и $\boldsymbol{ic_j^f}$ в граф $\boldsymbol{ic_{i'}^f}$. Если $\boldsymbol{\delta > \gamma}$, то $\boldsymbol{m_i + m_j - \gamma + \delta > m_i + m_j}$ (то есть $\boldsymbol{\mu(v_i \oplus v_j) > \mu(v_i) + \mu(v_j)}$ при заданных $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$). Следовательно, Свойство 9 выполняется.

6.2.4 Важные соображения

Соображения, рассматриваемые в отношении FLOC, перечислены ниже.

- FLOC — это более точный показатель размера метода, чем LOC. FLOC образуется суммированием нужных значений метрики LOC.
- Совокупный размер всех методов, выполняющих некоторую функциональность, также может помочь спрогнозировать необходимое время и усилия для реализации, понимания (включая поиск зависимостей) и изменения требований, выполняемых методом, являющимся начальной точкой графа [7].
- Более высокие значения метрики FLOC метода увеличивают вероятность того, что через любую из его зависимостей (то есть каскадные ошибки или изменения, тестирование, реструктуризация) метод может быть изменен. Например, вероятность того, что какое-либо изменение повлияет на монолитную программу, равна 1.
- Методы, которые имеют больше зависимостей, предположительно, сложнее использовать повторно, они более специфичны для конкретного приложения.

6.2.5 Экспериментальные данные

Сводная статистика по метрике FLOC для всех систем представлена в табл. 4. В этой таблице показаны минимум, 1-й квартиль, медиана, 3-й квартиль и максимум для метрики FLOC. Она также показывает наиболее частое расстояние MFD между значениями метрик LOC и FLOC и долю методов со значением метрики, равным MFD, в общем количестве методов. В ней

также представлен коэффициент корреляции Кендалла между значениями метрик LOC и FLOC, а также его *p-значение*.

Средние значения метрик FLOC для всех корневых методов всех систем представлены в табл. 2.

Функции плотности метрики FLOC для всех систем представлены на рис. 4, а линейные диаграммы со значениями метрики FLOC и LOC для методов каждой системы, ранжированной по LOC, представлены на рис. 5.

Табл. 3. Сводная статистика для метрики FLOC
Table 3. Summary Statistics for the FLOC metric

Sys	MFD*	%MDF	$\tau^{\dagger}$	*p-val*	Min	1st Q	Med	3rd Q	Max
I_1	0	47	0.33	0	1	3	6	96	990
I_2	0	65	0.85	0	1	1	4	15	20,964
I_3	0	45	0.63	0	1	1	9	37	4,860
I_4	0	94	0.93	0	1	1	6	11	291
I_5	0	80	0.71	0	1	3	4	8	291
I_6	0	71	0.56	0	1	3	4	7	300
O_1	0	40	0.48	0	1	1	8	164	4,205
O_2	0	34	0.35	0	1	3	16	92	5,621
O_3	0	49	0.65	0	1	1	5	32	2,940
O_4	0	60	0.70	0	1	1	4	17	1,589
O_5	0	56	0.59	0	1	1	3	17	555

**MFD: наиболее частое расстояние между FLOC и LOC (MFD: most frequent distance between FLOC and LOC)*

$^{\dagger}\tau$: коэффициент корреляции Кендалла между FLOC и LOC (τ: Kendall correlation coefficient between FLOC and LOC)

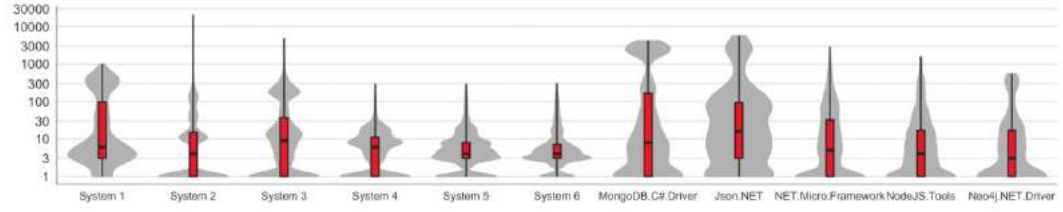


Рис. 4. Функция плотности для FLOC для всех систем
Fig. 4. Density function for FLOC from all systems

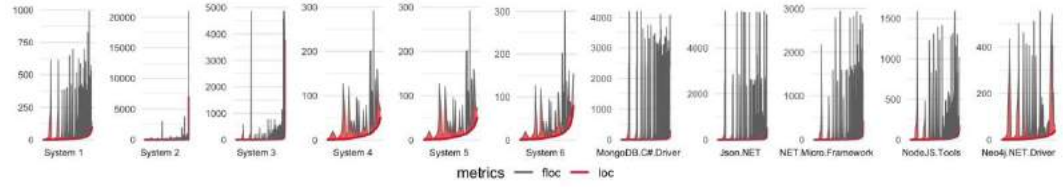


Рис. 5. Расстояние между значениями FLOC и LOC для ранжированных методов всех систем
Fig. 5. Distance between FLOC and LOC values for the ranked methods from all systems

6.3 Вычисление показателя хрупкости цикломатической сложности (FCYC)

Метрика FCYC – это число, показывающее значение консолидированной цикломатической сложности всех методов подграфа хрупкости данного метода.

6.3.1 Определение

Пусть $v_i \in V$ – метод, μ = FCYC, и $ic_i = \{v_1 \cdots v_n\}$ множество всех методов в ic_i^f, тогда:

$$m_i = \mu_i = \text{FCYC}_i = \sum_{k=1}^{n} \text{CYC}_k - (n-1) \quad (6)$$

Компонент $(n-1)$ в формуле корректирует результирующее число цикломатической сложности. Результат этой формулы представляет собой подстановку всех методов (их расширение подобно макросам) в корневой метод.

6.3.2 Теоретические основы

FCYC соответствует расширению метрики сложности CYC для одиночного метода. Метрика FCYC – это та же метрика CYC, но в применении к подграфу хрупкости косвенных связей этого метода. Оно соответствует сумме цикломатических сложностей всех методов, которые косвенно вызывает данный метод, включая его самого. Когда текст вызываемого метода встраивается в текст вызывающего метода, CYC результирующего кода равен сумме соответствующих им значений CYC минус 1. В результате вычитаемое в формуле (6) компенсирует лишние $(n-1)$ единиц. Эта метрика также учитывает зависимости, скрытые на этапе проектирования, что позволяет управлять размером и сложностью, а также следовать правилам инкапсуляции.

6.3.3 Аналитические оценки

Пусть $\boldsymbol{v_i, v_j \in V}$ – два разных произвольных метода системы $\boldsymbol{E}$. Переменные $\mathbf{CYC_i}$ и $\mathbf{CYC_j}$ будут i.i.d. (см. Предположение 1), то же самое мы предполагаем относительно переменных $\boldsymbol{m_i} =$ FCY$\mathbf{C_i}$ и $\boldsymbol{m_j} =$ FCY$\mathbf{C_j}$; следовательно, $\boldsymbol{\mu_i \neq \mu_j}$ с ненулевой вероятностью. Следовательно, Свойство 1 выполняется. Свойство 2 также выполняется (см. свойства в разделе 4). Существует ненулевая вероятность, что $\boldsymbol{\exists\, v_k \in V \mid \mu_i = \mu_k}$, соответственно, Свойство 3 выполняется. Функциональность, выполняемая методом, не определяет общее совокупное число цикломатической сложности для всех методов в его подграфе хрупкости, поскольку это проектное решение, не зависящее от функциональности, следовательно, Свойство 4 выполняется. Пусть $\boldsymbol{\mu_i = m_i}$ и $\boldsymbol{\mu_j = m_j}$, тогда $\boldsymbol{\mu(v_i \oplus v_j) = m_i + m_j - \delta}$, где $\boldsymbol{\delta}$ – общее совокупное число цикломатической сложности для общих методов в графах $\boldsymbol{if_i^f}$ и $\boldsymbol{ic_j^f}$. Максимальное значение $\boldsymbol{\delta}$ равно $\boldsymbol{\min(m_i, m_j)}$. Тогда $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu_i}$ и $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu_j}$, соответственно, Свойство 5 выполняется. Теперь, пусть $\boldsymbol{\mu_i = \mu_j}$, и $\boldsymbol{\exists\, v_k \in V}$ такой, что граф $\boldsymbol{ic_k^f}$ имеет совокупное цикломатическое число сложности $\boldsymbol{\beta}$ из общих методов с графом $\boldsymbol{ic_i^f}$ (см. Предположение 2) и совокупное число цикломатической сложности $\boldsymbol{\delta}$ с графом $\boldsymbol{ic_j^f}$, где $\boldsymbol{\beta \neq \delta}$. Тогда

$$\boldsymbol{\mu(v_i \oplus v_k) = \mu_i + \mu_k - \beta}$$
$$\boldsymbol{\mu(v_j \oplus v_k) = \mu_j + \mu_k - \delta}.$$

Таким образом, $\mu(v_i \oplus v_k) \neq \mu(v_j \oplus v_k)$, и Свойство 6 выполняется. Свойства 7 и 8 также выполняются (см. раздел 4). Пусть γ – совокупное значение метрики CYC методов для двух произвольных методов v_i и v_j, общих для графов ic_i^f и ic_j^f, и пусть δ будет накопленным значением метрики CYC новых программ, написанных для объединения графов ic_i^f и ic_j^f в

граф ic_i^f. Если $\delta > \gamma$, то ясно, что $m_i + m_j - \gamma + \delta > m_i + m_j$ (то есть, $\mu(v_i \oplus v_j) > \mu(v_i) + \mu(v_j)$ для заданных v_i и v_j). Следовательно, Свойство 9 выполняется.

6.3.4 Важные соображения

Соображения, рассматриваемые в отношении FCYC, перечислены ниже.

- FCYC — это более точный количественный показатель цикломатической сложности, чем CYC, поскольку он принимает во внимание значения CYC всех зависимых компонентов.
- Совокупная цикломатическая сложность всех методов, выполняющих некоторую работу, также может помочь спрогнозировать необходимое время и усилия для реализации, понимания (включая исследование подграфа хрупкости) и развития функциональности исходного метода [7].
- Методы с более высокими значениями метрики FCYC из-за своих зависимостей будут скорее всего более чувствительны к любым модификациям.
- Методы, с более сложными зависимостями, по-видимому, будет сложнее использовать повторно, они более специфичны для конкретного приложения.

6.3.5 Экспериментальные данные

Сводная статистика по показателю FCYC для всех систем представлена в табл. 5. В этой таблице показаны минимум, 1-й квартиль, медиана, 3-й квартиль и максимум для FCYC. Таблица также показывает наиболее часто отмеченное расстояние MFD между значениями метрик CYC и FCYC, долю методов, имеющих именно такое расстояние MFD, среди всех методов и коэффициент корреляции Кендалла между значениями метрик CYC и FCYC, а также *p-значение*.
Средние значения метрики FCYC для всех корневых методов всех систем показаны в табл. 2.

Табл. 5. Сводная статистика для метрики FCYC
Table 5. Summary Statistics for the FCYC metric

Sys	MFD*	%MDF	$\tau^{\dagger}$	*p-val*	Min	1st Q	Med	3rd Q	Max
I_1	0	51	0.4	0	1	2	2	34	314
I_2	0	67	0.7	0	1	1	2	8	1,706
I_3	0	55	0.62	0	1	1	3	12	592
I_4	0	95	0.94	0	1	1	2	3	89
I_5	0	82	0.8	0	1	1	2	2	89
I_6	0	75	0.73	0	1	1	2	2	106
O_1	0	50	0.38	0	1	1	3	53	1,718
O_2	0	37	0.29	0	1	2	8	50	2,940
O_3	0	61	0.63	0	1	1	2	11	1,208
O_4	0	67	0.68	0	1	1	2	8	671
O_5	0	63	0.54	0	1	1	2	6	231

**MFD: наиболее частое расстояние между FCYC и CYC (MFD: most frequent distance between FCYC and CYC)*

†τ: коэффициент корреляции Кендалла между FCYC и CYC (τ: Kendall correlation coefficient between FCYC and CYC)

Функции плотности метрики FCYC для всех систем показаны на **р**ис**.** 6, а линейные диаграммы со значениями метрик FCYC и CYC для методов каждой системы, ранжированной по CYC, представлены на рис. 7.

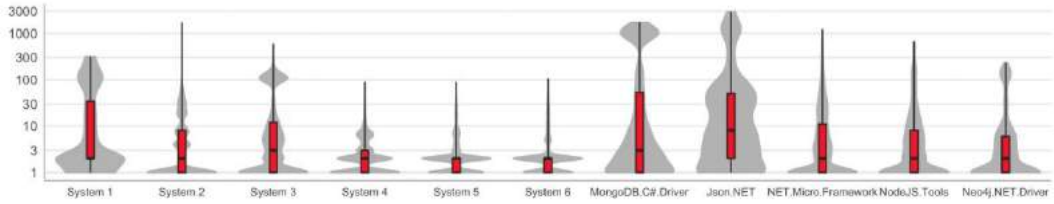


Рис. 6. Функция плотности для FCYC для всех систем
Рис. 6. Density function for FCYC from all systems

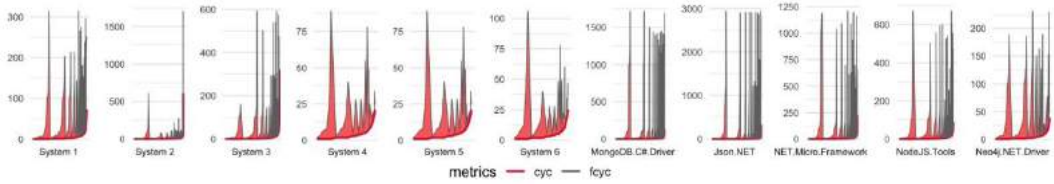


Рис. 7. Расстояние между значениями FCYC и CYC для ранжированных методов всех систем
Fig. 7. Distance between FCYC and CYC values for the ranked methods from all systems

6.4 Вычисление показателя жесткости методов (RNOM)

RNOM — количество методов в подграфе жесткости (или рискованности) данного метода.

6.4.1 Определение

Пусть $\boldsymbol{v_i} \in \boldsymbol{V}$ – метод, $\boldsymbol{\mu = RNOM}$, и $\boldsymbol{ic_i} = \{\boldsymbol{v_1} \cdots \boldsymbol{v_n}\}$ множество методов в $\boldsymbol{ic_i^r}$ (то есть подграф жесткости косвенных связей метода $\boldsymbol{v_i}$), значит:

$$m_i = \mu_i = \mathrm{RNOM}_i = \sum_{k=1}^{n} \mathrm{NOM}_i \qquad (7)$$

6.4.2 Теоретические основы

Метрика RNOM соответствует расширению метрики сложности FAN-IN для некоторого метода. По сути, эта метрика есть результат применения метрики FAN-IN к подграфу жесткости косвенных связей этого метода. Тем самым вычисляется количество методов, прямо или косвенно вызывающих метод, включая сам метод. В этом показателе также учитывается влияние клиентов, подвергшихся рефакторингу при проектировании, на возможности управления размером программ и их сложностью, а также на повышение возможностей повторного использования.

6.4.3 Аналитические оценки

Пусть $\boldsymbol{v_i, v_j} \in \boldsymbol{V}$ два разных произвольных метода в системе $\boldsymbol{E}$. Предполагается, что переменные *FAN-IN*$_i$ и *FAN-IN*$_j$ соответствуют определению i.i.d. (см. Предположение 1), то же самое касается переменных $\boldsymbol{m_i} = \mathrm{RNOM}_i$ и $\boldsymbol{m_j} = \mathrm{RNOM}_j$. Соответственно, с ненулевой вероятностью $\boldsymbol{\mu_i \neq \mu_j}$. Свойство 1 выполняется. Свойство 2 также выполняется (см. раздел 4). С ненулевой вероятностью $\exists\, \boldsymbol{v_k} \in \boldsymbol{V} \mid \boldsymbol{\mu_i = \mu_k}$, значит Свойство 3 выполняется. Функциональность, выполняемая методом, не определяет количество элементов в его подграфе жесткости, поскольку это проектное решение, не зависящее от этой функциональности, следовательно, Свойство 4 выполняется. Пусть $\boldsymbol{\mu_i = m_i}$ и $\boldsymbol{\mu_j = m_j}$, тогда $\boldsymbol{\mu(v_i \oplus v_j) = m_i + m_j - \delta}$, где $\boldsymbol{\delta}$ количество общих методов в графах $\boldsymbol{ic_i^r}$ и $\boldsymbol{ic_j^r}$. Максимальное значение $\boldsymbol{\delta}$ равно $\boldsymbol{\min(m_i, m_j)}$. Следовательно, $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu_i}$ и $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu_j}$, и Свойство 5 выполняется. Далее, пусть $\boldsymbol{\mu_i = \mu_j}$, и $\exists\, \boldsymbol{v_k} \in \boldsymbol{V}$ такой, что в граф $\boldsymbol{ic_k^r}$

входит $\boldsymbol{\beta}$ методов, общих с графом $\boldsymbol{ic_i^r}$ (см. Предположение 3), и $\boldsymbol{\delta}$ методов, общих с графом $\boldsymbol{ic_j^r}$, причем $\boldsymbol{\beta \neq \delta}$. Тогда,

$$\boldsymbol{\mu(v_i \oplus v_k) = \mu_i + \mu_k - \beta}$$
$$\boldsymbol{\mu(v_j \oplus v_k) = \mu_j + \mu_k - \delta.}$$

Следовательно, $\boldsymbol{\mu(v_i \oplus v_k) \neq \mu(v_j \oplus v_k)}$, и Свойство 6 выполняется. Свойства 7 и 8 также выполняются (см. раздел 4). Для двух методов $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$ пусть $\boldsymbol{\gamma}$ будет количеством методов, общих для графов $\boldsymbol{ic_i^r}$ и $\boldsymbol{ic_j^r}$, и пусть $\boldsymbol{\delta}$ будет количеством новых методов, добавленных в программу при реструктуризации двух графов $\boldsymbol{ic_i^r}$ и $\boldsymbol{ic_j^r}$ в новый граф $\boldsymbol{ic_i^r}$. Если $\boldsymbol{\delta > \gamma}$, то ясно, что $\boldsymbol{m_i + m_j - \gamma + \delta > m_i + m_j}$ (то есть $\boldsymbol{\mu(v_i \oplus v_j) > \mu(v_i) + \mu(v_j)}$ при заданных $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$). Следовательно, Свойство 9 выполняется.

6.4.4 Важные соображения

В отношении метрики RNOM можно высказать следующие соображения:

- Метрика RNOM более точно отражает сложность методов, чем показатель FAN-IN. В определенном смысле RNOM –это рекурсивный показатель FAN-IN.
- RNOM – это количество методов, косвенным образом решающих задачу. Эта метрика показывает, сколько времени и усилий потребуется для реализации, понимания (включая отслеживание цепочек вызовов методов) и развития фрагментов программ, которые используют метод, решающий задачу.
- Внесение изменений в метод с бóльшим значением RNOM с большей вероятностью повлияет на необходимость изменений в любом из зависимых методов. Подобный волновой эффект умножает заботы об исправлении ошибок, внесении изменении, проведении тестирования и рефакторинге в методы подграфа жесткости.
- Методы с большим числом зависимых методов, в большей степени считаются подходящими для повторного использования и менее специфичны для приложений.

6.4.5 Экспериментальные данные

Функции плотности метрики RNOM для всех систем показаны на рис. 8, а на рис. 9 представлены линейные диаграммы значений метрик RNOM и FAN-IN для методов во всех системах, ранжированных по FAN-IN.

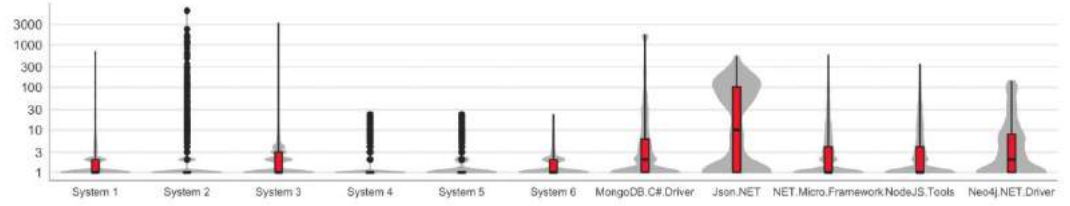


Рис. 8. Функция плотности для RNOM из всех систем
Рис. 8. Density function for RNOM from all systems

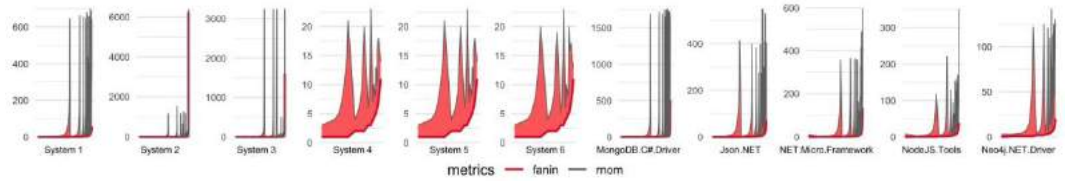


Рис. 9. Расстояние между значениями RNOM и FAN-IN для ранжированных методов всех систем
Fig. 9. Distance between RNOM and FAN-IN values for the ranked methods from all systems

Сводная статистика по метрике RNOM для всех систем представлена в табл. 6. В ней указаны минимум, 1-й квартиль, медиана, 3-й квартиль и максимум для RNOM. Она также показывает наиболее часто отмеченное расстояние MFD между значениями метрик FAN-IN и RNOM и долю методов MFD, имеющих именно такое расстояние MFD, среди всех методов. В ней также представлен коэффициент корреляции Кендалла между значениями метрик FAN-IN и RNOM, а также *p-значение*. Средние значения метрики RNOM для всех конечных методов всех систем приведены в табл. 2.

Табл. 6. Сводная статистика для метрики RNOM
Table 6. Summary Statistics for the RNOM metric

Sys	MFD*	%MDF	$\boldsymbol{\tau}^{\dagger}$	*p-val*	Min	1st Q	Med	3rd Q	Max
$\boldsymbol{I_1}$	1	84	0.94	0	1	1	1	2	697
$\boldsymbol{I_2}$	1	95	0.98	0	1	1	1	1	6,386
$\boldsymbol{I_3}$	1	76	0.9	0	1	1	1	3	3,238
$\boldsymbol{I_4}$	1	98	0.99	0	1	1	1	1	23
$\boldsymbol{I_5}$	1	93	0.98	0	1	1	1	1	23
$\boldsymbol{I_6}$	1	91	0.97	0	1	1	1	2	23
$\boldsymbol{O_1}$	1	68	0.83	0	1	1	2	6	1,756
$\boldsymbol{O_2}$	1	44	0.68	0	1	1	10	103	548
$\boldsymbol{O_3}$	1	68	0.86	0	1	1	1	4	594
$\boldsymbol{O_4}$	1	71	0.87	0	1	1	1	4	351
$\boldsymbol{O_5}$	1	57	0.76	0	1	1	2	8	142

**MFD: наиболее частое расстояние между RNOM и FAN-IN (MFD: most frequent distance between RNOM and FAN-IN)*

†τ: коэффициент корреляции Кендалла между RNOM и FAN-IN (τ: Kendall correlation coefficient between RNOM and FAN-IN)

6.5 Вычисление показателя жесткости числа строк текста (RLOC)

RLOC – это количество строк кода всех методов в подграфе жесткости данного метода.

6.5.1 Определение

Пусть $\boldsymbol{v_i} \in \boldsymbol{V}$ – метод, $\boldsymbol{\mu} = \mathbf{RLOC}$, и $\boldsymbol{ic_i} = \{\boldsymbol{v_1} \cdots \boldsymbol{v_n}\}$ множество всех методов в графе $\boldsymbol{ic_i^r}$, тогда:

$$m_i = \mu_i = \mathrm{RLOC}_i = \sum_{k=1}^{n} \mathrm{LOC}_k \tag{8}$$

6.5.2 Теоретические основы

RLOC соответствует расширению метрики размера отдельных методов LOC. RLOC – это метрика LOC в применении к подграфу косвенной связи жесткости этого метода (то есть она соответствует суммарному количеству строк кода во всех методах, которые могут прямо или косвенно вызывать метод, а также строк самого этого метода) . В этом показателе также учитывается влияние клиентов, подвергшихся рефакторингу при проектировании, на возможности управления размером программ и их сложностью, а также на увеличение возможностей повторного использования.

6.5.3 Аналитические оценки

Пусть $\boldsymbol{v_i}, \boldsymbol{v_j} \in \boldsymbol{V}$ два разных произвольных метода системы $\boldsymbol{E}$. Переменные $\boldsymbol{LOC_i}$ и $\mathbf{LOC_j}$ предполагаются обладающими свойством i.i.d. (см. Предположение 1), то же самое касается

переменных $\boldsymbol{m_i} = \text{RLOC}_{\boldsymbol{i}}$ и $\boldsymbol{m_j} = \text{RLOC}_{\boldsymbol{j}}$; тогда, с ненулевой вероятностью $\boldsymbol{\mu_i \neq \mu_j}$; Свойство 1 выполняется. Свойство 2 также выполняется (см. раздел 4). С ненулевой вероятностью $\exists\, \boldsymbol{v_k} \in \boldsymbol{V} \mid \boldsymbol{\mu_i = \mu_k}$, следовательно, Свойство 3 выполняется. Функциональность, выполняемая методом, не определяет общее количество строк кода во всех методах в его подграфе жесткости, поскольку это проектное решение, не зависящее от функциональности, следовательно, Свойство 4 выполняется. Пусть $\boldsymbol{\mu_i = m_i}$ и $\boldsymbol{\mu_j = m_j}$, тогда $\boldsymbol{\mu(v_i \oplus v_j) = m_i + m_j - \delta}$, где $\boldsymbol{\delta}$ – количество строк кода в общих методах графов $\boldsymbol{ic_i^r}$ и $\boldsymbol{ic_j^r}$. Максимальное значение $\boldsymbol{\delta}$ равно $\boldsymbol{\min(m_i, m_j)}$. Тогда, $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu(v_i)}$ и $\boldsymbol{\mu(v_i \oplus v_j) \geq \mu(v_j)}$, и Свойство 5 выполняется. Далее, пусть $\boldsymbol{\mu_i = \mu_j}$, и $\exists\, \boldsymbol{v_k} \in \boldsymbol{V}$ такой, что в граф $\boldsymbol{ic_k^r}$ входит $\boldsymbol{\beta}$ строк кода, общих с графом $\boldsymbol{ic_i^r}$ (см. Предположение 3), и $\boldsymbol{\delta}$ строк кода, общих с графом $\boldsymbol{ic_j^r}$, причем $\boldsymbol{\beta \neq \delta}$. Тогда,

$$\boldsymbol{\mu(v_i \oplus v_k) = \mu_i + \mu_k - \beta}$$
$$\boldsymbol{\mu(v_j \oplus v_k) = \mu_j + \mu_k - \delta}.$$

Следовательно, $\boldsymbol{\mu(v_i \oplus v_k) \neq \mu(v_j \oplus v_k)}$, и Свойство 6 выполняется. Свойства 7 и 8 также выполняются (см. раздел 4). Для любых двух методов $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$, пусть $\boldsymbol{\gamma}$ будет значением метрики LOC методов, общих для графов $\boldsymbol{ic_i^r}$ и $\boldsymbol{ic_j^r}$, и пусть $\boldsymbol{\delta}$ будет значением метрики LOC новых программ, написанных при реструктуризации графов $\boldsymbol{ic_i^r}$ и $\boldsymbol{ic_j^r}$ в новый граф $\boldsymbol{ic_{i'}^r}$. Если $\boldsymbol{\delta > \gamma}$, становится истинным отношение $\boldsymbol{m_i + m_j - \gamma + \delta > m_i + m_j}$ (то есть $\boldsymbol{\mu(v_i \oplus v_j) > \mu(v_i) + \mu(v_j)}$ при заданных $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$). Следовательно, Свойство 9 выполняется.

6.5.4 Важные соображения

Ниже приводится список перспектив, рассматриваемых в отношении RLOC:

- RLOC – это расширение LOC как метрики размера отдельного метода (то есть RLOC – это рекуррентное накопление LOC).
- Совокупный размер всех методов, которые косвенно зависят от некоторой нужной всем им функциональности, также может помочь спрогнозировать время и усилия, необходимые для разработки, понимания (включая локализацию зависимостей) и изменения требований, которые используют общий для всех метод.
- Более высокие значения RLOC некоторого метода увеличивают вероятность того, что изменение в нем затронет любую из зависящих от него частей программы (возможно возникновение каскадов ошибок или изменений, потребности в тестировании, реструктуризации).
- Считается, что методы с большим количеством зависимых методов легче использовать повторно; они менее специфичны для приложения.

6.5.5 Экспериментальные данные

Статистика метрики RLOC по всем системам представлена в табл. 7. В этой таблице показаны минимум, 1-й квартиль, медиана, 3-й квартиль и максимум для RLOC. Она также показывает наиболее часто отмеченное расстояние MFD между значениями метрик LOC и RLOC и долю методов MFD, имеющих именно такое расстояние MFD, среди всех методов. В ней также представлен коэффициент корреляции Кендалла между значениями метрик LOC и RLOC, а также *p-значение*.

Средние значения метрики RLOC для всех конечных методов всех систем показаны в табл. 2, а функции плотности метрики RLOC для всех систем изображены на рис. 10. Линейные диаграммы значений метрик RLOC и LOC для методов во всех системах, ранжированных по LOC, представлены на рис. 11.

Табл. 7. Сводная статистика для метрики RLOC
Table 7. Summary Statistics for the RLOC metric

Sys	MFD*	%MDF	$\tau^{\dagger}$	*p-val*	Min	1st Q	Med	3rd Q	Max
I_1	0	62	0.59	0	1	2	5	21	5,608
I_2	0	75	0.68	0	1	1	4	12	140,705
I_3	0	52	0.53	0	1	2	8	27	75,327
I_4	0	94	0.94	0	1	1	6	12	196
I_5	0	77	0.82	0	1	1	4	8	196
I_6	0	68	0.75	0	1	2	3	8	196
O_1	0	47	0.44	0	1	2	6	44	12,370
O_2	0	37	0.28	0	1	3	87	1,306	4,235
O_3	0	52	0.46	0	1	1	9	65	9,181
O_4	0	55	0.45	0	1	2	7	41	3,956
O_5	0	40	0.28	0	1	2	10	49	724

**MFD: наиболее частое расстояние между RLOC и LOC (MFD: most frequent distance between RLOC and LOC)*
†τ: коэффициент корреляции Кендалла между RLOC и LOC (τ: Kendall correlation coefficient between RLOC and LOC)

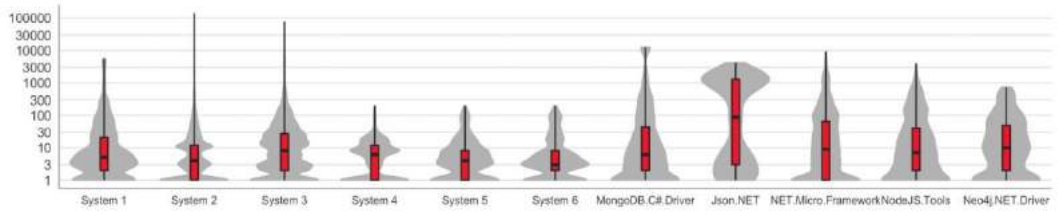


Рис. 10. Функция плотности для RLOC из всех систем
Fig. 10. Density function for RLOC from all systems

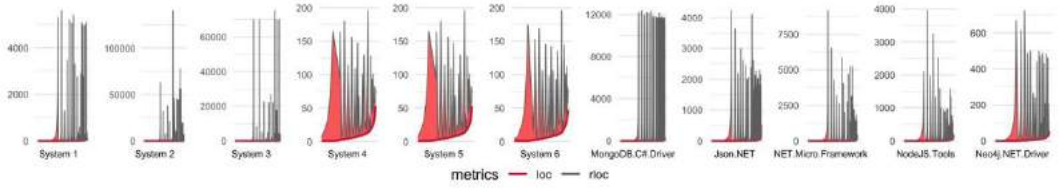


Рис. 11. Расстояние между RLOC и LOC для ранжированных методов всех систем
Fig. 11. Distance between RLOC and LOC values for the ranked methods from all systems

6.6 Вычисление показателя жесткости цикломатической сложности (RCYC)

RCYC – сумма значений сложности всех методов подграфа жесткости данного метода.

6.6.1 Определение

Пусть $\boldsymbol{v_i} \in \boldsymbol{V}$ – метод, $\boldsymbol{\mu = RCYC}$, и $\boldsymbol{ic_i} = \{\boldsymbol{v_1} \cdots \boldsymbol{v_n}\}$ – множество методов в графе $\boldsymbol{ic_i^r}$, тогда:

$$m_i = \mu_i = \mathrm{RCYC}_i = \sum_{k=1}^{n} \mathrm{CYC}_k - (n-1) \qquad (9)$$

В формуле (9) вычитаемое значение $(n-1)$ нужно для тех же целей, что и при вычислении метрики FCYC.

6.6.2 Теоретические основы

RCYC соответствует расширению метрики сложности CYC для метода. RCYC – это метрика CYC, примененная к подграфу косвенной связи жесткости этого метода (то есть она соответствует совокупной цикломатической сложности всех методов, которые могут прямо или косвенно вызывать метод, включая сложность самого этого метода). Вычитаемое в формуле (9) компенсирует лишние $(n-1)$ единиц. В этом показателе также учитывается влияние клиентов, подвергшихся рефакторингу при проектировании, на возможности управления размером программ и их сложностью, а также на повышение возможностей повторного использования.

6.6.3 Аналитические оценки

Пусть $\boldsymbol{v_i, v_j} \in \boldsymbol{V}$ – любые два разных метода системы $\boldsymbol{E}$. Предполагается, что переменные $\boldsymbol{CYC_i}$ и $\mathbf{CYC_j}$ обладают свойствами i.i.d. (см. Предположение 1), это же предположение относится к переменным $\boldsymbol{m_i} = \mathrm{RCY}\mathbf{C_i}$ и $\boldsymbol{m_j} = \mathrm{RCY}\mathbf{C_j}$. Тогда с ненулевой вероятностью $\boldsymbol{\mu_i} \neq \boldsymbol{\mu_j}$; Свойство 1 выполняется. Свойство 2 также выполняется (см. раздел 4).

Существует ненулевая вероятность того, что $\exists\ \boldsymbol{v_k} \in \boldsymbol{V} \mid \boldsymbol{\mu_i} = \boldsymbol{\mu_k}$, следовательно, Свойство 3 выполняется. Функциональность, выполняемая методом, не определяет цикломатическую сложность методов в его подграфе жесткости, поскольку это проектное решение, от функциональности не зависящее, Свойство 4 выполняется. Пусть $\boldsymbol{\mu_i} = \boldsymbol{m_i}$ и $\boldsymbol{\mu_j} = \boldsymbol{m_j}$, тогда $\boldsymbol{\mu}(\boldsymbol{v_i} \oplus \boldsymbol{v_j}) = \boldsymbol{m_i} + \boldsymbol{m_j} - \boldsymbol{\delta}$, где $\boldsymbol{\delta}$ – значение цикломатической сложности для методов, общих в графах $\boldsymbol{ic_i^r}$ и $\boldsymbol{ic_j^r}$. Максимальное значение $\boldsymbol{\delta}$ равно $\boldsymbol{\min(m_i, m_j)}$. Следовательно, $\boldsymbol{\mu}(\boldsymbol{v_i} \oplus \boldsymbol{v_j}) \geq \boldsymbol{\mu_i}$ и $\boldsymbol{\mu}(\boldsymbol{v_i} \oplus \boldsymbol{v_j}) \geq \boldsymbol{\mu_j}$, и Свойство 5 выполняется. Далее, пусть $\boldsymbol{\mu_i} = \boldsymbol{\mu_j}$, и $\exists\ \boldsymbol{v_k} \in \boldsymbol{V}$ такой, что цикломатическая сложность методов, общих в графах $\boldsymbol{ic_k^r}$ и $\boldsymbol{ic_i^r}$, равна $\boldsymbol{\beta}$ (см. Предположение 3), а цикломатическая сложность методов, общих в графах $\boldsymbol{ic_k^r}$ и $\boldsymbol{ic_j^r}$, равна $\boldsymbol{\delta}$, причем $\boldsymbol{\beta} \neq \boldsymbol{\delta}$. Тогда,

$$\boldsymbol{\mu}(\boldsymbol{v_i} \oplus \boldsymbol{v_k}) = \boldsymbol{\mu_i} + \boldsymbol{\mu_k} - \boldsymbol{\beta}$$
$$\boldsymbol{\mu}(\boldsymbol{v_j} \oplus \boldsymbol{v_k}) = \boldsymbol{\mu_j} + \boldsymbol{\mu_k} - \boldsymbol{\delta}.$$

Таким образом, $\boldsymbol{\mu}(\boldsymbol{v_i} \oplus \boldsymbol{v_k}) \neq \boldsymbol{\mu}(\boldsymbol{v_j} \oplus \boldsymbol{v_k})$. Свойство 6 выполняется. Свойства 7 и 8 также выполняются (см. раздел 4). Для двух произвольных методов $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$, пусть $\boldsymbol{\gamma}$ будет значением CYC для методов, общих в графах $\boldsymbol{ic_i^r}$ и $\boldsymbol{ic_j^r}$, и пусть $\boldsymbol{\delta}$ будет значением CYC новых программ, написанных для объединения графов $\boldsymbol{ic_i^r}$ и $\boldsymbol{ic_j^r}$ в новый граф $\boldsymbol{ic_i^r}$. Если $\boldsymbol{\delta} > \boldsymbol{\gamma}$, то $\boldsymbol{m_i} + \boldsymbol{m_j} - \boldsymbol{\gamma} + \boldsymbol{\delta} > \boldsymbol{m_i} + \boldsymbol{m_j}$ (то есть, $\boldsymbol{\mu}(\boldsymbol{v_i} \oplus \boldsymbol{v_j}) > \boldsymbol{\mu_i} + \boldsymbol{\mu_j}$ при заданных $\boldsymbol{v_i}$ и $\boldsymbol{v_j}$). Следовательно, Свойство 9 выполняется.

6.6.4 Важные соображения

В отношении метрики RCYC можно высказать следующие соображения:

- RCYC – более точная метрика цикломатической сложности метода, чем CYC, поскольку она учитывает значения CYC методов-клиентов.
- Объединенная цикломатическая сложность всех методов, которые должны выполнять какую-то общую работу, также может позволить прогнозировать время и усилия, необходимые для программирования, понимания (включая обнаружение подграфа жесткости или рискованности) и сопровождения функций, в реализации которых участвует данный метод.
- Методы с бóльшими значениями метрики RCYC могут с большей вероятностью распространить влияние любых изменений на вызывающие их методы.
- Методы с более сложными зависимостями легче использовать повторно, они менее специфичны для приложения.

6.6.5 Экспериментальные данные

Сводная статистика по метрике RCYC во всех системах представлена в табл. 8. В этой таблице показаны минимум, 1-й квартиль, медиана, 3-й квартиль и максимум для RCYC. Она также показывает наиболее часто отмеченное расстояние MFD между значениями метрик CYC и RCYC и долю методов MFD, имеющих именно такое расстояние MFD, среди всех методов. В ней также представлен коэффициент корреляции Кендалла между значениями метрик CYC и RCYC, а также *p-значение*.

Табл. 8. Сводная статистика для метрики RCYC
Table 8. Summary Statistics for the RCYC metric

Sys	MFD*	%MDF	$\tau^{\dagger}$	*p-val*	Min	1st Q	Med	3rd Q	Max
I_1	0	73	0.61	0	1	1	2	7	1,517
I_2	0	82	0.74	0	1	1	2	4	29,226
I_3	0	58	0.59	0	1	1	2	6	11,872
I_4	0	97	0.95	0	1	1	2	3	77
I_5	0	90	0.83	0	1	1	2	2	77
I_6	0	85	0.76	0	1	1	2	2	77
O_1	0	60	0.48	0	1	1	2	14	3,620
O_2	0	42	0.31	0	1	1	48	733	2,193
O_3	0	57	0.47	0	1	1	3	22	3,362
O_4	0	59	0.5	0	1	1	3	18	1,674
O_5	0	49	0.3	0	1	1	4	23	273

**MFD: наиболее частое расстояние между RCYC и CYC (MFD: most frequent distance between RCYC and CYC)*

†τ: коэффициент корреляции Кендалла между RCYC и CYC (τ: Kendall correlation coefficient between RCYC and CYC)

Средние значения метрики RCYC для всех конечных методов всех систем приведены в табл. 2. Функции плотности метрики RCYC для всех систем показаны на рис. 12. Линейные диаграммы значений метрик RCYC и CYC для методов во всех системах, ранжированные по CYC, представлены на рис. 13.

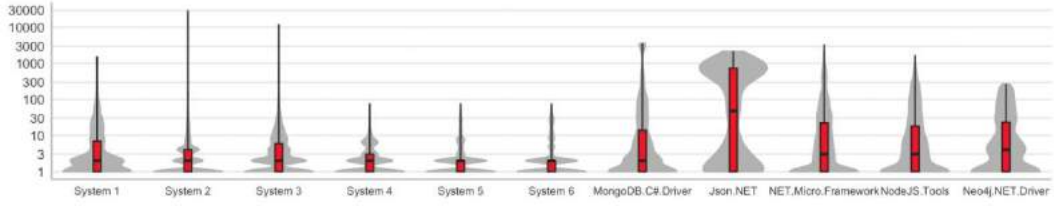


Рис. 12. Функция плотности для RCYC из всех систем
Рис. 12. Density function for RCYC from all systems

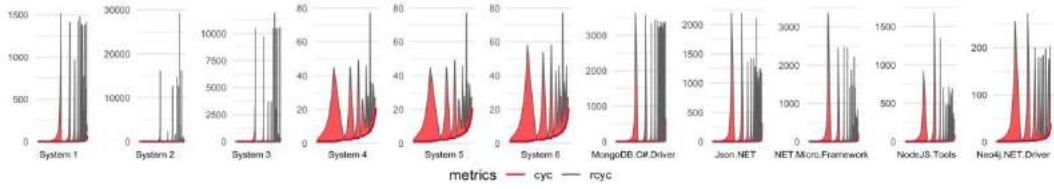


Рис. 13. Расстояние между RCYC и CYC для ранжированных методов всех систем
Fig. 13. Distance between RCYC and CYC values for the ranked methods from all systems

6.7 Обсуждение

В этом разделе обсуждаются рейтинги, присваиваемые метриками, аналитические результаты и результаты, имеющие отношение к управлению проектами.

6.7.1 Ранжирование с помощью метрик

Шесть представленных показателей косвенной связи измеряют различные и независимые атрибуты размера или сложности, используя либо хрупкость (неустойчивость) связи, либо ее жесткость (рискованность). Существуют три метрики хрупкости, основанные на одних и тех же подграфах хрупкости каждого метода, а также три метрики жесткости, основанные на одних и тех же подграфах жесткости. Нам могут возразить, заявив, что достаточно иметь только одну метрику хрупкости и одну метрику жесткости. В этой статье для подтверждения разницы между показателями использовался коэффициент корреляции Кендалла. В табл. 9 представлены коэффициенты для каждой системы и для каждой пары связанных показателей.

Табл. 9. Коэффициент корреляции Кендалла τ для каждой пары показателей
Table 9. Kendall's τ correlation coefficient for every pair of metrics

Sys	FNOM FLOC	FNOM FCYC	FLOC FCYC	RNOM RLOC	RNOM RCYC	RLOC RCYC
I_1	0.79	0.79	0.81	0.66	0.67	0.81
I_2	0.77	0.8	0.91	0.36	0.27	0.81
I_3	0.82	0.7	0.83	0.54	0.55	0.75
I_4	0.13	0.16	0.85	0.18	0.2	0.86
I_5	0.36	0.39	0.7	0.45	0.49	0.76
I_6	0.51	0.48	0.6	0.62	0.63	0.71
O_1	0.85	0.76	0.84	0.78	0.68	0.77
O_2	0.85	0.81	0.91	0.88	0.88	0.93
O_3	0.78	0.73	0.85	0.74	0.72	0.87
O_4	0.73	0.7	0.82	0.75	0.74	0.87
O_5	0.81	0.74	0.84	0.83	0.76	0.85

Метрики FLOC и RLOC являются парными метриками косвенных связей, связанными с атрибутами размера методов в подграфах хрупкости и жесткости. Это значит, что метрики FLOC и RLOC можно оставить в наборе метрик, поскольку они измеряют размер методов, а не их сложность. Кроме того, больший размер кода не означает более сложный код и наоборот. FLOC имеет значение корреляции τ, находящееся в диапазоне от 0,13 до 0,85 для FNOM, и значение τ, находящееся в диапазоне от 0,60 до 0,91 для FCYC. Аналогично, RLOC имеет τ от 0,18 до 0,88 для RNOM и τ от 0,71 до 0,93 для RCYC. Более высокие значения τ указывают на то, что соответствующая пара метрик присваивает методам в системе почти одинаковый рейтинг, но поскольку корреляция не идеальна, не все методы получают одинаковый рейтинг по обеим метрикам. С помощью FLOC и RLOC можно найти наиболее хрупкие или наиболее жесткие методы, рассматривая размеры косвенно связанных с ними методов.

6.7.2 Обсуждение аналитических результатов

Как показано в разделе 4, все предложенные метрики соответствуют расчетным критериям. Выявленные отношения между хрупкостью и жесткостью косвенных связей позволяют создавать робастные метрики, измеряющие размер и сложность программных систем.

6.7.3 Обсуждение результатов по управлению проектами

Администраторы и технические руководители процессами проектирования могут использовать набор метрик косвенных связей для оценки и управления размерами и сложностью программных проектов. Метрика FNOM может выявлять методы со многочисленными косвенными зависимостями. Вероятно, что именно эти методы реализуют наиболее сложные системные функции. Чтобы создать правильный проект и правильно его реализовать, именно этим методам следует уделять особое внимание,

Некоторая функциональность может приводить к существенному числу зависимостей, но метрика FNOM не сумеет этого выявить, однако, метрика FLOC обнаружит, что они реализованы многими и многими строками кода. Размер, измеряемый суммирующей метрикой LOC, учитывающей зависимости некоторого метода, поможет обнаружить такую функциональность и определить, не нужно ли разделить некоторые крупные фрагменты на более мелкие части, или провести рефакторинг программ. Какие-то другие функции могут быть рассредоточены по множеству длинных зависимостей, их можно обнаружить с помощью метрик FNOM и FLOC.

Метрика FCYC может помочь обнаружить функциональность, граф управления которой состоит из множества линейно независимых путей. Такое знание может оказаться полезным при разработке тестовых примеров для проверки функциональности и принятии решения о необходимости рефакторинга. Информация в графе ICG_i^f метода v_i также дает представление о способах реализации его функциональной структуры.

Остальные три метрики жесткости позволяют анализировать и обнаруживать атрибуты размера и сложности программ с учетом зависимых или повторно используемых методов. Метрика RNOM позволяет узнать количество методов, косвенно зависящих от данного метода, и дает фактическую степень повторного использования этого многим нужного метода. Благодаря этому пониманию специалист по сопровождению может изучить все зависимые методы и определить, какие из них необходимо изменить при корректировке переиспользуемого метода. Эти зависимые методы должны подвергнуться дополнительному тестированию. Если значение RNOM становится слишком большим для какого-либо метода v_i, специалист по сопровождению программ может проверить его подграф жесткости ICG_i^r и определить необходимость рефакторинга.

Принципы модульности и повторного использования широко используются разработчиками программного обеспечения для борьбы со сложностью программ. Следовательно, переиспользуемые программные компоненты встречаются нередко. Изменение такого рода методов чревато побочными эффектами в каждой строке кода, которая косвенно их переиспользует. И это кроме тех дополнительных усилий, которые необходимы для понимания того, какой из этих зависимых методов будет затронут изменениями. В результате могут стать необходимыми дополнительные изменения в зависимых методах, внедрение в программы новых скрытых ошибок или затруднение с обнаружением старых, рост потребностей в дополнительных ресурсах на проведение тестирования, или, в самом худшем случае, полный рефакторинг связанных методов. Метрика жесткости RLOC позволяет оценить подобные потенциальные последствия в строках кода. Объединив метрики RLOC и RNOM, можно обнаружить те методы и их подграфы ICG^r, которые имеют большое количество методов-клиентов, большой размер кода в строках кода методов-клиентов или и то, и другое одновременно.

Наконец, метрика RCYC может помочь измерить количество линейно независимых путей, на которых переиспользуется некоторый служебный метод. Эта метрика даст более точную оценку усилий, необходимых для тестирования метода после его изменения, или убедит в необходимости рефакторинга из-за технических недоработок.

6.8 Угрозы достоверности

Конструктивная достоверность. Проблема конструктивной достоверности связана с тем, действительно ли измеряем то, что хотим. Предлагаемый набор метрик построен на основе хорошо известных по литературе метрик: FAN-IN, FAN-OUT, LOC и CYC. Каждая метрика вычисляется суммированием метрик хрупкости и жесткости методов вдоль ребер графа косвенных связей ICG. Подграфы хрупкости отражают синтаксическую структуру разделенной на части и инкапсулированной функциональности, тогда как подграфы жесткости имеют дело со статическими шаблонами переиспользования. Обе статические структуры кода воплощают атрибуты размера и сложности, связанные с реализуемой ими функциональностью, и являются результатом хорошей практики проектирования, направленной на управление сложностью. Чтобы снизить вероятность того, что метрики хрупкости и жесткости не будут измерять ничего, кроме собственных базовых показателей (то есть FAN-OUT для FNOM, LOC для FLOC и так далее), к каждой паре базовых и косвенных метрик связи мы применили корреляционный анализ Кендалла. Каждая из шести возможных пар метрик хрупкости и жесткости (то есть FNOM-FLOC, FNOM-FCYC и FLOC-FCYC, RNOM-RLOC, RNOM-RCYC и RLOC-RCYC) подверглась одинаковому анализу. Это дало гарантию, что информация, извлеченная каждой из метрик, не является избыточной. Был также проведен анализ значений метрик прямых и косвенных связей с использованием линейных диаграмм этих значений для методов, ранжированных по значению прямой метрики, а затем по значению метрики косвенных связей. Это должно показать, что введенные метрики показывают результаты, отличные от значений соответствующих базовых показателей.

Внутренняя достоверность. В данном исследовании предполагается, что каждая из базовых метрик, применяемых к каждому методу в системе, представляет собой независимую и одинаково распределенную дискретную случайную величину (то есть нельзя предсказать значения метрик одного метода на основе метрик других методов). Мы предполагаем то же самое в отношении метрик косвенных связей. Чтобы гарантировать справедливость этого предположения, мы полагаемся на тот факт, что структура каждого подграфа косвенных связей вытекает из решений проектировщиков и разработчиков программного обеспечения, а не из значений метрик. Мы также ссылаемся на то, что метрики удовлетворяют всем критериям достоверности, предложенным в работе [34].

Внешняя достоверность. Внешняя достоверность относится к обобщению результатов исследования. Мы изучили шесть промышленных систем от двух разных компаний, две из них были разработаны по классической транзакционной схеме “Модель-Представление-Контроллер” (Model-View-Controller, MVC), три – на основе самостоятельно созданной инфраструктуры быстрой разработки приложений (Rapid Application Development, RAD), которая сама рассматривается в качестве системы № 6. Мы также изучили пять систем с открытым исходным кодом. Языком программирования этих систем является C#, что возможно оставляет в стороне системы, написанные с использованием других языков программирования. Кроме того, можно было пропустить некоторые косвенные связи в системе из-за ошибок при анализе исходных текстов. Мы снижаем эти риски, подсчитывая значения метрик с помощью официального компилятора, как это будет делаться с любым другим языком, который будет использован в будущем. Более того, косвенные связи между методами, функциями или модулями в большинстве современных языков программирования аналогичны и служат той же цели.

7. Заключение

В этой статье не только представлены метрики, которые имеют прочную теоретическую основу и были тщательно изучены, но эти метрики также применены к исследованию нескольких реальных коммерческих проектов, использовавших язык C#, и к программным

системам с открытым исходным кодом. В статье представлена полезная информация о том, как эти метрики можно использовать при сопровождении программного обеспечения, а также предлагаются рекомендации по их эффективной интеграции. Далее следует некоторый краткий обзор этих рекомендаций.

Опытный, квалифицированный руководитель проекта для оценки времени и усилий, необходимых для понимания и сопровождения функциональности методов объектов может использовать метрику FNOM. Эту информацию можно использовать для сравнения требований к ресурсам различных методов и систем. Кроме того, FNOM может выявлять методы, которые с набольшей вероятностью подвержены влиянию ошибок и изменений в связанных с ними программах. Это поможет своевременно выделить ресурсы для тестирования их функциональности после проведения модификаций. Благодаря собранной информации руководитель проекта может принимать обоснованные решения и оптимизировать распределение ресурсов, обеспечивая эффективное и результативное проведение проекта.

Помимо оценки технических недоработок, этот подход может помочь принять решение о необходимости проведения рефакторинга программных систем.

Также могут помочь достижению тех же целей метрики FLOC и FCYC, одним и тем же методам системы они дают оценки, отличающиеся от оценок FNOM. Эти метрики могут выявить более крупные или более сложные процедуры с точки зрения количества строк кода или цикломатической сложности, а это может помочь принять решения о приоритетах в рефакторинге.

Для оценки хрупкости системы рекомендуется использовать все метрики в совокупности. Сравнивая среднюю хрупкость методов разных систем, руководство проектов может принимать обоснованные решения о перераспределении ресурсов между различными системами. Высокий уровень хрупкости может также указывать на ограниченную возможность повторного использования кода, поскольку более сложный и специфичный код имеет тенденцию быть менее пригодным для повторного использования. Более того, подграф хрупкости методов, которые необходимо изменить, может проявить функциональность, которая требует поддержки при сопровождении.

Метрики жесткости (рискованности) в первую очередь ориентированы на измерение возможности повторного использования, размера и сложности подграфов жесткости методов. Каждая из этих метрик – RNOM, RLOC и RCYC – обеспечивает различное ранжирование методов, их можно комбинировать и определять, какие из них имеют зависимый код, который либо более обширен, либо более сложен, чем другие. Особенно жесткие методы (имеющие высокий риск зависимости от обращающихся к ним методов), выявленные с помощью этих показателей, требуют большего внимания с точки зрения обновления и тестирования по сравнению с менее жесткими методами (с низким уровнем такого риска). Разработчики могут использовать эти метрики для определения приоритетности усилий по сопровождению и тестированию, гарантируя, что высоко рискованные методы особенно тщательно модифицируются и тестируются перед включением в работу. Используя метрики жесткости, команды разработчиков программного обеспечения могут заранее выявлять и устранять проблемы, влияющие на качество и сопровождение программного обеспечения.

Список литературы / References

[1]. M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski, “Metrics and laws of software evolution-the nineties view”, Proceedings Fourth International Software Metrics Symposium (1997) pp. 20–32.

[2]. Almeyda and A. Dávila, “Process improvement in software requirements engineering: A systematic mapping study”, Programming and Computer Software 48, 513–533 (2022).

[3]. ISO/IEC 14764, Software Engineering – Software Life Cycle Processes – Maintenance, Standard (International Organization for Standardization, 2006(E)).
[4]. Priyadarshi Tripathy Naik and Kshirasagar, Software Evolution and Maintenance: A Practioner's Approach (John Wiley & Sons, Inc, 2015) p. 393.
[5]. Shyam R. Chidamber and Chris F. Kemerer, "Towards a Metrics Suite for Object Oriented Design", in OOPSLA '91 Conference proceedings on Object-oriented programming systems, languages, and applications (ACM Digital Library, Phoenix, Arizona, USA, 1991) pp. 197–211.
[6]. Wei Li and Sallie Henry, "Object-Oriented Metrics that Predict Maintainability", Journal of Systems and Software 23, 111–122 (1993).
[7]. Shyam R. Chidamber and Chris F. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering 20, 476–493 (1994).
[8]. Lionel Briand, Prem Devanbu, and Walcelio Melo, "An investigation into coupling measures for C++", in ICSE '97 Proceedings of the 19th international conference on Software (ACM Digital Library, 1997) pp. 412–421.
[9]. Thomas Zimmermann, Peter Weisgerber, Stephan Diehl, and Andreas Zeller, "Mining Version Histories to Guide Software Changes", in Proceedings of the 26th International Conference on Software Engineering (IEEE Computer Society, 2004) pp. 563–572.
[10]. Ewan Tempero and Paul Ralph, "A Framework for Defining Coupling Metrics", Science of Computer Programming, 1–17 (2018).
[11]. Timothy C. Lethbridge and R. Lagani`ere, Object-Oriented Software Engineering: Practical Software Development Using UML and Java, 2nd ed. (McGraw-Hill, 2005) p. 561.
[12]. Hong Yul Yang, Measuring Indirect Coupling, Ph.D. thesis, University of Auckland (2010).
[13]. Shari Lawrence Pfleeger and Shawn Bohner, "A Framework for Software Maintenance Metrics", in Proceedings of the Conference on Software Maintenance (1990) pp. 320–327.
[14]. Denys Poshyvanyk, Andrian Marcus, Rudolf Ferenc, and Tibor Gyimóthy, "Using information retrieval based coupling measures for impact analysis", Empirical Software Engineering 14, 5–32 (2009).
[15]. Gabriele Bavota, Bogdan Dit, Rocco Oliveto, Massimilano Di Penta, Denys Poshyvanyk, and Andrea De Lucia, "An empirical study on the developers' perception of software coupling", in Proceedings of the International Conference on Software Engineering (ICSE '13) (IEEE, 2013) pp. 692–701.
[16]. A. M. Frolov, "A hybrid approach to enhancing the reliability of software", Programming and Computer Software 30, 18–24 (2004).
[17]. Johann Eder, Gerti Kappel, and Michael Schrefl, Coupling and Cohesion in Object-Oriented Systems, Tech. Rep. 1 (University of Klagenfurt, Austria, 1992).
[18]. M Hitz and B Montazeri, "Measuring coupling and cohesion in object-oriented systems", Proceedings of the International Symposium on Applied Corporate Computing 50, 75–76 (1995).
[19]. Thomas Zimmermann and Nachiappan Nagappan, "Predicting defects using network analysis on dependency graphs", Proceedings of the 30th International Conference on Software Engineering, 531 (2008).
[20]. Nasib S. Gill and Balkishan, "Dependency and interaction oriented complexity metrics of component-based systems", ACM SIGSOFT Software Engineering Notes 33, 1 (2008).
[21]. V. N. Kasyanov, "Graph applications in programming", Programming and Computer Software 27, 146–164 (2001).
[22]. L.C. Briand, J. Wust, and H. Lounis, "Using coupling measurement for impact analysis in object-oriented systems", in Proceedings IEEE International Conference on Software Maintenance (ICSM '99) (IEEE Xplore, Oxford, England, UK, 1999) pp. 475–482.
[23]. Alan MacCormack, John Rusnak, and Carliss Baldwin, "Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis", Research Policy 41, 1309–1324 (2012).
[24]. M. Durán, R. Juárez-Ramírez, S. Jiménez, and C. Tona, "User story estimation based on the complexity decomposition using Bayesian networks", Programming and Computer Software 46, 569–583 (2020).
[25]. F. Valdés-Souto and Lizbeth Naranjo-Albarrán, "Improving the software estimation models based on functional size through validation of the assumptions behind the linear regression and the use of the confidence intervals when the reference database presents a wedge-shape form", Programming and Computer Software 47, 673–693 (2021).
[26]. Huan Li and Bing Li, "A pair of coupling metrics for software networks", Journal of Systems Science and Complexity 24, 51–60 (2011).

[27]. Ran Mo, Yuanfang Cai, Rick Kazman, Lu Xiao, and Qiong Feng, “Decoupling level: A New Metric for Architectural Maintenance Complexity”, Proceedings of the 38th International Conference on Software Engineering - ICSE ’16, 499–510 (2016).
[28]. Saleh Almugrin, Waleed Albattah, and Austin Melton, “Using indirect coupling metrics to predict package maintainability and testability”, Journal of Systems and Software 121, 298–310 (2016).
[29]. Robert Lagerstr¨om, Carliss Baldwin, Alan MacCormack, Dan Sturtevant, and Lee Doolan, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 10379 LNCS (2017) pp. 53–69.
[30]. N. I. V’yukova, V. A. Galatenko, and S. V. Samborskii, “Dynamic program analysis tools in gcc and clang compilers”, Programming and Computer Software 46, 281–296 (2020).
[31]. A. A. Timakov, “Information flow control in software db units based on formal verification”, Programming and Computer Software 48, 265–285 (2022).
[32]. Robert C. Martin, Agile Software Development: Principles, Patterns, and Practices (Pearson Education, Inc., New Jersey, USA, 2003) p. 557.
[33]. Norman Fenton, “Software Measurement: A Necesary Scientific Basis”, IEEE Transactions on Software Engineering 20, 199–206 (1994).
[34]. Elaine J Weyuker, “Evaluating Software Complexity Measures”, IEEE Transactions on Software Engineering 14, 1357–1365 (1988).
[35]. Sriram Pemmaraju and Steven Skiena, Computational discrete mathematics: combinatorics and graph theory with Mathematica (Caambridge University Press, 2003) p. 497.
[36]. Thomas J. McCabe, “A Complexity Measur”, IEEE Transactions on Software Engineering SE-2, 308–320 (1976).
[37]. Harold N. Gabow, “Path-based depth-first search for strong and biconnected components”, Information Processing Letters 74, 107–114 (2000).
[38]. MongoDB, “.NET Driver for MongoDB” (2019).
[39]. Newtonsoft, “Json.NET: Popular high-performance JSON Framework for .NET” (2019).
[40]. NETMF, “.NET Micro Framework Interpreter” (2019).
[41]. Microsoft, “Node.js tools for Visual Studio” (2019).
[42]. Neo4j, “Neo4j .NET Driver” (2019).
[43]. Maurice G. Kendall, Rank Correlation Methods, 4th ed. (Griffin London, London, England, 1970) p. 202.
[44]. All density functions in the Figures use log10 scale to avoid the graphics from squashing.
[45]. Except those methods whose difference between FNOM and FAN-OUT is the MFD (this applies to all other line plots as well).
[46]. M. V. Ksenzov, “Architectural refactoring of corporate program systems”, Programming and Computer Software 32, 31–43 (2006).
[47]. M. H. Halstead, “Toward a theoretical basis for estimating programming effort”, in ACM 1975 Annual Conference (ACM Digital Library, 1975) pp. 222–224.

Информация об авторах / Information about authors

Хосе НАВАС-СУ – бакалавр в области компьютерных наук Технологического института Коста-Рики (1994), магистр в области компьютерных наук из Технологического института Коста-Рики с отличием Magna Cum Laude (2017), кандидат наук в области инженерии в Технологическом институте Коста-Рики под руководством доктора Антонио Гонсалеса Торреса. Работал инженером-программистом в течение трех десятилетий, включая работу в многонациональных компаниях, таких как Accenture и GFT. Преподаватель кафедры компьютерных наук Коста-Риканского технологического института.

Jose NAVAS-SU is an Instructor at the Department of Computer Science of the Costa Rica Institute of Technology. He is a candidate for Ph.D. in Engineering at the Costa Rica Institute of Technology under the supervision of Dr. Antonio González Torres, he obtained a master’s degree in Computer Science (2017) from the Costa Rica Institute of Technology with the Magna Cum Laude distinction, and obtained a bachelor’s degree in Computer Science (1994) from the Costa Rica Institute of Technology. José has worked as Software Engineer for three decades, including working for multinational companies, such as Accenture and GFT.

Антонио ГОНСАЛЕС-ТОРРЕС – бакалавр в области компьютерных наук (1999), магистр в области компьютерных наук (2001) в Университете Коста-Рики, магистр в области интеллектуальных систем (2014) в Университете Саламанки (Испания), доктор философии в области компьютерных наук и автоматизации с отличием Summa Cum Laude, международная докторская степень (2015). В рамках подготовки докторской диссертации выполнял исследования в Открытом университете Великобритании. Доцент кафедры вычислительной техники Коста-Риканского технологического института. Имеет более чем 20-летний профессиональный опыт, в течение которого он работал в нескольких многонациональных компаниях, включая Walmart, Intel, Equifax, Global Exchange group (Eurodivisas) и Sykes. Параллельно с профессиональной работой работал инструктором Cisco и профессором университета, участвовал в нескольких исследовательских проектах. Сфера научных интересов: программная инженерия, разработка методов и инструментов визуальной аналитики и кибербезопасность.

Antonio GONZALEZ-TORRES is an Assistant Professor at the Department of Computer Engineering of the Costa Rica Institute of Technology. He obtained a Ph.D. in Computer Science and Automation with the Summa Cum Laude and the International Doctorate (2015) distinctions, and a master's degree in Intelligent Systems (2014) from the University of Salamanca (Spain). As part of his doctorate, he made a research stay at the Open University of the United Kingdom. In addition, he received a master's degree in Computer Science (2001) and a bachelor's degree in Computer Science (1999) from the University of Costa Rica. Dr. González has more than 20 years of professional experience, during which he has worked for several multinational companies, including Walmart, Intel, Equifax, Global Exchange group (Eurodivisas) and Sykes. In parallel with his professional work, he has worked as a Cisco instructor and university professor and has participated in several research projects. His area of expertise is software engineering, the design of visual analytics methods and techniques, and cybersecurity.

DOI: 10.15514/ISPRAS-2023-35(6)-4

Онтология архитектурных знаний в совместно локализованной “живой” среде

[1] *Х.Л. Роблес*, ORCID: 0000-0002-0695-1021 <*jose.robles99633@potros.itson.edu.mx*>
[2] *Х. Боррего*, ORCID: 0000-0001-7315-5693 <*gilberto.borrego@itson.edu.mx*>
[2] *Р.Р. Паласио*, ORCID: 0000-0002-4059-2149 <*ramon.palacio@itson.edu.mx*>
[3] *Ф. Кастильо-Баррера*, ORCID: 0000-0001-7821-5819 <*ecastillo@uaslp.mx*>

[1] *Технологический институт Соноры,*
Мексика, штат Сонора, Сьюдад-Обрегон.
[2] *Технологический институт Соноры,*
Мексика, штат Сонора, Навохоа.
[3] *Автономный университет Сан-Луис-Потоси,*
Мексика, Сан-Луис-Потоси.

Аннотация. Небольшие компании, ведущие “живую” разработку программного обеспечения, сталкиваются с новой реальностью удаленной разработки. В удаленном взаимодействии создаётся множество видеозаписей, извлекаемых из видеозвонков и используемых в последующей работе. Записанные видеозвонки содержат архитектурные знания, собранные на виртуальных встречах, они важны для компаний, сталкивающихся с проблемой испарения знаний. Однако в литературе редко встречаются предложения по управлению архитектурным знанием, имеющимся в видеозаписях. В статье предлагается решение для восстановления архитектурных знаний, содержащихся в видео, которое, следуя концепции конденсации архитектурных знаний, в качестве классификационной схемы использует онтологию. В соответствии с руководящими принципами Methontology была проведена валидация предложений по управлению архитектурными знаниями. Внедрение онтологии как схемы классификации представляет собой шаг вперед к достижению консолидации архитектурных знаний в гибкой среде разработки для микропредприятий.

Ключевые слова: микропредприятия; живая разработка; архитектурное знание; конденсация знания.

Для цитирования: Роблес Х. Л., Боррего Х., Паласио Р. Р., Кастильо-Баррера Ф. Онтология архитектурных знаний в совместно локализованной agile-среде. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 75–94. DOI: 10.15514/ISPRAS–2023–35(6)–4.

Благодарности: Эта исследование стало возможным, благодаря Мексиканскому национальному совету по науке и технологии CONACYT, Технологическому институту Соноры и Автономному университету Сан Луис Потоси.

Ontology for Architectural Knowledge Condensation in a Co-Localized Agile Environment

[1] *J.L. Robles*, ORCID: 0000-0002-0695-1021 <*jose.robles99633@potros.itson.edu.mx*>
[2] *G. Borrego*, ORCID: 0000-0001-7315-5693 <*gilberto.borrego@itson.edu.mx*>
[2] *R.R. Palacio*, ORCID: 0000-0002-4059-2149 <*ramon.palacio@itson.edu.mx*>
[3] *F. Castillo-Barrera*, ORCID: 0000-0001-7821-5819 <*ecastillo@uaslp.mx*>

[1] *Instituto Tecnológico de Sonora*,
Ciudad Obregón, Sonora, Mexico.
[2] *Technological Institute of Sonora*,
Navojoa, Sonora, Mexico.
[3] *Autonomous University of San Luis Potosi*,
San Luis Potosi, Mexico

Abstract. Agile software development companies considered small entities (VSE) face a new reality of remote development. Remote communication has generated many videos derived from video calls recorded for later reference. The video calls recorded contains architectural knowledge from virtual meetings and is essential for companies facing the knowledge vaporization problem. However, only some proposals in the literature can potentially manage AK in videos. This article proposes a solution to recover this architectural knowledge contained in videos using an ontology as a classification scheme, following the architectural knowledge condensation concept. We validated our proposal to manage architectural knowledge following the Methontology guidelines. Implementing an ontology as a classification scheme represents a step forward to achieving the condensation of architectural knowledge in an agile development environment for VSE.

Keywords: very small entities; agile development; architectural knowledge; condensation knowledge.

For citation: Robles J. L., Borrego G., Palacio R. R., Castillo-Barrera F. Ontology for architectural knowledge condensation in a co-localized agile environment. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 75-94 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-4.

Acknowledgements. This research work was possible thanks to CONACYT, the Instituto Tecnológico de Sonora, and the Universidad Autónoma de San Luis Potosí.

1. Введение

В настоящее время в компаниях[1] все чаще внедряются принципы “живой” (agile) разработки программного обеспечения. Ее популярность обусловлена в основном возможностью быстрого получения результатов [1] и тем, что работающая программа важнее сложной документации [2]; кроме того, люди и их общение важнее процессов и инструментов, присущих процессу разработки [2]. Однако чрезмерное внимание к кодированию может привести к пробелам в документации, которые представляют собой разновидность технического долга (technical debt), возникающего из-за пренебрежения или приостановки действия одного или нескольких активов, связанных с процессом разработки, то есть из-за недостаточности документации по разработке [3], часто оставляемой в незаконченном виде из-за нежелания тратить время и силы [4]. Такая ситуация влияет на архитектурное знание (Architectural Knowledge, AK), которое строится из архитектурных проектов и решений с их обоснованиями, на основе которых определяется архитектура программного обеспечения [5].

Отсутствие документации по архитектурному знанию может быть результатом следования ценностям манифеста живой разработки [2], и часто воспринимается разработчиками как проблема только после наступления последствий, например, таких, как потеря или “испарение” архитектурного знания [6]. Среди распространенных последствий “испарения”

[1] *https://info.digital.ai/rs/981-LQX-968/images/SOA15.pdf*

можно назвать следующие: неправильно понятые требования, задержки в процессе разработки и отсутствие технического видения проекта [4, 7-8].

Кроме того, в “живой” разработке предпочтение отдается личному взаимодействию, поскольку оно считается наиболее эффективным и действенным способом передачи информации внутри команды разработчиков [2] в общей для них коммуникационной среде [9]. Таким образом, следование манифесту может привести к преобладанию неявного архитектурного знания. Согласно Нонаке и Такеучи [10], для эффективного обмена знаниями необходимо пройти четыре этапа модели создания знания (SECI). Сюда входят следующие этапы:

- Обобществление (Socialization), когда архитектурное знание передается другому человеку устно (оставаясь неявным),
- Проявление (Externalization), когда архитектурное знание передается путем записи в нестандартные документы, становясь явным,
- Сборка (Combination), когда документы и артефакты формализуются путем применения правил, форматов или протоколов, облегчающих их интерпретацию (например, UML [11]), и
- Отчуждение (Internalization), когда документы и артефакты с формальным архитектурным знанием используются для передачи знаний другим людям, инициируя новый цикл.

Поскольку в “живой” разработке создание документации отходит на второй план, этап сборки знаний не доводится до конца, следовательно, архитектурное знание не формализуется [12]. Складывающаяся ситуация не дает возможности эффективно обмениваться знаниями, а это приводит к их “испарению” [6] и, как следствие, к появлению незрелого или неполного архитектурного проекта [13], который не соответствует уровню качества, необходимому для полноценной разработки программ [14-15]. Эти проблемы возникают из-за неявности управления архитектурным знанием или из-за неформальностей, либо чрезмерной специализации обозначений [16], допущенных в документации (например, в чертежах, технических спецификациях, диаграммах, эскизах).

Несмотря на это, неявное архитектурное знание считается очень важным, благодаря опыту разработчиков, который необходим для принятия архитектурных решений [17] и тоже является источником знаний для разработчиков. Однако если архитектурное знание не формализовано, то в будущем оно не сможет быть правильно понято, а значит, в итоге становится ненадежным источником архитектурного знания [7].

Отсутствие надежных источников архитектурного знания приводит к взаимозависимости в команде разработчиков при решении вопросов о программных проектах, создавая “склады знаний”, то есть взаимозависимость с опытными разработчиками (или старшими по должности в компании) [18]. Такая взаимозависимость часто становится раздражителем для разработчиков, являющихся источником архитектурного знания, поскольку им приходится много раз отвечать на одни и те же вопросы. Это приводит к разрушению личных отношений и негативно сказывается на общении разработчиков [19]. Возможным выходом из этой ситуации могло бы стать включение в практику формализация архитектурного знания (например, использование унифицированного языка моделирования (UML) [11]), однако, это может противоречить правилам “живой” разработки [20] и желаниям разработчиков [21]. Другим возможным решением может быть применение специальной среды разработки, например, среды Scaled Agile Framework (SAFe[2]), для работы в которой явно назначается ответственный за разработку архитектуры. Для крупных компаний, в силу их высокой штатной численности, это вполне осуществимо. В отличие от крупных компаний, небольшие

[2] *https://www.scaledagileframework.com*

компании или отделы разработки программного обеспечения (фактически, микропредприятия – Very Small Entities, VSE) не могут применять такие подходы из-за ограниченного количества сотрудников (согласно ISO/IEC-29110-4-2[3], микропредприятия имеют в штате не более 25 сотрудников). Это представляет собой существенную проблему, поскольку только на американском рынке 99% предприятий относятся к малому бизнесу (в том числе и микропредприятиям)[4]. Тем самым, микропредприятия полностью подвержены проблеме испарения архитектурного знания.

Таким образом, из-за нежелания разработчиков документировать, а также из-за того, что применение сред разработки или методов формального представления архитектурного знания для микропредприятий не представляется возможным, проявляется тенденция поиска архитектурного знания в репозиториях и на известных платформах, например, на Stack Overflow или Github [22-23]. Другой распространенной практикой является постоянный анализ исходного кода проектов с целью поиска архитектурного знания [24], однако этот процесс крайне медленный, неэффективный и чреват наличием ошибок, поскольку получаемые таким образом знания, как правило, не структурированы, неполны и непоследовательны. Таким образом, в обоих случаях эти методы могут нарушить привычный рабочий процесс разработчика и, следовательно, замедлить его, а проблема испарения останется латентной.

В последнее время появилась новая возможность уменьшить испарение архитектурных знаний. Поскольку среди компаний, ведущих живые разработки стал популярен удаленный режим работы (к этому подтолкнули ограничения из-за распространения COVID-19 [25]), увеличилось количество виртуальных встреч, и эти видеозвонки часто записываются для последующего использования [26]. В этих записях может содержаться значительное архитектурное знание из сессий с вопросами-ответами (QA sessions) и периодических совещаний, на которых решаются важные вопросы и принимаются архитектурные решения. Однако извлечение необходимых знаний может быть затруднено из-за отсутствия механизма организации и классификации видеозаписей, полученных в ходе виртуальных встреч; таким образом, эти знания тоже подвержены риску испарения. Такая ситуация грозит стать постоянной, поскольку многие компании, занимающиеся разработкой программного обеспечения, решили продолжить работу в удаленном режиме, ощутив новые возможности и преимущества такого способа организации работы [27-28].

Чтобы облегчить извлечение знаний из записей виртуальных встреч может оказаться полезной концепция конденсации этих знаний [12]. Эта концепция заключается в сборе архитектурных знаний, разбросанных на различных носителях, их классификации и облегчении доступа к ним с помощью поискового механизма [12]. Однако знания часто распространяются на различных носителях, платформах, в виде различных артефактов, что делает их классификацию сложной [29]. Знания могут распространяться и в виде записей видеозвонков. Реализация механизма классификации архитектурных знаний облегчила бы реализацию концепции конденсации знаний, а согласно работе [29], в онтологиях эта реализация действительно возможна.

В этой работе мы предлагаем цикл конденсации архитектурного знания, основанный на исходной концепции конденсации [12] и следующий рекомендациям модели SECI [10] и Интегрированного цикла управления знаниями (Integrated knowledge management cycle [30]). Цикл конденсации архитектурного знания включает в себя три этапа:

(I) Сбор информации (Acquisition), при котором из различных источников усваивается архитектурное знание, которое тем самым приобретается людьми.

[3] *https://www.iso.org/standard/77735.html*

[4] *https://www.forbes.com/advisor/business/small-business-statistics/*

(II) Классификация (Classification), при которой собранные архитектурные знания классифицируются по определенной схеме, и

(III) Объединение знаний (Sharing), при котором архитектурное знание передается другим людям для начала нового цикла (см. рис. 1).

Из рис. 1 видно, что некоторые активности напрямую связаны с конкретным состоянием модели SECI, например, групповая виртуальная встреча, предполагающая групповое взаимодействие посредством видеозвонка, где архитектурное знание записью видеозвонка трансформируется из неявного в явное. Запись видеозвонка становится источником знания, эта запись может быть использована для получения и классификации знания, а впоследствии также и для обмена знаниями. Следуя этим рекомендациям, мы использовали цикл конденсации знаний для анализа возможностей такой конденсации применительно к конкретной работе, для определения этапов цикла, которые можно реализовать, и для управления сферой применения архитектурного знания. Сформулировав наши предложения по циклу конденсации знаний и проанализировав предложения, опубликованные другими авторами, мы, стремясь добиться полноценной конденсации знаний, предложили решение, которое соответствует руководящим принципам цикла конденсации архитектурных знаний, ориентированным на видеозаписи, создаваемые по видеозвонкам и размещаемыми на различных платформах, содержащих такие знания. Наше предложение основано на концепции онтологий, которую мы рассматриваем в качестве неотъемлемой части механизма классификации архитектурных знаний. Целью этой статьи является реализация онтологии, способной организовать и облегчить поиск этих видеозаписей, в независимости от их происхождения и формата, содействуя поиску и доступу к знаниям в них, следуя рекомендациям цикла конденсации архитектурных знаний. Мы представляем сценарий нашего онтологического решения, чтобы пояснить реализацию концепции конденсации знаний при помощи механизма классификации, умеющего размечать видеозаписи (например, такого, как система социальных тегов), обеспечивая управление этими записями и доступ к ним путем реализации механизма поиска.

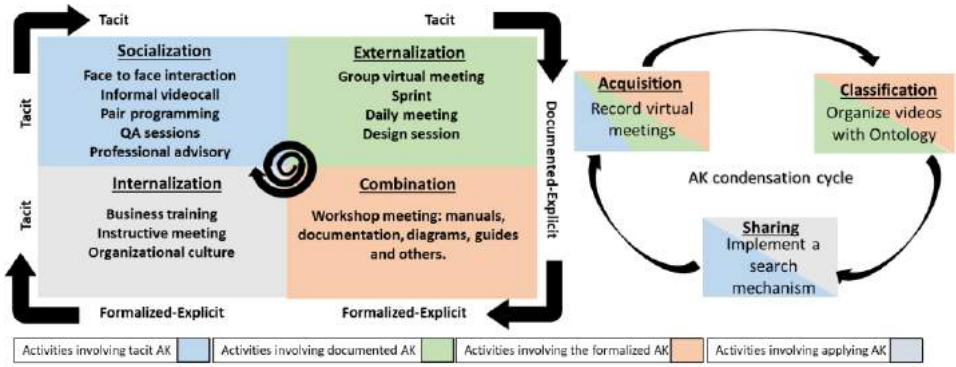


Рис. 1. Взгляд на "живые" активности модель SECI и цикл конденсации. Отображено, какие состояния знаний рассматриваются на каждом цикле конденсации
Fig 1. View of the agile activities related to the state of knowledge through the SECI model and the condensation cycle. We show which knowledge states are considered in each phase of the condensation cycle

Остальная часть статьи построена следующим образом. В разделе 2 представлены смежные работы, посвященные механизмам обмена архитектурными знаниями, использованию видеозаписей для обмена знаниями и предложению новой онтологии. В разделе 3 представлены наши предложения по сценарию решения поставленной задачи. В разделе 4 представлена методология построения предложенной онтологии. В разделе 5 представлены

результаты, полученные в соответствии с этой методологией. В разделе 6 представлено обсуждение полученных результатов, а в разделе 7 – выводы и предложения по дальнейшим работам.

2. *Смежные* работы

Конденсация архитектурного знания – это процесс сбора и классификации знаний, снижающий риск их потери и обеспечивающий доступ к ним для их совместного использования. В ходе данного исследования мы проанализировали работы по управлению архитектурным знанием через поддержку платформ как источников знаний, ввели цикл конденсации знаний, основанный на оригинальной концепции конденсации архитектурного знания [12] и интегрированном цикле управления знаниями Далкира [30], включающем состояния архитектурного знания модели SECI [10], чтобы объяснить трансформацию знаний на каждом этапе (см. рис. 1).

2.1 Предложения по управлению архитектурным знанием

Управление архитектурным знанием основано на спирали знаний [10] и имеет состояния, в которых знания существуют в неявном (I) и явном (II) видах. Однако согласно модели SECI [10], анализируя активности на каждом этапе, можно выделить “явно документированные” подсостояния архитектурного знания, когда это знания могут существовать в неформализованных документах или в виде конспективных записей (например, заметки на наклейках). Существует также “явно формализованные” подсостояния, когда правила или форматы обеспечивают создание и интерпретацию артефактов и документов (например, UML [11]).

В работе [31] предложена схема, характеризующая потерю знаний, вызванную текучкой кадров. Они выделяют полную потерю, когда носитель неявных знаний отсутствует постоянно, и частичную, когда она носит временный или внезапный характер. Они описывают 20 последствий потери знаний, объединенных в четыре темы: отсутствие руководства, опора на документацию, работа с коллегами и воссоздание знаний. Авторы рекомендуют своевременно оформлять документацию для предотвращения потери знаний, привлекая коллег к обсуждению проблем и стратегий их решения. По сути работа представляет собой руководство по документированию для предотвращения потери знаний.

В работе [32] предложена программная среда для управления проектными решениями по архитектуре крупных программных проектов. Эта среда предназначена для решения конкретных задач – извлечения и классификации проектных решений, аннотирования архитектурных элементов, выработки рекомендаций по альтернативным вариантам решений и назначения экспертов для принятия проектных решений. Для реализации этих задач необходимо проводить планирование и оценку, итеративно используя при этом научный, исследовательский подход к проектированию. Среда позволяет соотнести систему управления архитектурными знаниями с этапами проектирования, разработки и сопровождения. Она помогает заинтересованным сторонам документировать проектные решения и извлекать уроки из уже выполненных проектов.

С другой стороны, в работе [33] предлагается инструмент для принятия архитектурных решений, которым документируются решения и проблемы заинтересованных сторон. Инструмент записывает решения по пяти позициям: взаимоотношения, вовлечение заинтересованных сторон, сила, хронология и детали. Это помогает разработчикам архитектуры анализировать и управлять решениями, однако для его более эффективного использования заинтересованным сторонам на адаптацию может потребоваться время. Для небольших команд, ведущих живые разработки с прямым общением, процесс документирования можно упростить, что сократит количество решений. В работе [34] предлагается метод документирования архитектурного знания, названный подходом C3A

(CA[5] Agile Architecture) и предназначенный для фаз выработки архитектурных решений и проектирования жизненного цикла разработки программного обеспечения. Предлагаемый авторами метод основан на использовании эталонной архитектуры. Он заключается в регистрации шаблона для написания характеристик каждого программного компонента с описанием каждого элемента, необходимого для развития процесса разработки. Авторы приходят к выводу о том, что архитектура CSA позволяет сопоставлять процесс разработки с элементами системной архитектуры. Такой подход может помочь устранить разрыв между стратегическим мышлением организации, занимающейся разработкой, и тактикой проводимой ею "живой" реализации. Аналогичным образом, система поддержки принятия решений по разработке архитектуры (Architectural Design Decision Support System, ADDSS) представляет собой веб-инструмент, который помогает создавать и документировать архитектурные проектные решения и их связи с другими программными артефактами [35]. В работе [36] предлагается система Knowledge Trace Retrieval (KTR), предназначенная для извлечения элементов решения проектных задач из деловой электронной почты. Хотя инструменты и методы управления знаниями являются промышленными стандартами, их использование в небольших проектах не является регулярным. Система помогает пользователям извлекать следы знаний из большого объема различных электронных писем, полученных в рамках прошлых проектов. Для поиска знаний система использует расширенный контекст (данные о проекте, компетенции и профили пользователей), методы машинного обучения и алгоритмы ранжирования. Аналогичным образом в работах [37-38] для сокращения количества документации предложен метод репертуарной сетки (Repertory Grid Technique, RGT), который помогает предотвратить потерю архитектурного знания за счет документирования архитектурных решений.

Для решения проблемы фиксации и обмена архитектурными решениями в работе [39] представлены высокоуровневые истории проектирования (High-level Design stories, HLDs), которые представляют собой небольшие артефакты, содержащие архитектурные решения проекта, контекст, предположения, анализ качественных характеристик и нерешенные вопросы. Истории HLDs являются модульными, они создаются и уточняются в ходе архитектурно-ориентированного процесса разработки для подтверждения решений и проблем, помогая получить подробный глобальный проект архитектуры. Еще одним предложением по поиску архитектурного знания является дополнение Slack под названием TaggerBot [40], которое представляет собой виртуального тематического помощника, призванного уменьшить потерю знаний, неявно возникающих при разработке программного обеспечения. Помощник использует следы архитектурного знания в сообщениях и взаимодействиях через платформу Slack и реализует механизм классификации знаний на основе социальных тегов. Однако, как и в более ранних предложениях, цель была сосредоточена в основном на документировании знаний.

По этой причине некоторые авторы предложили повторное использование знаний в качестве быстрого решения проблемы поиска знаний, особенно в процессе разработки. В работах [41-42] повторное использование кода рассматривается как необходимое для понимания проблем в процессе разработки программного обеспечения. Однако, хотя разработчики могут увеличить свои знания и принимать архитектурные решения на основе консультаций по текстам программ [24, 26], консультации по архитектурному знанию обусловлены необходимостью документирования [43, 24], вызванной задолженностью по документации [4]. Заинтересованные стороны постоянно ищут наилучший способ обмена и сохранения архитектурного знания [26], приспосабливая решения к своей среде разработки, например, записывая виртуальные встречи и обмениваясь видеороликами для ответов на вопросы. В этом смысле в работе [44] предлагается метод анализа различий между использованием

[5] *https://www.broadcom.com/*

видеоматериалов для обучения и подготовки к работе и ручного обучения без видеоматериалов. Анализ направлен на повышение производительности труда путем минимизации времени и затрат на обучение работников за счет использования видеоматериалов в качестве источника обучения. Результаты показали, что использование видеоматериалов в качестве источника обучения значительно повышает производительность труда, минимизируя время и стоимость обучения работников. По мнению авторов работы [45], видеоматериалы могут рассматриваться как эффективный инструмент обмена информацией в любой отрасли и могут способствовать обучению, развитию критического мышления и сотрудничеству. Хотя существуют работы, в которых использование видеоматериалов рассматривается как источник знаний и механизм коммуникации, большинство проанализированных предложений ориентировано на применение видеоматериалов для тренировок персонала или в образовательных целях, что далеко от потребностей микропредприятий, работающих в гибкой среде, где видеоматериалы необходимы как средство консультаций по вопросам архитектурного знания.

В табл. 1 приведены краткие сведения о предложениях авторов работ, рассмотренных в этом разделе. В столбце “Уровень знаний” указан уровень управления архитектурными знаниями в соответствии с моделью SECI [10]. В столбце “Проверено?” указано, сообщалось ли в той или иной статье о валидации предложений авторов. В столбце “Этап” указаны этапы цикла конденсации архитектурного знания, охваченные предложениями, где С – это сбор знаний, К – классификация знаний, а О – обмен знаниями.

Согласно модели SECI [10], все найденные работы по поддержке управления архитектурным знанием в живых средах используют это знание в *явно задокументированном* состоянии (см. табл. 1). Как утверждается в работе [20], это происходит так, поскольку в этих предложениях знание, как правило, фиксируется неформально: в живых средах механизмы такого рода считаются слишком навязчивыми из-за того, что заинтересованные стороны создают документацию во время, отведенное прежде всего на разработки, отвлекаясь от “живого” процесса. Несомненно, удаленная работа повлияла на интенсивность обмена архитектурным знанием в живых средах [46], по этой причине возникли усилия по его минимизации, однако они предпринимались только на одном или двух этапах (сбор/классификация) цикла конденсации знаний и не касались этапа обмена знаниями. Можно также отметить, что во многих предложениях не был указан размер компании, на которую оно было ориентировано. Некоторые из них прошли валидацию, показав целесообразность сбора и/или классификации архитектурных знаний.

Табл. 1. Классификация связанных работ
Table 1. Related work classification

Название предложения	Уровень знаний	Тип предложения	Вид компании	Проверено?	Этап
Робиллард [31]	Явно документированный	Фреймворк	Не указан	Нет	С
RGT [37-38]	Явно документированный	Технология	Любой	Да	С
C3A [34]	Явно документированный	Подход	Не указан	Нет	С
Бхат [32]	Явно документированный	Фреймворк	Крупный	Нет	С
ADDSS [35]	Явно документированный	Инструмент	Не указан	Да	С
KTR [36]	Явно документированный	Инструмент	Не указан	Да	С
Мантейфель [43]	Явно документированный	Модель	Не указан	Нет	С
HLD [39]	Явно документированный	Модель	Не указан	Нет	С
TaggerBot [40]	Явно документированный	Инструмент	Крупный	Да	С, К
Типпаннавар [44]	Явно документированный	Метод	Не указан	Нет	С
Веддер-Вайс [45]	Явно документированный	Обучение	Образовательный	Да	С

Согласно имеющимся в литературе данным, существуют предложения, непосредственно связанные с управлением архитектурными знаниями (рис. 2). Рассматривая этап сбора знаний цикла конденсации, можно найти предложения по механизмам управления неявных знаний (например, [37-38]), явных знаний (например, [35-36]), переиспользуемых знаний ([41-42]), которые предполагают обращение к прикладным знаниям [24] или ранее принятым решениям [43], подчеркивая использование этих артефактов в качестве источника знаний [26], как в случае записанных видеозвонков, к которым можно обратиться позже. Что касается применения механизмов классификации и организации, то для конденсации содержащейся в них информации необходимо иметь записи знаний, хранящиеся в репозиториях. Найденные предложения, потенциально способные выявить архитектурные знания, обычно относятся к стадиям сбора и классификации знаний (документированные архитектурные знания) и, как правило, нуждаются в развитии или реализации в реальном сценарии. Поэтому следующий этап – обмен архитектурными знаниями – обычно пропускается или выполняется с использованием неформальных и очень специальных документов [12, 16], что нарушает этап объединения знаний модели SECI [10], вызывает задолженность по документации [4] и приводит к испарению знаний.

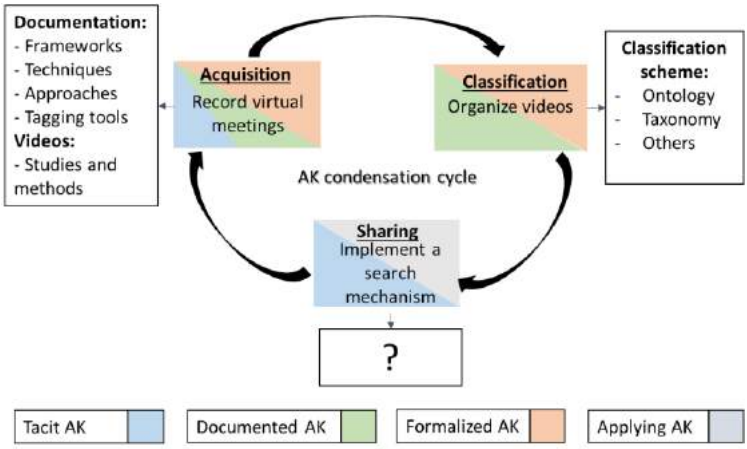


Рис. 2. Предложения по циклу конденсации знаний. Белые поля для каждого этапа цикла содержат предложения, представленные в смежных работах, посвященных соответствующему этапу. Предложения для этапа объединения знаний отсутствуют. В рамках каждого из этапов цикла конденсации показаны основные виды деятельности, связанные с нашим предложением, а также состояния и подсостояния, в которых находится архитектурное знание

Fig. 2. Proposals through the AK condensation cycle. The white boxes for each cycle phase contain proposals presented in the related work focused on the corresponding phase. There are no proposals for the Sharing phase. Within each of the phases of the condensation cycle, the main activities related to our

2.2 Онтологии как средство классификации архитектурных знаний

Одним из способов повторного использования архитектурного знания является применение онтологий, которые представляют собой формальное описание связанных друг с другом обобщенных понятий [47]. Онтология позволяет повторно использовать знания путем прояснения неоднозначностей [26] и способна облегчить процесс конденсации знаний, поскольку вводит словарь для обмена информацией и помогает принимать более конкретные архитектурные решения [48]. Онтологии возникают как инструмент управления знаниями, поддерживающий представление, обработку, хранение и извлечение знаний [49], что необходимо для разработки программного обеспечения [50].

Хотя некоторые из проанализированных здесь предложений направлены на сохранение знаний, в живой разработке программ сохранить его удается редко из-за нарушений цикла знаний [20], которые мешают формализации архитектурных знаний. Складывающаяся ситуация приводит к тому, что разработчики стремятся реализовать неявные механизмы управления знаниями или создают документы с неформальными обозначениями. Именно поэтому необходимость в механизме, формализующем документы и артефакты архитектурного знания, создаваемые в процессе живой разработки, побудила нас разработать онтологию, с помощью которой можно устанавливать семантические связи между документами и артефактами, независимо от их происхождения и формата. В работе [29] авторы в своем исследовании показывают, как онтология может классифицировать знания, помогая в поиске и сохранении знаний из различных источников, в организации знаний и их соотнесении с их создателями. Эта семантическая связь влияет на то, как разработчики принимают решения при возникновении различных вопросов или проблем. Поскольку разработчикам нужна надежная и точная информация [51], они обычно ищут эти решения в различных источниках знаний (помимо обращения к экспертам) [52]. В работе [49] предлагается онтология сбора знаний о среде разработки программного обеспечения для более успешного проведения его тестирования. Предложение касается тестировщиков и разработчиков, участвующих в тестировании программ, и включает базу знаний о контексте разработки, дающую возможность обращаться к этим знаниям, когда они необходимы, и способствующую принятию высокоэффективных решений, связанных с тестированием.

Можно сделать вывод о том, что будет интересно рассмотреть, каким образом онтология может способствовать конденсации архитектурных знаний, записанных в видеоформатах, которые, как правило, при проведении “живой” разработки содержат соглашения разработчиков и их важные решения, связанные с их деятельностью по разработке.

3. Предложение по управлению архитектурным знанием для микропредприятий, ведущих “живую” разработку

В этом разделе мы описываем проблему взрывного роста числа видеоматериалов и трудностей поиска архитектурных знаний на основе ранее проведенного экспериментального исследования [26]. Эта трудности проистекают из большого количества видеозаписей виртуальных встреч сотрудников микропредприятий, ведущих разработки по “живым” правилам.

Вместо того чтобы делать заметки или вести протоколы заседаний, чтобы сохранить информацию о проекте (принятые решения, ответы на вопросы, решенные проблемы и так далее), которая может содержать архитектурное знание, заинтересованные стороны ведут видеозаписи виртуальных встреч. Чтобы компенсировать недостаток личного общения, заинтересованные стороны обычно проводят много виртуальных встреч, а это приводит к появлению большого количества многочасовых видеозаписей. Поиск конкретного знания в этих файлах может быть трудным, поскольку заинтересованным сторонам приходится вести поиск во многих видеозаписях, во всех файлах, минута за минутой. Это приводит к тому, что заинтересованные стороны тратят слишком много времени, надолго прерывая свою основную деятельность по разработке [20]. Из-за необходимости просмотра большого количества проверяемых видеофайлов, заинтересованные стороны часто не могут найти нужную им информацию. Это обстоятельство может привести к тому, что информацию приходится запрашивать у коллег, отправляя им текстовые сообщения или даже делая видеозвонки (что усугубляет проблему), часто вызывая у них раздражение, поскольку часто запрашивается информация, уже обсуждавшуюся на прошедших встречах [20].

Основываясь на концепции конденсации архитектурного знания [12] и учитывая потребности микропредприятий в управлении знаниями, мы представляем предлагаемое нами решение с учетом удаленной “живой” разработки программного обеспечения [26], используя

преимущества записи виртуальных встреч, которые происходят в коллективах и в которых члены коллективов сохраняют важные разговоры, содержащие архитектурное знание. Первым шагом цикла конденсации знаний является получение этих знаний, поэтому видеозаписи обязательно должны размещаться в репозиториях (например, на Google Drive) с включенным общим доступом, позволяющим обращаться к видеозаписям из любого источника.

На виртуальных совещаниях заинтересованные стороны имеют возможность помечать (ставить теги) конкретные видеофрагменты, содержащие ценные знания, для последующего их поиска. Эти теги должны соответствовать существующим метатегам в классах онтологии, поскольку в качестве классификационной основы предлагаемое нами решение предполагает использование онтологии. Онтология представляет собой модель знаний, устанавливающую смысловые связи между тегами знаний и заранее определенной иерархической структурой. Таким образом, использование тегов, связанных с онтологией, позволяет автоматически организовать знания, спрятанное в видеофайлах виртуальных встреч. Благодаря использованию онтологии этап классификации цикла конденсации архитектурного знания выполняется достаточно эффективно.

Более того, присущие онтологии характеристики дают определенные преимущества для реализации механизмов поиска. Онтология позволяет осуществлять систематический поиск, охватывая всю онтологию посредством иерархических сущностей. Теги могут быть связаны с классами и подклассами, что повышает удобство запросов к онтологии. Интеграция механизма поиска на основе онтологии позволяет завершить цикл конденсации знаний, упростить управление знаниями за счет хорошо организованных видеоматериалов, содержащих потенциальное знание. Для реализации предложенного нами решения очень важно создать схему классификации, поэтому мы и выбрали онтологию. Разработка онтологии предполагает проведение сложного процесса построения абстракций, в ходе которого определяются отношения, основанные на требованиях управления знаниями к организации записанных виртуальных встреч, хранящихся в репозиториях. Следующим логическим шагом является создание онтологии для очерчивания области знаний, полностью охватывающей все этапы цикла конденсации знаний. Такой целостный подход направлен на реализацию эффективного управления знаниями в рамках живой разработки для микропредприятий.

4. Методика разработки онтологии

Для проведения данного исследования мы следовали пяти этапам, рекомендованным технологией Methontology [53] и работой [29]. Далее мы опишем каждый этап, выполняемые на них работы и ожидаемые от них результаты.

Этап спецификации состоит в определении требований к онтологии. Мы определили требования на основе ранее проведенного экспериментального исследования [26]. Мы составили спецификацию требований к онтологии (Ontology Requirements Specification Document, ORSD), в которой описали назначение, область применения, язык реализации, предполагаемых конечных пользователей и возможные варианты использования онтологии.

Этап концептуализации: мы организовали и структурировали информацию, получив модель знаний, представленную в виде таксономии.

Этап формализации: определив модель знаний, мы создали онтологию с помощью редактора онтологий Protégé, определили классы и реализовали семантические связи между ними, в результате чего получилась онтологическая сущность.

Этап оценки: мы провели верификацию и валидацию онтологии с помощью набора компетентностных вопросов (Competency Questions, CQ), который был определен в соответствии с рекомендациями Methontology и в соответствии с нашим сценарием. Мы решили использовать механизм логического вывода Pellet, учитывая положительные

результаты, полученные при оценке онтологий OWL [54]. В процессе валидации мы убедились, что онтология соответствует своему назначению, а в процессе верификации выяснили что она удовлетворяет необходимым нам требованиям и корректно функционирует.

Этап сопровождения начинается с расширения исходной онтологии, предложенной в работе [29]. Этот этап также включает в себя обновление онтологии в соответствии с изменениями в домене или потребностями управления видеоинформацией. В рамках изменений, реализованных при обслуживании онтологии, мы добавили новые свойства данных и объектов, сделав акцент на актуальности семантических отношений архитектурных знаний, содержащихся в видеофайлах. В итоге мы получили полный список данных и свойств объектов онтологии.

5. Результаты

В данном разделе представлены результаты выполнения каждого этапа описанной выше методики.

Этап спецификации: результаты этапа спецификации являются частью документа ORSD[6], который определяет назначение и область применения онтологии, тем самым устанавливая параметры и ограничения, которые она будет иметь в процессе разработки.

Этап концептуализации: в рамках результатов этапа концептуализации мы создаем таксономию, которая объясняет отношения между объектом архитектурного знания и некоторым членом команды разработчиков. В то же время команда в рамках конкретного проекта может потребовать или создать артефакты архитектурного знания. В полученной таксономии мы показываем элементы, которые определяют организованные объекты и придают смысл отношениям, которые они имеют друг с другом; например, член команды может иметь различные роли и опыт, артефакты могут иметь различные форматы и размещаться на различных платформах, а проекты, к которым эти артефакты относятся, могут вестись на разных языках программирования.

Этап формализации: на основе классификационной структуры таксономии с помощью инструмента Protégé мы определили следующие классы (см. рис. 3.A): Artifacts (представление характеристик артефакта и место его размещения), Layers (представление архитектурного слоя, к которому относятся артефакты), Project (представление типа проекта, к которому относятся артефакты) и Team (представление члена команды разработчиков, связанного с артефактами: создатель и консультант). Связи между классами онтологии осуществлялись с помощью свойств объектов (рис. 3.B).

Этап оценки: проведена валидация и верификация онтологии с использованием логического вывода Pellet (рис. 4.A) со следующими вопросами:

CQ1. Какие видеоролики создал “имя участника”?
CQ2. Какие видеоролики использовал “имя участника”?
CQ3. Сколько тегов создал “имя участника”?
CQ4. Где расположено видео “имя видео”?
CQ5. Сколько видеороликов было создано в проекте “название проекта”?
CQ6. Сколько видеороликов было создано членом команды “имя участника”?
CQ7. Сколько видеороликов создал “имя участника» в проекте “название проекта”?
CQ8. Где в “имя видео” находится “имя тега”?

[6] *https://drive.google.com/file/d/1TPeMuI7udG7rtsx2jQO20xxIjGakqQMT/view?usp=sharing*

Эти вопросы были получены из сценария, представленного в разделе 3, который был основан на ранее проведенном эксперименте с разработчиками живых проектов [26]. Для проверки целостности нашей онтологии и подтверждения логической структуры мы использовали раздел DL Query программы Protégé (рис. 4.B). Использование механизм логического вывода облегчает оценку целостности определенных классов и их семантических связей. На рис. 4 видно, что процесс валидации и верификации с использованием вопросов по компетенциям прошел успешно, это подтверждает согласованность структуры онтологии и ее готовность ответить на любой другой вопрос того же типа.

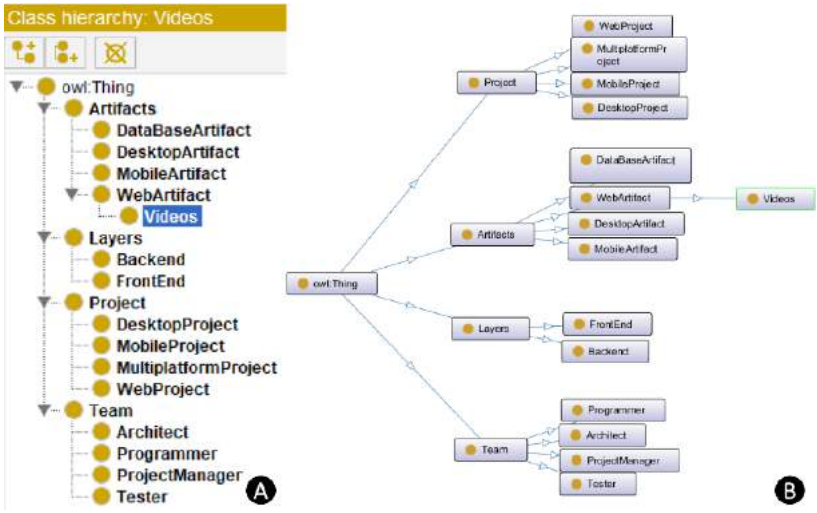


Рис. 3. Представления онтологии: (A) Вид иерархии классов. (B) Вид онтографа Protégé
Fig. 3. Ontology views: (A) Classes hierarchy view. (B) Protégé ontograph view

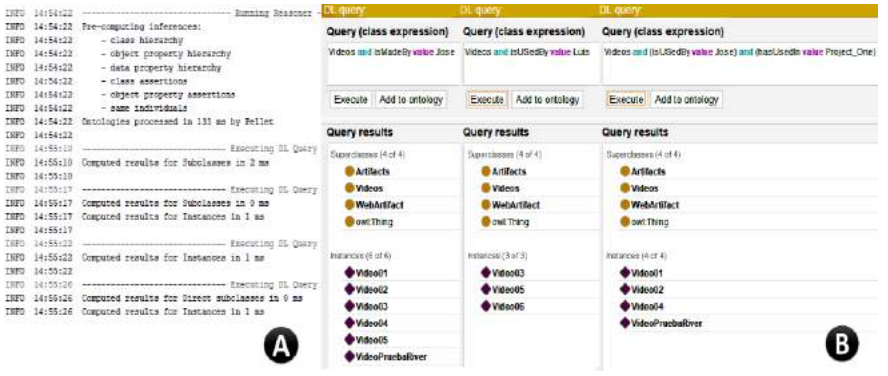


Рис. 4. Оценка онтологии:
(A) Результаты работы консоли системы Protégé. (B) Результаты DL-запросов
Fig. 4. Ontology evaluation: (A) Protégé system console results. (B) DL query results

Этап сопровождения: результаты этапа сопровождения начинаются с расширения онтологии, предложенного в работе [29]. Мы создали сущность “Video” как подкласс класса “WebArtifact”, чтобы классифицировать содержащее архитектурное знание видео, созданное на основе записей виртуальных встреч. Для класса “Programmer” мы создали суперкласс “Team” и добавили сущности “Architect”, “ProjectManager” и “Tester”, созданные в одной иерархии с классом “Programmer”, чтобы расширить возможности классификации и определить область архитектурного знания. Кроме того, в класс “Project” добавлены

подклассы “MultiplatformProject” для организации гибридных или мультиплатформенных проектов. Полная структура онтологии представлена на рис. 3.

Что касается свойств данных, то мы добавили новые атрибуты в “artifactDescription”. Мы добавили атрибут “artifactFormat” для определения формата видео, атрибут “artifactLocation” для определения платформы хранения видео, атрибут “artifactTag” для определения меток видеоконтента на временной шкале, а также атрибут “artifactCreationDate” для определения даты создания. Для свойства “projectLanguage” мы добавили атрибут “projectPlatform”, определяющий платформу проекта. Наконец, для свойства “userDescription” мы добавили атрибут “userRole” для определения ролей заинтересованных сторон в проекте. Для свойств объекта реализован атрибут “hasTaggedBy”, определяющий того, кто создает метку в видеоролике. Полный список данных и свойств объектов приведен ниже:

ArtifactDescription: *artifacLocation, artifactTag, artifactFormat, artifactLanguaje, artifactSubject, artifactCreationDate.*

ProjectDescription: *projectPlatform, projectName, projectLanguage*.

UserDescription: *userRole, userEmai, userExperience, userGende, userName.*

6. Обсуждение результатов исследования

Микропредприятия, работающие в стиле “живой” разработки, сталкиваются с серьезной проблемой, связанной с архитектурными знаниями, генерируемыми в ходе виртуальных встреч, которые обычно записываются в видеофайлы. Мы определили цикл конденсации знаний, с помощью которого смогли заметить, что большая часть опубликованных работ сосредоточена на предложениях, облегчающих документирование знаний и оставляющих в стороне усилия по классификации и облегчению доступа к знаниям, что необходимо для восстановления знаний, разбросанных на различных носителях, их классификации и облегчения доступа к ним, то есть конденсации знаний [12].

В ходе анализа литературы мы обнаружили предложения, направленные на смягчение последствий потери знаний. Такие предложения обусловлены важностью восстановления знаний, обычно размещенных на различных носителях. В качестве примера можно привести предложение СЗА [34] – артефакт, предназначенный для создания документации проектов на этапе разработки архитектуры. СЗА определяет, какие аспекты необходимо документировать, и для каждого аспекта определяет в документе отдельный раздел, что облегчает его создание. Следует, однако, помнить, что модель SECI [10] определяет состояние, называемое явно-документированным, которое относится к знаниям, зарегистрированным в нестандартных форматах; таким образом, СЗА не обеспечивает корректной интерпретации знаний в будущем, поскольку при его создании не соблюдаются никакие правила или протоколы. Со временем может возникнуть риск того, что архитектурное знание перестанут понимать или забудут, то есть оно испарится [12].

Напротив, предлагаемое нами решение использует в качестве механизма классификации онтологию и позволяет осуществлять регистрацию знаний в классифицированном и организованном виде, поскольку содержит формальную структуру, определяемую семантическими отношениями и иерархией классов, то есть онтологию, что позволяет осуществлять поиск знаний и облегчает доступ к ним путем реализации запросов через их классы. В отличие от работ, встречающихся в литературе, наше предложение позволяет придать регистрации знаний большую формальность, обеспечивая доступность знаний в случае необходимости. Это свойство снижает вероятность того, что знания будут утеряны и перестанут быть понятными, поскольку онтология позволяет семантически связать знания с рабочей средой микропредприятия, что создает контекст использования и происхождения знаний путем создания их иерархической организации. Таким образом, мы могли бы генерировать новые предложения по управлению знаниями, реализуя механизмы поиска,

ориентированные на документирование, классификацию и облегчение доступа к знаниям, хранящимся в проектах разработки программного обеспечения.

Интересным предложением для механизма классификации является помощник TaggerBot [40], в котором применен метод классификации архитектурного знания на основе таксономии. Однако в этой классификации рассматриваются только концепции, и проверить работоспособность такого механизма классификации можно только путем проведения оценок с помощью контролируемых экспериментов или в реальных сценариях, что требует значительных усилий и больших ресурсных затрат. Наше предложение может быть проверено инструментами логического вывода и компетентными вопросами, то есть проверка реализуемости онтологии требует меньших усилий, чем создание таксономии. Таким образом, онтология, составляющая суть нашего предложения, является базой для дальнейших запросов и выводов, подтверждающих возможность классификации новых типов артефактов.

Реализация нашей онтологии обеспечивает классификацию знаний, делая более удобным его извлечение. Однако остается необходимость реализации механизма разметки знаний тегами, подобный механизму, используемому в TaggerBot. Этот механизм должен быть основан на социальном тегировании и иметь ограничения, препятствующие взрывному росту числа тегов. В отличие от предложения TaggerBot, использующего реляционную базу данных, запросы ограничены конкретной архитектурой. Предлагаемое нами решение обеспечивает гибкость в реализации технологий и архитектур для хранения знаний. В то же время поддержка реляционной базы данных может потребовать значительных усилий. Кроме того, для сопровождения и поддержки онтологии нужны методики и средства управления изменениями и редактирования.

Аналогичным образом, видео является средством хранения и обмена знаниями [44-45]. Однако встречающиеся в литературе предложения пока не посвящены управлению их содержимым. Несомненно, богатство этого вида артефактов для микропредприятий имеет большое значение. Поэтому наше предложение может облегчить классификацию видеоконтента за счет использования тегов, например TaggerBot [40], для маркировки фрагментов видео, которые содержат интересные АЗ, а соответствующие вопросы АЗ могут решаться заинтересованными сторонами. При использовании нашего предложения заинтересованным сторонам не нужно будет просматривать множество видеоматериалов, чтобы найти конкретный фрагмент АЗ, поскольку искомое АЗ можно будет найти без особых усилий, так как оно уже будет помечено и классифицировано. Таким образом, АЗ будут связаны с сущностями онтологии, оставляя АЗ доступным для консультации и обсуждения любой из заинтересованных сторон. Предложенная онтология обеспечивает гибкость запросов, если они находятся в пределах области МСП.

7. Выводы и дальнейшая работа

В представленной работе рассматривалась необходимость извлечения архитектурных знаний, которые микропредприятиями хранятся в виде видеоматериалов. Важность данной работы заключается в том, что необходимо выявить видеофрагменты, содержащие необходимую информацию для проектов микропредприятий, поскольку из-за большого количества хранящихся видеозаписей и отсутствия протоколов или правил их сохранения и обеспечения доступа к ним в будущем найти необходимые знания, содержащиеся в видеозаписях, не так просто.

Определение цикла конденсации АЗ было необходимо для анализа литературы. Это позволило нам определить объем предложений с точки зрения управления архитектурными знаниями в живых средах. Определив цикл, мы обнаружили, что большинство работ сосредоточено на качестве только одного этапа цикла, который заключается в сборе знаний,

оставляя в стороне классификацию и доступ к этим знаниям. Мы поняли, что необходимо найти способ повысить эффективность реализации всех трех этапов цикла.

Для этого, используя формальные рекомендации, предложенные в [53], мы построили онтологию, а затем следовали этой методологии, стараясь оценить свою онтологию. В результате мы поняли, что онтология пригодна для классификации видеороликов и их содержания. В отличие от других предложений, известных по литературе, наше решение позволяет классифицировать и давать легкий доступ к видеофрагментам, в которых содержится соответствующая проектная информация, связанная с архитектурным знанием. Кроме того, в онтологию заложен потенциал не только для управления видео артефактами, но и для управления любыми артефактами с архитектурным знанием. Наше решение создает возможность извлечения знаний, содержащихся в видеоматериалах микропредприятий, тем самым предотвращая риск испарения знаний.

В данной работе мы представили разработку онтологии для решения проблемы классификации архитектурных знаний. Наше предложение также способствует реализации механизма поиска, позволяющего дополнить цикл конденсации знаний. Разработанная онтология была проверена с помощью инструментов логического вывода и компетентностных вопросов, что позволяет сделать вывод о том, что онтология полностью пригодна для реализации в соответствии с поставленными задачами, и заключить, что наше предложение может быть реализовано с использованием механизма классификации в виде онтологии, прошедшей валидацию. Кроме того, в смежных работах мы не нашли предложений, которые можно использовать в качестве онтологии для классификации знаний, которая могла бы быть реализована в виде системы. Таким образом, наше предложение может стать шагом вперед в поиске программных решений, решающих поставленную задачу управления знаниями. В качестве дальнейшей работы следующим шагом будет разработка программного комплекса, реализующего разработанную в данной работе онтологию, для проверки правильности модели знаний и наличия потенциала разработанной онтологии для управления знаниями. Определение механизма включает в нашу онтологию механизм разметки тегами и механизм поиска, которые облегчают реализацию всех этапов цикла конденсации архитектурных знаний.

По этой причине необходимо найти механизм, который сопряжет нашу онтологию с механизмом тегирования и механизмом поиска, которые поспособствуют реализации всех этапов цикла конденсации знаний. В рамках дальнейшей работы мы также рассматриваем возможность внедрения генератора слов для системы тегирования, чтобы помочь заинтересованным сторонам в создании тегов, и подчеркиваем, что для стимулирования создания тегов заинтересованными сторонами может потребоваться внедрение социальной системы тегирования. Наконец, для выполнения всех этапов цикла конденсации архитектурных знаний необходима разработка веб-системы, использующей разработанную в данной работе онтологию.

Список литературы / References

[1]. P. Jain, A. Sharma, and L. Ahuja, “The Model for Determining Weight Coefficients of Maintainability Criteria in Agile Software Development Process,” in 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1–4.

[2]. K. Beck et al., “Manifesto for Agile Software Development,” The Agile Alliance, 2001.

[3]. W. Cunningham, “The WyCash portfolio management system,” in Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, 1992.

[4]. E. Tom, A. Aurum, and R. Vidgen, “An exploration of technical debt,” J. Syst. Softw., 2013.

[5]. P. Kruchten, P. Lago, and H. Van Vliet, “Building up and reasoning about architectural knowledge,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2006.
[6]. J. Bosch, “Software architecture: The next step,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2004.
[7]. H. Holz and F. Maurer, “Knowledge management support for distributed agile software processes,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2003.
[8]. A. R. N. Uikey, U. Suman, “A Documented Approach in Agile Software Development,” Int. J. Softw., vol. Vol. 2, no, pp. 13– 22., 2011.
[9]. R. K. Kavitha and M. S. I. Ahmed, “A knowledge management framework for agile software development teams,” in Proceedings of 2011 International Conference on Process Automation, Control and Computing, PACC 2011, 2011.
[10]. I. Nonaka and H. Takeuchi, “Knowledge-Creating Company,” Knowledge-Creating Co., 1995.
[11]. P. Stevens and R. J. Pooley, Using UML: Software Engineering with Objects and Components. Addison-Wesley, 2006.
[12]. G. Borrego, A. L. Morán, R. R. Palacio, A. Vizcaíno, and F. O. García, “Towards a reduction in architectural knowledge vaporization during agile global software development,” Inf. Softw. Technol., 2019.
[13]. Z. Li, P. Liang, and P. Avgeriou, “Architectural Debt Management in Value-Oriented Architecting,” in Economics-Driven Software Architecture, 2014.
[14]. S. Fraser et al., “Technical Debt: From Source to Mitigation,” in Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, & Applications: Software for Humanity, 2013, pp. 67–70.
[15]. Y. Guo and C. Seaman, “A Portfolio Approach to Technical Debt Management,” in Proceedings of the 2nd Workshop on Managing Technical Debt, 2011, pp. 31–34.
[16]. R. Hoda, J. Noble, and S. Marshall, “How much is just enough?,” 2010.
[17]. R. K. Jonkers and K. E. Shahroudi, “Reducing the Costs of Engineering Design Changes Through Adoption of a Decision Support and Knowledge Management System Early in the Design,” in 2019 IEEE International Systems Conference (SysCon), 2019, pp. 1–8.
[18]. L. S. D. Annunzio and T. Sy, “Challenges and Organizations : Top-Level and Mid-Level Managerss’ Perspectives,” in Human Resource Planning Society, 2006.
[19]. S. Ryan and R. V. O’Connor, “Acquiring and Sharing tacit knowledge in software development teams: An empirical study,” Inf. Softw. Technol., 2013.
[20]. G. Bortis, “Informal Software Design Knowledge Reuse,” in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, 2010, pp. 385–388.
[21]. J. Burge and D. C. Brown, “Reasoning with Design Rationale,” in Artificial Intelligence in Design ’00, 2000.
[22]. B. Vasilescu, V. Filkov, and A. Serebrenik, “StackOverflow and GitHub: Associations between software development and crowdsourced knowledge,” in Proceedings - SocialCom/PASSAT/BigData/EconCom/BioMedCom 2013, 2013.
[23]. J. Tantisuwankul et al., “A topological analysis of communication channels for knowledge sharing in contemporary GitHub projects,” J. Syst. Softw., 2019.
[24]. B. Selic, “Agile documentation, anyone?,” IEEE Softw., 2009.
[25]. M. Sako, “From remote work to working from anywhere,” Communications of the ACM. 2021.
[26]. Jose L. Robles; Gilberto Borrego; Ramon Palacio;, “Gestión del conocimiento arquitectónico posterior a COVID-19 en pequeñas entidades de desarrollo de software: un estudio cualitativo,” Abstr. Appl. 36 63 – 77, vol. 36, no. ISSN2007-2635, pp. 63–77, 2022.
[27]. A. Ozimek, “Remote Workers on the Move,” SSRN Electron. J., 2021.
[28]. A. Cetrulo, “Is remote working here to stay? Lessons and ideas for a post-pandemic future,” Sinappsi, 2021.

[29]. J. R. Martínez-García, F. E. Castillo-Barrera, R. R. Palacio, G. Borrego, and J. C. Cuevas-Tello, "Ontology for knowledge condensation to support expertise location in the code phase during software development process," IET Softw., 2020.
[30]. K. Dalkir, Knowledge management in theory and practice. 2013.
[31]. M. P. Robillard, "Turnover-Induced Knowledge Loss in Practice," in Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2021, pp. 1292–1302.
[32]. M. Bhat, K. Shumaiev, and F. Matthes, "Towards a Framework for Managing Architectural Design Decisions," in Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings, 2017, pp. 48–51.
[33]. C. Manteuffel, D. Tofan, P. Avgeriou, H. Koziolek, and T. Goldschmidt, "Decision architect - A decision documentation tool for industry," in Journal of Systems and Software, 2016.
[34]. E. Hadar and G. M. Silberman, "Agile Architecture Methodology: Long Term Strategy Interleaved with Short Term Tactics," in Companion to the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications, 2008, pp. 641–652.
[35]. M. Che, "An Approach to Documenting and Evolving Architectural Design Decisions," in Proceedings of the 2013 International Conference on Software Engineering, 2013, pp. 1373–1376.
[36]. R. Francois, M. Nada, and A. Hassan, "How to Extract Knowledge from Professional E-Mails," in 2015 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), 2015, pp. 687–692.
[37]. D. Tofan, M. Galster, and P. Avgeriou, "Reducing architectural knowledge vaporization by applying the repertory grid technique," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2011.
[38]. D. Tofan, M. Galster, and P. Avgeriou, "Capturing tacit architectural knowledge using the repertory grid technique (NIER track)," in Proceedings - International Conference on Software Engineering, 2011.
[39]. J. A. Diaz-Pace and A. J. Bianchi, "High-Level Design Stories in Architecture-Centric Agile Development," in 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), 2019, pp. 137–144.
[40]. G. Borrego, G. Salazar-Lugo, M. Parra, and R. Palacio, "Slack's knowledge classification mechanism for architectural knowledge condensation," in Proceedings - 6th Annual Conference on Computational Science and Computational Intelligence, CSCI 2019, 2019.
[41]. Q. Jiang, C. H. Tan, C. L. Sia, and K. K. Wei, "Followership in an open-source software project and its significance in code ReUse," MIS Q. Manag. Inf. Syst., 2019.
[42]. A. P. Koenzen, N. A. Ernst, and M. A. D. Storey, "Code Duplication and Reuse in Jupyter Notebooks," in Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC, 2020.
[43]. C. Manteuffel, P. Avgeriou, and R. Hamberg, "An exploratory case study on reusing architecture decisions in software-intensive system projects," J. Syst. Softw., 2018.
[44]. R. P. Tippannavar, V. N. Kulkarni, and V. N. Gaitonde, "Productivity Improvement at Actuator Assembly Section Using Manual and Video Work Study Techniques," in Lecture Notes in Mechanical Engineering, 2020.
[45]. D. Vedder-Weiss, A. Segal, and A. Lefstein, "Teacher Face-Work in Discussions of Video-Recorded Classroom Practice: Constraining or Catalyzing Opportunities to Learn?," J. Teach. Educ., 2019.
[46]. J. Spatero, "2 Years of Digital Transformation in 2 Months," Microsoft 365, 2020.
[47]. C. W. Shiang, F. S. Tee, A. A. Halin, N. K. Yap, and P. C. Hong, "Ontology reuse for multiagent system development through pattern classification," Softw. - Pract. Exp., 2018.
[48]. T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," Int. J. Hum. - Comput. Stud., 1995.

[49]. S. Vasanthapriyan, J. Tian, D. Zhao, S. Xiong, and J. Xiang, “An ontology-based knowledge management system for software testing,” in Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, 2017.

[50]. S. Vasanthapriyan, J. Tian, and J. Xiang, “A Survey on Knowledge Management in Software Engineering,” Proc. - 2015 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS-C 2015, pp. 237–244, 2015.

[51]. N. Bin Ali, “Is effectiveness sufficient to choose an intervention?: Considering resource use in empirical software engineering,” in International Symposium on Empirical Software Engineering and Measurement, 2016.

[52]. S. Farshidi, S. Jansen, R. de Jong, and S. Brinkkemper, “A decision support system for software technology selection,” J. Decis. Syst., 2018.

[53]. M. Fernandez, A. Gómez-Pérez, and N. Juristo, “Methontology: from ontological art towards ontological engineering,” in Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, 1997.

[54]. J. A. Khan and S. Kumar, “OWL, RDF, RDFS inference derivation using Jena semantic framework & pellet reasoner,” in 2014 International Conference on Advances in Engineering and Technology Research, ICAETR 2014, 2014.

Информация об авторах / Information about authors

Хосе Луис РОБЛЕС – магистр инженерных наук Научно-исследовательского центра высшего образования в Энсенаде. С 2020 года аспирант Технологического института Соноры по отделению компьютерных наук. Сфера научных интересов: программная инженерия, проектирование и архитектура, управление знаниями, живая разработка, веб-разработка, приложения онтологий в программной инженерии.

José Luis ROBLES – Master in Engineering Sciences by the Centro de Investigación Científica y de Educación Superior de Ensenada. Currently, he is a PhD student in the Department of Computer Science and Design at the Instituto Tecnológico de Sonora since 2020. His research interests include software engineering, design and architecture, knowledge management, agile development, web development, and applications of Ontologies in Software Engineering.

Хильберто БОРРЕГО – доктор наук, с 2019 года – профессор Технологического института Соноры. Сфера научных интересов: процессы программной инженерии, парадигма “живой” разработки, управление знаниями, архитектура программного обеспечения.

Gilberto BORREGO - Doctor in Sciences, Full-time Professor at the Technological Institute of Sonora, since 2019. He works on improvements in the software engineering processes particularly on the agile paradigm. Research interests: Agile software engineering, knowledge management, software architecture.

Рамон Рене ПАЛАСИО – доктор наук, с 2004 года – профессор Технологического института Соноры. Исследует процессы текущей производственной практики, проектирует, разрабатывает, оценивает технологии, пригодные для различных видов операционного окружения. Сфера научных интересов: программная инженерия, взаимодействие человек-машина, интернет вещей.

Ramón René PALACIO – Doctor in Sciences, Full-time Professor at the Technological Institute of Sonora, since 2004. He conducts studies to gain a better understanding of current work practices and, based on this understanding, designs, develops, and evaluates technologies appropriate for various work environments. Research interests: Software engineering, Human-computer interaction, and Internet of Things.

Франсиско КАСТИЛЬО-БОРРЕРА - доктор наук в области информационных технологий, с 2002 года профессор инженерной школы факультета компьютерных наук Автономного университета Сан-Луи́с-Потоси́. Сотрудник факультета компьютерных наук

Государственного университета. Сфера научных интересов: приложения онтологий в программной инженерии, интеллектуальные системы, основанные на логическом выводе.

Francisco-Edgar Castillo BARRERA - Doctor in Information Technologies, Full-time Professor at the Autonomous University of San Luis Potosí, School of Engineering, Department of Computer Science since 2002. He is a specialist in the Department of Computer Science of State University. His research interests include Ontology applications in Software Engineering and Intelligent Systems based on Logic.

DOI: 10.15514/ISPRAS-2023-35(6)-5

Язык программирования для обучения технологиям компиляции и трансформации

А.Е. Недоря, ORCID: 0000-0001-8998-7072 <aleksei.nedoria@yandex.ru>

Аннотация. Обучение студентов технологиям компиляции и трансформации является актуальным. В статье рассматриваются критерии выбора языка программирования для выполнения практических работ в качестве целевого, дается краткое описание языка Тривиль и рассматривается пригодность этого языка для использования в соответствии с критериями.

Ключевые слова: языки программирования; обучение технологии компиляции; разработка языков программирования; компиляторы; технологии компиляции.

Для цитирования: Недоря А.Е. Язык программирования для обучения технологиям компиляции и трансформации. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 95–102. DOI: 10.15514/ISPRAS–2023–35(6)–5.

Programming Language for Teaching Compilation and Transformation Technologies

A.E. Nedoria, ORCID: 0000-0001-8998-7072 <aleksei.nedoria@yandex.ru>

Abstract. Teaching students compilation and transformation technologies is in demand. The article discusses the criteria for choosing a programming language for practical work as a target language, gives a brief description of the Trivil language and examines the suitability of this language for using in accordance with these criteria.

Keywords: programming languages, language design; compilers; compilation technologies.

For citation: Nedoria A.E. Programming language for teaching compilation and transformation technologies. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 95-102 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-5.

1. Введение

Разработка языков программирования, компиляторов и инструментов разработки, не теряет своей актуальности, а скорее даже становится более актуальной, что обусловлено как минимум двумя факторами

1. лицензионные проблемы, связанные с глобальным политическим противостоянием;
2. и появление новых идей и новых требований, воплощенных, например, в языках Rust [1], Go [2], Swift [3], Mojo [4].

Соответственно, является актуальным обучение студентов технологиям компиляции и трансформации программ и технологиям разработки или расширения языков программирования.

Обучение технологиям должно, наряду с теорией, включать практические работы, например:

- разработка компилятора, или, скорее, отдельных проходов компилятора, включая анализ, оптимизации, генерацию кода;
- тестирование компилятора, включая проверку соответствия компилятора языку (compliance test);
- расширение языка, добавление новых конструкций языка;
- разработка среды исполнения, включая управление памятью и сборку мусора;
- разработка библиотек;
- разработка инструментов, к примеру, статического анализатора;
- проведение доказательства свойств языка, например, type system soundness;
- построение модели памяти или параллельный вычислений;
- и так далее.

Для каждой из таких работ нужен язык программирования, который используется, как полигон для студенческих работ. То есть для этого языка пишется или дорабатывается компилятор, разрабатывается среда выполнения, проводятся доказательства и так далее. Будем называть такой язык **целевым** языком.

2. Сложности выбора целевого языка

Очевидно, что выбор такого языка существенно влияет на качество обучения. При выборе существенным фактором является ограниченность времени (семестр, год), за который студент должен показать работающий результат. Практическая работа должна быть, с одной стороны, достаточно серьезной, с другой стороны она существенно ограничена по времени выполнения.

Какой язык может быть выбран в качестве целевого?

По сути, есть три варианта:

- существующий язык программирования;
- подмножество существующего языка;
- учебный (игрушечный) язык.

Замечу, что в этой статье я, во многом, опираюсь на свой студенческий опыт. Моя дипломная работа в НГУ в 1984 году называлась «Компилятор с языка Эдисон для МВК Эльбрус». Компилятор был написан на языке Эль-76 [5] группой из трех студентов в течение года. В качестве целевого языка был выбран язык Эдисон (Edison) [6], автором которого был Пер Бринч Хансен. Эта дипломная работа во многом определила мой профессиональный путь, позволив мне накопить опыт как в проектировании и разработке достаточно сложной программы, так и в управлении командой разработчиков.

В то время я не задумывался, почему руководитель моей дипломной работы выбрал именно этот язык, но сейчас очевидны три существенных достоинства:

- простота языка
- хорошая (и короткая, около 35 страниц) спецификация языка
- современность языка: описание языка появилось в 1981 году

Вернусь к выбору целевого языка программирования. Среди широко используемых языков, ни один из языков нельзя назвать простым. И даже, если мы не говорим о C++, и судим только по размеру спецификации языка, то спецификация языка Kotlin - более 300 страниц, Java – 800 страниц, Go – 107 страниц. Можно сказать, что Go относительно простой язык, но простым его назвать нельзя.

Для работы с любым таким языком студенту понадобится существенное время на изучение самого языка. При этом даже если студент до этого писал на этом языке, то этих знаний недостаточно, любая практическая работа из тех, что перечислены выше, потребует более глубокого понимания языка, а работы, связанные с разработкой или доработкой компилятора (или других инструментов), еще и знаний устройства компилятора и инструментов.

Остальные два варианта — выделение подмножества существующего языка или создание учебного языка представляются возможными, но очень трудоемкими для преподавателя. Преподавателю, кроме разработки подмножества или языка, придется написать компилятор или обрезать компилятор (для подмножества) и выполнить другие предварительные работы.

В этой работе, я предлагаю язык, который можно отнести как к 1-му варианту, так и к 3-му – это использование языка Тривиль [7], который разрабатывался для решения других задач, но является (как и Эдисон в свое время) простым, современным и описанным в короткой спецификации.

Замечу, что я не предлагаю использовать Тривиль, как язык для обучения программирования или как **инструментальный** язык, на котором студенты решают свою практическую задачу. Использование целевого языка в качестве инструментального возможно, для некоторых задач это может быть полезно, но не для всех.

3. Критерии оценки целевого языка

Прежде чем переходить к краткому описанию языка Тривиль, выделим критерии применимости некоторого языка (или подмножества) для использования в качестве целевого, чтобы потом оценить язык Тривиль по этим критериям.

На мой взгляд, важнейшим критерием является обозримость и достаточная простота целевого языка, что позволяет студенту:

- видеть свою задачу целиком;
- сосредоточится на своей задаче (а не на сложностях самого язык).

Думаю, что это очевидно, если рассматривать, например, практическую задачу разработки части компилятора или доказательство type system soundness.

В то же время, язык не должен быть слишком простым, чтобы обучение было эффективным.

Другими важными критериями являются:

- наличие спецификации языка разумного размера;
- современность синтаксиса и семантики языка;
- наличие компилятора на популярных платформах;
- простота и обозримость компилятора и инструментов;
- открытый код и отсутствие лицензионных ограничений.

В табл. 1 перечислены критерии для целевого языка в таблице со степенью важности критерия (очень важно, важно, полезно, не важно), и для сравнения рассмотрим степень важности для инструментального языка.

4. Язык программирования Тривиль

Работа над языком Тривиль была начата в ноябре 2022 года с целью получить язык и инструментальные средства минимально достаточные для удобной разработки нескольких компиляторов разрабатываемого семейства языков. К лету 2023 года было сделано (все исходные тексты в открытом репозитории [8]):

- описание языка [7],
- два компилятора - первый на Go, второй на Тривиле

- и набор стандартных библиотек.

Оба компилятора построены по классической архитектуре:

- Парсер строит AST
- Далее по AST проходит семантический анализ (два прохода)
- По корректному атрибутированному AST делается генерация в C99

Табл. 1. Критерии целевого и инструментального языков
Table 1. Criteria for the applicability of target and instrumental languages

Критерий	**Для целевого языка**	**Для инструментального языка**
Обозримость и простота языка	Очень важно	Полезно
Описание (спецификация) языка разумного размера	Очень важно	Полезно
Современность синтаксиса и семантики языка	Важно	Полезно
Открытый код компилятора и экосистемы, отсутствие лицензионных ограничений	Очень важно	Полезно
Простота (обозримость) компилятора и инструментов	Очень важно (но не для всех практических работ)	Не важно
Доступность на популярных платформах, удобство использования	Важно	Очень важно

Размеры кода:

- компилятор на Go: 11,200 строк
- компилятор на Тривиле: 10,500 строк
- runtime на C99: 1,800 строк
- библиотеки на Тривиле: 2,800 строк

Первый доклад об языке был сделан на семинаре STEP-2023 [9] в апреле 2023. При подготовке к семинару пришло понимание, что язык может быть использован в качестве целевого для обучения студентов.

Приведем несколько примеров, которые должны быть понятны без пояснений или с минимальными пояснениями. Начнем, традиционно, с программы “Привет, мир” (см. Листинг 1).

```
модуль привет
импорт "стд::вывод"
вход{
   вывод.ф("Привет, мир\n")
}
```

Листинг 1. Привет, мир
Listing 1. Hello, world

Немного более сложный пример, показывающий описание функций показан на Листинге 2, результат исполнения примера: `5! = 120`.

```
модуль факториал
импорт "стд::вывод"
фн Факториал(n: Цел64): Цел64 {
    надо n > 1 иначе вернуть 1
    вернуть n * Факториал(n - 1)
```

```
}
вход{
    пусть № = 5
    вывод.ф("$;! = $;\n",  №, Факториал(№))
}
```

Листинг 2. Факториал
Listing 2. Factorial

В рамках статьи сделать подробное описание языка не представляется возможным (см. [7]), в табл. 2 перечислены его основные черты.

Табл. 2. Основные черты языка
Table 2. The main features of the language

Типы	**Описания**

• Байт, Цел64, Слово64
• Вещ64
• Лог
• Символ (unicode)
• Строка, Строка8
• тип вектора: []T
• тип класса: **класс** (база) {}
• может быть тип: **мб** T
	• **тип** T = тип
	• **конст** к (: T)? = знач
	  ○ **конст** к = 1
	  ○ **конст** к: Байт = 1
	• **пусть** п(: T)? (= | :=) знач
	  ○ **пусть** № = 1 // val
	  ○ **пусть** №: Байт := 1 // var
	• описания функций и методов

Описание функций
`фн Факториал(n: Цел64): Цел64 {`
`  // операторы`
`}`
	Описание методов
	`фн (сб: Сборщик) добавить строку(ст: Строка){`
	`  // операторы`
	`}`

Операторы	**Выражения**
• :=, ++, -- • **если** усл {} **иначе** {} • **надо** усл **иначе** (завер | {}) • **выбор** выражение {} • **выбор** тип перем = выр {} • **пока** усл {} • **цикл** перем **среди** век {} • **прервать** • **вернуть** знач? • **авария**(“описание”)	• +, -, *, /, % • =, #, <, <=, >, >= • логические: &, |, ~ (not) • битовые: :&, :|, :\ (xor), :~, <<, >> • преобразование типа: (: • подтверждение типа: ^ • конструктор вектора: T[1, 2, 3] • конструктор класса: T{имя: "Вася" }

При разработке языка не ставилась задача придумать новые конструкции языка, скорее, как и предполагает имя языка, ставилась задача использовать проверенные конструкции. Тем не менее, в языке есть интересные особенности, обеспечивающие удобство и безопасность программирования:

- Лаконичный синтаксис
- Обязательная инициализация полей и переменных
- Изменяемые и неизменяемые поля и переменные
- Поздняя инициализация для полей
- Простой вывод типов (type inference)
- Создание экземпляров классов и векторов с помощью конструкторов (Go composite literals)
- Отсутствие явных ссылок (ссылочные типы - класс, вектор, Строка)
- Безопасная работа со ссылками (null safety)
- Операторы “надо” и “выбор” по типу

- Безопасные вариативные и полиморфные параметры
- Обобщенные модули
- Русскоязычный синтаксис и ключевые слова, пробелы в идентификаторах

Часть из этих особенностей показана в следующих примерах.

Пример: реализации класса «Сборщик» (string builder), текст взят из библиотеки «стд::строки» (см. Листинг 3).

```
модуль строки

тип Байты = []Байт // тип: вектор байтов
тип Сборщик* = класс { // экспортированный класс Сборщик
    байты = Байты[]  // конструктор вектора длины 0
    число-символов := 0
}
фн (сб: Сборщик) добавить строку*(ст: Строка) {/*...*/ }
фн (сб: Сборщик) строка*(): Строка {
    вернуть сб.байты(:Строка) // преобразование к UTF-8 строке
}
```

Листинг 3. Класс «Сборщик»
Listing 3. Class "StringBuilder"

Второй пример из библиотеки «строки» показывает реализацию форматного вывода, класс «Разборщик» (см. Листинг 4).

```
модуль  строки
тип Символы = []Символ // вектор Unicode символов
тип Разборщик = класс {
    сб: Сборщик = позже // поле с поздней инициализацией
    формат: Символы = позже
}
// Добавляет строку по формату
фн (сб: Сборщик) ф*(фс: Строка, аргументы: …*) {
    // конструктор экземпляра класса с инициализацией полей:
    пусть р = Разборщик{сб: сб, формат: фс(:Символы)}
    пока р.следующий() {
        надо №-арг < длина(аргументы)
        иначе авария("не достаточно аргументов")
        // здесь обработка аргумента
    }
}
```

Листинг 4. Класс «Разборщик»
Listing 4. "Parser"

Следующий пример показывает работу с может-быть типами (безопасными ссылками, null safety) на примере бинарного дерева (см. Листинг 5). Ссылки на под-узлы бинарного дерева описаны с помощью *может быть* типа, которые добавляет значение «пусто» к домену значений типа Узел. Функция «число узлов» выдает число узлов бинарного дерева. Операция `"^"` является аналогом операции `"!"` (non-null assertion) в таких языках, как Kotlin [10] и Typescript.

Надеюсь, что примеры дает некоторое общее понимание языка, подробнее, см. [8].

```
тип Узел = класс {
    № := 0
    лев: мб Узел := пусто
```

```
    прав: мб Узел := пусто
}
фн число узлов(у: мб Узел): Цел64 {
    если у = пусто { вернуть 0 }
    вернуть 1 + число узлов(у^.лев) +  число узлов(у^.прав)
}
```

Листинг 5. Бинарное дерево
Listing 5. Binary Tree

5. Оценка использования в качестве целевого языка

Дадим оценку языку Тривиль как целевому языку в соответствии с критериями определенными выше (см. табл. 3).

Табл. 3. Применимость языка Тривиль в качестве целевого языка
Table 3. Trivil's applicability as a target language

Критерий	**Для целевого языка**	**Тривиль**
Обозримость и простота языка	Очень важно	Да
Описание (спецификация) языка разумного размера	Очень важно	Да размер описания языка — 40 страниц
Современность синтаксиса и семантики языка	Важно	Да
Открытый код компилятора и экосистемы, отсутствие лицензионных ограничений	Очень важно	Да
Простота (обозримость) компилятора и инструментов	Очень важно	Да
Доступность на популярных платформах, удобство использования	Важно	Платформы: Windows, Linux, FreeBSD, MacOS IDE: слабая интеграция

Единственным пунктом, в котором Тривиль не полностью удовлетворяет критерию, это «удобство использования». В настоящее время полноценная интеграция языка в IDE отсутствует. Впрочем, такая интеграция может быть темой практической работы студентов.

Было бы интересно сравнить Тривиль по этим критериям с другими языками, используемыми в качестве целевых, например, с языком РуСи [11], авторы которого ставят, в качестве одной из задач: «облегчить изучение программирования школьниками и студентами». К сожалению, такой информации у меня нет. Сравнение с общеизвестными языками не имеет смысла, если нет информации об их использовании в качестве целевых.

6. Заключение

За год с начала разработки языка Тривиль, язык прошел практическую проверку — на нем был написан компилятор и набор библиотек. По ходу разработки, в язык было внесено несколько существенных улучшений. Кроме того, язык два раза обсуждался на семинаре STEP-2023, обратная связь от участников семинара была очень полезна.

Начиная с сентября 2023 года, язык проходит практическую проверку в качестве целевого языка. В Университете Иннополиса несколько студенческих команд используют Тривиль и его компиляторы в практических работах по генерации кода для нескольких платформ и по созданию теста на соответствие компилятора и языка (language compliance test suit).

На взгляд автора, Тривиль может стать основой полигона для обучения студентов в разных вузах. Под полигоном я понимаю набор языков, компиляторов, других инструментов, формальных моделей и доказательств, которые используются студентами и куда выкладываются результаты практических работ.

Это позволит использовать результаты в следующих работах, наращивая разнообразие задач, в том числе это позволит вносить соревновательный элемент, например, лучшие реализация, инструмент или доказательство. Для этого нужны те, кто заинтересован в повышении качества обучения и место, где собирается информация о практических работах и их результатах.

Список литературы / References

[1]. Klabnik S., Nichols C.. The Rust Programming Language. Available at: https://doc.rust-lang.org/book/title-page.html, accessed 29.10.2023.
[2]. The Go Programming Language Specification. Available at: https://golang.org/ref/spec, Version of Aug 2, 2023.
[3]. The Swift Programming Language. Available at: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/aboutthelanguagereference/, accessed 29.10.2023.
[4]. Mojo Programming Manual. Available at: https://docs.modular.com/mojo/programming-manual.html, accessed 29.10.2023.
[5]. Пентковский В. М. Автокод Эльбрус. Принципы построения языка и руководство к пользованию. М., Наука, 1982, 352 с.
[6]. Brinch Hansen, Per. The Design of Edison. Software: Practice and Experience Vol. 11, No. 4, 1981, pp 363-396. DOI: 10.1002/SPE.4380110404.
[7]. Язык программирования Тривиль, версия языка 0.9.2. Available at: https://gitflic.ru/project/alekseinedoria/trivil-0/blob?file=doc%2Freport%2Freport.pdf, accessed 29.10.2023.
[8]. Репозиторий язык программирования Тривиль и компиляторов. Available at: https://gitflic.ru/project/alekseinedoria/trivil-0, accessed 29.10.2023.
[9]. Российский гибридный семинар STEP-2023 по фундаментальным вопросам программной инженерии, теории и экспериментальному программированию. Available at: https://persons.iis.nsk.su/en/STEP-2023, accessed 29.10.2023.
[10]. Null Safety. Available at: https://kotlinlang.org/docs/null-safety.html, accessed 29.10.2023.
[11]. Терехов А. Н,, Терехов М. А.. Проект РуСи для обучения и создания высоконадежных программных систем. Available at: https://cyberleninka.ru/article/n/proekt-rusi-dlya-obucheniya-i-sozdaniya-vysokonadezhnyh-programmnyh-sistem, accessed 29.10.2023.

Информация об авторах / Information about authors

Алексей Евгеньевич НЕДОРЯ, к.ф.-м.н. Сфера научных интересов: языки программирования, программные экосистемы, технологии программирования, мульти-платформенное программирование, архитектурной программирования, компонентно-ориентированное программирование.

Aleksei Evgenevitch NEDORIA – Cand. Sci. (Phys.-Math.). Research interests: programming languages, software ecosystems, programming technologies, multi-platform programming, architecture programming, component-oriented programming.

DOI: 10.15514/ISPRAS-2023-35(6)-6

Статический анализ на основе обобщённого абстрактного синтаксического дерева

[1,2] *В.О. Афанасьев*, ORCID: 0000-0002-8036-0633 <*vafanasiev@ispras.ru*>
[1] *А.Е. Бородин*, ORCID: 0000-0003-3183-9821 <*alexey.borodin@ispras.ru*>
[1,4] *К.И. Вихлянцев* <*vishkosty@ispras.ru*>
[1,3] *А.А. Белеванцев*, ORCID: 0000-0003-2817-0397 <*abel@ispras.ru*>

[1] *Институт системного программирования РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
[2] *Национальный Исследовательский Университет Высшая Школа Экономики,*
Россия, 101000, г. Москва, ул. Мясницкая, д. 20.
[3] *Московский государственный университет имени М.В. Ломоносова,*
Россия, 119991, Москва, Ленинские горы, д. 1.
[4] *Московский физико-технический институт,*
Россия, 141701, Московская область, г. Долгопрудный, Институтский переулок д.9.

Аннотация. В работе описывается универсальное представление для абстрактного синтаксического дерева (АСД), подходящее для статического анализа нескольких языков программирования. Предлагаемая схема анализа состоит в сохранении промежуточного представления в виде обобщенного АСД из компиляторов соответствующих языков и последующим анализом сохраненных деревьев. Мы реализовали такое представление для Java, Kotlin и Python. В анализаторе обобщенного АСД реализовано 27 детекторов. Мы описываем сущности предлагаемого представления, особенности его реализации для поддержанных языков. Приводим экспериментальные результаты скорости и качества анализа, а также сравнения анализа на обобщенном АСД с анализом, выполненным ранее на АСД конкретного компилятора. В итоге подход демонстрирует некоторое ухудшение скорости анализа, но позволяет разделить построение АСД для анализа и реализацию детекторов, что упрощает разработку АСД-детекторов в случае, когда количество поддерживаемых анализатором языков становится значительным.

Ключевые слова: статический анализ; поиск ошибок; АСД; синтаксический анализ; Java; Kotlin; Python; Svace.

Для цитирования: Афанасьев В.О., Бородин А.Е., Вихлянцев К.И., Белеванцев А.А. Статический анализ на основе обобщённого абстрактного синтаксического дерева. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 103–120. DOI: 10.15514/ISPRAS–2023–35(6)–6.

Static Analysis Based on the Unified Abstract Syntax Tree

[1,2] *V.O. Afanasyev*, ORCID: 0000-0002-8036-0633 <*vafanasiev@ispras.ru*>
[1] *A.E.Borodin*, ORCID: 0000-0003-3183-9821 <*alexey.borodin@ispras.ru*>
[1,4] *K.I.Vihliantsev* <*vishkosty@ispras.ru*>
[1,3] *A.A. Belevantsev*, ORCID: 0000-0003-2817-0397 <*abel@ispras.ru*>

[1] *Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] *National Research University Higher School of Economics,*
20 Myasnitskaya str., Moscow, 101000, Russia.
[3] *Lomonosov Moscow State University,*
Leninskie Gory, Moscow, 119991, Russia.
[4] *Moscow Institute of Physics and Technology,*
9 Institutskiy Pereulok, Dolgoprudny, Moscow Oblast, 141701, Russia.

Abstract. The paper describes a unified representation for an abstract syntax tree (AST) suitable for static analysis of several programming languages. The proposed analysis scheme consists of saving an intermediate representation in the form of a unified AST from compilers of the corresponding languages and subsequent analysis of the saved trees. We have implemented this described representation for Java, Kotlin and Python. The unified AST analyzer has 27 checkers. In the paper we present structure and entities of our unified AST, provide more details regarding language specifics that have to be reflected in the UAST representation. We give extensive experimental results that show UAST generation and analysis speed, analysis quality, and comparison with the old scheme of analyzing compiler ASTs where applicable. As a result, we see that we observe some degradation of analysis speed, but we pay it for the separation of AST construction and checkers' implementation. This separation allows easier support of many languages in the analyzer, where one can just generate UAST and support the required checker once within the UAST infrastructure instead of implementing a checker once per language.

Keywords: static analysis; search for defects; AST; parsing; Java; Kotlin; Python; Svace.

For citation: Afanasyev V.O., Borodin A.E., Vihliantsev K.I., Belevantsev A.A. Static analysis based on the unified abstract syntax tree. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 103-120 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-6.

1. Введение

В данной работе мы описываем универсальный статический анализатор, позволяющий единообразно описывать легковесные детекторы ошибок для множества языков программирования. Анализатор использует единое абстрактное синтаксическое дерево (АСД) для всех языков и в настоящее время поддерживает Java, Kotlin и Python. Реализация выполнена в рамках инструмента статического анализа Svace [1-3].

Анализ на основе АСД для каждого исходного файла получает на вход дерево, описывающее синтаксис анализируемого кода. Вершинами в этом дереве являются операторы языка, а листьями – соответствующие операнды.

Для анализа АСД в Svace использовался подход, при котором используется АСД компилятора соответствующего языка, и в рамках компилятора реализуются детекторы для поиска ошибок. Исключение – анализ АСД для языков C и C++, который выполняется на основе Clang Static Analyzer (CSA) [4]; с одной стороны, Clang Static Analyzer использует то же АСД, что и сам Clang, а с другой, помимо самого анализа АСД, в CSA можно создавать детекторы на основе статического символьного выполнения.

Основным недостатком схемы с реализацией детекторов внутри компиляторов является независимая кодовая база каждого детектора. Это, в свою очередь, приводит к необходимости реализовывать каждый детектор отдельно, невозможно создать общую

библиотеку для детекторов разных языков. Реализованные детекторы получаются неконсистентными и работают немного по-разному для разных языков. Как правило, детекторы в разных компиляторах реализуют разные люди с разным опытом. Это приводит к тому, что улучшение определенного детектора для одного языка не приводит к улучшению того же детектора для другого. Очевидным следствием является необходимость поддерживать больше кода и высокая стоимость разработки.

Общая идея анализатора на основе обобщённого АСД заключается в том, что в компиляторах остаётся только код для генерации АСД, которое имеет единый формат для всех языков. Все детекторы при этом реализованы в отдельном анализаторе.

Подобная схема работоспособна при определенных условиях:

- Типы узлов для абстрактных синтаксических деревьев поддерживаемых узлов должны пересекаться. В таком случае код обработки общих узлов разделяется между языками, а специфические для каждого языка узлы могут обрабатываться отдельно. При этом правомерно использовать разные компромиссы при создании анализатора: специфические узлы могут обрабатываться по-своему для полного учета особенностей языка, могут выражаться через общие для получения некоего «среднего» варианта обработки, а могут вообще пропускаться.
- Нужно учитывать потенциальное замедление анализа, т.к. компилятор должен сохранить АСД для последующего анализа.

В данной статье мы опишем, как был реализован наш анализатор обобщенного АСД, с какими проблемами мы столкнулись, сравним недостатки таких анализаторов с их преимуществами. Оценим целесообразность использования нашего подхода.

Для языков Java и Kotlin анализатор Svace уже имел реализации на основе АСД в компиляторах Javac и Kotlinc [5] соответственно. Мы сравним оба подхода.

Для языка Python у нас не было другого варианта анализа, и на примере этого языка мы оценим сложность добавления поддержки нового языка в анализатор на обобщённом абстрактном синтаксическом дереве (далее UAST).

2. Схема анализа с UAST

Общая схема UAST-анализа приведена на рис. 1 и содержит две фазы: трансляцию и анализ.

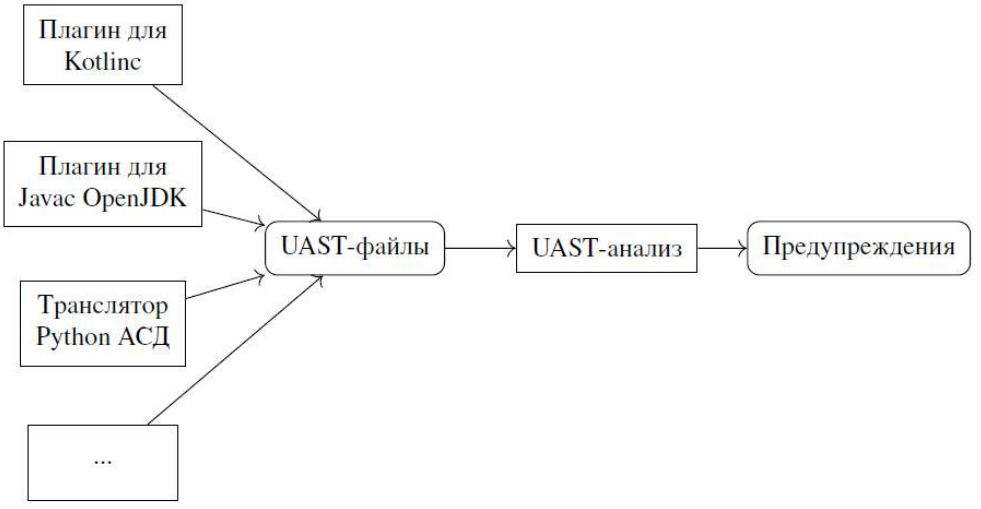


Рис. 1. Схема UAST-анализатора
Fig. 1. Scheme of the UAST analyser

Во время фазы трансляции для каждого языка запускаются трансляторы – программы, получающие на вход исходную программу и конвертирующие каждый файл с исходным кодом этой программы в файл в формате UAST. Помимо этого, трансляторы параллельно могут выполнять и другие действия, например, генерировать внутреннее представление для глубокого межпроцедурного анализа, выполняемого основным движком Svace.

Трансляторы для поддерживаемых нашим анализатором языков реализованы следующим образом:

- Трансляция AST языка Java осуществляется при помощи плагина компилятора Javac, использующего Compiler Tree API [6];
- Трансляция AST языка Kotlin осуществляется при помощи плагина компилятора Kotlinc, использующего PSI [7];
- Трансляция AST языка Python осуществляется при помощи скрипта на языке Python, использующего библиотеку ast [8].

Процесс анализа программы в Svace разделён на две фазы: контролируемую сборку [9] и анализ. На этапе сборки строится промежуточное представление программы, которое является входными данными для анализа. Такое разделение позволяет производить сборку и анализ на разных машинах, что удобно, т.к. к ним могут предъявляться разные требования. Для сборки требуется правильно настроить окружение. А для анализа может требоваться определённое количество оперативной памяти и желательно большее количество процессоров для его ускорения.

На фазе анализа происходит чтение промежуточных файлов в формате UAST, и для каждого такого файла происходит запуск ряда детекторов. То, какие детекторы будут запущены, зависит от языка оригинальной программы. В нашем анализаторе реализованы как детекторы, которые работают для всех языков (т.е. логика их работы не зависит от самого языка), так и специфичные для одного или нескольких поддерживаемых языков. Нами было реализовано 27 детекторов, реагирующих на следующие ситуации:

- Неявное создание кортежа из одного значения (Python);
- Плохой клон кода (Java, Kotlin, Python);
- Использование метода `wait` вне цикла, из-за чего программа уязвима к так называемому ложному пробуждению – spurious wakeup (Java, Kotlin);
- Ловимое исключение имеет тип `NullPointerException` (Java, Kotlin);
- Слишком общий тип ловимого исключения (Java, Kotlin);
- Отступ инструкции, следующей за условным оператором, создаёт впечатление вложенности, что вводит в заблуждение (Java, Kotlin);
- Пустой `catch`/`except`-блок (Java, Kotlin, Python);
- Пустой `synchronized`-блок (Java);
- Управление из ветви оператора `switch` может перейти к другой ветви (Java);
- Неправильные значения аргументов при создании диапазона (Kotlin);
- Результат операции не зависит от значений аргументов (Java, Kotlin, Python);
- Реализация метода `Iterator::hasNext` вызывает метод `Iterator::next` (Kotlin);
- Реализация метода `Iterator::next` не бросает исключение `NoSuchElementException` (Kotlin);
- Отсутствие `default`-ветви в операторах `switch` и `when` (Java, Kotlin);

- Использование изменяемого значения в качестве значения аргумента по умолчанию у функции (Python);
- Перекрытие имён (Java, Kotlin);
- Излишние или недостижимые проверки на тип в `when`-выражении (Kotlin);
- Использование оператора `return` в finally-блоке (Java, Kotlin, Python);
- Дублирующийся код в ветвлениях (Java, Kotlin, Python);
- Неправильный порядок аргументов при вызове функции (Java, Kotlin, Python);
- Неправильный порядок операций при использовании тернарного оператора (Java);
- Слишком общий тип бросаемого исключения (Java, Kotlin);
- Недостижимый `catch`-блок для `checked`-исключения (Java);
- Результат операции сравнения не используется (Kotlin);
- Использование неправильного значения в качестве аргумента оператора `synchrëonized` (Java);
- Сравнение по ссылке там, где ожидалось сравнение по значению (Java, Kotlin);
- Точка с запятой, стоящая после оператора `if`, что делает его бессмысленным (Java).

Все детекторы работают на уровне одного файла. То есть ошибки, для поиска которых требуется знание кода из разных файлов и/или модулей, в текущей реализации не обнаруживаются. Межмодульные ошибки также можно обнаруживать – для этого для каждого языка необходимо добавить этап связывания, который будет сопоставлять символы и типы из разных файлов и модулей. Данное ограничение было сделано намеренно, чтобы упростить реализацию.

В частности, подобный подход позволил реализовать кэш для анализа. В ходе такого анализа для каждого дерева строится хэш по содержимому. Этот хэш уникально описывает АСД. После анализа АСД результаты в виде предупреждений сохраняются в кэше, где хэш является ключом. Если при следующем анализе в кэше уже есть результаты для данного ключа, то анализ не запускается, а сразу возвращаются сохранённые результаты. Анализ с кэшем для случая, когда меняется небольшое количество файлов анализируемой программы, позволяет существенно ускорить результирующий анализ, т.к. реальный анализ будет выполняться только для изменившихся файлов.

3. Описание сущностей UAST-анализатора

UAST-анализ оперирует тремя основными сущностями: узлы АСД, символы и типы. Каждый узел:

- Хранит номер строки и столбца (позицию в коде) для первого и последнего символа, относящегося к узлу;
- Может иметь связанный тип (если это предусмотрено логикой данного узла). К примеру, с узлом идентификатора переменной будет связан тип данной переменной;
- Может иметь связанный символ (если это предусмотрено логикой данного узла). К примеру, узел идентификатора переменной будет иметь символ, указывающий на данную переменную;
- Может иметь дополнительные атрибуты, представленные другими узлами, типами, символами и прочими объектами (числами, строками, коллекциями и т.д.). К примеру, узел определения функции хранит в себе список узлов для всех инструкций данной функции, а узел константы хранит значение этой константы.

В нашем представлении используется 101 конкретный узел, из которых:

- 57 используются для двух или трёх языков;
- 9 используются только для Java;
- 13 используются только для Kotlin;
- 22 используются только для Python.

Помимо конкретных узлов, в нашем представлении имеется 17 общих узлов, от которых наследуются все прочие (к примеру, узел-«родитель» для всех выражений). Данные узлы позволяют детекторам абстрагироваться от специфики конкретного языка, реализуя только обработчики для общих надузлов.

Символы являются сущностями программы, которые можно отличить друг от друга по уникальному идентификатору (обычно имени[1]) и области видимости. Например:

- Локальные и глобальные переменные;
- Поля и свойства классов/структур/перечислений;
- Классы/структуры/перечисления;
- Параметры функций/методов/конструкторов;
- Функции/методы/конструкторы.

Под типами в UAST понимаются статические типы компилируемых языков – UAST-анализатор оперирует только теми типами, которые были известны в момент компиляции. Для интерпретируемых языков и языков с динамической типизацией информация о типах редко может быть определена статически, поэтому все типы считаются неизвестными, и информация о них никак не используется UAST-анализатором. Так происходит для языка Python (типы известны только для частных случаев вроде литералов, определений функций и определений пользовательских типов).

4. Особенности отдельных языков

4.1 Java

Как описывалось выше, транслятор для языка Java реализован в виде плагина компилятора Javac, использующего Compiler Tree API. Вся необходимая информация о сущностях предоставляется компилятором, и единственной задачей является лишь правильная трансляция этих сущностей в сущности UAST. В частности, несмотря на то, что UAST-анализ работает на уровне одного файла, информацию о типах и символах, определённых в других файлах, но используемых в текущих, удаётся сохранить в полном объёме.

Тем не менее, при реализации транслятора для языка Java мы столкнулись с рядом проблем.

Во-первых, некоторые сущности в АСД компилятора Java представлены одними и теми же узлами, из-за чего может возникать неоднозначность. К примеру, локальные переменные методов, поля классов и параметры функций представляются узлом `VariableTree`, а для определений классов, интерфейсов, аннотаций, перечислений и `record`-классов используется узел `ClassTree`. Подобного рода неоднозначности недопустимы в UAST – это сильно усложняет анализ, так как детекторам, реагирующим на конкретные типы узлов, приходится отличать подобные случаи самостоятельно. Поэтому такие сущности компилятора необходимо транслировать в разные сущности UAST, отличая их либо по наличию модификаторов (как в случае с `ClassTree`), либо по внешнему контексту (как в случае с `VariableTree`).

[1] Иногда уникальным идентификатором символа выступает некоторое целое число. Это важно для анонимных функций и классов, не имеющих имени, но для которых UAST-анализатор имеет символы и должен уметь их различать.

Второй проблемой при трансляции АСД языка Java является то, что АСД, предоставляемое посредством Compiler Tree API, не всегда соответствует исходному коду. В частности, мы столкнулись со следующими проблемами:

1. Для классов, в которых отсутствует конструктор, создаётся фиктивный конструктор без аргументов;
2. В классах, в которых отсутствует явный вызов `super-` или `this`-конструктора, добавляется неявный вызов `super`-конструктора без аргументов;
3. Элементы `record`-классов генерируются как поля внутри тела класса.

Решения этих проблем в нашем случае такие:

1. Фиктивные определения конструкторов отличаются от явных конструкторов по модификатору `GENERATEDCONSTR` у соответствующего символа компилятора. Такие определения не транслируются в обобщенное представление;
2. Вызовы методов, которые называются `this` или `super`, не имеют аргументов и имеют неизвестную позицию в исходном коде, считаются синтетическими и не транслируются в обобщенное представление;
3. `Record`-классы генерируются следующим образом. Ищется группа из последовательных полей класса, находящихся в самом верху соответствующего узла АСД, которые имеют моодификаторы `GENERATED_MEMBER`, `RECORD`, `PARAMETER` и не являются статическими. Эти поля считаются элементами `record`-класса, стоящими в его заголовке. Все прочие определения считаются определениями в теле `record`-класса.

Отметим также, что данный транслятор можно было бы реализовать не как плагин компилятора, а как надстройку над инструментом TreeSitter [10]. Мы не используем данный инструмент по ряду причин:

- TreeSitter предоставляет доступ только к АСД исходной программы. Информация о типах и символах данным инструментом не предоставляется, поэтому генерация данной информации должна быть возложена на транслятор. В нашем же подходе вся информация о типах и символах предоставляется самим компилятором и требует лишь трансляции в унифицированное представление;
- Svace во время фазы контролируемой сборки запускает компиляторы исходных языков, генерирующие промежуточное представление для основного анализа. Подключение плагина позволяет переиспользовать построенную компилятором информацию, требуемую UAST-анализатором, без каких-либо дополнительных расходов. При реализации транслятора на базе TreeSitter повторный парсинг и построение АСД занимали бы дополнительное время.

4.2 Kotlin

Как и транслятор для языка Java, транслятор для языка Kotlin реализован в качестве плагина к компилятору этого языка.

Одной из отличительных особенностей языка Kotlin является то, что почти все конструкции языка являются выражениями. Например, код приведённый в листинге 1, является полностью корректным. Операторы `continue` и `return` в языке Kotlin являются выражениями, благодаря чему они могут быть использованы как операнды в других операциях (в данном случае – при приведении типов и суммировании). Из-за подобных особенностей языка Kotlin, все конструкции, являющиеся выражениями, приходится считать выражениями и в UAST-анализаторе.

```
fun test() {
    for (i in (-5..5)) {
```

```
            println(i)
            val x = continue as Int +
                return as Int
            println(x)
        }
    }
```

Листинг 1. Использование continue и return в качестве выражений
Listing 1. Using continue and return as expressions

Язык Kotlin допускает перегрузку операторов. В частности, можно перегрузить операторы составного присваивания для встроенных типов, что допускает код из листинга 2. Это приводит к тому, что в UAST-анализаторе мы считаем, что слева от присваивания может стоять любое выражение даже в тех языках, которые такого не допускают[2].

```
fun test() {
    1 += 1
}
operator fun Int.plusAssign(value: Int) {
    println(this + value)
}
```

Листинг 2. Перегрузка оператора составного присваивания в Kotlin
Listing 2. Augmented assignment operator overloading in Kotlin

4.3 Python

Для понимания особенностей, с которыми пришлось столкнуться при реализации поддержки UAST для Python, необходимо рассмотреть, как исходный код программы на Python компилируется в байт-код. В качестве компилятора мы выбрали CPython – наиболее распространённую, эталонную реализацию языка программирования Python.

CPython компилирует исходный код в байт-код в несколько шагов [11]:

1. Разбиение исходного кода на токены;
2. Преобразование последовательности токенов в АСД;
3. Трансформация АСД в последовательность команд;
4. Построение графа потока управления и его оптимизация;
5. Генерация байт-кода на основе графа потока управления.

Построение таблицы символов компилятором CPython является промежуточным этапом при переходе со 2 на 3 шаг, к которому мы не можем получить доступа[3]. Поэтому UAST-представление строится на основе 2-го шага компиляции без сгенерированной компилятором CPython таблицы символов. Это приводит нас к необходимости создать и поддерживать собственную промежуточную таблицу символов при трансформации АСД в UAST-формат.

Для создания корректной промежуточной таблицы символов были учтены основные особенности языка Python [12]:

1. Области видимости переменных (блоки кода);

[2] Альтернативным решением этой проблемы может являться то, что в анализаторе будут иметься два узла для присваивания – общее, допускающее все выражения в левой части, и второе, допускающее в левой части только некоторые выражения. При этом второй узел будет наследником первого.

[3] Доступ к этой информации можно получить при наличии собственного модифицированного компилятора. Но поддержка своей версии компилятора – куда более трудоёмкая задача, чем представленная далее реализация.

2. Правила привязки переменных;
3. Специальные инструкции языка global [13] и nonlocal [14].

В языке Python существуют три области видимости: область видимости модуля (файла) или глобальная, область видимости класса и область видимости функции. Если объявить переменную в текущей области видимости, то она перекроет собой все переменные с таким же именем, которые расположены в областях видимости более высокого уровня.

Когда переменная используется в левой части выражения присваивания, она привязывается к текущей области видимости (см. листинг 3). В иных случаях сначала переменная будет искаться в текущей области видимости, а потом на верхних уровнях, заканчивая областью видимости модуля. Если ни в одной области видимости не будет найдено переменной с таким именем, то во время выполнения интерпретатором будет выброшено исключение.

```
a = 3
b = 7

def func():
    a = 13
    print(a) # 13
    b += 1 # NameError:
           #   name 'b' is not defined
```

Листинг 3. Правила привязки переменных
Listing 3. Name binding rules

Инструкции языка `global` и `nonlocal` позволяют изменить значение переменных, объявленных во внешних областях видимости. Инструкция `global` даёт указание интерпретатору искать переменную в области видимости модуля, в то время как инструкция `nonlocal` говорит о том, что переменная должна искаться во внешней области видимости.

```
 1| a = 3
 2|
 3| def func():
 4|     b = 7
 5|
 6|     def nested_func():
 7|         nonlocal b
 8|         b += 1
 9|
10|     nested_func()
11|
12|     global a
13|     a += b
14|
15| func()
16| print(a)  # 11
```

Листинг 4. Специальные инструкции языка global и nonlocal
Listing 4. Special global and nonlocal language statements

Инструкция nonlocal на 7 строчке листинга 4 даёт указание интерпретатору использовать в области видимости функции `nested_func` переменную с именем `b` из области видимости функции `func`, то есть переменную `b`, объявленную на 4 строчке. В то время как инструкция `global` указывает интерпретатору на необходимость искать переменную с именем `a` в

глобальной области видимости, поэтому в области видимости функции `func` будет использоваться переменная `a`, объявлена в самой первой строчке листинга.

Другой особенностью, с которой мы столкнулись, является то, что язык Python имеет динамическую типизацию переменных. Следовательно, в АСД отсутствует информация о типах. Для моделирования такой ситуации в UAST введён универсальный тип – UAnyType, который указывает на то, что тип значения может быть любым. Исключениями являются типы, которые связаны с объявлением функции или класса, а также константные выражения:

```
1| def func():
2|     ...
3|
4|
5| class Cls:
6|       ...
7|
8|
9| func = smth
```

Листинг 5. Типизация объявлений в UAST для Python
Listing 5. Typing of declarations in UAST for Python

То есть для примера из листинга 5 символ для идентификатора `func` на строке 1 будет иметь функциональный тип, а символ идентификатора `Cls` на строке 5 – тип класса. Однако при использовании переменной с идентичным именем в левой части присваивания на строке 9 транслятор создаст в динамической таблице символов новую переменную с именем `func` и типом `UAnyType`.

5. Результаты

5.1 Время анализа

При использовании UAST добавляется дополнительный этап в виде построения универсального промежуточного представления. Более того, в нашей реализации это представление сохраняется на диск. Поэтому можно ожидать замедление анализа.

В табл. 1 представлены данные времени сборки и анализа для проектов с открытым исходным кодом[4]. Для оценки Java анализа мы использовали исходный код ОС Android 11, для оценки Kotlin – исходный код компилятора Kotlinc версии 1.5.30, для Python – проект PyTorch версии 2.1.0.

Как можно заметить, замедление сборки на Java и Kotlin проектах очень незначительное – замедление для AOSP-11 составило около 3%, для Kotlinc-1.5.3 – около 2%. Сборка для PyTorch замедлилась примерно в три раза – скорее всего, это связано с тем, что транслятор для Python написан на языке Python.

[4] Сборка проектов проводилась на вычислительной машине с характеристиками: CPU: 16 cores 2.1GHz, RAM: 256Gb. Анализ проектов проводился на вычислительной машине с характеристиками: CPU: 8 cores 3.4GHz, RAM: 64Gb.

Табл.1. Результаты времени сборки и анализа
Table 1. Build and analysis time results

Проект	Язык	Размер, MLOC	Время сборки без UAST, мин	Время сборки с UAST, мин	Время анализа без UAST, мин	Время анализа с UAST, мин
ОС Android 11	Java	33	152	157	67	69
Kotlinc-1.5.30	Kotlin + Java	1.9 + 1.0	97	99	19	20
PyTorch-v2.1.0	Python	1	6	19	5	5

5.2 Сравнение результатов для Java и Kotlin

Мы вручную просмотрели результаты анализов для языков Java и Kotlin. Табл. 2 и 3 содержат сводные результаты нашей разметки. Для детекторов, которыми были выдано не более 50 предупреждений, были размечены все предупреждения. Для детекторов, которые выдали более 50 предупреждений, было размечено по 50 случайно выбранных предупреждений. В таблицах представлена информация только для детекторов, которыми было выдано хотя бы одно предупреждение. В среднем процент истинных предупреждений для Java и Kotlin составил более 90%.

Табл.2. Результаты разметки для Kotlinc-1.5.30
Table 2. Kotlinc-1.5.30 analysis results

Детектор	Количество истинных	Количество ложных	Процент истинных
Плохой клон кода	2	4	33%
Слишком общий тип ловимого исключения	50	0	100%
Пустой catch/except-блок	50	0	100%
Отсутствие default-ветви в операторах `switch` и `when`	42	0	100%
Управление из ветви оператора `switch` может перейти к другой ветви	0	1	0%
Реализация метода `Iterator::next` не бросает исключение `NoSuchElementException`	12	3	80%
Перекрытие имён	37	13	74%
Дублирующийся код в ветвлениях	19	0	100%
Неправильный порядок операций при использовании тернарного оператора	0	1	0%
Неправильный порядок аргументов при вызове функции	7	0	100%
Суммарно	**219**	**22**	**91%**

Практически все UAST-детекторы имеют качество не хуже, чем их неунифицированные версии. Из приведённых детекторов только два пострадали от унификации: детектор перепутанных местами аргументов и детектор плохих клонов кода.

Так как ранее в Svace использовался подход с анализом без унифицированного АСД, правильно было бы сравнить старую и новую реализации и определить, не вызвала ли унификация каких либо ухудшений. В табл. 4 и 5 представлены результаты сравнения детекторов для старой (без UAST) и новой (с UAST) реализаций. Сравнивались только

детекторы, которые выдали как минимум одно истинное предупреждение в старой реализации. В последнем столбце таблицы приведена доля истинных срабатываний старой реализации, которые были найдены и в новой.

Табл.3. Результаты разметки для ОС Android 11
Table 3. Android 11 analysis results

Детектор	Количество истинных	Количество ложных	Процент истинных
Плохой клон кода	40	10	80%
Использование метода `wait` вне цикла, из-за чего программа уязвима к т.н. spurious wakeup	40	5	89%
Слишком общий тип ловимого исключения	50	0	100%
Пустой catch/except-блок	50	0	100%
Ловимое исключение имеет тип `NullPointerException`	50	0	100%
Отступ инструкции, следующей за условным оператором, создаёт впечатление вложенности, что вводит в заблуждение	11	5	69%
Отсутствие default-ветви в операторах `switch` и `when`	50	0	100%
Пустой synchronized-блок	18	0	100%
Управление из ветви оператора switch может перейти к другой ветви	50	0	100%
Результат операции не зависит от значений аргументов	50	0	100%
Использование оператора return в finally-блоке	8	0	100%
Перекрытие имён	44	6	88%
Дублирующийся код в ветвлениях	49	1	98%
Неправильный порядок операций при использовании тернарного оператора	22	28	44%
Слишком общий тип бросаемого исключения	46	0	100%
Недостижимый catch-блок для checked-исключения	0	2	0%
Неправильный порядок аргументов при вызове функции	13	3	81%
Использование неправильного значения в качестве аргумента оператора synchronized	24	2	92%
Суммарно	**615**	**62**	**91%**

Качество первого детектора снизилось из-за того, что изначальная его версия, реализованная нами в компиляторе языка Java, работала на уровне нескольких файлов. Реализация же в UAST-анализаторе работает исключительно на уровне одного файла, соответственно, ошибочные случаи, в которых вызывается функция из другого файла, новой реализацией не обнаруживаются.

Качество детектора плохих клонов кода снизилось, так как изначально существовало две совершенно разных реализации этого детектора: одна для языка Java, другая – для Kotlin. В качестве эталонной реализации была выбрана версия для языка Java, и по большей части реализация в UAST-анализаторе основывается на ней. Соответственно, ухудшение данного детектора на Kotlin-проектах – ожидаемый эффект.

В листинге 6 приведён пример одного из срабатываний на ОС Android 11 (код отредактирован для компактности). В данном примере программист хотел проверить значение переменной `uri` на `null` и в зависимости от этого получить либо пустую строку, либо результат вызова метода `toString`. Но приоритет операций в языке Java отличается от того, что подразумевал автор – сначала будет выполнена операция конкатенации строк, затем сравнение со значением `null`, а только потом тернарный оператор. Таким образом, результат этой операции всегда будет равен `uri.toString()`.

Табл.4. Сравнение результатов с и без UAST-анализатора для ОС Android 11
Table 4. Comparison of analysis results with and without UAST analyser for Android 11

Детектор	Истинных без UAST	Истинных с UAST	Процент совпавших истинных
Плохой клон кода	40	40	100%
Использование метода `wait` вне цикла, из-за чего программа уязвима к т.н. spurious wakeup	40	40	100%
Слишком общий тип ловимого исключения	50	50	100%
Пустой catch/except-блок	50	50	100%
Ловимое исключение имеет тип `NullPointerException`	50	50	100%
Отступ инструкции, следующей за условным оператором, создаёт впечатление вложенности, что вводит в заблуждение	4	11	100%
Отсутствие default-ветви в операторах switch и when	50	50	100%
Пустой synchronized-блок	18	18	100%
Управление из ветви оператора switch может перейти к другой ветви	50	50	98% (49 из 50)
Результат операции не зависит от значений аргументов	49	50	100%
Использование оператора return в finally-блоке	8	8	100%
Дублирующийся код в ветвлениях	18	49	100%
Слишком общий тип бросаемого исключения	46	46	100%
Неправильный порядок аргументов при вызове функции	26	13	42% (11 из 26)
Использование неправильного значения в качестве аргумента оператора synchronized	23	24	96% (22 из 23)

Табл.5. Сравнение результатов с и без UAST-анализатора для Kotlinc-1.5.30
Table 5. Comparison of analysis results with and without UAST analyser for Kotlinc-1.5.30

Детектор	Истинных без UAST	Истинных с UAST	Процент совпавших истинных
Плохой клон кода	6	2	17% (1 из 6)
Слишком общий тип ловимого исключения	49	50	100%
Пустой catch/except-блок	50	50	100%
Отсутствие default-ветви в операторах switch и when	42	42	100%
Реализация метода `Iterator::next` не бросает исключение `NoSuchElementException`	12	12	100%

```
Log.w(TAG,
    "Exception for URI:" + uri == null ? "" : uri.toString(), e);
```

Листинг 6. Пример срабатывания анализатора на ОС Android 11
Listing 6. Example of emitted warning for Android 11

5.3 Анализ Python-проектов

Процесс добавления языка программирования Python в схему UAST-анализатора занял примерно 800 человеко-часов. Это время включает в себя написание модуля трансляции из Python АСД в UAST, дополнение общих для всех языков детекторов в соответствии с семантическими особенностями языка и добавление двух детекторов, работающих только для языка Python: детектора неявного создания кортежа из одного значения и детектора, находящего использования изменяемых значений в качестве значений аргумента по умолчанию. Суммарная кодовая база для поддержки языка Python составила примерно 4 тысячи строк кода, из которых 3 тысячи на Python и 1 тысяча на Java.

В табл. 6 приведены результаты разметки для проекта PyTorch. Общая доля истинных срабатываний составила 98%.

В листингах 7 и 8 представлены примеры срабатываний UAST-детекторов для Python (код отредактирован для компактности).

Табл.6. Результаты разметки для pytorch-v2.1.0
Table 6. Pytorch-v2.1.0 analysis results

Детектор	Количество истинных	Количество ложных	Процент истинных
Неявное создание кортежа из одного значения	50	0	100%
Плохой клон кода	14	4	78%
Пустой catch/except-блок	50	0	100%
Результат операции не зависит от значений аргументов	50	0	100%
Использование изменяемого значения в качестве значения аргумента по умолчанию у функции	4	0	100%
Использование оператора return в finally-блоке	2	0	100%
Дублирующийся код в ветвлениях	34	0	100%
Неправильный порядок аргументов при вызове функции	11	0	100%
Суммарно	**215**	**4**	**98%**

```
class Test:
  def __init__(self, lst=[1, 2, 3]):
    self.lst = lst


t1 = Test()
print(t1.lst)  # [1, 2, 3]
t1.lst.append(4)
print(t1.lst)  # [1, 2, 3, 4]

t2 = Test()
print(t2.lst)  # [1, 2, 3, 4]
```

Листинг 7. Пример срабатывания детектора, обнаруживающего изменяемые значения в качестве аргументов по умолчанию
Listing 7. Example of a warning emitted by the mutable default argument checker

```
primals = tree_map(
    lambda x: x if isinstance(x, torch.Tensor) else x, primals
)
```

Листинг 8. Пример срабатывания детектора дублирующихся ветвей
Listing 8. Example of a warning emitted by the similar branches checker

6. Похожие работы

Использование обобщенных абстрактных синтаксических деревьев является популярным подходом. Во многих случаях такой подход используется IDE для синтаксического анализа и рефакторинга. IDE X-Develop [15] имеет АСД для нескольких языков (C#, Java,Visual Basic, ASP, XML, JavaScript), которое может использоваться для рефакторинга программы.

Среда TexMo [16] имеет UAST, содержащий только общие лексические элементы, а отношения между строками описываются отдельными связями вида «ключ-значения». Среда не производит никакого существенного анализа, а лишь помогает отслеживать связи между элементами программы, в том числе на разных языках.

Компания JetBrains в своих IDE предоставляет фреймворк, называемый UAST [17]. Он является языконезависимой прослойкой над АСД конкретных языков, которая позволяет разрабатывать плагины, в том числе системы сборки, линтеры, статические анализаторы для языков, работающих поверх JVM. При помощи данного фреймворка упрощается реализация и поддержка данных инструментов, так как они реализуются лишь единожды сразу для нескольких языков. Тем не менее, интегрированный в среду разработки анализ имеет ограничения на сложность, так как должен выполняться в режиме реального времени и не мешать работе пользователя.

Инструмент DMS [18] предназначен для написания спецификаций и их реализаций на разных языках программирования, а также поддержания связей между ними и выполнения трансформаций кода. Программа DMS была применена к ряду коммерческих приложений на Java и C++, в которых выполнялся поиск клонов кода, упрощение директив препроцессора, генерация кода для микроконтроллеров.

Инструмент Bauhaus [19] для анализа и обратной инженерии программ представляет АСД поддерживаемых языков в виде графа, узлы которого формируют единую иерархию классов. Разработчики реализовали базовые консервативные анализы потока данных и управления поверх этого графа, включая анализ указателей, а также выполнили поиск взаимных блокировок и мертвого кода.

В [20] предлагается использовать представление на основе языка XML, содержащее основную информацию о классах программы и связях между ними. Представление генерируется парсерами соответствующих языков. В прототипе системы XML-представление генерируется для языков Java и Python и используется для анализаиерархии классов.

Другим подходом является построение UAST и предоставление декларативного языка для описания ошибочных ситуаций. В частности, в анализаторе Klocwork используется язык KAST [21, 22], расширяющий XQuery, для реализации АСД-детекторов.

Альтернативным решением является использование промежуточного представления, специализированного только для одного языка. Предыдущие версии анализатора Svace использовали такой подход. Хотя Svace имеет общее представление для анализируемых языков, это представление не включает в себя АСД. Поэтому каждый АСД-анализатор реализуется независимо для поддерживаемых языков программирования. В данной статье мы уже описали плюсы и минусы такого подхода.

Теоретически детекторы, реализованные на АСД от компилятора, имеют больше информации об исходной программе и могут работать быстрее. На практике при нехватке ресурсов на поддержку темп развития АСД-детекторов для разных языков был различным, некоторые редкие детекторы практически не развивались.

Инструмент Infer [23, 24] имеет промежуточное представление SIL, используемое при проведении анализа. Это представление не включает в себя синтаксис программы. Для реализации АСД-детекторов Infer используется декларативный язык для написания линтеров [25]. Этот язык, основанный на АСД компилятора Clang, ограничен языками, поддерживаемыми этим компилятором. Он недоступен для Java и C#, для которых возможен

анализ на основе SIL-а. Более того, он объявлен устаревшим (deprecated). Таким образом, отсутствие общего представления в данном случае помешало создать АСД-детекторы для Java и C#.

7. Заключение

Создание универсального статического анализатора оказывается возможным для довольно отличающихся языков программирования, таких как Kotlin и Python. Анализ с обобщенным представлением позволяет упростить разработку детекторов наоснове поиска шаблонов в АСД.

На наш взгляд, такой подход, имея незначительные ухудшения в скорости анализа, предпочтительнее, чем реализация отдельных анализаторов для каждого языка, т. к. позволяет разделить проблему построения АСД-представления и реализацию детекторов. Благодаря этому сложность реализации пропорциональна $N + M$, где N – количество поддерживаемых языков программирования, а M – количество детекторов. Для независимой реализации эта сложность пропорциональна их произведению $N \times M$, что становится существенным по мере роста количества поддерживаемых языков.

При наличии АСД-представления написание детекторов является относительно несложной проблемой. Тем не менее, на примере Infer видно, что эта работа не производится. Таким образом, использование общего представления является существенным для анализатора, поддерживающего несколько языков программирования.

Список литературы / References

[1]. В. П. Иванников, А. А. Белеванцев, А. Е. Бородин, В. Н. Игнатьев, Д. М. Журихин, А. И. Аветисян и М. И. Леонов. Статический анализатор Svace для поиска дефектов в исходном коде программ. Труды ИСП РАН , 26(1):231—250, 2014. DOI: 10.15514/ISPRAS-2014-26(1)-7.

[2]. A. Belevantsev, A. Borodin, I. Dudina, V. Ignatiev, A. Izbyshev, S. Polyakov, E. Velesevich and D. Zhurikhin. Design and development of Svace static analyzers. In 2018 Ivannikov Memorial Workshop (IVMEM), pp.3—9. IEEE, 2018.

[3]. A.E. Borodin and A. A. Belevantsev. A static analysis tool Svace as a collection of analyzers with various complexity levels. Proceedings of the Institute for System Programming of the RAS, 27(6):111—134, 2015.

[4]. Clang static analyzer. URL: https://clang-analyzer.llvm.org (дата обращения 02.10.2023).

[5]. V. Afanasyev, S. Polyakov, A. Borodin and A. Belevantsev. Kotlin from the point of view of static analysis developer. Programming and Computer Software, 49(7):549—558, 2023. DOI: 10.1134/S0361768823070022.

[6]. Compiler tree api. Английский. URL: https://docs.oracle.com/javase/8/docs/jdk/api/javac/tree/index.html (дата обращения 18.09.2023).

[7]. Intellij platform plugin sdk. Psi elements. Английский. URL: https://plugins.jetbrains.com/docs/intellij/psi-elements.html (дата обращения 18.09.2023).

[8]. Ast — abstract syntax trees. Английский. URL: https://docs.python.org/3/library/ast.html (дата обращения 18.09.2023).

[9]. A. Belevantsev, A. Izbyshev and D. Zhurikhin. Monitoring program builds for svace static analyzer. System administrator, (7-8):135—139, 2017.

[10]. Treesitter. Introduction. Английский. URL: https://tree-sitter.github.io/tree-sitter/ (дата обращения 18.12.2023).

[11]. Python developer’s guide. Compiler design. Английский. URL: https://devguide.python.org/internals/compiler/ (дата обращения 28.09.2023).

[12]. Execution model. Английский. URL: https://docs.python.org/3.8/reference/executionmodel.html#execution-model (дата обращения 13.10.2023).

[13]. The global statement. Английский. URL: https://docs.python.org/3.8/reference/simple_stmts.html#the-global-statement (дата обращения 27.10.2023).

[14]. The nonlocal statement. Английский. URL: https://docs.python.org/3.8/reference/simple_stmts.html#the-nonlocal-statement (дата обращения 27.10.2023).
[15]. D. Strein, H. Kratz и W. Lowe. Cross-language program analysis and refactoring. In 2006 Sixth IEEE International Workshop on Source Code Analysis and Manipulation, pp 207—216. IEEE, 2006.
[16]. R.-H. Pfeiffer и A. Wasowski. Texmo: a multi-language development environment. In European Conference on Modelling Foundations and Applications, pp 178—193. Springer, 2012.
[17]. Uast – unified abstract syntax tree. Intellij platform plugin sdk. Английский. URL: https://plugins.jetbrains.com/docs/intellij/uast.html (дата обращения 03.10.2023).
[18]. I. D. Baxter. Dms: program transformations for practical scalable software evolution. In Proceedings of the International Workshop on Principles of Software Evolution, pp 48—51, 2002.
[19]. A. Raza, G. Vogel и E. Plodereder. Bauhaus–a tool suite for program analysis and reverse engineering. In Reliable Software Technologies–Ada-Europe 2006: 11th Ada-Europe International Conference on Reliable Software Technologies, Porto, Portugal, June 5-9, 2006. Proceedings 11, pp 71—82. Springer, 2006.
[20]. М. В. Зубов, А. Н. Пустыгин и Е. В. Старцев. Применение универсальных промежуточных представлений для статического анализа исходного программного кода. Доклады Томского государственного университета систем управления и радиоэлектроники , (1 (27)):64—68, 2013.
[21]. С. В. Сыромятников. Декларативный интерфейс поиска дефектов по синтаксическим деревьям: язык kast. Труды Института системного программирования РАН , 20:51—68, 2011.
[22]. Н. Л. Луговской и С. В. Сыромятников. Применение языка kast для преобразования исходного кода и автоматического исправления дефектов. Труды Института системного программирования РАН , 25:51—66, 2013.
[23]. D. Harmim, V. Marcin и O. Pavela. Scalable static analysis using Facebook Infer. I, VI-B, 2019.
[24]. D. Harmim. Advanced static analysis of atomicity in concurrent programs through Facebook Infer. Proceedings of the Excel@ FIT'21.
[25]. Ast language (al). Английский. URL: https://fbinfer.com/docs/checker-linters/ (дата обращения 02.10.2023.

Информация об авторах / Information about authors

Виталий Олегович АФАНАСЬЕВ – магистрант факультета компьютерных наук НИУ ВШЭ, сотрудник ИСП РАН. Сфера научных интересов: компиляторные технологии, статический анализ, JVM языки.

Vitaly Olegovich AFANASYEV – master's student at the Faculty of Computer Science, NRU HSE, employee of ISP RAS. Research interests: compiler technologies, static analysis, JVM languages.

Алексей Евгеньевич БОРОДИН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН. Сфера научных интересов: статический анализ исходного кода программ для поиска ошибок.

Alexey Evgenevich BORODIN – Cand. Sci. (Phys.-Math.) - researcher. Research interests: static analysis for finding errors in source code.

Константин Игоревич ВИХЛЯНЦЕВ – студент факультета ФРКТ Физтеха, сотрудник ИСП РАН. Сфера научных интересов: компиляторные технологии, статический анализ, язык Python.

Konstantin Igorevich VIHLIANCEV – PSRECT MPTI student, ISP RAS researcher. Research interests: compiler technologies, static analysis, Python.

Андрей Андреевич БЕЛЕВАНЦЕВ – доктор физико-математических наук, ведущий научный сотрудник ИСП РАН, профессор МГУ. Сфера научных интересов: статический анализ программ, оптимизация программ, параллельное программирование.

Andrey Andreevich BELEVANTSEV – Dr. Sci (Phys.-Math.), Leading Researcher at ISP RAS, Professor at MSU. Research interests: static analysis, program optimization, parallel programming.

DOI: 10.15514/ISPRAS-2023-35(6)-7

Обнаружение возможной перезаписи переменных вследствие использования функций нелокальных переходов

[1, 2] *Н.Ю. Шугалей,* ORCID: 0009-0000-9310-8317, <*shugaley@ispras.ru*>
[1] *В.А. Иванишин,* ORCID: 0000-0002-9784-2911, <*vlad@ispras.ru*>
[1] *А.В. Монаков,* ORCID: 0000-0001-5118-0054, <*amonakov@ispras.ru*>

[1] *Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
[2] *Московский физико-технический институт (национальный исследовательский университет), 117303, г. Москва, ул. Керченская, д.1 А, корп. 1.*

Аннотация. Причиной возникновения неопределенного поведения является исходный код, написанный с нарушением стандарта языка Си. Неопределенное поведение приводит к появлению уязвимостей в программном обеспечении. Одним из распространенных источников неопределенного поведения является некорректное использование функций нелокальных переходов (в частности setjmp и longjmp). В данной работе рассмотрены средства обнаружения такого типа неопределенного поведения, реализованные в основных современных компиляторах (GCC, Clang, MSVC). Сделаны выводы о том, что эти средства обладают существенными недостатками либо вовсе отсутствуют в отдельных компиляторах. Описана реализация нового метода компиляторной диагностики рассматриваемого неопределенного поведения. Приведенный в работе метод обладает точностью, достаточной для практического применения на реальных проектах. Рассмотрены преимущества представленного решения над похожими существующими.

Ключевые слова: компиляторы; си; нелокальные переходы; setjmp; volatile; неопределенное поведение; статический анализ.

Для цитирования: Шугалей Н.Ю., Иванишин В.А., Монаков А.В. Обнаружение возможной перезаписи переменных вследствие использования функций нелокальных переходов. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 121-134. DOI: 10.15514/ISPRAS–2023–35(6)-7.

Detecting Potentially Clobbered Variables due to the Use of Nonlocal Jumps Functions

[1, 2] *N.U. Shugaley,* ORCID: 0009-0000-9310-8317, <*shugaley@ispras.ru*>
[1] *V.A. Ivanishin,* ORCID: 0000-0002-9784-2911, <*vlad@ispras.ru*>
[1] *A.V. Monakov,* ORCID: 0000-0001-5118-0054, <*amonakov@ispras.ru*>

[1] *Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] *Moscow Institute of Physics and Technology (National Research University),*
1 A, Kerchenskaya st., Moscow, 117303, Russia.

Abstract. The reason of undefined behavior is source code written in violation of the C language standard. Undefined behavior leads to vulnerabilities in software. One of the common sources of undefined behavior is an incorrect use of functions for nonlocal jumps (in particular setjmp and longjmp). This paper considers the

means of detecting this type of undefined behavior which are implemented in the major modern compilers (GCC, Clang, MSVC). We conclude that these means either have significant disadvantages or are absent in some compilers. This paper presents the implementation of a new method of compiler warning of the considered undefined behavior. The described method is accurate enough for practical application on real projects. We consider the advantages of the proposed solution over similar existing ones.

Keywords: compilers; c; nonlocal jumps; setjmp; volatile; undefined behavior; static analysis.

For citation: Shugaley N.U., Ivanishin V.A., Monakov A.V. Detecting potentially clobbered variables due to the use of nonlocal jumps functions. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 6, 2023. pp. 121-134 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-7.

1. Введение

Механизм работы функций семейства `setjmp`

Функция `setjmp` обладает свойством двойного возврата. При первом непосредственном вызове `setjmp` сохраняет информацию о текущем состоянии вызова (значения регистров) в специальный буфер с типом `jmp_buf` (в дальнейшем сам буфер будем также обозначать как `jmp_buf`). Таким образом, `setjmp` динамически устанавливает точку, в которую позже функция `longjmp` передает управление. После прямого вызова `setjmp` происходит нормальный возврат из него с нулевым возвращаемым значением. После этого `jmp_buf` передается в `longjmp` для осуществления нелокального перехода и восстановления исходного состояния. Происходит возврат не из `longjmp`, а второй возврат из `setjmp` – анормальный (англ. abnormal) возврат с ненулевым возвращаемым значением.

Пара функций `sigsetjmp` / `siglongjmp` аналогична по поведению паре `setjmp` / `longjmp` за исключением того, что предоставляет предсказуемую обработку маски сигналов процесса. Реализовано это с помощью отдельного передаваемого аргумента, который явно отвечает за сохранение маски сигналов. Для дальнейшего повествования данное различие не имеет значения, поэтому при упоминании пары `setjmp` / `longjmp` подразумевается также и пара с префиксом `sig`.

Неопределенное поведение вследствие перетирания переменных

Компилятор может оптимизировать доступ к переменным, размещая их на регистры. А `longjmp` восстанавливает значения регистров вместе с указателем на стек и счетчиком команд. Следовательно, при выполнении некоторых условий значение переменной с автоматическим классом памяти после `longjmp` будет иметь неопределенное значение. Это значение зависит от размещения компилятором переменной в ходе оптимизаций. Условия возникновения неопределенного значения после вызова `longjmp` [1]:

- Переменная локальна для функции, внутри которой происходит непосредственный вызов `setjmp`;
- Ее значение изменяется между вызовами `setjmp` и `longjmp`;
- Она не объявлена как `volatile`.

Если соблюдены эти условия и неопределенное значение переменной читается после вызова `longjmp`, то мы имеем неопределенное поведение.

Наиболее простой способ устранить данное неопределенное поведение без изменения общей логики программы – объявить такую переменную как `volatile`. Дело в том, что компилятор гарантирует не оптимизировать доступ к `volatile` переменной, то есть каждое чтение и запись такой переменной будут выполняться непосредственно из памяти, а не из регистра процессора. Таким образом, перетирание регистров перестает влиять на значение переменной. Значение переменной будет сохраняться после вызова `longjmp`.

2. Постановка задачи

Актуальность работы

Функции `setjmp` / `longjmp` могут использоваться для:

- Обработки ошибок в глубоко вложенных функциях – аналог исключений из C++. Используется в исходном коде на языке Си, когда C++ не применим: встроенные системы с жесткими ограничениями по памяти [2], старый программный код. Является наиболее распространенным механизмом применения.
- Передачи управления от обработчиков сигналов в определенную точку программы. Не рекомендуется, поскольку может привести к неопределенному поведению в результате использования non-async-signal-safe функций [3].
- Реализация сопрограмм [4].

В дистрибутиве Alpine Linux 3.16 данные функции встречаются в исходном коде около 800 пакетов программ из репозиториев main и community. Всего эти два репозитория содержат 6250 пакетов.

Постановка задачи

Целью данной работы является разработка метода обнаружения случаев неопределенного поведения в языке Си при использовании функций нелокальных переходов (англ. nonlocal jumps или nonlocal gotos), в частности `setjmp` и `longjmp`. Рассматривается только описанное выше неопределенное поведение, связанное с перезаписью переменных.

Данная работа нацелена на реализацию предупреждения в компиляторе, а не в полноценном статическом анализаторе. Это накладывает некоторые ограничения на возможности диагностики. Если функции определены в разных единицах трансляции, то в общем случае компилятор не может проанализировать определение вызываемой функции в точке ее вызова. Искомый метод диагностики должен быть полезным, даже когда применение межпроцедурного анализа невозможно.

Диагностика должна быть применима для реальных программ, написанных с использованием функций нелокальных переходов. Общее число срабатываний должно быть реальным для ручного рассмотрения и исправления, а количество ложно-положительных срабатываний должно быть минимизировано. Для этого на основе анализа реального программного кода должны быть выделены основные шаблоны использования `setjmp`, а на их основе сформулированы и реализованы необходимые эвристики.

Данная задача включает в себя следующие подзадачи:

- Анализ существующих решений и выявление их недостатков.
- Разработка более точного алгоритма и эвристик, повышающих точность диагностики.
- Реализация этого алгоритма в Безопасном компиляторе [5], разрабатываемом в ИСП РАН на основе GCC (опция `-Wclobbered`).
- Тестирование реализации сборкой популярного дистрибутива ОС Linux.

3. Обзор существующих решений

Существующие решения

В GCC с версии 4.3.0 (2008 г.) добавлена опция `-Wclobbered` (включена в `-Wextra`). Опция включает компиляторную диагностику рассматриваемого неопределенного поведения. Данная диагностика реализована на промежуточном представлении RTL [6].

В компиляторах Clang и MSVC данная диагностика отсутствует.

Недостатки существующих решений

Основной недостаток реализации на RTL – зависимость от размещения переменной компилятором. Разные компиляторы, в том числе и разные версии одного компилятора, могут как помещать переменную на регистр в ходе оптимизаций, так и сохранять ее непосредственно в памяти. Во втором случае, анализируя RTL, мы лишаемся возможности диагностики неопределенного поведения. Перенос реализации с RTL на GIMPLE [7], [8] позволил бы обнаруживать потенциальные проблемы без привязки к результатам оптимизации переменной с автоматическим классом памяти. Такой перенос устраняет рассмотренную зависимость от текущего процесса компиляции конкретным компилятором.

Кроме того, реализация на GIMPLE позволит анализировать возвращаемое значение `setjmp` и различать нормальный и анормальный возвраты из `setjmp`. Благодаря этому можно подавить распространенное в GCC ложно-положительное срабатывание со следующим сценарием:

1) Вызов `setjmp`;
2) Запись в локальную не `volatile` переменную;
3) Вызов `longjmp` с соответствующим `jmp_buf`;
4) Запись в эту переменную (или вовсе отсутствие дальнейшего использования переменной).

Очевидно, что данный сценарий не приводит к неопределенному поведению, так как неопределенное значений переменной нигде не используется.

Таким образом, существующая в GCC реализация диагностики на RTL является малоприменимой для реальных проектов ввиду приведенных выше фактов, следствием которых являются многочисленные ложно-отрицательные и ложно-положительные срабатывания [9].

4. Решение: новый алгоритм на GIMPLE

4.1 Детали реализации функций нелокального перехода в GIMPLE

В GIMPLE нелокальные переходы моделируются с помощью функции `abnormal_dispatcher`. Ее вызов происходит в отдельном базовом блоке, который обозначим как B_{ab}. Пользовательская функция всегда обладает единственным B_{ab} (либо его вообще нет, если функция не вызывает `setjmp`). С остальными базовыми блоками B_{ab} соединен специальными анормальными ребрами:

- Исходящими в базовые блоки с вызовом `setjmp` (обозначим такой блок как B_{sj});
- Входящими от базовых блоков с вызовом любой другой функции.

На рис. 1 показан пример ГПУ [10] с B_{ab}. На этом и последующих примерах ГПУ имеет упрощенный схематичный вид. Обозначения ребер здесь и далее:

- Пунктирные линии – анормальные ребра;
- Сплошные линии – нормальные (все остальные) ребра.

Непосредственно перед вызовом `setjmp` для каждой живой локальной не `volatile` переменной компилятор создает phi-функцию. Такую phi-функцию кратко обозначим как `phi0`.

На рис. 2:

- B_{ab} находится в блоке <bb 5>;
- B_{sj} находится в блоке <bb 2> (в нем же `phi0` переменной `i`).

4.2 Фундаментальное ограничение алгоритма на GIMPLE

Фундаментальное ограничение диагностики на GIMPLE – зависимость от наличия phi-функции `phi0`. В случае отсутствия `phi0` мы имеем потенциальное ложно-отрицательное срабатывание. В частности, `phi0` не обладают регистровые ассемблерные переменные из-за их текущей реализации в GCC.

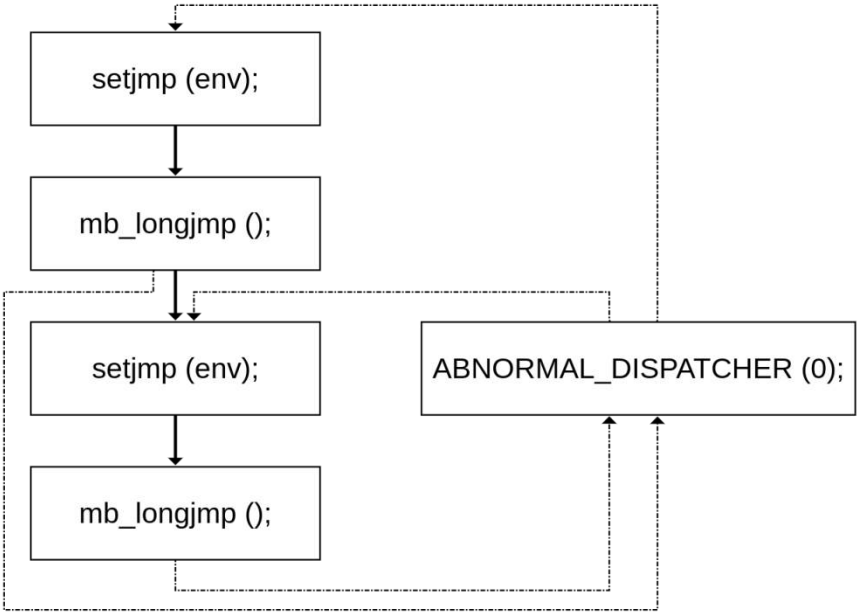


Рис. 1. Пример ГПУ с abnormal_dispatcher
Fig. 1. Example of CFG with abnormal_dispatcher

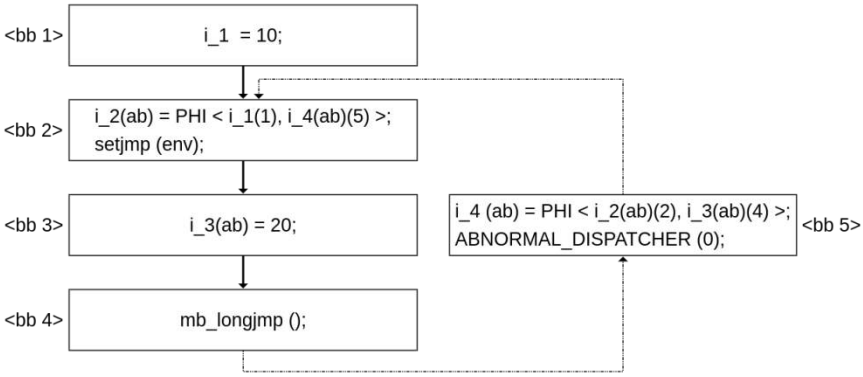


Рис. 2. Пример ГПУ с phi0
Fig. 2. Example of CFG with phi0

В теории также источником ложно-отрицательных срабатываний могут быть поля структур, которые GCC SRA [11] не скаляризирует (не заменяет на локальные переменные), и, как следствие, они не имеют `phi0`, однако другой компилятор может их скаляризировать.

4.3 Базовый алгоритм

Краткий базовый алгоритм новой реализации диагностики `-Wclobbered` на GIMPLE:

1) Поиск B_{ab}. Если у анализируемой функции нет B_{ab}, то диагностика завершается.
2) Вычисление M_{ab} – множества базовых блоков, достигающих B_{ab} (кроме B_{sj}). Здесь и далее достижимость определяется с помощью алгоритма DFS [12]. На данном шаге анормальные ребра считаются проходимы в ходе DFS обхода, на остальных – нет.
3) Вычисление M_{sj} – множества базовых блоков, достижимых из B_{sj}.
4) Вычисление $M_{sj,\,a}$ – множества базовых блоков, достижимых из B_{sj} и совместимых с анормальным возвратом (производится анализ предиката возвращаемого значения `setjmp`).
5) Вычисление M_{int} – пересечения M_{ab} и M_{sj}. То есть M_{int} является множеством базовых блоков между `setjmp` и `abnormal_dispatcher`.
6) Поиск в M_{int} переопределения каждой переменной, обладающей `phi0`. Реализовано это посредством рекурсивного обхода аргументов `phi0`. Если в M_{int} обнаружены константа или явное определение аргумента, то искомое переопределение найдено. Если нет, то диагностика для данной переменной завершается.
7) Поиск использований `phi0` в $M_{sj,\,a}$. Если такое использование найдено, то на переопределение из шага 6 выдается срабатывание как на источник потенциального неопределенного поведения.

Таким образом, весь базовый алгоритм можно обобщить тремя следующими пунктами:

1) Шаг 1: поиск B_{ab}. Наличие B_{ab} – условие запуска алгоритма.
2) Шаги 2-4, 6: поиск переопределения переменной, обладающей `phi0`, между `setjmp` и `abnormal_dispatcher`.
3) Шаги 5, 7: поиск использования `phi0` после анормального возврата.

Если найдено переопределение и использование, то компилятор выдает срабатывание.
Пример поиска переопределения (рис. 3):

- M_{ab} – visited справа;
- M_{sj} – visited слева;
- M_{int} – visited с двух сторон (в <bb 3> искомое на шаге 6 переопределение).

Пример поиска использования (рис. 4):

- $M_{sj,\,a}$ – visited;
- Нет искомого на шаге 7 использования `phi0`, достижимого через $M_{sj,\,a}$.

4.4 Аналогия с алгоритмом анализа циклов

Можно провести некоторую аналогию между базовым алгоритмом и алгоритмом анализа циклов [13]. Рассматриваются все циклы, которые:

- Начинаются с B_{sj};

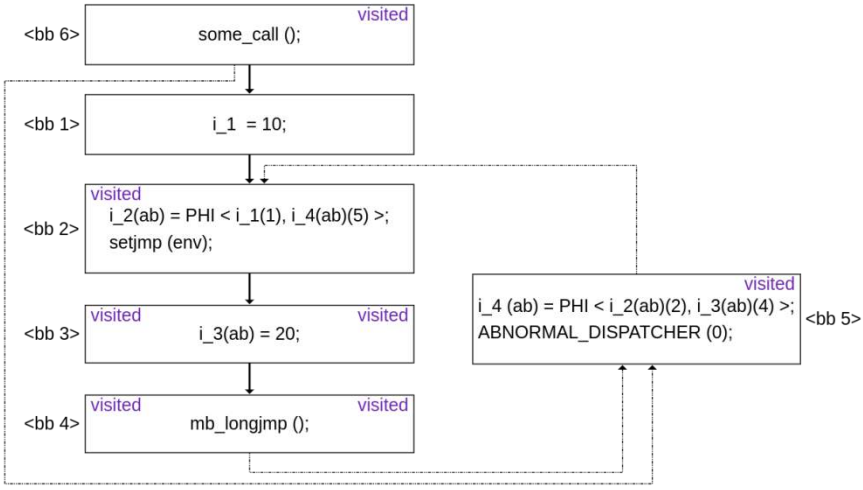


Рис. 3. Поиск переопределения
Fig. 3. Search of redefinition

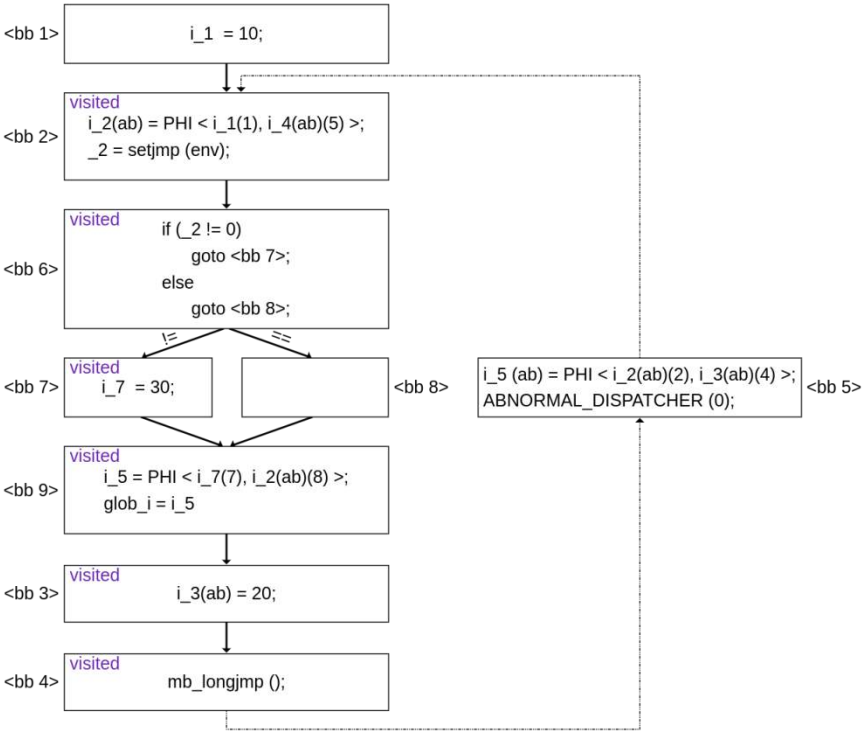


Рис. 4. Поиск использования
Fig. 4. Search of usage

- Включают в себя B_{ab} по произвольному пути;
- Возвращаются в B_{sj}.

Происходит поиск переменных, которые являются живыми на входе и изменяются в таком цикле. Полученный в результате поиска набор кандидатов на срабатывание фильтруется: остаются лишь те определения из цикла, которые в конечном итоге используются после анормального возврата из `setjmp`.

4.5 Эвристика

Данный алгоритм обеспечивает широкое покрытие на исходном коде программ, однако никак не считается с реальным практическим применением `setjmp` программистами.

Рассмотрим следующие мотивационные примеры (см Листинг 1 и Листинг 2).

```
if (setjmp (...) == 0)
   foo ();
else
   handle_longjmp ();
bar ();
```

Листинг 1. Первый шаблон использования в реальном коде
Listing 1. The first pattern of using in real code

```
if (setjmp (...) != 0)
   handle_longjmp ();
bar ();
```

Листинг 2. Второй шаблон использования в реальном коде
Listing 2. The second pattern of using in real code

Принципиальное различие между двумя этими шаблонами:

- Листинг 1: есть отдельный условный блок кода, выполняющийся только после нормального возврата;
- Листинг 2: нет такого условного блока.

На основе анализа исходного кода реальных программ были выделены именно такие шаблоны использования `setjmp`. Сформулируем следующую эвристику:

- Вызов `longjmp` не ожидается из условного блока, выполняющегося только после анормального возврата. Обычно программист не ожидает еще один `longjmp` (по крайней мере, с тем же `jmp_buf`) из кода, отвечающего за обработку предыдущего `longjmp`.
- Вызов `longjmp` ожидается из условного блока, выполняющегося только после нормального возврата, если он есть (листинг 1). При наличии такого выделенного кода программист ожидает `longjmp` именно из него.

В приведенных примерах в базовом алгоритме без эвристики `longjmp` ожидался из любого вызова, так как на шаге 6 переопределение искалось в M_{int}. На шаге 5 M_{int} строится на основе M_{sj}, содержащего все достижимые из B_{sj} блоки. С эвристикой же:

- Листинг 1: `longjmp` ожидается из вызова `foo` (реализуются оба пункта эвристики);
- Листинг 2: `longjmp` ожидается из вызова `bar` (реализуется только первый пункт эвристики, так как условного блока нормального возврата нет).

На рис. 5 и рис. 6 жирным шрифтом выделены базовые блоки, из которых, согласно эвристике, ожидается `longjmp`.

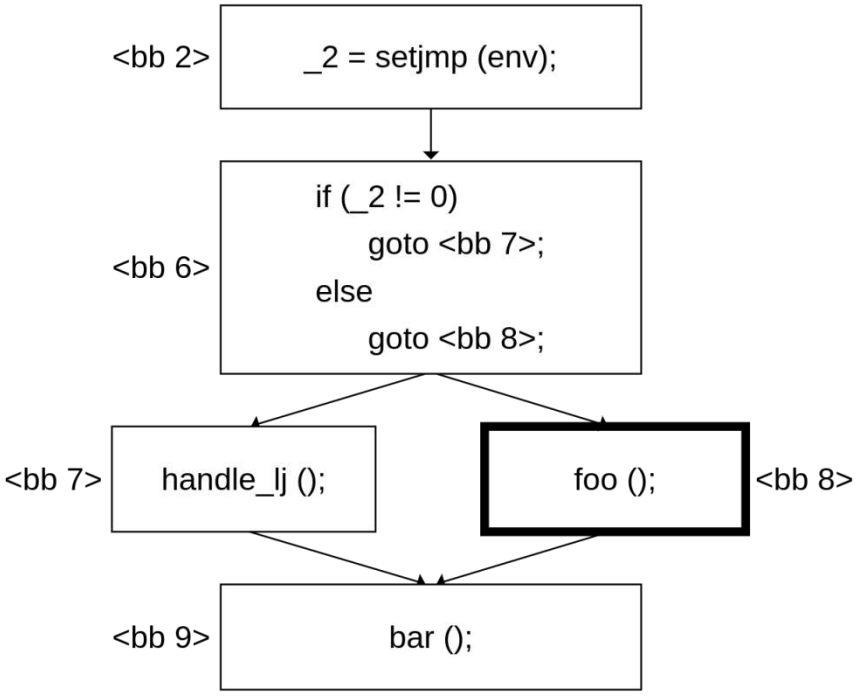


Рис. 5. Эвристика для листинга 1
Fig. 5. Heuristic for listing 1

Таким образом, суть эвристики – определить базовые блоки, где `longjmp` на практике ожидается, то есть сузить множество M_{int}. Для внедрения эвристики модифицируются шаги 4 и 5 базового алгоритма.

Шаг 4 разбивается на два подпункта:

- Вычисление $M_{\text{sj, a}}$ – множества базовых блоков, достижимых из B_{sj} и совместимых с анормальным возвратом (шаг 4 из базового алгоритма).
- Вычисление $M_{\text{sj, n}}$ – множества базовых блоков, достижимых из B_{sj} и совместимых с нормальным возвратом.

На шаге 5, помимо непосредственно вычисления M_{int}, оно также сужается (уточняется с учетом эвристики) до $M_{\text{int, h}}$. Для вычисления $M_{\text{int, h}}$ к шагу 5 добавляются следующие подпункты, соответствующие одному из шаблонов из листингов 1 и 2:

- Если $M_{\text{sj, n}}$ не насыщенное подмножество $M_{\text{sj, a}}$, то $M_{\text{int, h}}$ есть пересечение M_{int} и разности $M_{\text{sj, n}}$ и $M_{\text{sj, a}}$. Соответствует шаблону из листинга 1.
- Если $M_{\text{sj, n}}$ насыщенное подмножество $M_{\text{sj, a}}$, то $M_{\text{int, h}}$ есть пересечение M_{int} и $M_{\text{sj, n}}$. Соответствует шаблону из листинга 2.
- Под понятием насыщенного подможество здесь подразумевается: множество базовых блоков A является насыщенным подмножеством множества базовых блоков B, если разность A и B является пустым множеством или содержит только пустые базовые блоки.
- На шаге 6 (поиск переопределения) вместо M_{int} используется новое $M_{\text{int, h}}$.

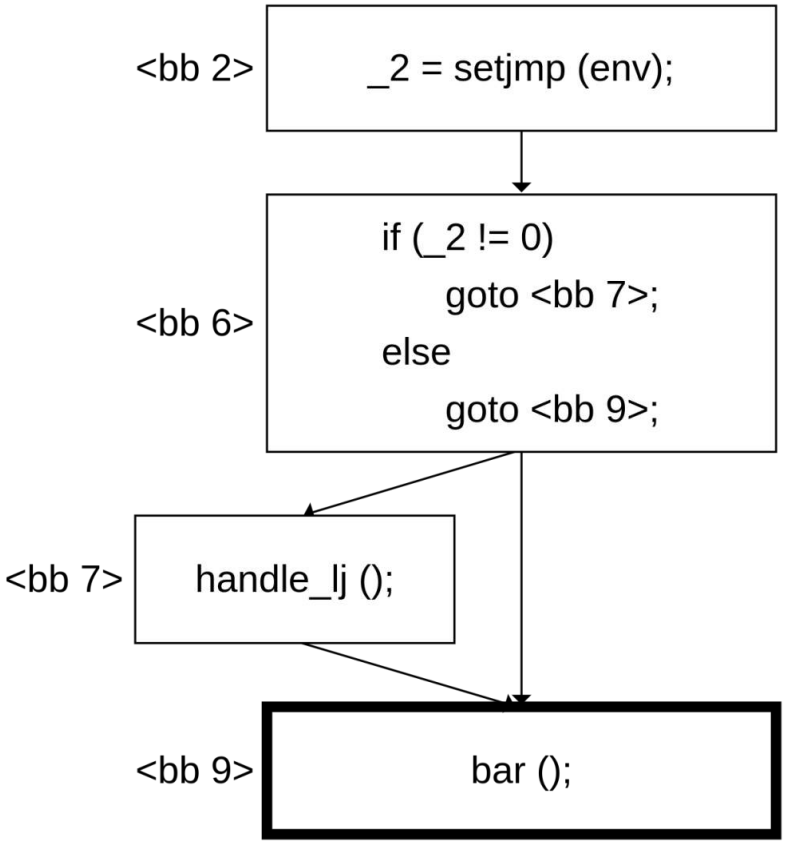


Рис. 6. Эвристика для листинга 2
Fig. 6. Heuristic for listing 2

4.6 Дополнительное уточнение алгоритма с эвристикой

С эвристикой может возникнуть ситуация, когда найденное на шаге 6 переопределение окажется присваиванием возвращаемого значения функции, внутри которой ожидается потенциальный `longjmp`. Данная ситуация может спровоцировать ложно-положительное срабатывание. Рассмотрим следующее выражение:

```
var = mb_longjmp ();
```

Листинг 3. Пример выражения, которое может привести к ложно-положительному срабатыванию в алгоритме с эвристикой
Listing 3. The example of an expression that may lead to a false positive

Рассмотрим ситуацию, когда базовый блок с данным выражением является крайним в $M_{\text{int, h}}$. Под крайним здесь имеется в виду то, что ни один из наследников этого базового блока не принадлежит $M_{\text{int, h}}$. То есть из вызовов внутри наследников по эвристике не может произойти `longjmp`. Тогда и присваивания переменной `var` в случае вызова `longjmp` внутри `mb_longjmp` не произойдет. Таким образом, шаг 6 базового алгоритма дополняется следующим образом:

- Базовый блок с искомым переопределением должен содержаться в $M_{\text{int, h}}$, и хотя бы один из наследников данного базового блока также должен содержаться в $M_{\text{int, h}}$.

4.7 Учет области действия `jmp_buf`

Рассмотрим следующий пример (Листинг 4).

```
{
   jmp_buf env;
   setjmp (env);

   ... some code ...

   mb_longjmp_local (env);
   var = 10;
   no_longjmp ();
}
```

Листинг 4. Пример кода, который может привести к ложно-положительному срабатыванию
Listing 4. The example of a code that may lead to a false positive

Здесь `jmp_buf env` объявлен локальным. Тогда `env` может достигнуть `longjmp` внутри некоторого вызова, только если он явно передается в этот вызов. Таким образом, после переопределения переменной `var` вызов `longjmp` невозможен. Аналогичные примеры можно привести для `jmp_buf` с любой областью действия. Для уточнения алгоритма для этого случая планируется модификация шага 2 базового алгоритма:

- Вычисление $M_{\text{sj, jb}}$ – множества базовых блоков, достигающих B_{ab} с учетом возможности утечки `jmp_buf` в вызов функции (то есть анализ сигнатуры и места определения вызываемой функции исходя из области действия `jmp_buf`). Например, если `jmp_buf` локальный, как в примере, то его утечка в некоторый вызов возможна, только если он явно передается в этот вызов.

На остальных шагах вместо M_{sj} используется новое $M_{\text{sj, jb}}$.

5. Тестирование и анализ результатов

Тестирование заключалось в сборке дистрибутива Alpine Linux 3.16. Для сборки использовался Безопасный компилятор на базе GCC 11.2.1 с флагами `-O2 -Safe2` (опция `-Safe2` принудительно включает `-Werror=clobbered`). Безопасным компилятором собираются около 3700 пакетов дистрибутива, из которых около 700 используют `setjmp`.

Количество пакетов со срабатыванием `-Wclobbered`:

- Алгоритм на RTL (GCC): 125 пакетов.
- Алгоритм на GIMPLE базовый: 102 пакета (по сравнению с RTL устранено 14 ложно-отрицательных и 37 ложно-положительных).
- Алгоритм на GIMPLE с эвристикой: 85 пакетов.
- Алгоритм на GIMPLE с учетом области действия `jmp_buf` на данный момент находится в разработке.

Большинство срабатываний нового алгоритма на GIMPLE были определены как истинные в рамках существующих ограничений. Небольшое количество выявленных ложно-положительных срабатываний вызвано сложными для анализа случаями. Например, сюда относятся запутанные условия совместимости кода с анормальным возвратом (большое количество нетривиально пересекающихся предикатов). В дальнейшем данные ошибки планируется исправить.

Разработчикам ряда пакетов было сообщено об обнаруженных срабатываниях. Нами были предложены изменения программного кода, устраняющие потенциальное неопределенное поведение. В 5 пакетах предложенные изменения были приняты и включены разработчиками в более новые версии пакетов.

6. Заключение

В рамках данной работы были рассмотрены существующие и предложен новый метод обнаружения случаев неопределенного поведения в языке Си при использовании функций нелокальных переходов. Были решены следующие задачи:

- Рассмотрены существующие методы. Ввиду их недостатков они были признаны неэффективными для практического применения на реальных проектах.
- Разработан и описан новый алгоритм диагностики, а также эвристика, повышающая точность срабатываний на реальном коде. Фундаментальное ограничение данного алгоритма – зависимость он наличия phi-функции перед вызовом `setjmp`. Если у некоторой переменной нет такой phi-функции, то ее анализ невозможен.
- Представленный алгоритм с эвристикой реализован в Безопасном компиляторе, разрабатываемом в ИСП РАН на основе GCC.
- Реализация была протестирована сборкой Alpine Linux 3.16. Общее количество пакетов со срабатыванием: 85.

Дальнейшая планируемая в рамках задачи работа:

- Подробный анализ срабатываний.
- Исправление обнаруженных недостатков.
- Внедрение учета области действия `jmp_buf` в текущую реализацию.

Список литературы / References

[1]. C11 Standard ISO/IEC 9899:2011 // Programming language – C. – 2011. – P. 561. – International standard.
[2]. Herity, Dominic. C++ in embedded systems: Myth and reality / Dominic Herity // Embedded Systems Programming. – 1998. – Vol. 11, no. 2. – Pp. 48–71.
[3]. ISO/IEC/IEEE 9945:2009 // Portable Operating System Interface (POSIX®) Base Specifications, Issue 7. – 2009. – P. 37. – International standard.
[4]. Xu, Xiao. Research on coroutine-based process interaction simulation mechanism in c++ / Xiao Xu, Ge Li // AsiaSim 2012: Asia Simulation Conference 2012, Shanghai, China, October 27-30, 2012. Proceedings, Part III / Springer. – 2012. – Pp. 178–187.
[5]. Baev, Roman Vyacheslavovich. Prevention of vulnerabilities arising from optimization of code with Undefined Behavior / Roman Vyacheslavovich Baev, Leonid Vladlenovich Skvortsov, Evgeny Alekseevich Kudryashov, Ruben Arturovich Buchatskiy, Roman Aleksandrovich Zhuykov // Proc. Inst. Syst. Program. RAS. – 2021. – Vol. 33, no. 4. – Pp. 195–210.
[6]. Novillo, Diego. GCC Internals-Internal Representations / Diego Novillo // GCC IR-2. – 2007.
[7]. The internals of the GNU compilers. – Accessed: 2023-11-04. Available at:https://gcc.gnu.org/onlinedocs/gccint/.
[8]. Merrill, Jason. Generic and gimple: A new tree representation for entire functions / Jason Merrill // Proceedings of the 2003 GCC Summit. – 2003. – Pp. 171–180.
[9]. [9/10/11/12 Regression] clobbered by longjmp warning ignores the data flow // Bug 21161, GCC. – Accessed: 2023-11-04. Available at: https://gcc.gnu.org/bugzilla/show_bug.cgi?id=21161.
[10]. Aho, Alfred V. Compilers: principles, techniques, and tools / Alfred V Aho, Ravi Sethi, Jeffrey D Ullman // Addison-wesley Reading – 2007. – Vol. 2. - Pp. 399 - 410.
[11]. Jambor, Martin. The new intraprocedural scalar replacement of aggregates / Martin Jambor //GCC Developers' Summit. – 2010. – Vol. 47.
[12]. Tarjan, Robert. Depth-first search and linear graph algorithms / Robert Tarjan // SIAM journal on computing. – 1972. – Vol. 1, no. 2. – Pp. 146–160.
[13]. Haghighat, Mohammad R. Symbolic analysis for parallelizing compilers / Mohammad R Haghighat. No. 1880. – Springer Science & Business Media, 1995.

Информация об авторах / Information about authors

Никита Юрьевич ШУГАЛЕЙ – старший лаборант кафедры Системного программирования Московского физико-технического института. Сфера научных интересов: компиляторные технологии, безопасность программного обеспечения, методы статического анализа кода, оптимизация программ.

Nikita Yurievich SHUGALEY – senior laboratory technician of the Department of system programming of the Moscow Institute of Physics and Technology. Research interests: compiler technologies, software security, methods of static code analysis, program optimization.

Владислав Анатольевич ИВАНИШИН – научный сотрудник компиляторного отдела Института системного программирования. Научные интересы включают в себя компиляторные технологии, операционные системы, безопасность программного обеспечения, методы статического анализа кода, оптимизацию программ.

Vladislav Anatolevich IVANISHIN – researcher at the Compiler Department of the Institute for System Programming of the RAS. Research interests include compiler technologies, operating systems, software security, methods of static code analysis, and program optimization.

Александр Владимирович МОНАКОВ – старший научный компиляторного отдела Института системного программирования. Сферой научных интересов является компиляторные технологии, оптимизация программ.

Alexander Vladimirovich MONAKOV – researcher at the Compiler Department of the Institute for System Programming of the RAS. Research interests: compiler technologies, program optimization.

DOI: 10.15514/ISPRAS-2023-35(6)-8

Инструмент для поиска гонок по данным RaceHunter

Е.А. Герлиц, ORCID: 0000-0002-1747-075X <*gerlits@ispras.ru*>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Гонки по данным - это ошибки в многопоточных программах, возникающие, когда два потока выполняют доступ к одной ячейке памяти без синхронизации. Гонки по данным трудно находить и отлаживать. В этой статье представлен метод динамического поиска гонок по данным RaceHunter, который выполняет мониторинг многопоточной программы, выявляет конфликтующие доступы к памяти и систематически проверяет их на гонки по данным. RaceHunter не выдаёт ложных сообщений о гонках по данным, когда программа использует нестандартные примитивы синхронизации или неизвестные способы синхронизации потоков, и может находить гонки по данным, которые упускают другие подходы. Инструменты динамического поиска гонок по данным могут использоваться для мониторинга длительных выполнений программы либо для проверки относительно непродолжительных выполнений, инициированных тестами. Последнее - основной сценарий использования RaceHunter.

Ключевые слова: гонка по данным; динамический анализ; синхронизация потоков; состояние гонки; параллельная программа; многопоточная программа.

Для цитирования: Герлиц Е.А. Инструмент для поиска гонок по данным RaceHunter. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 135–156. DOI: 10.15514/ISPRAS–2023–35(6)–8.

RaceHunter Dynamic Data Race Retector

Gerlits E. A., ORCID: 0000-0002-1747-075X<*gerlits@ispras.ru*>
Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn Tt., Moscow, 109004, Russia

Abstract. Data races are a class of concurrency errors where two threads access a shared memory location without proper synchronization. Data races are hard to reveal and debug. This paper presents RaceHunter - a dynamic data race detection technique which monitors executions of shared memory concurrent programs, discovers pairs of conflicting memory accesses and systematically verifies them for data races. RaceHunter does not report false data races when the target software exploits non-standard synchronization primitives or unknown synchronization protocols and can find data races missed by other techniques. Dynamic data race detectors can monitor continuous, e.g. real-life, program executions or they can verify relatively short program executions, e.g. organized by system tests. The latter is the primary use case scenario for RaceHunter.

Keywords: data race; dynamic data race detection; dynamic analysis; concurrent error; concurrency error; thread synchronization; race condition; concurrent program, shared-memory program, multithreaded program; racehunter.

For citation: Gerlits E. A. RaceHunter dynamic data race detector. *Trudy ISP RAN/Proc. ISP RAS*, 2023, vol. 35, issue 6, pp. 135 – 156. DOI: 10.15514/ISPRAS–2023–35(6)–8.

1. Введение

Настоящая работа посвящена теме верификации программного обеспечения. В качестве объекта верификации выступают многопоточные программы, т.е. параллельные программы с общей памятью. Помимо ошибок, присущих последовательным программам, в многопоточных программах дополнительно возникают новые типы ошибок. Одним из них являются гонки по данным.

Определение 1 (Гонка по данным). *Гонка по данным - это ситуация в многопоточной программе, когда одновременно осуществляются два конфликтующих доступа к памяти.*

Определение 2 (Конфликтующие доступы к памяти). *Два доступа к памяти конфликтуют, если:*

1. *Их осуществляют разные потоки выполнения.*
2. *Участки памяти, к которым происходят обращения, пересекаются.*
3. *Один из доступов выполняет запись.*
4. *Один из доступов выполняется неатомарно.*

Гонка по данным может приводить к некорректным вычислениям и исключительным ситуациям в программе. Поэтому выявление гонок по данным является актуальной задачей.

Для её решения к настоящему времени предложено несколько подходов. В них можно условно выделить два направления - статический анализ [1—3] и динамический анализ. К статическим относятся подходы, в которых код программы исследуется без её выполнения. Динамические подходы ищут гонки по данным во время выполнения программы.

Востребованность обоих направлений обоснована их преимуществами и недостатками. Ключевым преимуществом статического анализа является более полный перебор потенциально возможных выполнений параллельной программы. Однако статические подходы могут выдавать

ложные предупреждения о гонках по данным. В случае большого их количества, обнаружение реально возмозможных гонок по данным существенно затрудняется.

Метод поиска гонок по данным, изложенный в данной работе, относится к динамическому анализу. В существующих динамических подходах можно условно выделить три направления:

1. Happens-before [8] - мониторинг выполнения параллельной программы с отслеживанием частичного порядка на множестве доступов к памяти в этом выполнении. Этот подход реализует, например, инструмент TSan [4].
2. Lock-set - мониторинг выполнения параллельной программы с вычислением множеств захваченных блокировок на доступах к памяти. Этот подход реализует инструмент Eraser [5].
3. Breakpoint-watchpoint - непосредственное обнаружение гонок по данным при помощи точек останова и наблюдения. Этот подход реализуют инструменты DataCollider [6] и KCSAN [7].

Все три направления востребованы ввиду их преимуществ и недостатков. Метод, который изложен в данной работе, использует breakpoint-watchpoint подход, ключевым преимуществом которого является то, что для вердикта о гонке по данным он не отслеживает события синхронизации. Поэтому он применяется к системному программному обеспечению, такому как операционные системы и виртуальные машины, где зачастую используются способы синхронизации, которые не удовлетворяют lock-unlock семантике и/или при применении которых трудно автоматически строить полное отношение happens-before. Например, это барьеры памяти, запрет переключения потоков, запрет прерываний.

1.1 Мотивация

Изначально breakpoint-watchpoint подход был реализован в инструменте DataCollider. Инструмент периодически устанавливает точки останова на некотором количестве инструкций доступа к памяти. Каждая инструкция выбирается случайным образом из множества инструкций доступа к памяти в программе[1]. Обработчик точки останова:

1. Читает данные, к которым обращается инструкция.
2. Устанавливает точку наблюдения на участок памяти, к которому обращается инструкция.
3. Выполняет ожидание в течение небольшого периода времени.
4. Удаляет точку наблюдения.
5. Второй раз читает данные, к которым обращается инструкция.
6. Сообщает о гонке по данным, если сработала точка наблюдения либо изменились данные.
7. Произвольным образом выбирает инструкцию доступа к памяти и устанавливает точку останова на неё.

Предполагается, что точка наблюдения срабатывает, когда происходит доступ к участку памяти, который пересекается с наблюдаемым.

Видно, что в подходе DataCollider доступы к памяти, на которые устанавливаются точки останова, выбираются случайным образом. Следствием этого является то, что инструмент может пропускать инструкции доступа к памяти, которые участвуют в гонке по данным. Следовательно, подход может пропускать гонки по данным.

Рассмотренный подход получил своё развитие в инструменте KCSAN. Точки останова и точки наблюдения инструмент реализует программно на основе статического инструментирования.

[1] Некоторые инструкции предварительно исключаются из множества. Например, те, что несомненно обращаются только к автоматической памяти.

Функции, инструментирующие инструкции доступа к памяти, выполняют действия обработчика точки останова, если они выполняются N-ми по порядку данным ядром процессора, где N генерируется случайным образом.

В подходе KCSAN инструкции доступа к памяти для проверки на гонки по данным также выбираются случайным образом. Следовательно, этот вариант breakpoint-watchpoint подхода также может пропускать гонки по данным.

Гипотеза 1. *Если доступы к памяти для проверки на гонки по данным выбирать не случайным образом, а систематически проверять только конфликтующие доступы к памяти (определение 2), то это позволит более эффективно выявлять гонки по данным.*

1.2 Структура работы

Задача работы сформулирована в разделе 2. Основные этапы подхода RaceHunter последовательно изложены в разделе 3. Расчёт объёма памяти, потребляемой инструментом RaceHunter в худшем случае, дан в разделе 4. О применении RaceHunter на практике рассказано в разделе 5. Сравнение RaceHunter с существующими подходами к поиску гонок по данным выполнено в разделе 6. Выводы по результатам работы сделаны в разделе 7. В разделе 8 обозначены направления будущих исследований. В приложении A приведены вспомогательные алгоритмы - функции, используемые алгоритмами и формулами из основных разделов.

2. Постановка задачи

Предложить динамический подход для поиска гонок по данным в многопоточных программах, который:

- Должен систематически проверять конфликтующие доступы к памяти на гонки по данным.
- Не должен выдавтать ложных предупреждений о гонках по данным при наличии не наблюдаемых событий синхронизации.

3. Подход

На рис. 1 (fig. 1) изображена схема подхода RaceHunter к поиску гонок по данным. Основные этапы подхода следующие:

1. **Инструментирование.** В определённые места кода программы компилятор автоматически вставляет вызовы функций из библиотеки поддержки выполнения RaceHunter. Это позволяет отслеживать события в программе во время её выполнения и реагировать на них.
2. **Мониторинг.** Выполнить программу. Во время выполнения для каждого потока отслеживать и записывать события: доступ к памяти, вызов функции, завершение функции, порождение потока, завершение потока и др. Результат мониторинга - трасса событий в программе.
3. **Анализ трассы**. Результат - множество целей для проверки на гонки по данным, где цель - это пара описателей конфликтующих доступов к памяти.
4. **Провокация гонки по данным.** Для каждой цели повторно выполнить программу, установив программные точки останова на двух доступах к памяти, соответствующих описателям из цели. В обработчиках точек останова ожидать выполнения второй точки останова. Если вторая точка останова сработала, то обнаружена гонка по данным.

Для провокации гонки по данным программа должна быть выполнена повторно. Подразумева-

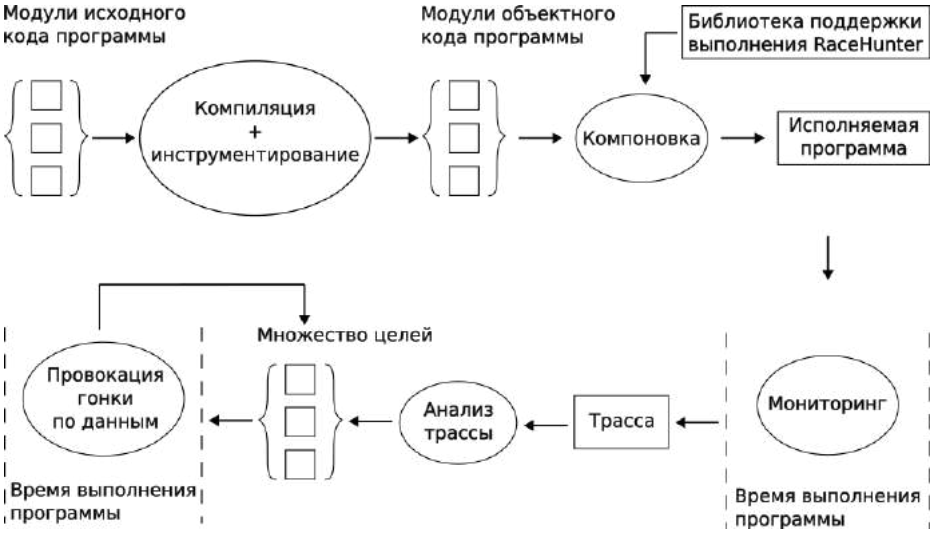


Рис. 1. Схема подхода RaceHunter
Fig. 1. General scheme of RaceHunter approach

ется, что повторные выполнения - это не произвольные выполнения, а попытки воспроизвести то выполнение, которое зафиксировано на этапе мониторинга. Для этого необходимо (но не достаточно):

1. Выполнить программу в том же начальном состоянии. Соответственно, для применения RaceHunter необходимо иметь способ перевода программы в начальное состояние либо её перезапуска.
2. Подать на вход программе те же входные данные.

Из-за непостоянности скорости исполнения потоков ядрами процессора места переключений потоков планировщиком в повторных выполнениях программы могут отличаться. Следственно, даже при выполнении двух вышеуказанных необходимых условий повторное выполнение многопоточной программы может отличаться от исходного. По мере изложения будем пояснять, как RaceHunter учитывает этот недетерминизм, свойственный многопоточным программам.

Дополнительные факторы недетерминизма, такие как использование случайных чисел и взаимодействие с окружением, по-возможности должны быть устранены на время верификации (детерминированы), чтобы повысить её эффективность.

3.1 Инструментирование

RaceHunter отслеживает события в программе и обновляет своё состояние при их возникновении. Для этого код RaceHunter внедряется в проверяемую программу:

1. На этапе компиляции программы в автоматическом режиме выполняется статическое инструментирование инструкций доступа к памяти, а также точек входа и выхода из функций, т.е. непосредственно перед этими инструкциями вставляются вызовы соответствующих функций из библиотеки поддержки выполнения RaceHunter. Код этой библиотеки должен быть скомпонован с кодом верифицируемой программы.
   Инструментирование реализовано на уровне промежуточного представления LLVM IR [9] компилятора Clang [10]. В целом инструментирование схоже с тем, что выполняет

инструмент TSan [11] и другие инструменты [12]. Основное преимущество текущей реализации инструментирования в RaceHunter состоит в том, что собственные инструкции программы не изменяются, а только добавляются новые инструкции - вызовы функций из библиотеки поддержки выполнения RaceHunter.

2. В качестве оптимизации функции для работы с памятью, такие как $memset$, $memcpy$, $memove$ [13] и др., автоматически подменяются на функции из библиотеки поддержки выполнения RaceHunter. Это позволяет множество обращений к памяти из этих функций моделировать одним обращением. Эту оптимизацию можно отключить, если важно не изменять собственные инструкции программы.
3. Вручную дополняется код некоторых системных функций, например, создание потока.

3.2 Мониторинг

На этапе мониторинга RaceHunter отслеживает ряд событий в программе и записывает их в трассу событий.

Обозначение 1 (Множество целых неотрицательных чисел)**.** $\mathbb{N}_0 = \mathbb{N} \cup 0$.

Определение 3 (Доступ к памяти)**.** *Доступ к памяти - это кортеж*[2] $< I, D, L, R, W, A >$*:*

- $I \in \mathbb{N}_0$ *- адрес инструкции доступа к памяти.*
- $D \in \mathbb{N}_0$ *- адрес начала сегмента данных, к которому осуществляется доступ.*
- $L \in \mathbb{N}$ *- длина в байтах сегмента данных, к которому осуществляется доступ.*
- $R \in 0, 1$ *- чтение (1) или нет (0).*
- $W \in 0, 1$ *- запись (1) или нет (0).*
- $A \in 0, 1$ *- атоматрный доступ (1) или нет (0).*

Обозначение 2 (Множество всевозможных доступов к памяти)**.** *Обозначим* Λ *множество доступов к памяти, соответствующих определению 3.*

Определение 4 (Вызов функции)**.** *Вызов функции - это кортеж* $< I, F >$*:*

- $I \in \mathbb{N}_0$ *- адрес инструкции вызова функции.*
- $F \in \mathbb{N}_0$ *- адрес вызываемой функции.*

Обозначение 3 (Множество всевозможных вызовов функций)**.** $\Gamma = \{< I, F >: I, F \in \mathbb{N}_0\}$.

Определение 5 (Событие)**.** *Событие - это кортеж* $< K, V >$*.* $K \in 1..4$ *- тип события. Семантика значений* K *следующая: 1 - доступ к памяти, 2 - вызов функции, 3 - завершение функции, 4 - порождение потока.*

$$V = \begin{cases} K = 1 & V \text{ - доступ к памяти по определению 3} \\ K = 2 & V \text{ - вызов функции по определению 4} \\ K = 3 & V \in \mathbb{N}_0 \text{ - адрес функции} \\ K = 4 & V \text{ - идентификатор потока по определению 8} \end{cases}$$

Обозначение 4 (Множество всевозможных событий)**.** *Обозначим* Σ *множество событий, соответствующих определению 5.*

[2]Формально кортеж - это конечная последовательность элементов или функция с областью определения $\{i \in \mathbb{N} : i \leq L\}$, где $L \in \mathbb{N}$ - количество элементов кортежа, и областью значений совпадающей с множеством элементов кортежа. Для наглядности в работе на элемент E кортежа t будем ссылаться через $t.E$.

Определение 6 (Трасса потока)**.** *Трасса потока - это кортеж* $< U, E, C >$*:*

- U *- идентификатор потока по определению 8.*
- $E : \mathbb{N} \to \Sigma$ *- конечная последовательность событий в потоке.*
- $C \in \mathbb{N}_0$ *- количество порождённых потоков.*

Определение 7 (Трасса программы)**.** *Трасса программы* $Trace$ *- это конечное множество трасс потоков по определению 6.*

В определениях 5 и 6 фигурирует идентификатор потока. Подходящий идентификатор должен удовлетворять следующим требованиям:

- Не должно быть двух потоков с одинаковым идентификатором в любом выполнении программы.
- В повторных выполнениях программы на этапе провокации гонки по данным поток должен иметь тот же идентификатор.

Библиотека поддержки выполнения RaceHunter предоставляет интерфейс, с помощью которого инструменту может быть указан способ вычисления идентификатора текущего потока. Например, в качестве идентификатора потока можно взять адрес начальной функции потока, если в программе не может быть двух потоков с одной начальной функцией.

По умолчанию RaceHunter использует свой внутренний способ идентификации потоков. Этот способ исходит из того, что программа начинает исполняться с одним потоком и любой поток может порождать новые потоки.

Определение 8 (Идентификатор потока по умолчанию)**.** *Идентификатор потока - это частично определённая функция* $\mathbb{N} \to \mathbb{N}$*. Функция моделирует конечную последовательность натуральных чисел.*

Определение 9 (Идентификатор начального потока)**.** $\emptyset \to \mathbb{N}$.

Алгоритм 1 обрабатывает событие порождения одним потоком другого непосредственно перед фактическим порождением потока в программе. Обработка остальных событий в целом сводится к добавлению информации о них в трассу потока, в котором эти события происходят.

Алгоритм 1. Обработка события порождения потока на этапе мониторинга

Вход: $u_1 : \mathbb{N} \to \mathbb{N}$ - идентификатор текущего потока
$trace$ - трасса программы по определению 7.
Выход: $trace$ - обновлённая трасса программы. Трассу программы параллельно пополняют все потоки. Остальные переменные алгоритма локальные.

1: $t_1 \in trace : equal(t_1.U, u_1)$ {Трасса текущего потока}
2: $u_2 : Dom(u_1) \cup \{|Dom(u_1)| + 1\} \to \mathbb{N}$ {Строим идентификатор порождаемого потока}

$$u_2(i) = \begin{cases} \forall i \in Dom(u_1) & u_1(i) \\ i = |Dom(u_1)| + 1 & t_1.C + 1 \end{cases}$$

3: $t_2 =< u_2, \emptyset \to \Sigma, 0 >$ {Порождаемый поток ещё не имеет событий и не порождал потоков}

4: $t_3 =< t_1.U, t_1.E, t_1.C + 1 >$ {Новая трасса текущего потока}
5: $trace = trace \setminus \{t_1\} \cup \{t_2, t_3\}$ {Обновление трассы программы}

Утверждение 1. *Алгоритм 1 порождает трассу программы, в которой все трассы потоков имеют различные идентификаторы.*

Доказательство. Вычисляемый алгоримом 1 идентификатор потока соответствует единственному пути в генеалогическом дереве потоков от начального потока до порождаемого потока. Узел этого дерева - порядковый номер потока-потомка у потока-родителя. □

Замечание 1. *В данной работе в параллельных алгоритмах будем выделять и нумеровать шаги. Положим, что шаг алгоритма является атомарным действием, которое может изменить состояние алгоритма. Например, в алгоритме 1 ровно 5 шагов. Также положим, что параллельное выполнение шагов алгоритма потоками эквивалентно некоторому их последовательному выполнению [14]. Используемая нами семантика выполнения параллельных алгоритмов соответствует семантике выполнения параллельных алгоритмов в языке PlusCal [15].*

3.3 Анализ трассы

Цель анализа трассы - определить конфликтующие пары доступов к памяти и сформировать из них цели для проверки на гонки по данным. Конфликтующие доступы к памяти RaceHunter ищет по определению 2. Формализуем его.

Определение 10 (Конфликтующие доступы к памяти)**.** *Доступ к памяти a_1 из трассы потока t_1 и доступ к памяти a_2 из трассы потока t_2 конфликтуют, если выполнена конъюнкция следующих условий:*

1. $\neg equal(t_1.U, t_2.U)$
2. $a_1.D = a_2.D \vee (a_1.D < a_2.D \wedge a_2.D < a_1.D + a_1.L)$
   $\vee (a_2.D < a_1.D \wedge a_1.D < a_2.D + a_2.L)$
3. $a_1.W = 1 \vee a_2.W = 1$
4. $a_1.A = 0 \vee a_2.A = 0$

Однако данных о доступе к памяти по определению 3 недостаточно для идентификации целевого доступа к памяти в повторном выполнении программы на этапе провокации гонки по данным. К примеру, целевая инструкция доступа к памяти ($a_1.I$, $a_2.I$) может выполняться несколько раз и несколькими потоками. Также равенство адресов данных ($a_1.D$, $a_2.D$) в разных запусках программы в общем случае не гарантируется.

Идентификация целевых доступов к памяти дополнительно осложняется тем, что повторное выполнение многопоточной программы может быть не идентично тому, что наблюдалось на этапе мониторинга. Поэтому описатель целевого доступа к памяти должен не только идентифицировать целевой доступ к памяти, но и допускать вариации в выполнении программы.

Определение 11 (Описатель доступа к памяти)**.** *Описатель доступа к памяти - это кортеж $< U, B, I, S, N >$:*

- *U - идентификатор потока по определению 8.*
- *$B \in \{0, 1\}$ - значение 0 (ложь) - любой поток, значение 1 (истина) - поток с идентификатором U.*
- *$I \in \mathbb{N}_0$ - адрес инструкции доступа к памяти.*
- *$S : \mathbb{N} \to \Gamma$ - верхняя часть стека вызовов функций во время доступа к памяти (функция, вызванная последней, является первым элементом последовательности). $Dom(S) = \emptyset$ означает любой стек вызовов функций.*

- $N \in \mathbb{N}$ - *порядковый номер доступа к памяти в трассе потока.*

В определении 11 подразумевается, что N нумерует доступы к памяти, соответствующие остальным условиям, т.е. доступы к памяти из потока с идентификатором U, выполненные инструкцией с адресом I, когда стек потока был равен S.

Обозначение 5. *Обозначим Ψ множество всевозможных описателей доступа к памяти по определению 11.*

Определение 12 (Цель для проверки на гонки по данным)**.** *Цель $Target$ это кортеж $< D_1 \in \Psi, D_2 \in \Psi >$ - пара описателей (по определению 11) доступов к памяти (по определению 3) из трассы программы (по определению 7), которые конфликтуют (по определению 10).*

Алгоритм 2. Построение описателя доступа к памяти

Вход: u - идентификатор потока по определению 8, который выполняет доступ к памяти
i - порядковый номер доступа к памяти в трассе потока с идентификатором u
$trace$ - трасса программы по определению 7
Выход: d - описатель доступа к памяти по определению 11

$d.U = u$
$t_u \in trace : equal(t_u.U, u)$
$d.I = t_u.E(i).V.I$
$d.B = \exists t \in trace : \neg equal(t.U, u) \wedge (\exists j \in Dom(t.E) : t.E(j).K = 1 \wedge t.E(j).V.I = d.I)$
$p = \{j : j \in Dom(t_u.E) \wedge t_u.E(j).K = 1 \wedge t_u.E(j).V.I = d.I\}$ {Доступы, выполненные d.I}

$s = \{(j, st) : j \in p \wedge st = stack(u, j, trace)\}$ {Стеки вызовов функций для доступов из p}
$d.S = top(s(i), \{s(j) : j \in p \wedge j \neq i\})$ {Верхняя часть стека, отличающая i от остальных}

$d.N = |\{j \in p : j < i$ {Доступы к памяти, выполненные инструкцией d.I ранее доступа i}
$\wedge\, Dom(d.S) \subset Dom(s(j))$
$\wedge\, \forall k \in Dom(d.S) : s(j)(|Dom(s(j)| - k + 1) = d.S(k)\}| + 1$

Алгоритм 2 строит описатель доступа к памяти по определению 11. Основная идея алгоритма 2 состоит в том, чтобы проверять необходимость добавления очередной порции данных к описателю для идентификации целевого доступа к памяти. Необходимой информацией является только адрес инструкции доступа к памяти.

Замечание 2. *Алгоритм 2 разделяет задачу идентификации целевого доступа к памяти на две подзадачи - идентификацию потока, который выполнил целевой доступ к памяти, и идентификацию доступа к памяти в рамках этого потока. Для некоторых приложений трудно подобрать идентификаторы для потоков, которые бы сохранялись в повторных выполнениях. В таком случае как сам алгоритм 2, так и описатель доступа к памяти (определение 11) можно модифицировать.*

Алгоритм построения множества целей по определению 12 по трассе программы по определению 7 выполняет поиск пар событий доступа к памяти ($K = 1$ по определению 5), которые конфликтуют по определению 10, и строит для каждого доступа из пары описатель по определению 11. В данной работе этот алгоритм решено не приводить с целью экономии места, так как его вклад в прояснение метода RaceHunter несущественен.

3.4 Провокация гонки по данным

Для каждой цели по определению 12 из множества целей, построенного по результатам анализа трассы, программа выполняется повторно. Провокация гонки по данным состоит в организации одновременного выполнения доступов к памяти, указанных в цели.

Определение 13 (Состояние потока)**.** *Состояние потока - это кортеж* $< U, S, N, L, C >$*:*

- U *- идентификатор потока по определению 8.*
- $S : \mathbb{N} \to \Gamma$ *- стек вызовов потока.*
- $N : \Psi \to \mathbb{N}_0$ *- количество обнаруженных доступов к памяти, соответствующих описателю.*
- $C \in \mathbb{N}_0$ *- количество порождённых потоков.*

Определение 14 (Состояние программы)**.** *Состояние программы это кортеж* $< T, A, W, V, X >$*:*

- T *- конечное множество состояний потоков по определению 13*
- $A : \Psi \to \Lambda$ *- функция выдаёт целевой доступ к памяти по его описателю.*
- $W : \Psi \to \{0, 1\}$ *- доступ к памяти, соответствующий описателю, произошёл (*1 *- истина) либо нет (*0 *- ложь).*
- $V \in \{-1, 0, 1\}$ *- вердикт о гонке по данным:* 0 *- нет гонки по данным,* 1 *- гонка по данным,* -1 *- вердикт ещё не получен.*
- $X \in \mathbb{N}$ *- таймаут ожидания.*

Этап провокации гонки по данным реализуется набором обработчиков событий. Обработчик события доступа к памяти представлен алгоритмом 4, а обработчики остальных событий приведены в алгоритме 3. Предполагается, что обрабатываемые события возникают в параллельных потоках. Семантика параллельного выполнения обработчиков событий сформулирована в замечании 1.

Алгоритмы 3 и 4 стартуют в начальном состоянии программы $< \{ts_0\}, a, w, -1, x >$:

- $ts_0 = \{< \emptyset \to \mathbb{N}, \emptyset \to \Gamma, \{(Target.D_1, 0), (Target.D_2, 0)\}, 0 >\}$ - начальное состояние начального потока программы.
- $a = \{(Target.D_1, < 0, 0, 0, 0, 0, 0 >), (Target.D_2, < 0, 0, 0, 0, 0, 0 >)\}$.
- $w = \{(Target.D_1, 0), (Target.D_2, 0)\}$.
- $x \in \mathbb{N}$ - некоторое натуральное число.

В алгоритме 4 таймаут ожидания моделируется циклом, увеличивающим счётчик до достижения им значения $ps.X$ (строки 15-17). В реализации это может быть что-то иное, например, ожидание по времени либо ожидание события.

С одной стороны, величина $ps.X$ должна быть достаточно большой, чтобы дождаться второго доступа к памяти. С другой стороны, избыточное ожидание неоправданно увеличивает время верификации.

Если всё-таки решено повысить вероятность дождаться второго доступа к памяти, то для этого цикл в строках 15-17 алгоритма 4 можно выполнять немного дольше, чем длительность этапа мониторинга для данного теста. Предположим, что имеется функция, выдающая текущее системное время. Тогда длительность выполнения кода - это разница между двумя значениями системного времени. Обозначим длительность этапа мониторинга для теста i через m_i. Для измерения длительности выполнения цикла возьмём $ps.X = c$, где c - некоторое большое целое, например, $c > 10^7$, а остальные проверки из условия цикла исключим. Получим l -

Алгоритм 3. Обработка событий на этапе провокации гонки по данным

Вход: u - идентификатор текущего потока по определению 8
e - событие по определению 5
$target$ - цель по определению 12
ps - состояние программы по определению 14
Выход: ps - новое состояние программы. Состояние программы параллельно обновляют все потоки. Остальные переменные алгоритма локальные.

1: $ts \in ps.T : equal(ts.U, u)$
2: **if** $e.K == 2$ {Вызов функции} **then**
3: $ts_2 =< ts.U, push(ts.S, e.V), ts.N, ts.C >$ {Добавляем вызов в стек вызовов}
4: $ps =< ps.T \setminus \{ts\} \cup \{ts_2\}, ps.A, ps.W, ps.V, ps.X >$
5: **else if** $e.K == 3$ {Завершение функции} **then**
6: $ts_2 =< ts.U, pop(ts.S), ts.N, ts.C >$ {Удаляем последний вызов из стека}
7: $ps =< ps.T \setminus \{ts\} \cup \{ts_2\}, ps.A, ps.W, ps.V, ps.X >$
8: **else if** $e.K == 4$ {Порождение потока} **then**
9: $u_2 : Dom(u) \cup \{|Dom(u)| + 1\} \to \mathbb{N}$ {Строим идентификатор порождаемого потока}

$$u_2(i) = \begin{cases} \forall i \in Dom(u) & u(i) \\ i = |Dom(u)| + 1 & ts.C + 1 \end{cases}$$

10: $n : \Psi \to \mathbb{N}_0,\ n = \{(target.D_1, 0), (target.D_2, 0)\}$
11: $ts_2 =< ts.U, ts.S, ts.N, ts.C + 1 >$ {Новое состояние текущего потока}
12: $ts_3 =< u_2, \emptyset \to \Gamma, n, 0 >$ {Начальное состояние порождаемого потока}
13: $ps =< ps.T \setminus \{ts\} \cup \{ts_2, ts_3\}, ps.A, ps.W, ps.V, ps.X >$
14: **end if**

нижняя оценка длительности выполнения цикла при $ps.X = c$. Тогда $ps.X = \lceil c * m_i/l \rceil$ для теста i.

Отметим, корректное вычисление l подразумевает учёт особенностей вычислительной системы, например, отключение кэшей процессора на время измерения.

4. Оценка размера трассы программы в худшем случае

Для длительных выполнений программы объём памяти для хранения конфликтующих пар доступов к памяти на этапе анализа трассы и состояния программы на этапе провокации гонок по данным пренебрежимо мал по сравнению с размером трассы программы, собираемой на этапе мониторинга. Поэтому сосредоточимся на оценке размера трассы программы в худшем случае.

Пусть для хранения целого числа достаточно b бит. Будем подсчитывать только размер полезных данных без учёта памяти, потребляемой для организации структуры данных.

Трасса программы - это конечное множество трасс потоков по определению 7. В трассу потока входит три элемента - идентификатор потока по определению 8, целочисленный счётчик количества порождённых потоков и последовательность событий в потоке.

Пусть в выполнении n потоков. Наибольший объём памяти для хранения идентификаторов потоков достигается, когда каждый поток порождает ровно один поток: $n(n-1)b/2$ бит. Для хранения счётчиков количества порождённых потоков, очевидно, достаточно bn бит.

По определению 5 всего 4 типа событий. Из них только 3 необходимо сохранять в трассе программы - это доступ к памяти (1), вызов функции (2) и завершение функции (3). Таким

Алгоритм 4. Обработка события доступа к памяти на этапе провокации гонки по данным

Вход: u - идентификатор текущего потока по определению 8
e - событие по определению 5
$target$ - цель по определению 12
ps - состояние программы по определению 14
Выход: ps - новое состояние программы. Состояние программы параллельно обновляют все потоки. Остальные переменные алгоритма локальные.

1: $ts \in ps.T : equal(ts.U, u)$
$n = ts.N$
2: **if** $e.K == 1$ {Доступ к памяти} **then**
3: **for all** $j \in Dom(target)$ {Кортеж target - это функция $\{1, 2\} \to \Psi$} **do**
4: $d_1 = target(j), d_2 = target(|Dom(target)| - j + 1)$
5: **if** $d_1.I = e.V.I \wedge (d_1.B = 0 \vee equal(d_1.U, ts.U))$
$\wedge\, (Dom(d_1.S) = \emptyset \vee isTop(d_1.S, ts.S))$ {Доступ соответствует описателю d_1} **then**
6: $n_2 : \Psi \to \mathbb{N}_0,\ n_2 = \{(d_1, n(d_1) + 1), (d_2, n(d_2))\}$ {Увеличиваем счётчик}
7: $ts_2 =< ts.U, ts.S, n_2, ts.C >$ {Новое состояние текущего потока}
8: $ps =< ps.T \backslash \{ts\} \cup \{ts_2\}, ps.A, ps.W, ps.V, ps.X >$ {Обновляем трассу программы}
9: **if** $d_1.N = n(d_1) + 1$ {Порядковый номер доступа соответствует d_1} **then**
10: **if** $ps.W(d_1) \neq 1$ **then** {Первый раз обнаружили целевой доступ}
$a : \Psi \to \Lambda, a = \{(d_1, e.V), (d_2, ps.A(d_2)\}$ {Сохраняем целевой доступ}
$w : \Psi \to \{0, 1\}, w = \{(d_1, 1), (d_2, ps.W(d_2)\}$ {Сохраняем, что доступ найден}
$ps =< ps.T, a, w, ps.V, ps.X >$ {Обновляем трассу программы}
else
go to 3 {Проверка доступа на соответствие $target.D_2$ или завершение}
end if
11: **if** $ps.W(d_2) = 1 \wedge conflict(ps.A(d_1), ps.A(d_2)) \wedge ps.V = -1$ {Гонка} **then**
12: $ps =< ps.T, ps.A, ps.W, 1, ps.X >$ {Сохраняем в трассе, что гонка найдена}
go to 22 {Завершение - обнаружена гонка по данным}
13: **end if**
14: $i = 0$ {Счётчик цикла ожидания доступа к памяти, соответствующего d_2}
15: **while** $i < ps.X \wedge ps.V = -1 \wedge$
$(\exists t \in ps.T : \neg equal(t.U, ts.U) \wedge Dom(t.S) \neq \emptyset)$ **do**
16: $i = i + 1$
17: **end while**
18: **if** $ps.V = -1$ **then** $ps =< ps.T, ps.A, ps.W, 0, ps.X >$ **end if** {Гонки нет}
19: **end if**
20: **end if**
21: **end for**
22: **end if**

образом, для хранения типа события с тремя возможными значениями достаточно 2 бита.

Рассмотрим событие доступа к памяти по определению 5. Нет необходимости сохранять тип операции (чтение или запись) и сведения об атомарности в трассу программы для каждого доступа к памяти. Эти данные могут быть извлечены на этапе инструментирования из LLVM IR, ассоциированы с адресом инструкции в машинном коде и далее с доступом к памяти в трассе программы.

Длина сегмента данных, к которому осуществляется доступ, может вычисляться во время вы-

полнения (случай, когда множество доступов к памяти из функций $memset$, $memcpy$, $memove$ и других моделируется одним доступом к памяти). Таким образом, для хранения события доступа к памяти по определению 5 достаточно $3b + 2$ бит на адрес инструкции, адрес данных, длину сегмента данных и тип события.

Для хранения события вызова функции достаточно $2b + 2$ бит на адрес инструкции вызова функции, адрес вызываемой функции и тип события.

В предположении, что событие завершения функции в трассе программы относится к предыдущему событию вызова функции, приходим к заключению, что хранить адрес завершаемой функции нет необходимости. Следовательно, для хранения события завершения функции достаточно 2 бита (тип события).

Итого, для выполнения программы с n потоками, m доступами к памяти и k вызовами (и завершениями) функций в худшем случае достаточно $n(n+1)b/2 + m(3b+2) + k(2b+4)$ бит.

5. Эксперименты с RaceHunter

Инструмент RaceHunter разрабатывался с целью обнаруживать гонки по данным, которые могут быть упущены существующими breakpoint-watchpoint подходами. Полноценное экспериментальное исследование, подтверждающее эффективность RaceHunter, и индустриальное применение инструмента впереди. Цель экспериментов с RaceHunter, выполненных в настоящей работе, заключается в практическом подтверждении способности RaceHunter выявлять гонки по данным.

В качестве целевого программного обеспечения для экспериментов была выбрана операционная система реального времени [16]. Эта операционная система реализует стандарт ARINC 653. В этом стандарте сформулированы требования к программному интерфейсу, который операционная система предоставляет приложениям. В частности, этот программный интерфейс включает набор функций для обмена сообщениями между потоками. В экспериментах в реализацию этих функций вносилась ошибка, приводящая к гонке по данным, и проверялось, что RaceHunter обнаруживает эту гонку по данным.

Для каждого эксперимента был разработан тест. В тесте два потока с одинаковым приоритетом выполняются параллельно разными ядрами процессора. Один поток осуществляет посылку сообщения, а второй неограниченно ожидает прихода сообщения. Тесты были разработаны таким образом, чтобы две инструкции доступа к памяти, образующие гонку по данным, реально выполнялись.

Эксперимент 1. *Объявить глобальную неатомарную переменную целочисленного типа. В реализации функции посылки сообщения записать в переменную значение. В реализации функции приёма сообщения прочитать значение переменной. Доступ к переменной осуществлять без организации взаимного исключения.* ***Результат:*** *обнаружена гонка по данным.*

Эксперимент 2. *В реализации функции приёма сообщения переместить операцию захвата примитива синхронизации так, чтобы чтение одной из разделяемых переменных выполнялось без взаимного исключения.* ***Результат:*** *обнаружена гонка по данным.*

Эксперимент 3. *В реализации функции посылки сообщения переместить операцию захвата примитива синхронизации так, чтобы чтение одной из разделяемых переменных выполнялось без взаимного исключения.* ***Результат:*** *обнаружена гонка по данным.*

В табл. 1 приведены основные показатели статистики верификации при помощи RaceHunter, полученной по результатам экспериментов. Операционная система выполнялась в эмуляторе qemu [17]. Эмулировалась система на кристалле с 4 ядрами по 100 МГц и 128 Мбайт оперативной памяти.

Табл. 1. Статистические показатели верификации (средние значениия)
Table 1. Some statistical indicators of verification (average values)

Величина	**Значение**
Количество потоков в тесте	4
Количество доступов к памяти	696
Количество вызовов функций	129
Количество конфликтующих пар доступов к памяти	50
Объём памяти, потреблённой RaceHunter	$\sim$ 14 Кб
Время выполнения теста без RaceHunter	$\sim$ 2 сек.
Время верификации с RaceHunter	$\sim$ 93 сек.

Отметим, текущая редакция RaceHunter создавалась с целью апробировать новый подход к поиску гонок по данным. Поэтому её можно считать рабочим прототипом. От индустриального инструмента она отличается неоптимальным потреблением памяти и алгоритмами, при разработке которых важнейшим критерием была простота реализации.

6. Сравнение с существующими подходами

RaceHunter - это инструмент динамического анализа, который создавался с целью обнаруживать гонки по-данным, упускаемые существующими breakpoint-watchpoint подходами. Оказалось, RaceHunter способен обнаруживать гонки по данным, упускаемые lock-set и happens-before подходами.

Утверждение 2. *Существуют гонки по данным, которые может пропустить lock-set подход, но обнаруживает RaceHunter.*

Доказательство. Для доказательства достаточно привести пример программы и соответствующей гонки по данным в ней. Программа из листинга 1 содержит гонку по данным по адресу переменной a.

```
int a = 0, b = 0;
mutex ma, mb;

void t1(void) {
    lock(&ma);
    a = 1;
    unlock(&ma);
    lock(&mb);
    b = 1;
    unlock(&mb);
}

void t2(void) {
    lock(&mb);
    if(b == 1) {
        unlock(&mb);
        lock(&ma);
        a = 2;
        unlock(&ma);
    }
    else {
        unlock(&mb);
        a = 3;
    }
}
```

Листинг 1. Псевдокод программы с гонкой по данным
Listing 1. Pseudocode of a program with a data race

Lock-set подход пропускает эту гонку по данным на последовательном выполнении потоков

t_1, t_2, так как в этом случае t_2 выполняет ветку if, в которой доступ к a выполняется с захватом блокировки.

RaceHunter обнаруживает эту гонку на последовательном выполнении t_1, t_2, потому что видит доступ к a из двух потоков на этапе мониторинга и ожидает перед $t_1 : a = 1$ на этапе провокации гонки по данным, что приводит к выполнению в t_2 ветки $else$, в которой доступ к a выполняется без захвата блокировки. □

Утверждение 3. *Существуют гонки по данным, которые может пропустить happens-before подход, но обнаруживает RaceHunter.*

Доказательство. Для доказательства достаточно привести пример программы и соответствующей гонки по данным в ней. Программа из листинга 2 содержит гонку по данным по адресу переменной a.

```
int a = 0;
atomic int b = 0;
atomic int c = 2;

void t1(void) {
    a = 1;
    b = 1;
}

void t2(void) {
    if(b != c)
        a = 2;
}
```

Листинг 2. Псевдокод программы с гонкой по данным
Listing 2. Pseudocode of a program with a data race

Happens-before подход пропускает эту гонку по данным на последовательном выполнении потоков t_1, t_2. На этом выполнении пара операторов $(t_1 : b = 1,\ t_2 : b \neq c) \in happens{-}before$. Также в соответствии с программным порядком $(t_1 : a = 1,\ t_1 : b = 1) \in happens - before \wedge (t_2 : b \neq c,\ t_2 : a = 2) \in happens - before$. Следовательно, на последовательном выполнении потоков t_1, t_2 по транзитивности happens-before $(t_1 : a = 1,\ t_2 : a = 2) \in happens - before$.

RaceHunter обнаруживает эту гонку на последовательном выполнении t_1, t_2, потому что перед $t_1 : a = 1$ успешно дожидается $t_2 : a = 2$. □

Усиление способности выявлять гонки по данным в подходе RaceHunter достигается за счёт увеличения времени верификации (в основном из-за повторных запусков) и неограниченного потребления памяти на этапе мониторинга. Табл. 2 содержит результаты сравнения ключевых характеристик динамических подходов для поиска гонок по данным. Обозначения, используемые в таблице: RH - RaceHunter, HB - happens-before, LS - lock-set, BW - breakpoint-watchpoint.

Табл. 2. Сравнение RaceHunter с существующими динамическими подходами к поиску гонок по данным
Table 2. RaceHunter vs. existing dynamic data race detection techniques

Характеристика	**RH**	**HB**	**LS**	**BW**
Не выдаёт ложных гонок из-за не наблюдаемой синхронизации	+	-	-	+
Не упускает гонки из-за случайных проверок доступов к памяти	+	+	+	-
Потребляет фиксированный объём памяти	-	+	+	+
Не тратит время на повторные выполнения программы	-	+	+	+
Находит гонки, упускаемые другими подходами	+	+	+	+

Можно провести некоторую аналогию между подходом RaceHunter и фаззингом [18—20]:

- Фаззер начинает свою работу с некоторого начального множества входных данных программы. RaceHunter в свою очередь применяется к некоторому множеству выполнений многопоточной программы.
- Фаззер мутирует входные данные программы. RaceHunter в свою очередь провоцирует новые выполнения многопоточной программы.

Однако в отличие от RaceHunter фаззер добавляет некоторые мутированные входные данные к начальному множеству входных данных для получения новых входных данных из них.

Идея двухэтапного динамического метода поиска гонок по данным, в котором в первом выполнении определяются возможные гонки по данным, а в последующих выполнениях эти гонки по данным подтверждаются путём воспроизведения, была ранее сформулирована и реализована в работе [21]. Алгоритм второго этапа в методе [21]:

- Организует некоторый последовательный порядок выполнения инструкций в программе, случайным образом выбирая поток для выполнения очередной инструкции в управляемом планировщике выполнения потоков.
- Идентифицирует доступы к памяти исключительно по оператору программы, который его выполняет. Поэтому приостанавливает каждый поток перед выполнением любого из двух целевых операторов программы (на которых обнаружена гонка по данным на первом этапе метода) до тех пор, пока гонка по данным не будет подтверждена (конфликт с одним из ранее приостановленных доступов к памяти), либо все потоки не остановятся.

В результате действий алгоритма второго этапа, которые не нацелены на воспроизведение выполнения программы, полученного по результатам первого этапа, может получиться новое выполнение программы, на котором гонка по данным, обнаруженная на первом этапе, не воспроизведётся. Однако проверка всех доступов к памяти, выполняемых целевыми операторами программы, может показать большую эффективность, чем подход RaceHunter, в случаях, когда описатель доступа к памяти по определению 11 (или иному) не идентифицирует целевой доступ к памяти в повторном выполнении программы из-за её недерминированного выполнения.

Дополнительно, управление планировщиком потоков на втором этапе метода [21] существенно ухудшает масштабирование метода на длительные выполнения программы либо на большое количество выполнений программы.

Метод [21] изначально разрабатывался для параллельных Java программ с общей памятью (shared memory concurrent Java programs). Позже в работе [22] авторы реализовали схожий метод для параллельных программ с распределённой общей памятью (distributed shared memory concurrent programs).

В методе [22] доступы к памяти также идентифицируются исключительно по оператору программы, который их выполняет. Алгоритм второго этапа отличается от метода [21]. Задачи на вычислительных узлах выполняются свободно за исключением обработки двух целевых операторов доступа к памяти. Обработчик последовательно:

1. Выполняет проверку на гонку по данным с одной из ранее приостановленных задач.
2. Если проверка не подтвердила гонку по данным, то приостанавливает текущую задачу на небольшой период времени. Точнее задача приостанавливается с некоторой вероятностью. Приостанавливая очередную задачу, обработчик понижает вероятность приостановки последующих задач.

Алгоритм второго этапа в методе [22] также может не подтверждать гонки по данным, обнаруженные первым этапом, из-за того, что целевые доступы к памяти идентифицируются не уникально, в выполнение программы вносятся задержки, не нацеленные на воспроизведение выполнения, полученного на первом этапе, и задачи приостанавливаются на целевых

операторах не всегда, а с некоторой вероятностью.

По аналогии с базовым методом [21] метод [22] может показать большую эффективность, чем подход RaceHunter в случае, когда описатель доступа к памяти по определению 11 не идентифицирует целевой доступ к памяти на этапе провокации гонки по данным ввиду недетерминированного выполнения программы. А уменьшение вероятности проверки доступов к памяти на гонки по данным на втором этапе метода [22] нацелено на улучшение масштабируемости метода.

7. Выводы

В настоящей работе предложен новый метод динамического поиска гонок по данным RaceHunter. RaceHunter развивает идею о непосредственном обнаружении гонок по данным при помощи точек останова и наблюдения, ранее реализованную в инструментах DataCollider [6] и KCSAN [7], идеей систематически выявлять и проверять конфликтующие доступы к памяти на гонки по данным.

Этап проверки пар конфликтующих доступов к памяти на гонки по данным (этап провокации гонки по данным) отличается от методов [21] и [22] попыткой идентифицировать целевую пару доступов к памяти в повторном выполнении программы уникальным образом, учитывая однако возможное недетерминированное выполнение многопоточной программы при помощи описателя доступа к памяти.

RaceHunter не выдаёт ложных предупреждений о гонках по данным при наличии не наблюдаемых событий синхронизации, поэтому может применяться к программному обеспечению, в котором используются любые способы синхронизации потоков. Это особенно актуально для системного программного обеспечения, такого как операционные системы и виртуальные машины. В частности, уже имеется опыт применения RaceHunter к операционной системе реального времени [16].

Усиление способности обнаруживать гонки по данным в подходе RaceHunter достигается в том числе за счёт повторных выполнений программы. Следствием этого является увеличение времени верификации.

Кроме того, подход RaceHunter не обеспечивает фиксированное потребление памяти, поэтому не нацелен на очень длительный (в пределе бесконечный) мониторинг выполнения программы. По этой причине RaceHunter более подходит для использования в системах непрерывной интеграции и разработки (Continous Integration and Development) совместно с тестами, которые задействуют параллельное выполнение потоков в программе.

8. Направления будущих исследований

Актуальным представляется экспериментальное исследование эффективности инструмента RaceHunter в сравнении с существующими динамическими подходами к поиску гонок по данным, в особенности с [7] и [22].

В текущей редакции RaceHunter конфликтующие пары доступов к памяти ищутся по определению 10. Применение анализа потоков, подобного [23], анализа блокировок, подобного [5], и др. потенциально способно сократить количество целей для этапа провокации гонок по данным и, следовательно, ускорить верификацию. Для этих видов анализа актуально оценить ускорение верификации и исследовать их влияние на эффективность выявления гонок по данным.

Подходу RaceHunter необходимо повторно выполнять программу на этапе провокации гонок по данным. Актуальным видится исследование применимости в RaceHunter существующих методов воспроизведения выполнения многопоточных программ [24—27].

Список литературы / References

[1]. M. Naik, A. Aiken и J. Whaley. Effective static race detection for java. В *Proceedings of the 27th ACM SIGPLAN Conference on Programming Language Design and Implementation*, страницы 308—319, 2006.

[2]. P. Andrianov и V. Mutilin. Scalable thread-modular approach for data race detection. В *International Workshop on Frontiers in Software Engineering Education*, страницы 371—385. Springer, 2019.

[3]. P. Pratikakis, J. S. Foster и M. Hicks. Locksmith: practical static race detection for c. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 33(1):1—55, 2011.

[4]. K. Serebryany и T. Iskhodzhanov. Threadsanitizer: data race detection in practice. В *Proceedings of the Workshop on Binary Instrumentation and Applications*, страницы 62—71, 2009. DOI: 10.1145/1791194.1791203.

[5]. S. Savage, M. Burrows, G. Nelson, P. Sobalvarro и T. Anderson. Eraser: a dynamic data race detector for multithreaded programs. *ACM Trans. Comput. Syst.*, 15:391—411, 1997. DOI: 10.1145/265924.265927.

[6]. J. Erickson, M. Musuvathi, S. Burckhardt и K. Olynyk. Effective data-race detection for the kernel. В *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, 2010.

[7]. The kernel concurrency sanitizer, The Linux Kernel Organization. URL: https://docs.kernel.org/dev-tools/kcsan.html (дата обращения 26.10.2023).

[8]. L. Lamport. *Time, clocks, and the ordering of events in a distributed system*. 2019, страницы 179—196. DOI: 10.1145/3335772.3335934.

[9]. LLVM language reference manual. URL: https://llvm.org/docs/LangRef.html (дата обращения 20.11.2023).

[10]. Clang: a c language family frontend for llvm. URL: https://clang.llvm.org (дата обращения 20.11.2023).

[11]. K. Serebryany, A. Potapenko, T. Iskhodzhanov и D. Vyukov. Dynamic race detection with llvm compiler: compile-time instrumentation for threadsanitizer. В *International Conference on Runtime Verification*, страницы 110—114. Springer, 2011.

[12]. D. Marino, M. Musuvathi и S. Narayanasamy. Literace: effective sampling for lightweight data-race detection. В *Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation*, страницы 134—143, 2009.

[13]. Information technology Portable Operating System Interface (POSIX ®) Base specifications. Standard, International Organization for Standardization, Geneva, CH, сент. 2009.

[14]. Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE transactions on computers*, 100(9):690—691, 1979.

[15]. L. Lamport. The pluscal algorithm language. В *International Colloquium on Theoretical Aspects of Computing*, страницы 36—60. Springer, 2009.

[16]. V. Cheptsov и A. Khoroshilov. Robust resource partitioning approach for arinc 653 rtos.

[17]. F. Bellard. Qemu, a fast and portable dynamic translator. В *USENIX annual technical conference, FREENIX Track*, том 41, страница 46. Califor-nia, USA, 2005.

[18]. C. Holler, K. Herzig и A. Zeller. Fuzzing with code fragments. В *21st USENIX Security Symposium (USENIX Security 12)*, страницы 445—458, 2012.

[19]. Libfuzzer – a library for coverage-guided fuzz testing. URL: https://llvm.org/docs/LibFuzzer.html (дата обращения 26.10.2023).

[20]. S. Sargsyan, J. Hakobyan, M. Mehrabyan, M. Mishechkin, V. Akozin и S. Kurmangaleev. Isp-fuzzer: extendable fuzzing framework. В *2019 Ivannikov Memorial Workshop (IVMEM)*, страницы 68—71, 2019. DOI: 10.1109/IVMEM.2019.00017.

[21]. K. Sen. Race directed random testing of concurrent programs. В *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*, страницы 11—21, 2008.

[22]. C.-S. Park, K. Sen, P. Hargrove и C. Iancu. Efficient data race detection for distributed memory parallel programs. В *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, страницы 1—12, 2011.

[23]. J. Mellor-Crummey. On-the-fly detection of data races for programs with nested fork-join parallelism. В *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing*, страницы 24—33, 1991.

[24]. Leblanc. Debugging parallel programs with instant replay. *IEEE Transactions on Computers*, 100(4):471—482, 1987.

[25]. P. Dovgalyuk. Deterministic replay of system's execution with multi-target qemu simulator for dynamic analysis and reverse debugging. В *CSMR*, страницы 553—556, 2012.

[26]. R. H. Netzer. Optimal tracing and replay for debugging shared-memory parallel programs. В *Proceedings of the 1993 ACM/ONR workshop on Parallel and distributed debugging*, страницы 1—11, 1993.

[27]. D. F. Bacon и S. C. Goldstein. Hardware-assisted replay of multiprocessor programs. *ACM SIGPLAN Notices*, 26(12):194—206, 1991.

Информация об авторе / Information about author

Евгений Анатольевич ГЕРЛИЦ — научный сотрудник отдела технологий программирования ИСП РАН. Область научных интересов: методы контроля и обеспечения качества программного обеспечения, методы динамической и статической верификации и анализа программ, формальные методы.

Evgeny GERLITS — researcher at the Software Engineering Department of ISP RAS. Main research interests: software quality control and assurance, dynamic and static software verification and analysis, formal methods.

A. Вспомогательные алгоритмы

Алгоритм 5. Проверка, конфликтуют ли два доступа к памяти, в предположении, что они выполняются из разных потоков

Вход: $a_1, a_2 \in \Lambda$

Выход: истина, если доступы к памяти конфликтуют, иначе ложь

function conflict(a_1, a_2)

return $(a_1.W = 1 \vee a_2.W = 1)$
$\wedge (a_1.A = 0 \vee a_2.A = 0)$
$\wedge (a_1.D = a_2.D \vee (a_1.D < a_2.D \wedge a_2.D < a_1.D + a_1.L)$
$\vee (a_2.D < a_1.D \wedge a_1.D < a_2.D + a_2.L))$

end function

Алгоритм 6. Добавление элемента в конец последовательности

Вход: $s : \mathbb{N} \to \Theta$, где Θ - произвольное множество
$v \in \Theta$ - добавляемый элемент

Выход: новая последовательность с добавленным элементом

function push(s, v)

$s_2 : Dom(s) \cup \{|Dom(s)| + 1\} \to \Theta$

$$s_2(k) = \begin{cases} \forall k \in Dom(s) & s_2(k) = s(k) \\ k = |Dom(s)| + 1 & v \end{cases}$$

return s_2

end function

Алгоритм 7. Удаление конечного элемента последовательности

Вход: $s : \mathbb{N} \to \Theta$, где Θ - произвольное множество

Выход: новая последовательность с удалённым элементом

function pop(s)

$s_2 : Dom(s) \setminus \{|Dom(s)|\} \to \Theta, \forall k \in Dom(s) \setminus \{|Dom(s)|\} : s_2(k) = s(k)$

return s_2

end function

Алгоритм 8. Равенство конечных последовательностей натуральных чисел

Вход: $a : \mathbb{N} \to \mathbb{N}$, $b : \mathbb{N} \to \mathbb{N}$

Выход: истина, если последовательности равны, иначе ложь

function equal(a, b)

return $Dom(a) \subset Dom(b) \wedge Dom(b) \subset Dom(a) \wedge \forall i \in Dom(a) : a(i) = b(i)$

end function

Алгоритм 9. Восстановление стека вызовов функций в момент доступа к памяти по трассе программы

Вход: U - идентификатор потока, из которого выполняется доступ к памяти
J - порядковый номер доступа к памяти в трассе потока с идентификатором U
$Trace$ - трасса программы по определению 6
Выход: стек вызовов функций $\mathbb{N} \to \Gamma$

function stack(U, J, Trace)
  $T_U \in Trace : equal(T_U.U, U)$
  $s : \emptyset \to \Gamma$
  return $rstack(T_U.S, J, 1, s)$
end function
function rstack(e, i, j, s)
  if j = i **then**
    return s
  else if e(j).K = 2 {вызов функции} **then**
    return $rstack(e, i, j + 1, push(s, e(j).V))$
  else if e(j).K = 3 {завершение функции} **then**
    $s_2 : Dom(s) \setminus \{|Dom(s)|\} \to \Gamma, \forall k \in Dom(s) \setminus \{|Dom(s)|\} : s_2(k) = s(k)$
    return $rstack(e, i, j + 1, s_2)$
  else
    return $rstack(e, i, j + 1, s)$
  end if
end function

Алгоритм 10. Проверка, является ли данная последовательность вызовов функций верхней частью стека вызовов функций

Вход: $s_1 : \mathbb{N} \to \Gamma$ - последовательность вызовов функций
$s_2 : \mathbb{N} \to \Gamma$ - стек вызовов функций
Выход: истина, если последовательность вызовов функций является верхней частью стека вызовов функций, иначе ложь

function isTop(s_1, s_2)
  return $Dom(s_1) \subset Dom(s_2) \wedge \forall k \in Dom(s_1) :$
  $s_1(k).I = s_2(|Dom(s_2)| - k + 1).I \wedge s_1(k).F = s_2(|Dom(s_2)| - k + 1).F$
end function

Алгоритм 11. Нахождение верхней части стека вызовов функций, которая отличает данный стек от заданного множества стеков

Вход: $s : \mathbb{N} \to \Gamma$ - стек вызовов функций
C - множество стеков вызовов функций
Выход: последовательность вызовов функций (верхняя часть стека вызовов s в обратном поряке)

function top(s, C)
  $s_2 : \emptyset \to \Gamma$
  return $rtop(s, C, s_2, |Dom(s)|)$
end function
function rtop(s_1, C, s_2, i)
  if $i > 0 \wedge \exists s \in C : s(i) = s_1(i)$ **then**
    $s_3 : Dom(s_2) \cup \{|Dom(s_2)| + 1\} \to \Gamma$

$$s_3(k) = \begin{cases} \forall k \in Dom(s_2) & s_2(k) \\ k = |Dom(s_2)| + 1 & s_1(i) \end{cases}$$

    return $rtop(s_1, \{s \in C : s(i) = s_1(i)\}, s_3, i - 1)$
  else
    return s_2
  end if
end function

DOI: 10.15514/ISPRAS-2023-35(6)-9

Классификация текста растрового документа по признаку начертания

[1,2] *Д.Е. Копылов,* ORCID: 0009-0000-6348-4004 <*it-daniil@yandex.ru*>
[1,2] *А.А. Михайлов,* ORCID: 0000-0003-4057-4511 <*mikhailov@icc.ru*>

[1] *Институт динамики систем и теории управления СО РАН,*
664033, Россия, г. Иркутск, ул. Лермонтова, д. 134,
[2] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. При выделении логической структуры документов используются ряд свойств, одним из которых является полужирное начертание слов текста. Полужирным начертанием в документах часто выделяют заголовки, определяемые слова, названия колонок в таблицах. В данной работе предложен метод классификации текста по жирности начертания, который состоит из последовательности шагов. На первом шаге проводится бинаризация всего изображения. Целью данного шага является разделение пикселей изображения на пиксели текста и фона. Вторым шагом проводится оценка каждого слова. В качестве результата возвращается величина, характеризующая толщину основного штриха символа в данном слове. На последнем шаге проводится кластеризация оценок на два кластера: жирный текст и обычный. Предложенный метод был реализован и протестирован на трех наборах данных, исходный код опубликован в открытом репозитории.

Ключевые слова: анализ документов; растровые документы; классификация текста.

Для цитирования: Копылов Д.Е., Михайлов А.А. Классификация текста растрового документа по признаку начертания. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 157–166. DOI: 10.15514/ISPRAS–2023–35(6)–9.

Classification of Printed Text on Raster Documents

[1,2] *D.E. Kopylov,* ORCID: 0009-6348-0000-4004 <*it-daniil@yandex.ru*>
[1,2] *A.A. Mikhailov,* ORCID: 0000-0003-4057-4511 <*mikhailov@icc.ru*>

[1] *Matrosov Institute for System Dynamics and Control Theory*
of the Siberian Branch of the Russian Academy of Sciences,
134, Lermontova st., Irkutsk, 664033, Russia.
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. When highlighting the logical structure of documents, a number of properties are used, one of which is the bold style of text words. In documents, headings, defined words, and column names in tables are often highlighted in bold. This paper proposes a method for classifying text by boldness, which consists of a sequence of steps. The first step is binarization of the entire image. The purpose of this step is to separate the image pixels into text and background pixels. The second step is to evaluate each word. The result is returned a value characterizing the thickness of the main stroke of the character in the given word. At the last step, the ratings are clustered into two clusters: bold text and regular. The proposed method was implemented and tested on three data sets, and the source code was published in an open repository.

Keywords: document analysis, raster documents, text classification.

For citation: Kopylov D.E., Mikhailov A.A. Classification of printed text on raster documents. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 157-166 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-9

1. Введение

Человеческая деятельность неразрывно связана с обработкой документов. Люди ежедневно имеют дело с товарными чеками, инструкциями, брошюрами и т.п. С гораздо большим объемом информации работают предприятия и организации. Процессы, протекающие в них, осуществляются за счет документооборота. К таким документам можно отнести приказы, кадровые документы, счета фактур и т.д.

С развитием компьютеров работать с документами стало гораздо проще. Сейчас компьютер способен обрабатывать автоматически некоторые типы документов, но большую часть до сих пор нельзя обработать без участия человека. Научить компьютер «понимать документы» (document understanding) – это задача, которую пытаются решить многие исследователи. Усилия исследователей направлены на разные стороны этой проблемы, важной из которых является восстановление логической структуры документа.

Для построения логической структуры документа используется различные свойства текста, одним из таких свойств является полужирное начертание (в статье «жирное начертание» будет использоваться как синоним к полужирному начертанию). Жирным начертанием выделяются заголовки или определяемые слова. Часто это одно из самых главных свойств, а порой единственное, по которым можно восстановить структуру документа. При обработке документов в форматах DOC, HTML, XML и т.п. получить информацию о жирности не составляет труда. Это не относится к документам в растровом формате, в которых информация не представлена в явном виде. Говоря о жирном начертании слов, заданных в виде изображения, нужно понимать, что речь идет о толщине символов (конкретно о толщине основных штрихов, которые вносят основной вклад в восприятия символа). Насыщенность символа определяется по основным штрихам и несет полезную информацию, только если есть слова менее насыщенные. Так, например, если текст целиком написан жирным шрифтом, то в таком случае он является основным. На восстановление логической структуры в этом случае полужирное начертание никак не повлияет.

В схожих работах, как правило, классифицируются шрифты в целом. Для этого обычно используются подходы на основе нейронных сетей. Так делают авторы работ, например, [1]–[2]. Для распознавания шрифтов хорошие результаты показывают нейронные сети с архитектурой трансформер.

Цель нашего исследования является классификация текста исключительно по признаку жирного начертания. Использовать упомянутые нейронные сети и из полученной информации брать только начертание будет избыточно сложным. Обучение нейронной сети исключительно для распознавания жирного текста на изображение может осуществляться только для документов, шрифты которых заранее известны (причем жирное начертание одного не похоже на обычное начертание другого).

В связи с этим, для решения задачи выбран подход с использованием математических моделей и статистик. Таким же образом поступают авторы работы [3]. Авторы предлагают разные алгоритмы для определения жирного шрифта, курсива, высоты символа. В работе приводятся несколько примеров работы разработанных алгоритмов, но отсутствует этап тестирования. Помимо этой работы, нам не удалось найти исследования, содержащие алгоритмы и их тестирование.

2. Предлагаемая схема работы метода

Нами была предложена схема работы подхода представленная на рис. 1. Предложенный метод был реализован на языке Python и опубликован в открытом репозитории исходных кодов[1]. Далее кратко будет передана идея работы метода, а в подпунктах 2.1 – 2.3 подробно будут описаны ключевые шаги метода.

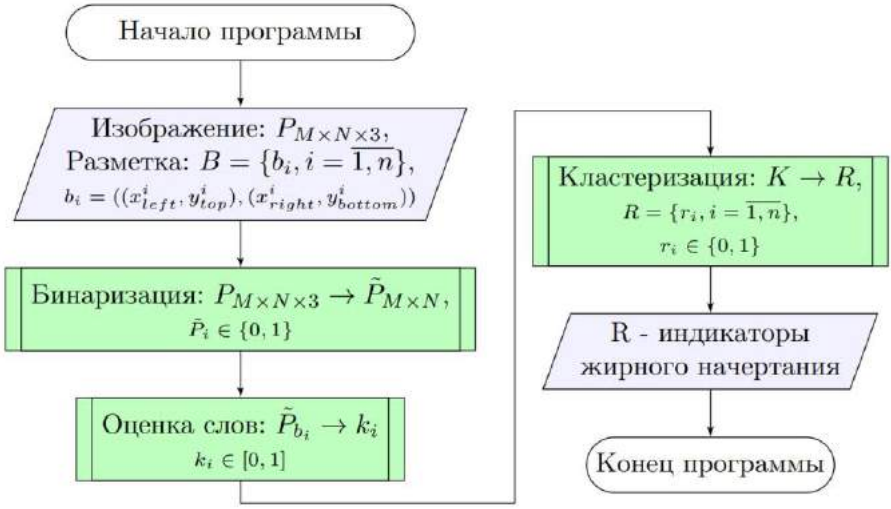


Рис. 1. Схема работы метода
Fig. 1. Scheme of method

На вход подается цветное изображение и координаты слов. На выходе алгоритм возвращает набор индикаторов, характеризующих жирный шрифт это или нет.

На первом шаге работы проводится бинаризация всего изображения. Целью данного шага является разделение пикселей изображения на пиксели текста и фона.

Вторым шагом проводится оценка каждого слова. В качестве результата возвращается величина, характеризующая толщину основного штриха символа в данном слове.

На последнем шаге проводится кластеризация оценок на два кластера: жирный текст и обычный.

2.1 Бинаризация

Для определения размеров требуется знать границу объекта, это относится и к толщине штриха. В процессе оценки так или иначе будет возникать вопрос о принадлежности пикселя к фону или символу. Для этого используется бинаризация, после которой становится ясно, где находятся границы символа. Неправильное выставление порога для бинаризации может привести к ухудшению изображения. Так, например, при высоком пороге бинаризации велик шанс сделать обычный текст неотличимым от жирного. Также стоит помнить, что фон на изображениях не обязательно будет абсолютно белым, а текст абсолютно черным. Исходя из вышесказанного, стоит отдавать предпочтение адаптивной бинаризации.

При работе с текстом распространена бинаризация Otsu [4], которая минимизирует взвешенную дисперсию двух классов (оттенков пикселей текста и пикселей фона). В нашей схеме используется адаптивная бинаризация «выделение впадины» [5]. В отличие от бинаризации Otsu, она берет в учет «впадину», возникающую в спектре изображения. В бимодальных распределениях один из пиков может иметь вес в несколько раз больше, чем

[1] https://github.com/Dann38/bold_classifier

другой. В таком случае бинаризация Otsu выставит порог, ближе к пику с наибольшим весом. Такой эффект возникает, например, при идеальном белом фоне и размытом тексте. Обрабатывая такой текст с помощью бинаризации Otsu, получаем весь текст похожий на жирное начертание. В случае бинаризации методом «выделения впадины» порог сдвигается в сторону «впадины» на графике спектра изображения, что избавляет от этого неприятного эффекта. Для бинаризации «выделение впадины» формула расчета порога T выглядит следующим образом:

$$T = argmax_{t\in(0,255)}\omega_3[\omega_1(t)\mu_1^2(t) + \omega_2(t)\mu_2^2(t)],$$

где $\omega_1(t), \omega_2(t)$ – вес темных и светлых пикселей, $\mu_1(t), \mu_2(t)$ – математическое ожидание темных и светлых пикселей, $\omega_3(t)$ – вес впадины. Для впадины дополнительно необходимо указать радиус. В предлагаемой схеме используется радиус равный 5. На данных, которые приведены в разделе тестирование, такой радиус показал себя лучше всего. Вопрос выбора бинаризации и ее параметров достоин отдельного исследования.

2.2 Оценка жирности начертания слова

Для оценки жирности начертания в данной работе используются статистические величины и их комбинации. Негативное влияния на статистические оценки оказывает шум и выбросы. К таким шумам можно отнести белое пространство вне области строчных букв – это область между верхними и нижними выносными элементами символов. С точки зрения математики, бинаризованное изображение — это матрица, состоящая из 0 и 1. Необходимо найти номер строки матрицы, где заканчиваются верхние выносные элементы, а также номер, где начинаются нижние выносные элементы. Сделать это можно, используя следующие формулы:

$$i_{top} = argmax_{i=\overline{1,m}}(p_{i-1} - p_i), i_{bottom} = argmax_{i=\overline{1,m}}(p_{i+1} - p_i),$$

$$p_i = mean_i(P),$$

где P – матрица бинаризованного изображения слова с числом строк равным m , $mean_i(\cdot)$– среднее значение i-й строки матрицы.

Недостатком данных подходов могут служить акронимы, цифры и текст, написанный в верхнем регистре. При этом стоит отметить, что маловероятно, что акроним из 2–5 букв будет являться, например, отдельно стоящим заголовком.

Другим примером шума служат пробелы между буквами. Для их удаления, рассчитывается среднее значение для каждого столбца матрицы изображения, а затем удаляются те, где среднее близко к единице. Под близостью к единице понимается значение 0.95. Для символов “т”, “г” значение порога критично, так как верхняя часть этих символов может быть достаточно мала, и будет признана пробелом. Среднее значение в этой области находится в диапазоне 0.90-0.95. По этой причине и выбирается значение 0.95.

Говоря непосредственно о способах оценки и статистических характеристиках с ними связанных, можно выделить три подхода:

1) Среднее значение интенсивности пикселей;
2) Медианное значение интенсивности пикселей;
3) Отношение периметра к площади символов.

Достаточно простым является первый вариант. Чем ниже оценка, тем более насыщенное слово. Чем более насыщенное слово, там вероятнее оно имеет жирное начертание. Недостатком такого подхода является то, что некоторые символы являются более насыщенными, чем другие.

Более устойчивой к таким выбросам является медиана, лежащая в основе второго подхода.

В третьем подходе используется идея, что оценить насыщенность слова можно как отношение двух характеристик, а именно:

- периметр символов (контур),
- площади символов.

В третьем подходе насыщенность измеряется как среднее значение, но при этом в учет берется форма символов. Учет формы обусловлен тем, что для прорисовки букв используется разное число пикселей. Так, например, для буквы «г» нужно меньше темных пикселей, чем для буквы «в». Оценка находится по следующей формуле:

$$w = \frac{c}{s},$$

где c – характеристика контура, s – характеристика площади.

Для удобства и эффективности вычислений эти оценки заменяются их эквивалентами. Для оценки площади используется оценка:

$$s = 1 - mean(P), P = \{p_{ij}\}$$

а для периметра оценка:

$$c = mean(dP), dP = \{p_{i+1,j} - p_{ij}\},$$

где P – матрица изображения слова, p_{ij} – элемент, стоящий на пересечении i-й строки и j-го столбца, dP – аналог изображения контура, $mean(\cdot)$– среднее значение.

При изменении начертания слова, периметр у букв изменяется непропорционально площади. В таком случае у полужирного начертания по сравнению с обычным значение w будет ниже. При прочих равных условиях данный подход показывает себя лучше, чем оценка среднего значения или поиск медианы в выборке пикселей изображений (рис. 2).

2.3 Кластеризация

После получения оценок последним шагом метода является разбиение слов на кластеры. Основной проблемой при проведении кластеризации является определения числа кластеров. В работе делается допущение, что на странице присутствует только два типа начертания. Свойство курсивного начертания не учитывается. Наличие промежуточной насыщенности также исключается, так как является неоправданно сложным. Начертание формул и рукописного текста на документе считается неизвестным и не учитывается при оценке подходов. Таким образом, количество кластеров не может превышать двух. Возможность наличия всего одного кластера обусловлена тем, что весь текст может быть написан обычным начертанием. Для определения числа кластеров проверяется гипотеза об однородности выборки. В данной работе используется критерий Duda и Hart [6].

Для оценки определяется значение

$$F_{\frac{2}{1}} = \frac{w_2}{w_1},$$

где w_1 – сумма квадратов внутрикластерного расстояния между двумя классами, w_2– сумма квадратов отклонения от среднего значения всех оценок. Это значение сравнивается с:

$$F_{\text{кр}} = 1 - \frac{2}{\pi \cdot p} - z_{(1-\alpha)} \cdot \sqrt{2 \cdot \frac{1 - \frac{8}{\pi^2 \cdot p}}{n \cdot p}}$$

где p – размерность кластеризуемого пространства, $z_{1-\alpha}$ – квантиль нормального распределения, n – число кластеризуемых объектов. При $F_{\text{кр}} < F_{\frac{2}{1}}$ отвергается гипотеза о

присутствии двух кластеров, а значит, нет оснований утверждать, что на изображении присутствует жирный шрифт.

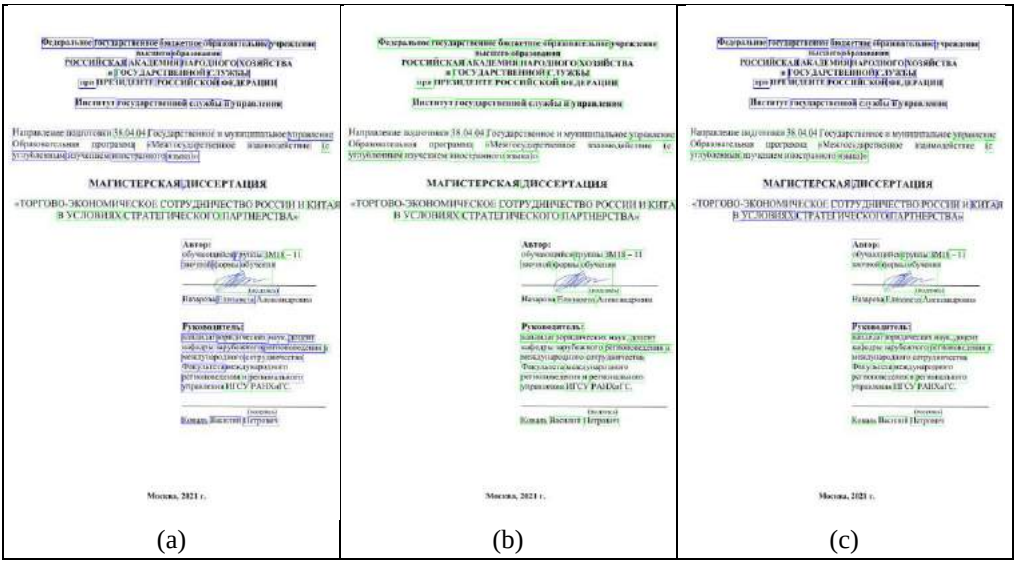


Рис. 2. Результат работы классификатора (зеленым – обычный шрифт, синим – жирный) с использованием в качестве оценки
(a) – среднего значения, (b) – медианного значения, (c) – отношение периметра к площади
Fig. 2. The result of the classifier (green – regular, blue – bold) using as an estimate
(a) – the average value, (b) – the median value, (c) – the ratio of the perimeter to the area

В качестве параметра для описанного выше критерия нужно задать уровень значимости α (вероятность того, что будет отвергнута гипотеза, хотя она верна). В работе выбирается $\alpha = 0.15$. В рамках данной работы параметр подробно не изучался, но при таком значении α чаще будет не найден жирный текст, чем обычный ошибочно выделен жирным.

В работе рассматриваются три метода кластеризации:

- Кластеризация k-средних;
- Спектральная кластеризация;
- Агломеративная кластеризация.

Первые две кластеризации являются достаточно распространёнными [7]. Третий вид кластеризации менее известен. Агломеративная кластеризация выигрывает по сравнению со спектральной и кластеризацией k-средних за счет того, что изначально каждая оценка слова — это отдельный кластер. На каждом следующем шаге объединяются те кластеры, которые ближе друг к другу. На последнем шаге остаются два кластера, которые имеют расстояние между друг другом больше, чем между любым элементом из кластера и его ближайшим соседом. Учитывая, что слова состоят из символов с разной насыщенностью, начинать объединять их лучше по близости оценок.

Кластеризация проводится над векторами, что позволяет помимо оценки текущего слова добавить информацию об оценках предыдущего и следующего слова. Шанс того, что слово окажется написано жирным начертанием выше, если вокруг него слова написаны жирным начертанием. Каждому слову ставится в соответствие вектор:

$$x_i = (k_i, \frac{k_{i-1}+k_i+k_{i+1}}{3}), k_0 = k_1, k_{n+1} = k_n, i = 1, n,$$

где k_i – оценка i-слова, k_{i-1} – оценка предшествующего слова, k_{i+1} – оценка следующего слова.

3. Тестирование

Для тестирования использовались три набора данных, размеченных вручную.

- Первый набор называется «ВКР», он состоит из 30 изображений в следующей пропорции: 5 изображений со сложным текстом (титульные листы и листы содержания), 20 изображений с обычным текстом на которых присутствует текст с жирным начертанием, 5 изображений, не содержащие жирный шрифт. Общее число слов 5420. Число слов жирным начертанием 410 (7.6% от всего текста)
- Второй набор называется «ГОСТ». Он состоит из 30 изображений, полученных при конвертировании PDF ГОСТа в изображение. Особенностью данного набора является то, что изображение имеет более низкое разрешение по сравнению с «ВКР», также кегель текста более мелкий. Общее количество слов 10769. Число слов жирным начертанием 685 (6.3% от всего текста).
- Третий набор данных называется «СКАН». Он состоит из 20 изображений, полученных как сканы одного из учебников. Особенностью таких данных является то, что в них фон не абсолютно белый и текст не абсолютно черный. Также присутствуют небольшие искажение слов. Общее количество слов 5055. Число слов жирным начертанием 407 (7.8% от всего текста).

При тестировании использовалась F_1-мера, являющаяся классической оценкой в анализе данных. Расчет проводился для жирного начертания. Дополнительно был рассчитан показатель точности $Accuracy$ для того, чтобы показать правильность определения значений в целом.

Как было описано ранее, метод оценки слов по жирности начертания состоит из подпрограмм. Ключевым шагом является оценка слов. В табл. 1 приведены сравнения трех вариантов оценок в зависимости от разных данных и кластеризации. Другой важный шаг, это кластеризация слов по оценкам. Сравнение разных методов кластеризаций приведено в табл. 2.

По табл. 2 можно сделать вывод, что по сравнению с представленными подходами, оценка жирности как отношение периметра к площади показал себя лучше в независимости от вида данных, и метода кластеризации.

В табл. 3 приведены значения Accuracy при оценке отношения периметра к площади. Из табл. 3 видно, что нельзя сказать, что один из методов кластеризации превосходит остальные. Для ВКР показала себя лучше агломеративная кластеризация. Эти документы были представлены в хорошем качестве. Для наборов данных СКАН и ГОСТ все не так однозначно.

Если сравнивать только спектральную кластеризацию и кластеризацию k-средних, то стоит отдать предпочтение кластеризации k-средних. Выбор между агломеративной кластеризацией и кластеризацией k-средних будет зависеть от типа документов.

Табл. 2. Оценки f1 и accuracy для трех видов оценок в зависимости от набора данных и метода кластеризации
Table 2. F1 and accuracy estimates for three types of estimates depending on the data set and clustering method

Датасет	**Кластеризация**	**Отношение периметра к площади**	**Среднее значение**	**Медианное значение**

		f1	accuracy	f1	accuracy	f1	accuracy
ВКР	**Агломеративная**	**0.72**	**0.95**	0.07	0.57	0.04	0.66
	k-средних	**0.67**	**0.93**	0.10	0.46	0.08	0.60
ГОСТ	**Агломеративная**	**0.09**	**0.83**	0.11	0.28	0.11	0.35
	k-средних	**0.20**	**0.84**	0.11	0.28	0.11	0.30

Табл. 3. Оценка показателя точности "accuracy" для трех методов кластеризации в зависимости от набора данных при использование оценку по отношению площади к периметру
Table 3. Accuracy estimation for three clustering methods depending on the data set when using an estimate with respect to area to perimeter

Наборы данных	Агломеративная кластеризация	Спектральная кластеризация	Кластеризация k-средних
ВКР	**0.95**	0.86	0.93
СКАН	0.86	0.83	**0.91**
ГОСТ	0.83	**0.84**	**0.84**

4. Заключение

В данной работе был предложен метод решения задачи классификации строк по признаку жирности начертания. Процесс классификации разбивается на этапы: предобработка изображения, оценка изображений слов и кластеризация слов по полученным оценкам.

В процессе предобработки изображения документа важным оказывается избавление от различных шумов и сохранение насыщенности символов. При оценке насыщенности слова необходимо помнить, что символ состоит из штрихов. Не все штрихи при изменении начертания могут меняться. В случае оценки штрихов необходимо делать акцент на основные из них. Способ оценки с учетом формы символов из всех предложенных в работе является наилучшим. По накопившемся оценкам слов, кластеризация слов может осуществляться с использованием разных методов. Для документов в хорошем качестве рекомендуется агломеративная кластеризация. В случае плохого качества стоит отдать предпочтение кластеризации k-средних.

Используя представленный подход, можно находить в тексте слова, написанные полужирным начертанием. Обладая этой информацией, упрощается процесс восстановления логической структуры документа. Таким способом можно находить заголовки, структуры ключ-значение, заголовки столбцов в таблицах и другие компоненты.

Список литературы / References

[1]. Sandy I.C., Voinea D., Popa A.I. CONTENT: Context Sensitive Transformer for Bold Words Classification. arXiv:2205.07683.

[2]. Bychkov O., Merkulova K., Dimitrov G., Zhabska Y., Kostadinova I., Petrova P., Petrov P., Getova I., Panayotova G. Using Neural Networks Application for the Font Recognition Task Solution. In Proc. of 55th International Scientific Conference on ICEST, 2020. pp. 167-170. doi: 10.1109/ICEST49890.2020.9232788.

[3]. Ladareanu L., Chiroiu V., Bratu, P., Magheti, I. Automatic Text Clustering and Classification Based on Font Geometrical Characteristics. In Proc. of 9th WSEAS International Conference on Automation and Information, 2008, pp. 468-473.
[4]. Otsu N. A threshold selection method from gray-level histograms // IEEE Trans. Sys., Man., Cyber. : journal. — 1979. — Vol. 9. — P. 62—66.
[5]. Xing J., Yang P., Qingge L. Automatic thresholding using a modified valley emphasis. IET Image Processing, vol. 14(3), 2020, pp. 536-544. doi: 10.1049/iet-ipr.2019.0176
[6]. Яцкив И., Гусарова Л. Методы определения количества кластеров при классификации без обучения. The Journal of Transport and Telecommunication Institute, vol. 4(1), 2003. pp. 23-28.
[7]. Бурков А. Машинное обучение без лишних слов. Санкт-Петербург, Питер, 2020, 192 с.

Информация об авторах / Information about authors

Даниил Евгеньевич КОПЫЛОВ – магистрант направления подготовки «Прикладная математика и информатика» Иркутского государственного университета, сотрудник Институт системного программирования им. В.П. Иванникова Российской академии наук, сотрудник Института динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук. Сфера научных интересов: прикладная математика, анализ данных.

Daniil Evgenievich KOPYLOV is master’s student of Irkutsk State University, employee of Ivannikov Institute for System Programming of the Russian Academy of Sciences, employee of Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences. Research interests: applied mathematics, data analysis.

Андрей Анатольевич МИХАЙЛОВ является старшим научным сотрудником лаборатории Комплексных информационных систем Института динамики систем и теории управления имени В.М. Матросова. Его научные интересы включают анализ электронных документов, распознавание образов.

Andrey Anatolievitch MIKHAYLOV is a senior researcher of the Laboratory of information systems of Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences. His research interests include document analysis, image recognition.

DOI: 10.15514/ISPRAS-2023-35(6)-10

Извлечение опорных тестовых наборов из спецификаций криптопротоколов на предметно-ориентированном языке

С.Е. Прокопьев, ORCID: 0000-0002-4410-2565 <*sepr@ispras.ru*>

Институт системного программирования РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
АО «НПК «Криптонит»,
Россия, 115114, Москва, Шлюзовая набережная, 4

Аннотация. Работа посвящена описанию инструмента тестирования безопасности реализаций криптографических протоколов, работающего на основе спецификаций, написанных на декларативном интероперабельном встроенном предметно-ориентированном языке (emdedded [in Haskell] DSL). Рассмотрена задача формирования качественных опорных тестовых наборов для использования в тестировании безопасности реализаций протоколов. Предложен метод решения этой задачи в контексте представленного инструмента.

Ключевые слова: криптографические протоколы; формальные спецификации криптографических протоколов; тестирование криптографических протоколов; тестирование на основе формальных спецификаций; EDSL; виртуальная машина SBSM/C2.

Для цитирования: Прокопьев С.Е. Извлечение опорных тестовых наборов из спецификаций криптопротоколов на предметно-ориентированном языке. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 167–178. DOI: 10.15514/ISPRAS-2023-35(6)-10.

Благодарности: Исследование поддержано грантом Минобрнауки России № 075-15-2020-788.

Extracting the Reference Test Suites from Cryptographic Protocol Specifications Written on a Domain-Specific Language

S.E. Prokopev, ORCID: 0000-0002-4410-2565 <sepr@ispras.ru>

Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
JSC "NPK Kryptonite",
4, Shluzovaya emb., Moscow, 115114, Russia

Abstract. The paper describes a tool for testing the security of cryptographic protocol implementations working on the basis of specifications written in a declarative interoperable domain-specific language implemented as EDSL (Embedded [in Haskell] DSL). The problem of forming high-quality reference test suites for testing the security of cryptoprotocol implementations is considered. A method of addressing this problem within the tool being developed is discussed.

Keywords: cryptographic protocol specifications, cryptographic protocol testing, specification-based testing, DSL embedded in Haskell.

For citation: Prokopev S.E. Extracting reference test suites from cryptographic protocol specifications written on a domain-specific language. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 167-178 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-10.

Acknowledgements. The work is supported by a grant from the Ministry of Education and Science of Russia.

1. Введение

Защита взаимодействия компонентов распределенных систем по каналам связи с использованием криптографических протоколов подразумевает проведение комплекса работ, связанных с разработкой спецификаций практических протоколов, оформляемых в виде документов типа RFC, а также анализом конформности, безопасности и совместимости программных реализаций протоколов. В рамках данных работ можно выделить ряд проблемных моментов:

- низкая степень формализации спецификаций протоколов (формально описываются только форматы сообщений и некоторые вычисления, при этом используемый формальный язык не является машинно-интерпретируемым);
- отсутствие признанных эффективных подходов к тестированию реализаций криптопротоколов, как в части проверки конформности, так и в части проверки безопасности (из-за сложных зависимостей сообщений протокола от всей предыстории выполнения протокола, реализации криптопротоколов являются очень неудобными объектами для стандартных промышленных фаззеров, основанных как на грамматиках (например, Peach), так и на использовании обратной связи по покрытию ветвей вычислений (например, фаззеры семейства AFL));
- отсутствие признанных эффективных методик проверки совместимости реализаций криптопротоколов от разных разработчиков.

Под эффективностью тестирования понимается выполнение требований, перечисленных в табл. 1 (для справки в таблице также даны оценки по трехбалльной шкале для ряда инструментов, построенных по сильно отличающимся друг от друга технологиям: tlsfuzzer (tf) [1], tlspuffin (tp) [2], UniTESK (un) [3] и NCT (nc) [4]).

Табл.1. Требования к инструментам тестирования реализаций протоколов
Table 1. Protocol testing tool requirements

	Требование	tf	tp	un	nc
1	Способность автоматически генерировать неожиданные неконформные сообщения	–	+	–	–
2	Способность автоматически генерировать неожиданные конформные сообщения	–	+	–	+
3	Способность генерировать неожиданные сообщения на уровне логики протокола	–	=	–	+
4	Способность различать сгенерированные конформные и неконформные сообщения	–	–	+	+
5	Наличие адекватного критерия качества тестирования	–	–	+	–
6	Прослеживаемость результатов тестирования к проверкам свойств безопасности из явно заданного перечня	+	–	+	+
7	Простота адаптации инструмента к новым протоколам	–	–	–	=
8	Скорость работы инструмента	+	+	+	–

Генерация неожиданных сообщений необходима для тестирования безопасности реализаций, при этом важно уметь генерировать сообщения, неожиданные не только на уровне мутаций на битовом уровне, но и на уровне логики протокола.

Способность отличать конформные сообщения от неконформных необходима для получения сильного тестового оракула: в ответ на отправку конформного сообщения инструмент ожидает нормальные ответы, а на отправку неконформного – сообщение об ошибке или разрыв соединения.

Критерий качества тестирования должен оценивать способность инструмента проникать в разнообразные (включая глубокие) состояния протокола, а также степень разнообразия генерируемых сообщений с точки зрения используемых сочетаний алгоритмов криптографических примитивов.

Прослеживаемость к заданному перечню свойств безопасности означает, что по результатам тестирования должно быть ясно, какие именно типы ошибок реализации искались.

Простота адаптации к новым протоколам означает, что с каждым новым протоколом трудоемкость проведения тестирования с использованием инструмента резко уменьшается.

Скорость работы инструмента измеряется как число выполненных тестов в секунду на одно ядро процессора (при наличии адекватного критерия качества тестирования данный критерий рассматривается как второстепенный).

Выполнение требований 4 и 5 из табл. 1 возможно только в рамках подхода к тестированию на базе выразительных формальных моделей, например, таких как расширенные автоматы (рис. 1).

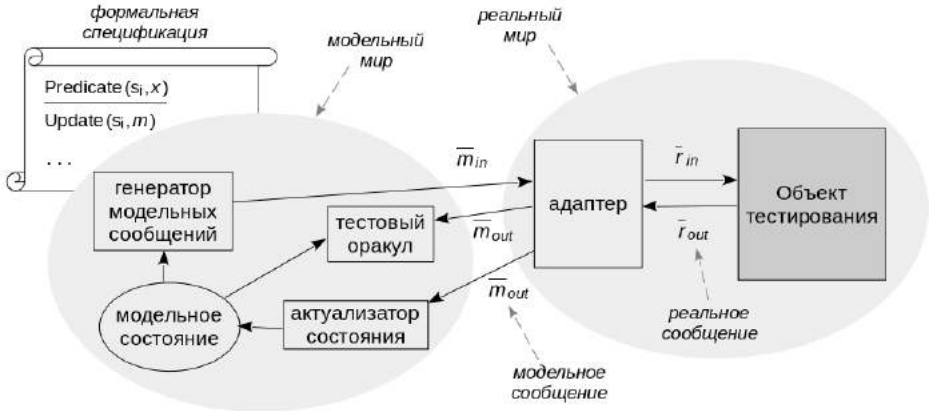


Рис. 1. Тестирование протоколов на основе расширенного автомата
Fig. 1. Protocol implementation testing based on an extended state machine

Под расширенными автоматами понимается семейство концептуально родственных друг другу автоматов с бесконечным числом состояний [5]. Упрощенно, модель протокола на базе расширенного автомата состоит из множества пар Predicate(s_i,x) / Update(s_i,m), где s_i – это состояние автомата, m = (src, dst, a_1, …, a_n) – это вектор полей модельного сообщения, включая адресата и отправителя сообщения.

Множество конформных сообщений протокола в состоянии s_i задается как множество всех векторов (src, dst, a_1, …, a_n) обращающих Predicate(s_i,x) в истину. В рамках методологии UniTESK извлечение этих решений проводится с помощью сценариев, которые необходимо дополнительно разрабатывать и прикладывать к спецификации протокола. В методе NCT решения извлекает SMT-решатель, что является медленной процедурой.

Основной проблемой здесь является сложность реализации в схеме на рис. 1 актуализатора состояния и адаптера: фактически для каждого протокола необходимо разработать

отдельную реализацию, совместимую с тестируемой реализацией на бинарном уровне, что противоречит требованию 7 из табл. 1.

Ранее в работах [6-7] автором был предложен высокоуровневый язык формальных спецификаций криптопротоколов – лаконичный, декларативный и выразительный. Язык основан на расширенном автомате SBSM, специально разработанном для предметной области криптографических протоколов, и реализован как встроенный предметно-ориентированный язык – то есть как EDSL (Embedded DSL), где в качестве хост-языка использован язык Haskell. Предложенный EDSL является интероперабельным, то есть совместимым с промышленными реализациями протоколов на уровне бинарных сообщений. Семантика языка задана на основе виртуальной машины SBSM/C2. Машина SBSM/C2 реализована на Haskell и обеспечивает возможность машинной интерпретации спецификаций на EDSL.

Так как язык спецификаций интероперабелен, то адаптер в схеме на рис. 1 не нужен. Кроме того, в реализацию виртуальной машины SBSM/C2 встроены функции автоматической актуализации модельного состояния.

В отличие от промышленных языков разработки, предложенный формальный язык спецификаций нацелен на максимально полное описание протокола: на описание всех разрешенных в RFC вариантов формирования сообщений протокола и их последовательностей. При этом ввиду того, что множество всех возможных траекторий выполнения протокола, задаваемых спецификацией, для промышленного криптопротокола (например, такого как TLS) огромно (и даже бесконечно), возникает задача автоматического извлечения из формальной спецификации небольшого по мощности, но качественного подмножества тестируемых траекторий.

2. Постановка задачи извлечения тестовых наборов из формальной спецификации на EDSL

Постановку задачи будем рассматривать на примере протокола TLS.

В записи протокола TLS (рис. 2) выделим три типа полей: информационные поля (белые прямоугольники), неинформационные поля (черные прямоугольники; это либо поля-константы, либо поля, которые автоматически вычисляются из других полей) и поля-последовательности (серые прямоугольники с пунктирными разделителями).

Рис. 2. Структура сообщений (записей) протокола TLS
Fig. 2. The structure of the records of the TLS protocol

Если взять сообщение ClientHello последовательности Handshake (рис. 3), то оно также содержит поля-последовательности: последовательность идентификаторов шифрнаборов и последовательность расширений сообщения ClientHello. В свою очередь, расширения сообщения ClientHello также могут включать в свой состав поля-последовательности, например, расширение Groups содержит последовательность двухбайтовых идентификаторов групп точек эллиптической кривой (рис. 4).

Фактически, структурное разнообразие конформных сообщений протокола TLS представляет собой разнообразие способов формирования последовательностей, входящих в состав сообщений: какие элементы и в каком порядке будут включены в каждую последовательность. И это в целом характерно для современных промышленных криптопротоколов. Например, в протоколе IPSEC IKEv2 можно обнаружить следующие

последовательности: payloads, SA proposals, transforms, configurations attributes, traffic selectors и др.

В EDSL при кодировании структур сообщений каждая такая последовательность описывается с помощью оператора Sequence, и этой последовательности назначается имя (рис. 5).

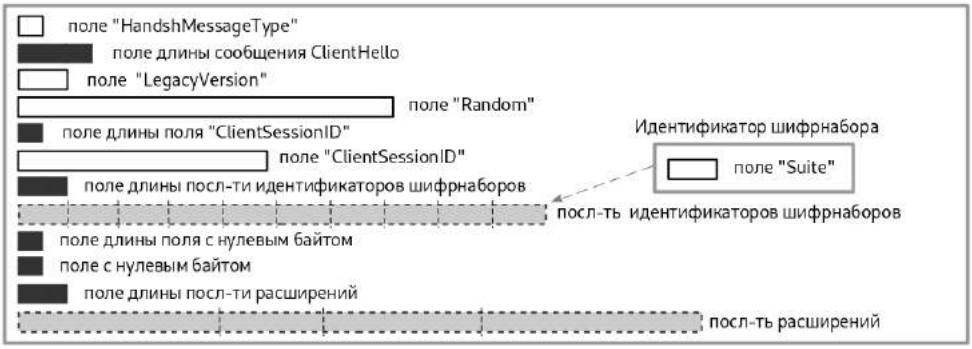


Рис. 3. Структура сообщения ClientHello подпротокола TLS Handshake
Fig. 3. The structure of the TLS ClientHello message

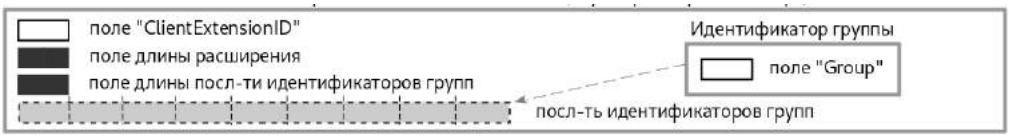


Рис. 4. Структура расширения Groups сообщения ClientHello
Fig. 4. The structure of the Groups extension of the TLS ClientHello message

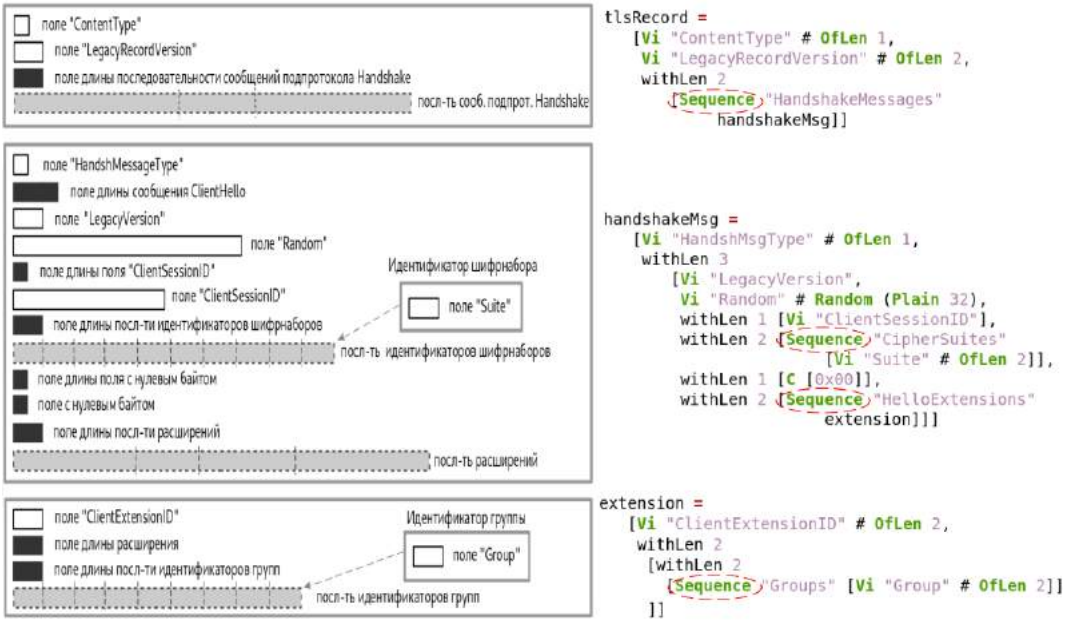


Рис. 5. Кодирование полей и последовательностей сообщений в EDSL
Fig. 5. Description of message fields and sequences in EDSL

С каждой последовательностью в спецификации ассоциировано дерево вариантов формирования элемента этой последовательности (рис. 6). Каждое ребро этого дерева

помечено парой Guard/Assign, где Guard – это гард (предикат), заданный на текущем состоянии автомата, Assign – список значений, назначаемых информационным полям формируемого элемента поля-последовательности. Проход по некоторому пути до конца дерева соответствует формированию одного элемента последовательности (при проходе будут присвоены значения всем информационным полям этого элемента). Путь в дереве может оканчиваться на Next (означает, что после формирования текущего элемента последовательности следует приступить к формированию следующего элемента), на Last (текущий формируемый элемент является последним) или на End (окончание формирования последовательности без формирования текущего элемента).

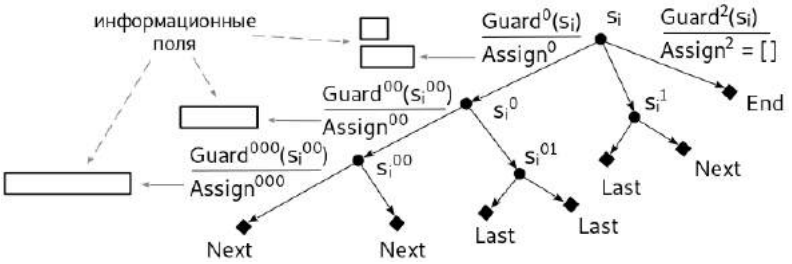


Рис. 6. Дерево вариантов формирования элемента последовательности
Fig. 6. Tree of variants of the sequence elements

При формировании конформных сообщений протокола тестировщику необходимо выбирать те пути, на которых все гарды обращаются в истину, а при формировании неконформных – все остальные. Таким образом, тестировщик всегда знает, какое сообщение он сформировал – конформное или неконформное.

Язык спецификаций предоставляет возможность гибкого управления выбором очередных элементов последовательности. Например, гард

```
ExistsIn ClientExtensions (Vi "ClientExtensionID" == [0x00,0x0a]) IsCurSess &&
Not (ExistsIn ClientExtensions (Vi "ClientExtensionID" == [0x00,0x33]) IsCurSess)
```

задает следующее условие: «в текущей сессии протокола сформированная ранее часть последовательности ClientExtensions содержит элемент, в котором информационному "ClientExtensionID" было присвоено значение [0x00,0x0a], и не содержит элементов, в которых информационному полю "ClientExtensionID" присвоено значение [0x00,0x33]».

Все возможные сочетания вариантов формирования последовательностей, содержащихся в сообщении TLS ClientHello, приводят к комбинаторному взрыву числа способов построения сообщения ClientHello (рис. 7).

На рис. 7 в виде $tpath_{seqName}^{x}$ обозначается путь с номером x в дереве вариантов формирования элемента последовательности с именем seqName (очевидно, что все такие *tpath-пути* в этом дереве можно перенумеровать). И это только одно сообщение протокола. Если составить из деревьев отдельных сообщений общее дерево траекторий протокола (рис. 8), то даже если ограничить число сессий протокола (например, тремя) и число сообщений в каждой сессии (например, двадцатью), то все равно обойти за разумное время все пути в этом дереве траекторий невозможно.

На рис. 8 тройка (clnt,srvr,msg) означает отправку участником clnt участнику srvr сообщения msg, сформированного в результате прохождения по некоторой последовательности tpath-путей.

Данное дерево траекторий задано неявно – это результат гипотетической работы автомата SBSM, моделирующего протокол, в недетерминированном режиме (то есть, когда в точках выбора вариантов продолжения траекторий автомат идет параллельно по всем путям). Вид этого дерева заранее не известен, вершины и помеченные ребра возникают динамически в процессе работы автомата.

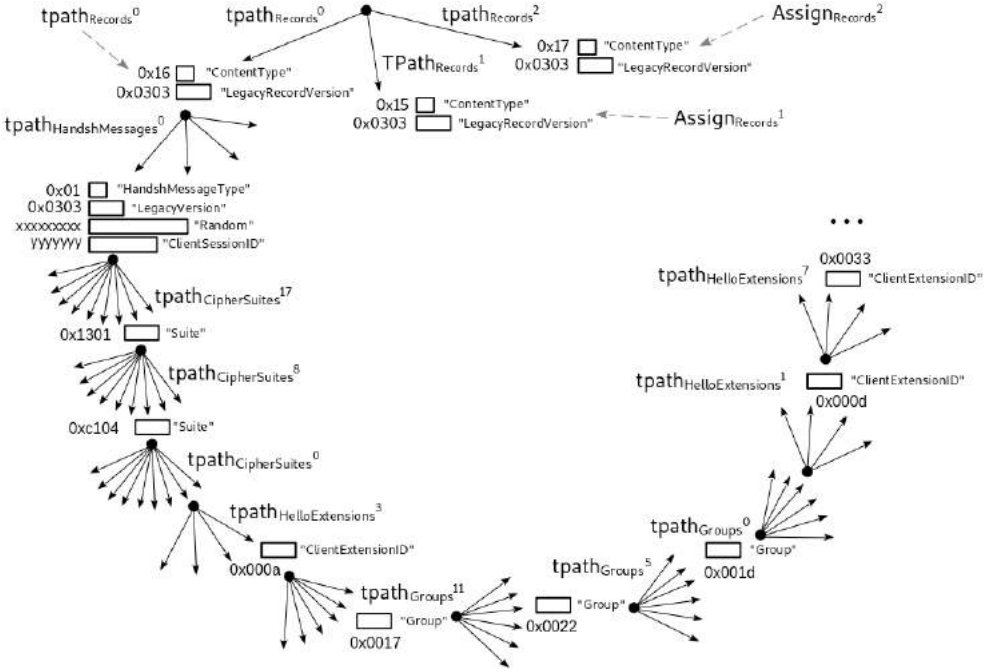


Рис. 7. Дерево вариантов формирования сообщения ClientHello
Fig. 7. Tree of variants of the ClientHello message

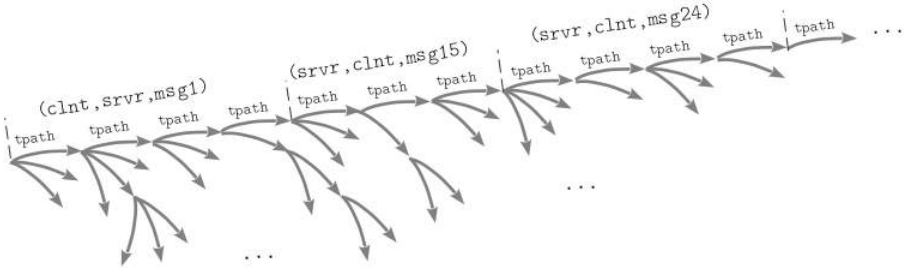


Рис. 8. Дерево траекторий протокола
Fig. 8. The tree of the protocol trajectories

Тестирование можно проводить методом случайного блуждания, то есть выбирать tpath-пути в развилках случайным образом. Однако ясно, что не все подмножества траекторий равноценны с точки зрения полезности при тестировании. Необходимо уметь автоматически выделять качественные подмножества траекторий, которые использовать и для проверки конформности, и для проверки безопасности реализаций.

В качестве критерия качества подмножества траекторий предлагается использовать критерий полноты покрытия типа MC/DC в следующей редакции: каждый гард рассматривается как формула, состоящая из атомарных предикатов (например, вида P1 && (P2 || P3)), и подмножество траекторий считается качественным, если каждый атомарный предикат каждого гарда спецификации побывал истинным, ложным и существенным (то есть значение гарда существенно зависит от этого атомарного предиката).

3. Метод факторизации множества траекторий тестирования

Процедура тестирования разбивается на две фазы. Цель первой фазы – провести тестирование конформности и одновременно факторизовать множество траекторий

протокола (остающееся огромным даже если ограничить длину траекторий) на классы эквивалентных траекторий и явно выписать эти классы в виде компактного автомата с конечным числом состояний. Вторая фаза – это тестирование безопасности реализации (фаззинг).

В рамках предлагаемого метода для борьбы с комбинаторным взрывом числа траекторий применяется последовательность приемов абстрагирования дерева траекторий протокола.

Шаг 1. В язык спецификаций добавлена возможность специальным образом помечать произвольные развилки дерева траекторий, давая тестировщику подсказки следующих двух типов:

тип 1: «эта развилка tpath-вариантов относится к последовательности, в которой нет зависимостей элементов ни от состояния, ни друг от друга; число вариантов формирования этой последовательности огромно, но все они распадаются на небольшое число классов эквивалентности»,

тип 2: «эта развилка tpath-вариантов относится к последовательности, в которой есть зависимости элементов друг от друга, но эти зависимости находятся исключительно на уровне гардов данной последовательности; число вариантов формирования этой последовательности большое, но не огромное, и среди них много эквивалентных».

Когда при обходе дерева траекторий тестировщик обнаруживает помеченную развилку, он продолжает обход лишь по одному ребру (первому или случайному), а остальные ребра откладывает в запас (рис. 9).

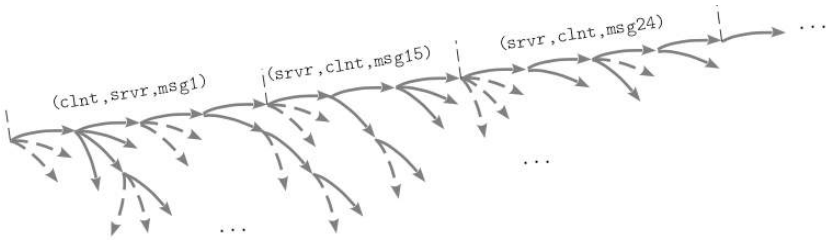


Рис. 9. Дерево траекторий протокола с отложенными ребрами
Fig. 9. The tree of the protocol with postponed edges

В случае развилки типа 2 тестировщик откладывает явно вычисленные tpath-варианты, а в случае развилки типа 1 – символическое описание отложенного множества tpath-вариантов.

Шаг 2. Вводится функция абстрагирования сообщения msg:

abstract(msg) = (true_atomic(msg), false_atomic(msg)),

где true_atomic – это мультимножество атомарных предикатов, входящих в состав гардов, встретившихся при формировании сообщения и обратившихся в истину, а false_atomic – это мультимножество остальных встретившихся гардов.

Для каждой вершины дерева, ребра сообщений msg, отображающиеся в одинаковые абстрактные сообщения, заменяются на одно ребро (рис. 10).

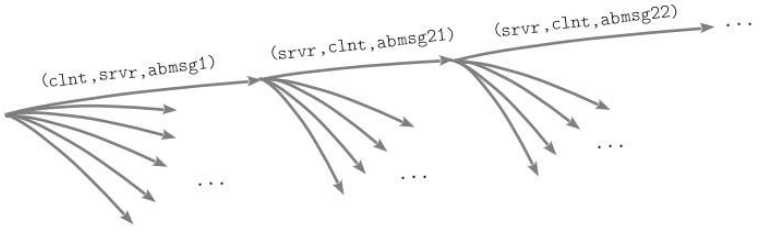


Рис. 10. Дерево траекторий протокола с абстрактными сообщениями
Fig. 10. The tree of the protocol with abstract messages as edges

Шаг 3. Наложим следующее условие: поведение объекта тестирования детерминировано, то есть на одни и те же воздействия он отвечает одинаково. Используя данное ограничение, будем строить дерево траекторий для конкретной конфигурации объекта тестирования: из всех ребер сообщений, которые потенциально мог бы отправить объект тестирования в некоторой вершине дерева, оставляем только то, которое было выбрано им фактически (рис. 11; здесь тестировщик работает в роли клиента, а объект тестирования – в роли сервера).

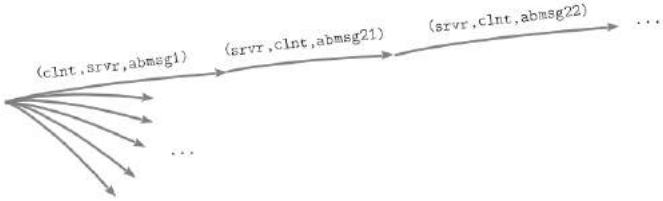


Рис. 11. Дерево траекторий протокола с детерминированным участником
Fig. 11. Customized tree of the protocol

Шаг 4. Ребра сообщений, отправляемых тестировщиком, и последующие ребра ответных флайтов (то есть ребра серии ответных сообщений) объекта тестирования заменяются на одно ребро (рис. 12), при этом флайты ответов могут быть пустыми.

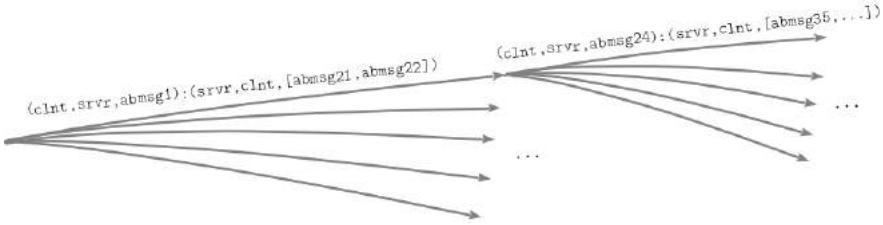


Рис. 12. Дерево траекторий протокола с флайтами сообщений
Fig. 12. The tree of the protocol with message flights as edges

Шаг 5. Согласно стандартному подходу, два состояния некоторого автомата, генерирующего сообщения, считаются эквивалентными, если из них «растут» одинаковые деревья сообщений. С учетом того, что длина каждой траектории в дереве бесконечна, данное сравнение возможно провести лишь на некоторых конечных поддеревьях этих деревьев.

В нашем методе точностью сравнения управляет специальный (изначально пустой) накопитель Precision, содержащий списки идентификаторов абстрактных сообщений abmsg, отправляемых тестировщиком. Эти списки интерпретируются как префиксы путей в дереве, изображенном на рисунке 1.12.

Абстрактным состоянием для состояния S_i тестировщика называется следующее конечное поддерево бесконечного дерева, растущего из вершины S_i: в первом слое оставлены все ребра, далее, для каждого элемента каждого префикса пути, входящего в Precision, дерево продлевается еще на один слой (рис. 13).

При тестировании тестировщик ходит по траекториям протокола, вычисляя и запоминая абстрактные сообщения и соединяя их ребрами. Если абстрактное состояние повторилось, то тестировщик далее по этой траектории не идет.

Когда все траектории завершатся попаданием в известное абстрактное состояние, будет получена текущая гипотеза конечного автомата тестирования (далее, автомат-гипотеза).

Шаг 6. Для текущего автомата-гипотезы осуществляется поиск контрпримеров двух типов:

1) опробование ребер, отложенных в запас: в помеченных развилках, в которых на шаге 1 было оставлено только одно tpath-ребро, это оставленное ребро заменяется на одно из tpath-ребер, отложенных в запас, после чего проверяется, приводит ли данная

замена к изменению автомата-гипотезы; если не приводит, то данное tpath-ребро записывается в класс эквивалентности первого tpath-ребра; если приводит, то это означает, что найден контрпример;

2) проход по различным путям автомата-гипотезы с целью поиска вычисленного абстрактного состояния, не совпадающего с абстрактным состоянием, которое должно быть в этой точке этого пути согласно автомату-гипотезе.

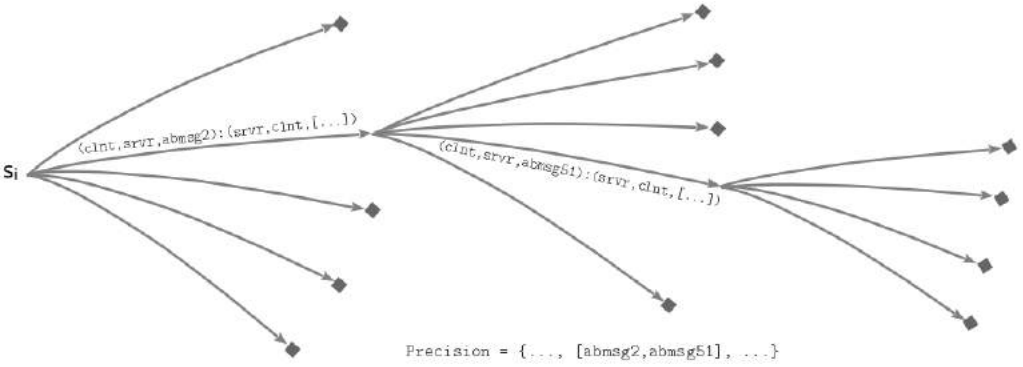


Рис. 13. Абстрактное состояние протокола
Fig. 13. Protocol abstract state

После нахождения контрпримера первого типа в помеченную развилку, в которой было найдено tpath-ребро-контрпример, добавляется это ребро (то есть в развилке теперь не одно, а два ребра), и шаги 1-5 проделываются заново. На последующих итерациях, при извлечении нового ребра из запаса последовательно проверяется, какому ребру из развилки он эквивалентен. Если никакому, то найден контрпример (новый класс эквивалентности) и т. д. После нахождения контрпримера второго типа в накопитель Precision добавляется путь в состояние тестировщика, на котором было выявлено несоответствие абстрактных сообщений, и шаги 1-5 проделываются заново для уточненных (refined) абстрактных состояний.

Отсутствие контрпримеров на некоторой итерации выполнения шагов 1-6 в течение продолжительного времени означает успешное завершение построения автомата тестового набора.

4. Заключение

Сходимость (постоянное увеличение уровня детализации автомата) и завершимость описанной итеративной процедуры тестирования на практических протоколах были проверены экспериментально на примере протоколов TLS и IPSec IKEv2. На рис. 14 представлены примеры автоматов-гипотез на четырех итерациях алгоритма при тестировании одной из конфигураций сервера OpenSSL.

Так как абстрагирование сообщений протокола и состояний автомата осуществляется на базе значений атомарных предикатов гардов, то представленный в настоящей работе метод построения опорного тестового набора нацелен на достижение критерия MC/DC в приведенной выше редакции. Однако автоматическое выполнение данного критерия не гарантируется. Непокрытые атомарные предикаты выводятся аналитику, который должен построить недостающие траектории с помощью специального инструмента ручного построения траекторий, после чего эти траектории будут автоматически встроены в опорный тестовый набор.

Построение опорного тестового набора означает завершение первой фазы тестирования реализации криптопротокола – фазы проверки конформности.

Далее этот тестовый набор будет использоваться во второй фазе тестирования – фазе проверки безопасности (фаззинге) – в качестве несущих траекторий мутационных воздействий на объект тестирования. Знание фаззером логического смысла содержимого полей сообщений протоколов (шифртекст, точка эллиптической кривой и т.д) обеспечит возможность применения предметно-ориентированных мутаций (например, отправку открытого ключа, не лежащего на согласованной эллиптической кривой) и исключит бессмысленные мутации (например, перебор искажений шифртекста или значений хэш-функции). Как следствие, может быть выполнено требование 6 из табл. 1 (прослеживаемость результатов тестирования к проверкам свойств безопасности из явно заданного перечня угроз).

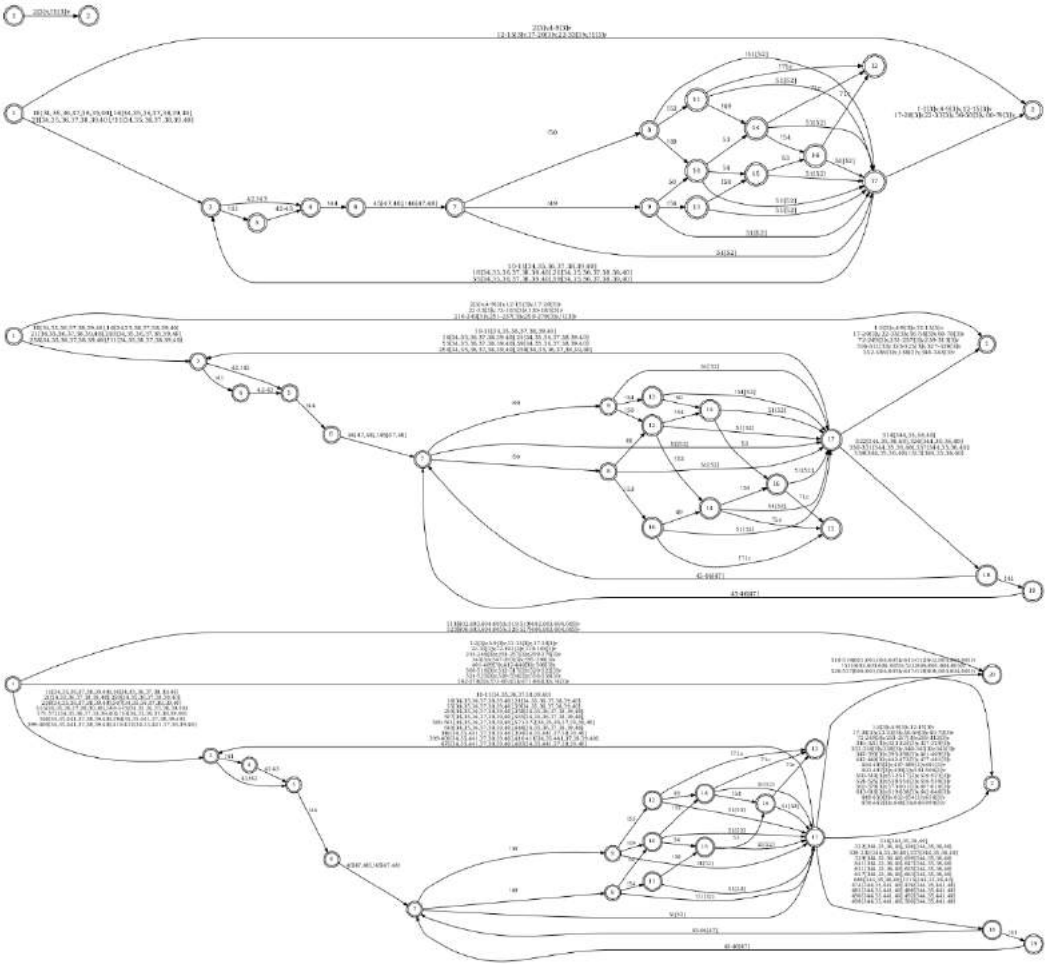


Рис. 14. Автоматы-гипотезы на разных итерациях метода
Fig. 14. Hypotheses of the automaton on the iterations of the method

Список литературы / References

[1]. H. Kario. Tlsfuzzer. Available at: https://github.com/tomato42/tlsfuzzer, accessed 10.11.2023.
[2]. M. Ammann, L. Hirschi, S. Kremer. DY Fuzzing: Formal Dolev-Yao Models Meet Cryptographic Protocol Fuzz Testing. https://eprint.iacr.org/2023/057.
[3]. Пакулин Н.В., Шнитман В.З., Никешин А.В. Разработка тестового набора для верификации реализаций протокола безопасности TLS // Труды Института системного программирования РАН, том 23, 2012 г., стр. 387-404.

[4]. Kenneth L. McMillan and Lenore D. Zuck. 2019. Formal specification and testing of QUIC. In Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM'19). ACM, New York, NY, USA, 14 pages.
[5]. ABZ – International Conference on Rigorous State Based Methods, Available at: https://abz-conf.org, accessed 10.11.2023.
[6]. Прокопьев С.Е. Формальный язык первичных спецификаций криптографических протоколов. Труды ИСП РАН. 2021;33(5):117-136. https://doi.org/10.15514/ISPRAS-2021-33(5)-7.
[7]. Prokopev, S. Cryptographic protocol conformance testing based on domain-specific state machine. J Comput Virol Hack Tech (2023). https://doi.org/10.1007/s11416-023-00474-1.

Информация об авторах / Information about authors

Сергей Евгеньевич ПРОКОПЬЕВ – старший научный сотрудник отдела технологий программирования ИСП РАН, руководитель направления сетевых протоколов безопасности лаборатории сетевой и информационной безопасности АО НПК «Криптонит».

Sergey Evgenevich PROKOPEV – research associate in the Department of Programming Technologies of the ISP RAS, senior researcher in the area of security protocols in the Laboratory of the Network Security of JSC NPK "Kryptonite".

DOI: 10.15514/ISPRAS-2023-35(6)-11

Анализ безопасности проекта национального стандарта «Нейросетевые алгоритмы в защищенном исполнении. Автоматическое обучение нейросетевых моделей на малых выборках в задачах классификации»

Г.Б. Маршалко, ORCID: 0009-0001-3499-3368 <*marshalko_gb@tc26.ru*>
Р.А. Романенков, ORCID: 0009-0000-2670-4709 <*romanenkov_ra@tc26.ru*>
Ю.А. Труфанова, ORCID: 0009-0003-5135-2196 <*trufanova_ua@tc26.ru*>
Технический комитет по стандартизации «Криптографическая защита информации», Россия, 127273, г. Москва, ул. Отрадная, 2Б стр.1.

Аннотация. В работе предложен метод проверки принадлежности обучающей выборке для нейросетевого алгоритма классификации из проекта национального стандарта, разработанного Омским государственным техническим университетом под эгидой Технического комитета по стандартизации «Искусственный интеллект» (ТК 164). Указанный метод позволяет определить, использовались ли данные при обучении нейронной сети, и направлен на нарушение свойства конфиденциальности обучающей выборки. Полученные результаты показывают, что описываемый стандартом механизм защиты нейросетевых классификаторов не обеспечивает заявленных свойств. Полученные результаты первоначально анонсированы на конференции Рускрипто'2023.

Ключевые слова: Статистическая классификация; нейронные сети; информационная безопасность; обучающая выборка; атака проверки принадлежности обучающей выборке; конфиденциальность; стандартизация.

Для цитирования: Маршалко Г.Б., Романенков Р.А., Труфанова Ю.А., Анализ безопасности проекта национального стандарта «Нейросетевые алгоритмы в защищенном исполнении. Автоматическое обучение нейросетевых моделей на малых выборках в задачах классификации». Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 179–188. DOI: 10.15514/ISPRAS–2023–35(6)–11.

Security Analysis of the Draft National Standard «Neural Network Algorithms in Protected Execution. Automatic Training of Neural Network Models on Small Samples in Classification Tasks»

G.B. Marshalko, ORCID: 0009-0001-3499-3368 <*marshalko_gb@tc26.ru*>
R.A. Romanenkov, ORCID: 0009-0000-2670-4709 <*romanenkov_ra@tc26.ru*>
J.A. Trufanova, ORCID: 0009-0003-5135-2196 <*trufanova_ua@tc26.ru*>
Technical Committee for Standardization «Cryptographic Information Protection», bld. 1, 2 B, Otradnaya st., Moscow, 127273, Russia.

Abstract. We propose a membership inference attack against the neural classification algorithm from the draft national standard developed by the Omsk State Technical University under the auspices of the Technical Committee on Standardization «Artificial Intelligence» (TC 164). The attack allows us to determine whether

the data were used for neural network training, and aimed at violating the confidentiality property of the training set. The results show that the protection mechanism of neural network classifiers described by the draft national standard does not provide the declared properties. The results were previously announced at Ruscrypto'2023 conference.

Keywords: Statistic classification; neural networks; informational security; training set; membership inference attack; confidentiality; standardization.

For citation: Marshalko G.B., Romanenkov R.A., Trufanova J.A., Security analysis of the draft national standard «Neural network algorithms in protected execution. Automatic training of neural network models on small samples in classification tasks». *Trudy ISP RAN/Proc. ISP RAS,* vol. 35, issue 6, 2019. pp. 179-188 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-11.

1. Введение

Работа посвящена исследованию безопасности проекта национального стандарта «Автоматическое обучение нейросетевых моделей на малых выборках в задачах классификации» [2] и [3], разработанного в 2022 году под эгидой Технического комитета по стандартизации «Искусственный интеллект» (ТК 164) Омским государственным техническим университетом.

Для технологий искусственного интеллекта, и, в частности для получивших в настоящее время наибольшее распространение технологий машинного обучения, как и для любой другой информационной технологии, необходимо рассматривать вопросы безопасности. Эти технологии обладают той особенностью, что алгоритм решения задачи (обученная модель), например, задачи классификации, получается в процессе ее решения и существенным образом зависит от входных данных (обучающей выборки), а не фиксирован заранее. Это создает новые способы построения атак на системы, использующие технологии искусственного интеллекта, не актуальные для других технологий.

К настоящему моменту известен широкий перечень угроз информационной безопасности при использовании методов машинного обучения [1], а именно:

1. Угрозы нарушения конфиденциальности данных:
    1.1. Извлечение данных о параметрах обученных моделей;
    1.2. Извлечение данных об обучающей выборке из обученных моделей;
2. Угрозы нарушения доступности данных:
    2.1. Искажение (отравление) обучающей выборки с целью ухудшения качества модели;
3. Угрозы нарушения целостности данных:
    3.1. Формирование т. н. состязательных входных данных, некорректно обрабатываемых (например, классифицируемых) моделью.

Модели машинного обучения, предназначенные для решения задач классификации данных, извлекают из входного примера некоторый вектор признаков, который затем используется для определения принадлежности к заданным заранее классам. Способы отнесения вектора признаков к тому или иному классу разнятся в зависимости от архитектуры модели машинного обучения. Например, одним из подходов является выделение на этапе обучения эталонного вектора признаков каждого класса. В процессе эксплуатации системы получаемые вектора признаков сравниваются с эталонными векторами и, соответственно, входные данные классифицируются в зависимости от степени близости вектора признаков к одному эталонных векторов. Однако при такой постановке задачи узким местом является необходимость постоянного хранения эталонного вектора в памяти системы, что может приводить к появлению угрозы их утечки с последующим восстановлением соответствующих им входных данных обучающей выборки.

Известно научное направление, связанное с построением защищенных (робастных) нейросетевых моделей, целью которых является встраивание защиты непосредственно в алгоритм ее работы. Технически это реализуется через выбор определенной архитектуры модели, особого алгоритма обучения и специально подобранных входных данных обучающей выборки. В большинстве случаев, как показывает практика, такой подход позволяет усложнить реализацию атак, но не может обеспечить гарантированную защиту.

Механизм построения защищенного классификатора, описываемый в проекте национального стандарта «Автоматическое обучение нейросетевых моделей на малых выборках в задачах классификации», является одним из вариантов построения защищенной нейросетевой модели и предназначен в соответствии с разделами 5 и 6 проекта стандарта [2] для защиты от указанных выше атак.

В рамках настоящей работы показано, что описываемый в проекте стандарта механизм уязвим к типу атак, направленных на извлечение данных об обучающей выборке из обученной модели — атаке проверки принадлежности обучающему множеству [4]. В атаках такого типа нарушитель имеет доступ к обученной модели и набору данных, среди которых, возможно, есть те, которые были использованы для обучения модели. Нарушитель, подавая последовательно на вход модели элементы набора данных, пытается определить на основе значений выходов, принадлежали ли данные обучающему множеству. С теоретической точки зрения возможность построения таких атак связана с наличием у нейросетевых моделей так называемого эффекта запоминания, заключающегося в том, что обученная модель срабатывает на входных данных из обучающей выборки «лучше», чем на произвольных данных. Практическая возможность реализации такой атаки возникает, например, в системах отменяемой биометрии [5] при их реализации на мобильных устройствах. Действительно, необходимость идентификации пользователя по биометрическим признакам без доступа к сети Интернет требует хранения контрольных биометрических шаблонов пользователей на мобильном устройстве, в том числе, в виде защищенных нейросетевых моделей [5], к которым относится и предлагаемый в проекте стандарта механизм. В случае компрометации (утери) устройства пользователем, нарушитель получает возможность непосредственного доступа к хранящейся на устройстве модели и способен проводить атаку проверки принадлежности обучающему множеству перебирая доступные ему, например, из социальных сетей входные биометрические образы потенциальных владельцев устройства.

2. Обозначения и краткое описание работы предложенного в проекта стандарта механизма

Исследуемый способ построения нейросетевых моделей (в терминах стандарта – нейросетевой преобразователь, НКП) предназначен для реализации классификаторов, на вход которых подается предварительно извлеченный из входных данных каким-либо способом вектор признаков, и рассматривает два класса входных признаков:

- «Свой» – класс признаков, которые должны корректно классифицироваться обученным НКП (например, соответствующие изображению лица конкретного человека);
- «Чужие» – произвольные входные признаки, не относящиеся к классу «Свой» (например, соответствующие случайной выборке без возвращения из множества изображений лиц различных людей).

Для целей исследования мы будем также использовать термин «Другой» – выборка из некоторого другого класса, не являющегося «Своим».

- $\overline{a(k)}$ – вектор признаков k-го входного объекта, если нет двусмысленности, то k будем опускать;

- a_j – j-й признак вектора признаков $\bar{a}$;
- $\bar{a'}$ – вектор мета-признаков;
- a'_{j^*} – j^*- мета-признак вектора мета-признаков $\bar{a'}$;
- n – количество признаков;
- m_j и σ_j – математическое ожидание и среднеквадратичное отклонение j-ого признака для класса «Свой»;
- μ_j и δ_j – нормирующие коэффициенты, вычисляемые как математическое ожидание и среднеквадратичное отклонение j-го признака для класса «Чужие»;
- K_G – количество тренировочных примеров образа «Свой»;
- K_I – количество тренировочных примеров образов «Чужой»;
- η – количество входов корреляционного нейрона;
- ω_j – вес корреляционного нейрона под номером j;
- $C_{j,t}$ – коэффициент корреляции между признаками с номерами j и t.

Признаки $\overline{a(k)}$ в соответствии с [2,3] должны иметь распределение, близкое к нормальному, что может быть достигнуто путем подбора и настройки алгоритма извлечения признаков из входных данных, например, вариационного автоэнкодера.

Мета-признаки вычисляются следующим образом:

$$a'_{j^*} = a'_{t,j} = f(a_t, a_j) = \left| \left|\frac{a_t}{\delta_t}\right|^{0.9} - \left|\frac{a_j}{\delta_j}\right|^{0.9} \right|,$$

где

$$\delta_i = \delta_{I,i} = \sqrt{\frac{1}{n}\sum_{K_I}\left(a_{I.i} - \mu_{I,i}\right)^2},$$

$$\mu_{I,i} = \frac{1}{n}\sum_{K_I} a_{I,i}.$$

Значение нейрона вычисляется следующим образом:

$$y = \sqrt{\frac{1}{\eta}\omega_i(a'_i - m')^2},$$

$$m' = \frac{1}{\eta}\sum_{i=1}^{\eta} a'_i.$$

Выбор мета-признаков при обучении каждого нейрона осуществляется псевдослучайным образом с учетом значения коэффициента корреляции между ними. Конкретные границы на такие значения приведены в тексте проекта стандарта [2], однако они не принципиальны для реализации предлагаемого метода и поэтому опущены в настоящей статье.

Веса синапсов вычисляются следующим образом:

$$\omega_i = \frac{|m''_{G,i} - m''_{I,i}|}{\sigma''_{G,i}, \sigma''_{I,i}},$$

где $m''_{G,i}, m''_{I,i}$ −математические ожидания, а $\sigma''_{G,i}, \sigma''_{I,i}$ −среднеквадратичные отклонения значений i-го мета-признака второго порядка:

$$a''_i = \left(a'_i - m'\right)^2.$$

Упрощенно говоря, вес нейрона хранит информацию о расстоянии между центром класса «Свой» и центром класса «Чужие» для соответствующих признаков. Значение функции выхода нейрона имеет вид:

$$\phi(y) = \begin{cases} 3, y < T_{left}; \\ 2, T_{left} \le y < T_{middle}; \\ 1, T_{middle} \le y < T_{right}; \\ 0, y \ge T_{right}. \end{cases}$$

Границы $T_{left}, T_{middle}, T_{right}$ подбираются таким образом, чтобы на выходе выполнялись соотношения $P(0) \approx P(1) \approx P(2) \approx P(3)$ в предположении о нормальности распределения выхода каждого нейрона. О том, какое именно состояние активации (далее – $\phi(y)$) соответствует гипотезе «Свой», известно только на этапе обучения корреляционных нейронов, злоумышленник не обладает этой информацией, так как она не сохраняется после настройки нейрона.

Далее выбирается хэш-таблица для кодирования полученных значений, но ее выбор также не принципиален для дальнейшего изложения, поскольку она является частью обученного НКП и не является секретной.

Таким образом, обученный преобразователь хранит значения следующих параметров:

- связи корреляционных нейронов с мета-признаками;
- нормирующие коэффициенты признаков δ_i;
- веса нейронов;
- границы $T_{left}, T_{middle}, T_{right}$;
- хэш-таблица для кодирования выходных значений нейронов.

На рис. 1 приведена схема автоматического обучения нейросетевых моделей, представленная в проекте стандарта [3].

3. Идея атаки

В основе предлагаемой атаки проверки принадлежности обучающему множеству лежит следующая гипотеза: обучение различных НКП на близких входных данных, например, на различных изображениях одного и того же человека, должно давать в некотором смысле «близкие» обученные нейросетевые преобразователи.

Будем предполагать, что объекты истинного класса «Свой» находятся в выборке, доступной нарушителю. Тогда нарушитель может реализовать атаку с помощью следующей последовательности действий:

- извлечь из атакуемого НКП связи корреляционных нейронов с мета-признаками, границы $T_{left}, T_{middle}, T_{right}$, нормирующие коэффициенты признаков δ_i;
- используя каждый доступный ему набор данных (из некоторого класса) в качестве класса «Свой», попытаться обучить новый НКП с фиксированными связями корреляционных нейронов с мета-признаками и нормирующими коэффициентами δ_iполученными из атакуемого НКП;
- выбрать среди полученных границ $T_{left}, T_{middle}, T_{right}$ те, которые ближе всего в некоторой метрике к соответствующим границам атакуемого (исходного) НКП.

Можно ожидать, что класс входных данных, который дает наиболее близкие к исходным границы, и был использован для обучения атакуемого нейрона изначально. То есть объекты соответствующего класса относятся к истинному классу «Свой» атакуемого нейрона.

Для проверки этой гипотезы рассмотрим далее ситуацию, когда у нарушителя есть множество данных из двух классов: «Свой» и «Другой», а также НКП, обученный изначально на классе «Свой».

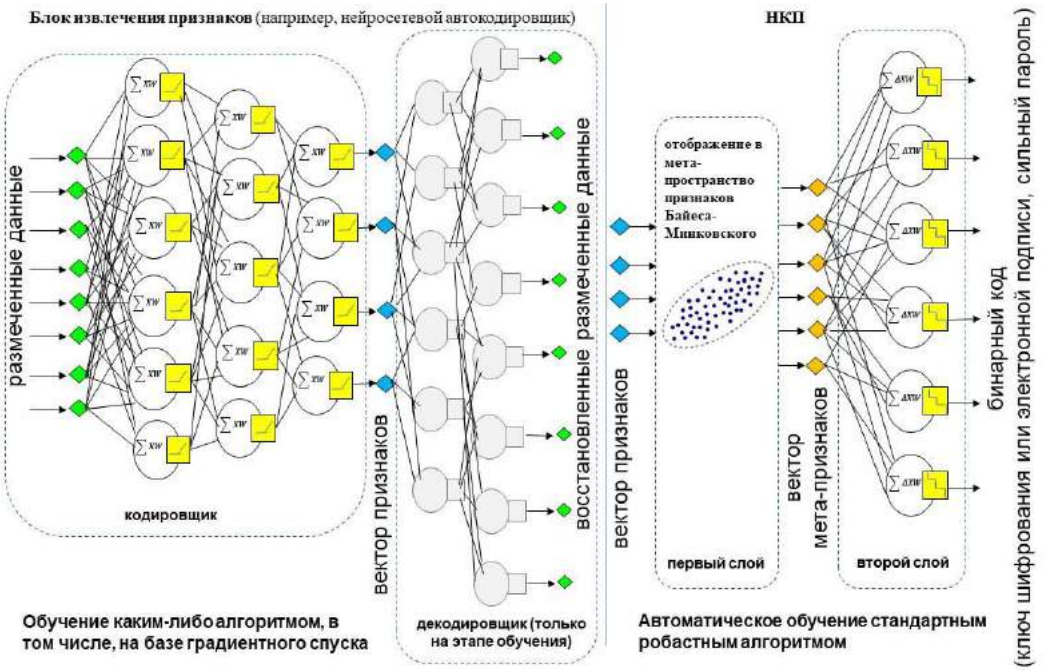


Рис. 1: Схема автоматического обучения нейросетевых моделей
Fig. 1: Scheme of automatic training of neural network models

4. Алгоритм атаки

Пусть у нас есть обученный НКП, а значит нам известны связи, нормирующие коэффициенты и веса нейронов. Будем считать, что он был обучен на выборке G_1 примеров «Свой» и выборке I_1примеров «Чужой».

Сформируем следующие обучающие множества:

- G_1 – выборка примеров «Свой»;
- I_1 –выборка примеров «Чужой»;
- $G_1^{'}$ –еще одна выборка примеров «Свой» из той же генеральной совокупности, из которой выбиралось множество G_1;
- G_2 – выборка примеров «Другой» из другой генеральной совокупности;
- I_2 – выборка примеров «Чужой» такая, что $\delta_{I_1,i} = \delta_{I_2,i}$.

Последнее равенство необходимо для того, чтобы обеспечить близость выборки I_1 к I_2. В табл. 1 приведены параметры выборок, использовавшихся при практической реализации атаки. Использование нормального распределения определено требованием проекта стандарта [3] о виде распределения входных признаков. Интервал использовавшихся значений математического ожидания выбран в соответствии с параметрами, использованными в контрольном примере [3].

Проект национального стандарта накладывает достаточно жесткие ограничения на распределение входных данных, а именно предполагается, что каждый входной признак имеет нормальное распределение [2] и [3]. Такой эффект действительно наблюдается для определенных вариантов построения НКП, например, тогда, когда на ход к нему поступают данные с вариационного автоэнкодера.

Для проведения экспериментов входные данные формируются в интервале $[0; 14]$ как указано в контрольном примере процедуры обучения НКП стандарта [2] Для каждой реализации в эксперименте при выработке множеств G_1 и $G_1^{'}$ для каждого признака фиксируются параметры нормального распределения $N\big(\mu_1(i), \sigma_1(i)\big), i = 1, \dots, 128$, где значение математического ожидания принадлежит указанному диапазону, далее вырабатывается нужное количество векторов признаков. Для множества G_2 фиксируются параметры нормального распределения $N\big(\mu_2(i), \sigma_2(i)\big), i = 1, \dots, 128$ с теми же ограничениями, далее вырабатывается нужное количество векторов признаков. Таким образом моделируются выборки векторов из классов «Свой» и «Другой». Как было отмечено ранее выбор вида функции распределения признаков определяется требованиями стандарта [2], однако нетрудно видеть, что предложенный в данной статье метод будет работать и для других классов распределений признаков в случае, если они имеют конечные математические ожидания.

Для множеств I_1 и I_2 при выработке каждого вектора фиксируется $N\big(\mu_3(i), \sigma_3(i)\big), \ i = 1, \dots, 128$, а затем из него делается выборка одного значения, что моделирует выборку случайных векторов (класс «Чужой»).

Табл. 1. Значение параметров выборки
Table 1. The values of the parameters of the training sets

Номер эксперимента	Параметры нормального распределения для выборок G_1 и $G_1^{'}$	Параметры нормального распределения для выборок I_1 и $I_1^{'}$	Параметры нормального распределения для выборок G_2
1	$\mathcal{N}(\mu, 1), \mu = 2,3$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2,3$
2	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 4$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 4$
3	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 5$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 5$
4	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 6$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 6$
5	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 7$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 7$
6	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 8$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 8$
7	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 9$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 9$
8	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 10$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 10$
9	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 11$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 11$
10	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 12$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 13$	$\mathcal{N}(\mu, 1), \mu = 2, \dots, 12$

Тогда эксперимент можно описать следующим образом:

1) Сформировать множества $G_1, G_1^{'}, G_2, I_1$ и I_2;
2) Обучить НКП-1 на исходных множествах G_1 и I_1, в результате чего получаются наборы $T_{left}^1, T_{middle}^1, T_{right}^1$;
3) Обучить НКП-2 на множествах $G_1^{'}$ и I_2, используя связи нейронов с мета-признаками и нормирующие коэффициенты из НКП-1, в результате чего получаются наборы $T_{left}^2, T_{middle}^2, T_{right}^2$;

4) Обучить НКП-3 на множествах G_2 и I_2, используя связи нейронов с мета-признаками и нормирующие коэффициенты из НКП-1, в результате чего получаются наборы $T^3_{left}, T^3_{middle}, T^3_{right}$;
5) Вычислить значение статистики F (описана далее) и сравнить его с граничным значением.

Эксперимент повторяется несколько раз (в представленных далее результатах эксперимент повторялся 50 раз). Таким образом, мы строим статистический критерий различения классов «Свой» и «Другой» относительно обученного НКП.

Обучение НКП-2 и НКП-3 (см. табл. 2) отличается от обучения НКП-1 тем, что мы не оцениваем корреляцию признаков для выбора связей, а берем значения связей из НКП-1. Возможны следующие варианты для каждого нейрона в НКП с номером i.

Табл. 2. Возможные варианты обучения нейронов в эксперименте
Table 2. Possible variants neurons training

Нейрон НКП-2	Нейрон НКП-3
обучен	обучен
обучен	не обучен
не обучен	обучен
не обучен	не обучен

Нас интересуют нейроны, соответствующие первой строке, т.е. которые обучились для всех трех НКП.

Для каждого такого нейрона (пусть их будет d штук) на шаге 4 вычисляются следующие статистики:

$$z_2(i) = max\left(\left|T^1_{left}(i) - T^2_{left}(i)\right|, \left|T^1_{middle}(i) - T^2_{middle}(i)\right|, \left|T^1_{right}(i) - T^2_{right}(i)\right|\right), i = 1, \dots, d,$$

$$z_3(i) = max\left(\left|T^1_{left}(i) - T^3_{left}(i)\right|, \left|T^1_{middle}(i) - T^3_{middle}(i)\right|, \left|T^1_{right}(i) - T^3_{right}(i)\right|\right), i = 1, \dots, d.$$

Далее вычисляется результирующая статистика:

$$F = \sum_{i=1}^{d} I\{z_2(i) < z_3(i)\}$$

Если полученное значение статистики стабильно больше $\frac{d}{2}$ для всех проведенных экспериментов, то гипотеза верна, и мы можем реализовать атаку проверки принадлежности обучающему множеству. В табл. 3 приведены средние значения статистики $F > \frac{d}{2}$ для каждого эксперимента, которые показывают успешность предложенной атаки.

5. Заключение

Для алгоритма обучения нейросетевого преобразователя из проекта национального стандарта «Нейросетевые алгоритмы в защищенном исполнении. Автоматическое обучение нейросетевых моделей на малых выборках в задачах классификации» предложен статистический критерий, который позволяет реализовать атаку проверки принадлежности обучающему множеству.

Проведены экспериментальные исследования, которые показали, что на входных данных, соответствующих ограничениям, приведенным в проекте стандарта, критерий позволяет корректно определять входные данные, использовавшиеся для обучения атакуемого нейросетевого преобразователя.

Табл. 3. Результаты эксперимента
Table 3. Experimental results

Номер эксперимента	Количество «успешно» обученных нейронов НКП-2	Количество «успешно» обученных нейронов НКП-3	Среднее значение статистики F	Число запусков, в которых значение статистики $F > \frac{d}{2}$
1	128	128	68	45
2	128	128	78	48
3	128	128	81	49
4	128	128	72	46
5	128	128	91	50
6	128	128	88	50
7	128	128	83	49
8	128	128	77	48
9	128	128	65	46
10	128	128	93	50

Список литературы / References

[1]. Гуселев А.М., Маршалко Г.Б., «Проблемы безопасности систем машинного обучения», в сборнике трудов МИТСОБИ'2021, стр. 23-24.
[2]. Первая редакция проекта национального стандарта «Искусственный интеллект. Нейросетевые алгоритмы в защищенном исполнении. Автоматическое обучение нейросетевых моделей на малых выборках в задачах классификации», Технический комитет по стандартизации «Искусственный интеллект» (ТК 164), 2022.
[3]. Сулавко, А.Е. «Защищенный режим исполнения искусственного интеллекта на базе автоматически обучаемых сетей автокорреляционных нейронов», Технический отчет, ОмГТУ, Омск, 2021, 101 с.
[4]. R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership Inference Attacks against Machine Learning Models, 2017 IEEE Symposium on security and privacy (SP), pp. 3-18, 2017.
[5]. Manisha, N. Kumar, Cancelable biometrics: a comprehensive survey, Artificial intelligence review, 53, 3403-3446, 2020.

Информация об авторах / Information about authors

Григорий Борисович МАРШАЛКО – эксперт, Технический комитет по стандартизации "Криптографическая защита информации". Область научных интересов: защита информации, криптография, биометрическая идентификация.

Grigory Borisovich MARSHALKO – Expert, Technical committee for standardization "Cryptography and security mechanisms". Research interests: information security, cryptography, biometric identification.

Роман Александрович РОМАНЕНКОВ – эксперт, Технический комитет по стандартизации "Криптографическая защита информации". Область научных интересов: защита информации, моделирование случайных величин, прикладные методы математической статистики.

Roman Alexandrovich ROMANENKOV – Expert, Technical committee for standardization "Cryptography and security mechanisms". Research interests: information security, cryptography, modeling of random variables, applied mathematical statistics.

Юлия Анатольевна ТРУФАНОВА – эксперт, Технический комитет по стандартизации "Криптографическая защита информации". Область научных интересов: защита информации, биометрическая идентификация, машинное обучение.

Julia Anatolievna TRUFANOVA – Expert, Technical committee for standardization "Cryptography and security mechanisms". Research interests: information security, biometric identification, machine learning.

DOI: 10.15514/ISPRAS-2023-35(6)-12

Параллельная реализация алгоритма исправления нарушений антенных правил в маршруте OpenLane

Д.А. Булах, ORCID: 0000-0002-6270-8616 <*dima@pkims.ru*>
А.В. Коршунов, ORCID: 0000-0002-2470-5836 <*korsh84@yandex.ru*>

Национальный исследовательский университет МИЭТ,
124498, Россия, г. Москва, г. Зеленоград, пл. Шокина, д.1

Аннотация. Открытый маршрут проектирования интегральных схем OpenLane позволяет реализовать маршрут проектирования RTL-to-GDS, однако некоторые задачи остаются незакрытыми. Примером такой задачи является рассматриваемое в данной работе исправление нарушений антенных правил, средства детектирования которых входят в состав маршрута OpenLane. Было разработано программное обеспечение, которое позволяет на основе входных данных в виде файлов с информацией о составе библиотеки элементов и информации о размещении и трассировке схемы выполнить исправление нарушений антенных правил. Для разработанного программного обеспечения была реализована параллельная реализация, в работе приводятся результаты сравнения двух версий; показано, что выигрыш по времени составил более 60%. В статье рассматриваются как разработанные алгоритм и построенное на его основе программное обеспечение, способное встраиваться в открытый маршрут OpenLane, так и пример кода скрипта встраивания. Использование разработанного программного обеспечения позволяет исправлять значительную часть нарушений антенных правил, повышая тем самым процент выхода годных.

Ключевые слова: интегральная схема; открытый маршрут проектирования; антенный эффект; антенные правила.

Для цитирования: Булах Д.А., Коршунов А.В. Параллельная реализация алгоритма исправления нарушений антенных правил в маршруте OpenLane. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 189–198. DOI: 10.15514/ISPRAS–2023–35(6)–12.

Благодарности: Работа выполнена при финансовой поддержке Минобрнауки в рамках государственного задания на выполнение научно-исследовательской работы «Разработка методики прототипирования электронной компонентной базы на отечественных микроэлектронных производствах на основе сервиса MPW» (FSMR-2023-0008).

Parallel Implementation of the Algorithm for Correction of Antenna Rule Violations in OpenLane Design Flow

D.A. Bulakh, ORCID: 0000-0002-6270-8616 <*dima@pkims.ru*>
A.V. Korshunov, ORCID: 0000-0002-2470-5836 <*korsh84@yandex.ru*>

National Research University of Electronic Technology,
Shokin sq., bld.1, Zelenograd, Moscow, 124498, Russia.

Abstract. The open IC design flow OpenLane make available the RTL-to-GDS design flow to be implemented, but still some tasks remain unsolved. An example of such task is the correction of antenna rules violations. The detection tools for this procedure are the part of the OpenLane flow but it does not contain any tools for avoiding them. In this article the software is presented that has been developed and allows to avoid violations of antenna rules based on input data in the form of LEF file with information about the standard cell library and DEF file with information about the placement and routing of the design. A parallel implementation is also described; we show the results of more than 60% gain in time with using parallel version in comparison with sequential version. The article describes both the developed algorithm and the software built on its basis, the capability of embedding into an open OpenLane flow, and an example of embedding script code. The use of the developed software makes it possible to correct a significant part of antenna rules violations, thereby increasing the yield.

Keywords: integrated circuit; open design flow; antenna effect; antenna rules.

For citation: Bulakh D.A., Korshunov A.V. Parallel implementation of the algorithm for correction of antenna rule violations in OpenLane design flow. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 189-198 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-12.

Acknowledgements. This work was carried out with the financial assistance of the Ministry of Education and Science in the framework of state task FSMR-2023-0008 (Development of a methodology for prototyping electronic component base at domestic microelectronic production facilities on the basis of MPW service).

1. Введение

Открытый маршрут проектирования OpenLane представляет собой набор отдельных программ и скриптов с открытым исходным кодом, которые объединены в единый цикл запуска управляющим скриптом. Он позволяет довести проектируемое интегральное решение от кода на языке описания цифровой аппаратуры Verilog до представления в технологическом формате GDSII [1, 2]. Маршрут базируется на открытом маршруте OpenROAD [3, 4] и включает в себя программы для синтеза и оптимизации дизайнов, выполнения процедур для верификации промежуточных данных и анализа физических характеристик получаемых решений. Последовательность выполняемых проектных процедур в рамках открытого маршрута OpenLane показана на рис. 1 [1].

Несмотря на наличие большого числа программ, входящих в состав маршрута, он всё ещё неполон. Например, при выполнении этапа физической верификации перед выводом результата в формате GDSII среди прочих проверок выполняется вызов программы Antenna Rules Checker, которая предназначена для проверки нарушений антенных правил для дорожек металлизации. Однако, при этом в маршруте отсутствуют средства для исправления обнаруженных нарушений.

В данной работе описывается реализованный алгоритм устранения антенных нарушений, результаты использования механизма распараллеливания для ускорения его работы, а также подход к его использованию в составе открытого маршрута OpenLane.

2. Антенный эффект

Технология физического изготовления интегральных схем предусматривает выполнение ряда технологических операций, одной из которых является сухое травление, которое используется при прокладке трасс металлизации для соединения элементов на интегральной

схеме. Плазма, используемая при травлении, содержит высокоэнергетические ионы, которые могут в большом количестве накапливаться в формируемых участках металлизации. Общее количество накопленных ионов определяется технологией операции и зависит от формируемой площади поверхности металла. Если металлическая дорожка имеет достаточно длинный размер, может произойти накопление потенциала в количестве, достаточном для проявления антенного эффекта. Он возникает в случае, когда этот участок металлизации подходит к затвору транзистора, таким образом этот значительный накопленный потенциал может пробить затвор транзистора. Это может произойти, поскольку между затвором и подложкой происходит накопление значительной разницы потенциалов. Описываемая ситуация показана на рис. 2.

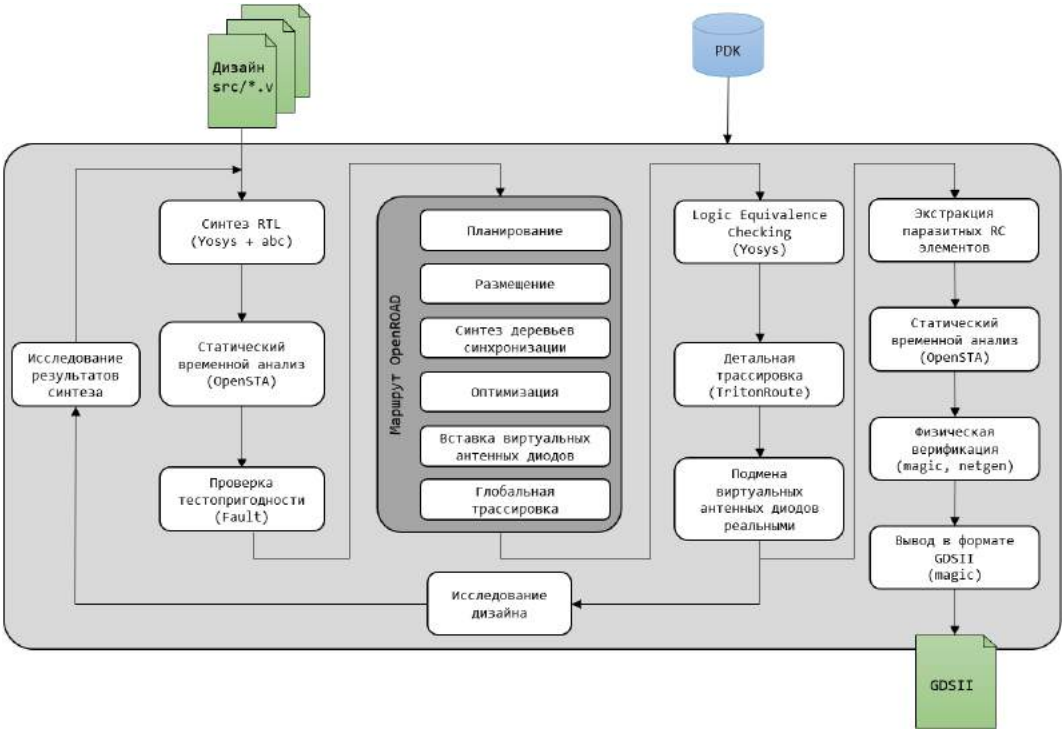


Рис. 1. Открытый маршрут проектирования цифровых интегральных схем OpenLane
Fig. 1. Open flow OpenLane for digital integrated circuits design

Модель получаемой физической структуры можно сравнить с моделью плоского конденсатора с большой разностью потенциалов на обкладках. При этом условно металл, накапливающий потенциал и подключённый к затвору транзистора, называется «антенной» [5].

Для решения этой проблемы в составе коммерческих САПР включены программы, анализирующие нарушения антенных правил и автоматически исправляющие топологию схемы [6]. Допустимые значения для антенных правил определяются записями в LEF-файлах, входящих в состав PDK. Каждое правило представляют из себя ограничение численного значения допустимой величины соотношения площадей металла и затвора и определяется формулой (1):

$$AntennaRatio = \frac{S_{Me}}{S_{Gate}} = \frac{Width_{Metal} \cdot Length_{Metal}}{Width_{Gate} \cdot Length_{Gate}}; \quad (1)$$

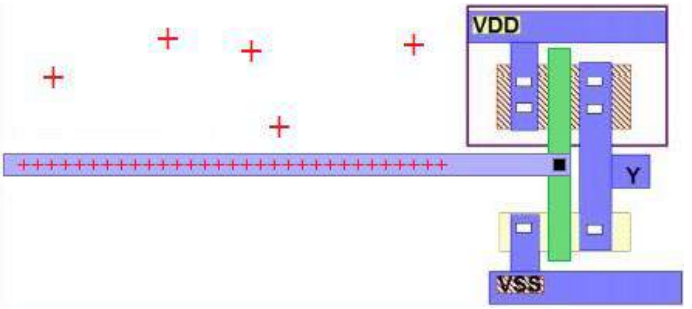


Рис. 2. Иллюстрация антенного эффекта
Fig. 2. Antenna effect illustration

3. Существующие пути устранения нарушений проверки антенных правил

Существует несколько решений для устранения нарушений антенных правил.

Первый подход заключается в размещении в непосредственной близости от затворов транзисторов, к которым подводится металл с нарушением антенных правил, диодов (они так и называются – антенные диоды), которые перетягивают на себя положительный потенциал в ходе формирования слоя металлизации.

Второй подход заключается в размещении в участках металлов специальных межслойных переходов, выглядящих как перемычки с переходом на верхний слой, называемых джамперми (от англ. «jump» - прыгать, перепрыгивать). Оба подхода показаны на рис. 3.

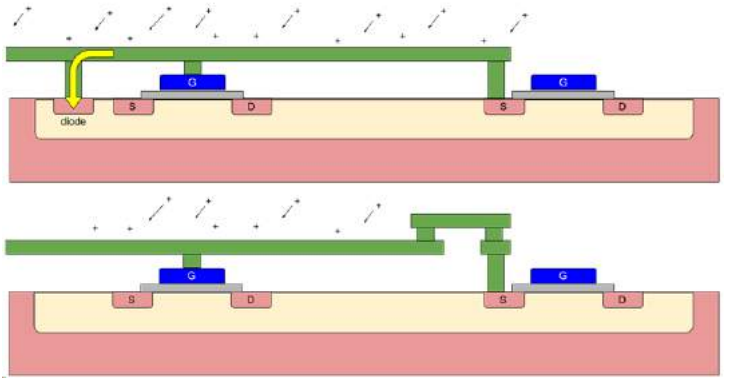


Рис. 3. Иллюстрация устранения эффекта: с помощью антенных диодов (вверху) и вставки перемычек (внизу)
Fig. 3. Illustration of antenna violations eliminations using antenna diodes (at the top) and jumpers (at the bottom)

Из рассмотренных вариантов маршрут OpenLane позволяет управлять только расстановкой антенных диодов путём задания значения специальной переменной, отвечающей в конфигурационном файле дизайна за алгоритм вставки антенных диодов (в файле config.tcl для этого используется переменная DIODE_INSERTION_STRATEGY). Всего реализовано 6 основных стратегий:

- 0 – без добавления антенных диодов;
- 1 – вставка диодов рядом с каждой ячейкой, соединения реализуются насильно;
- 2 – вставка диодов без подключений (fake diodes) рядом с каждой ячейкой с последующим подключением диодов только в тех местах, где проверка выявила нарушения антенных правил;

- 3 – используется программа FastRoute, которая на этапе глобальной трассировки пытается избежать формирования длин металлов, нарушающих антенные правила (имеется возможность задать число перетрассировок);
- 4 – используется эвристика, позволяющая оценить длины межсоединений ещё на этапе размещения;
- 5 – комбинация стратегий 2 и 4.

Несмотря на наличие в маршруте OpenLane возможности использования различных стратегий вставки антенных диодов, проведённые эксперименты показали, что, хотя включение алгоритмов вставки диодов и приводит к уменьшению количества нарушений, для дизайнов с числом вентилей от 100 штук никакой алгоритм вставки диодов не приводит к полному исчезновению таких нарушений. В табл. 1 показаны результаты применения различных стратегий для дизайнов s44 (121 ячейка) и picorv32a (9659 ячеек) из набора Google SkyWater PDK 130nm [7].

Табл. 1. Результаты применения различных стратегий вставки антенных диодов
Table 1. Results for different diode insertion strategies

Стратегия	Схема s44	Схема picorv32a
	Количество нарушений	
0	49	453
1	8	345
2	6	411
3	1	55
4	1	44
5	7	327

При этом, за счёт вставки антенных диодов, очевидно, существенно возрастает используемая площадь кристалла и повышается сложность и плотность трассировки.

Поскольку использование алгоритмов вставки антенных диодов не даёт полного исправления нарушений, был реализован алгоритм вставки перемычек, описываемый в данной работе.

4. Разработка алгоритма вставки перемычек

Работа алгоритма рассмотрена на примере использования PDK Google SkyWater 130nm [7].

Входными данными для работы программы являются DEF-файл с описанием результатов размещения и трассировки для дизайна, LEF-файл с описанием библиотеки стандартных ячеек, а также файл отчёта программы проверки нарушений антенных правил ARC, входящей в состав программ маршрута OpenLane. Пример участка файла отчёта о нарушении антенных правил приведён в листинге 1.

Из приведённого листинга видно, что нарушение произошло в цепи с именем net5, которая подходит к пину B_N экземпляра _102_ячейки sky130_fd_sc_hd__or2b_1 в первом металле.

Алгоритм работы программы строится на основе анализа входных файлов, в общем виде он показан на рис. 4. На первом этапе проводится проверка соответствия входных файлов: есть ли соответствие списка цепей с нарушениями из файла лога списку цепей трассировки из файла DEF. Затем выполняется обход всех цепей с нарушениями и для каждой такой цепи производится поиск самого длинного металла, подключённого к затворам транзисторов входного каскада ячейки (информация берётся из LEF-файлов).

. . .

```
Net net5
  _102_/B_N (sky130_fd_sc_hd__or2b_1)
    met2
    PAR:    8.64  Ratio:    0.00 (Area)
    PAR:   43.93  Ratio: 2778.20 (S.Area)
    CAR:  173.36  Ratio:    0.00 (C.Area)
    CAR:  870.17  Ratio:    0.00 (C.S.Area)

    met1
    PAR:  164.60  Ratio:    0.00 (Area)
    PAR:  826.11* Ratio:  400.00 (S.Area)
    CAR:  164.71  Ratio:    0.00 (C.Area)
    CAR:  826.25  Ratio:    0.00 (C.S.Area)
    ...
```

Листинг 1. Описание нарушения антенных правил для участка первого металла цепи net5
Listing 1. Description of antenna rule violation for the first metal of net5 net

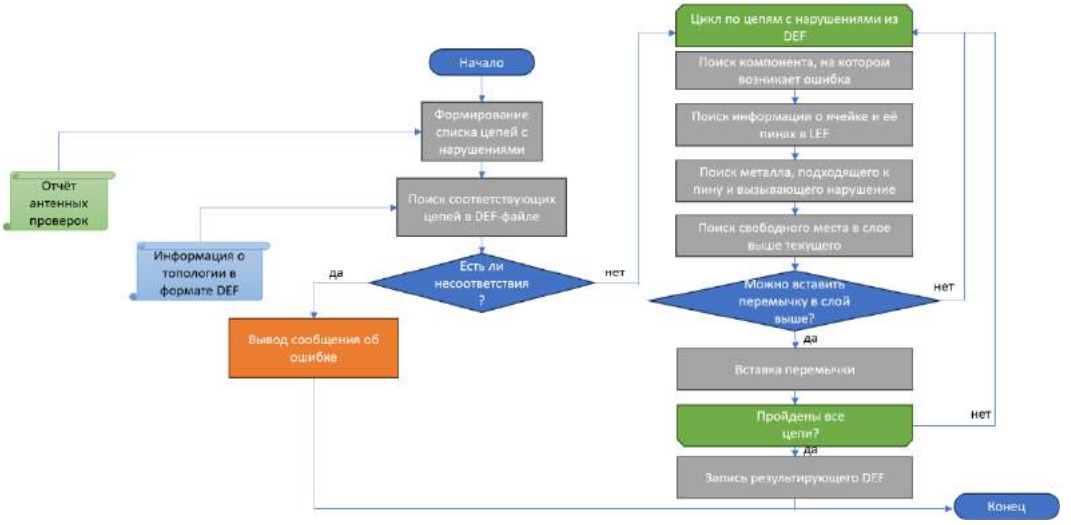


Рис. 4. Алгоритм работы программы
Fig. 4. The algorithm

Вставка межслойного перехода осуществляется следующим образом:

- для самого длинного сегмента происходит поиск всех пересечений в более высоком слое (если это возможно), поиск сегментов оканчивается формированием массива интервалов – доступных расстояний между пересекающими текущий слой металлами;
- в найденном массиве интервалов ищется самое большое расстояние, при это необходимым условием является то, что самый большой интервал должен быть больше, чем два расстояния шага металла в слое;
- если такой участок находится, в него происходит вставка межслойного перехода.

Алгоритм не способен исправить нарушение в одном из двух случаев:

1. если нарушение обнаружено на самом верхнем слое металла; очевидно, что в этом случае физически некуда вставлять межслойный переход (потому что нарушение обнаружено на самом верхнем уровне);
2. если над слоем с нарушением нет места для вставки перемычки из-за обилия

проходящих дорожек металлизации; такая ситуация встречается довольно редко, но, тем не менее, встречается для некоторых тестовых дизайнов.

В результате работы программы формируется выходной DEF-файл, участок такого файла показан на рис. 5. Из рисунка видно, что длинный участок металла первого слоя разбивается на две части с размещением на концах разбиения двух межслойных переходов и вставкой небольшого участка металла во втором слое, размер которого равен ширине шага трека в слое.

```
NETS 168 ;
    ...
  - net5 ( input5 X ) ( _096_ B_N ) ( _097_ A1 ) ( _098_ S1 )
       ( _102_ B_N ) ( _103_ A1 ) ( _104_ S1 ) + USE SIGNAL
    + ROUTED met2 ( 8970 197370 ) ( * 199070 )
    NEW met1 ( 155710 197030 ) ( * 197370 )
    NEW met2 ( 156630 194650 ) ( * 197030 )
    NEW met1 ( 155710 197030 ) ( 156630 * )
    NEW met1 ( 8970 197370 ) ( 155710 * )
    NEW met1 ( 140070 175270 ) ( 140530 * )
    NEW met2 ( 140990 170510 ) ( * 175270 )
    ...
```

```
NETS 168 ;
    ...
  - net5 ( input5 X ) ( _096_ B_N ) ( _097_ A1 ) ( _098_ S1 )
      ( _102_ B_N ) ( _103_ A1 ) ( _104_ S1 ) + USE SIGNAL
    + ROUTED met2 ( 8970 197370 ) ( * 199070 )
    NEW met1 ( 155710 197030 ) ( * 197370 )
    NEW met2 ( 156630 194650 ) ( * 197030 )
    NEW met1 ( 155710 197030 ) ( 156630 * )
    NEW met1 ( 8970 197370 ) ( 82170 * )
    NEW met1 ( 82170 197370 ) M1M2_PR
    NEW met2 ( 82170 197370 ) ( 82510 * )
    NEW met1 ( 82510 197370 ) M1M2_PR
    NEW met1 ( 82510 197370 ) ( 155710 * )
    NEW met1 ( 140070 175270 ) ( 140530 * )
    NEW met2 ( 140990 170510 ) ( * 175270 )
```

Рис. 5. Входной файл DEF и результат вставки перемычки в выходном файле DEF
Fig. 5. The input DEF file and the result of jumper insertion in output DEF file

5. Параллельная реализация алгоритма

На рис. 6 показаны результаты измерения времени работы составных частей программы для различных дизайнов.

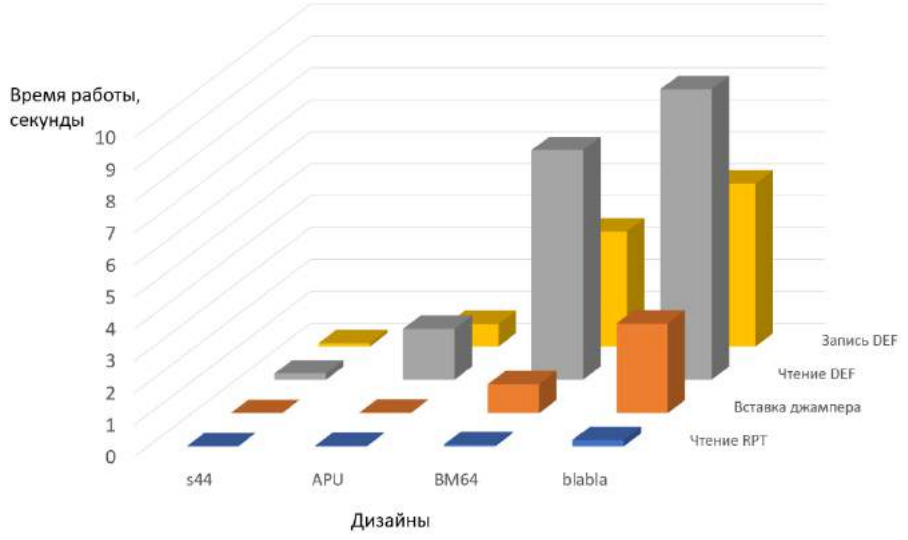


Рис. 6. Временные характеристики работы различных частей программы
Fig. 6. Different parts of the program temporal characteristics

Алгоритм обработки нарушений антенных правил достаточно хорошо распараллеливается как в части обработки участков металла для вставки перемычек, так и в части чтения входных данных. В части обработки участков металла он хорошо распараллеливается, поскольку при вставке перемычек отсутствует конкуренция по доступу к списку цепей для записи. Каждая операция работает со своим набором цепей и между различными потоками не возникает конфликта. В части чтения входных данных распараллеливание даёт сокращение суммарного

времени чтения входных данных. Распараллеливание реализовано с использованием библиотеки pthreads. Результаты распараллеливания с точки зрения времени работы показаны на рис. 7. Для всех тестовых наборов данных прирост производительности составляет не менее 60% при распараллеливании на 4 ядра.

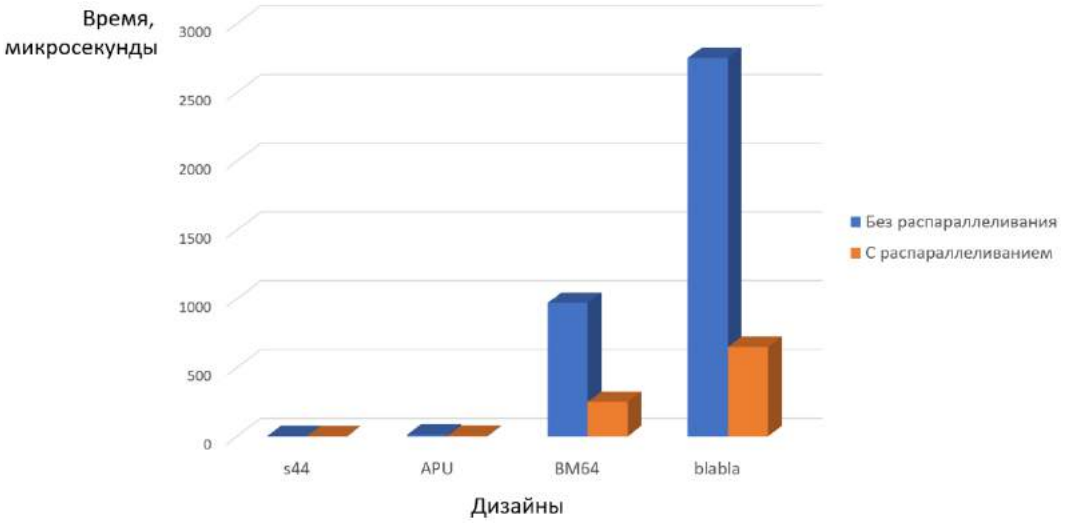


Рис. 7. Результаты сравнения времени выполнения последовательной (слева) и параллельной (справа) реализации алгоритмов для набора тестовых схем
Fig. 7. The comparison results of run time for sequential (on the left) and parallel (on the right) realizations of the algorithm for set of test designs

6. Использование программы в маршруте OpenLane

Разработанная программа встраивается в маршрут OpenLane и вызывается автоматически наряду с остальными программами маршрута. Схема модифицированного маршрута показана на рис. 8.

Управление очерёдностью вызова программ маршрута OpenLane управляет скрипт на языке Tcl. Для автоматизированного вызова скрипта необходимо запустить на выполнение программу, а затем повторно запустить выполнявшиеся ранее в скрипте проверки DRC и LVS. Эту процедуру нужно выполнять до тех пор, пока либо число обнаруженных нарушение не будет сведено к нулю, либо не будет достигнуто заданное фиксированное число итераций при неизменном результате. Кратко этот подход проиллюстрирован в листинге 2.

7. Выводы

Рассмотренный в данной статье алгоритм вставки межслойных переходов для исправления нарушений антенных правил позволяет исправить большую часть нарушений. Для схем из набора Google SkyWater PDK 130nm была получена следующая статистика:

- для небольших схем (до 10 тысяч вентилей) разработанный алгоритм позволяет исправлять 100% нарушений, при этом в случае выбора алгоритма разбиения по самому длинному металлу (без указания LEF файл) число итераций работы алгоритма доходило до трёх;
- для больших схем (от 10 до 25 тысяч вентилей) наблюдались ситуации, когда вставка межслойных переходов была невозможна из-за того, что либо над слоем металла, который нужно разбить нет места, либо нарушения встречаются в самом верхнем слое металлов.

Для устранения 100% нарушений самым оптимальным видится перестроение маршрута OpenLane таким образом, чтобы в его составе применялись программы размещения и трассировки, учитывающие возникновения антенных нарушений в ходе своей работы [8, 9].

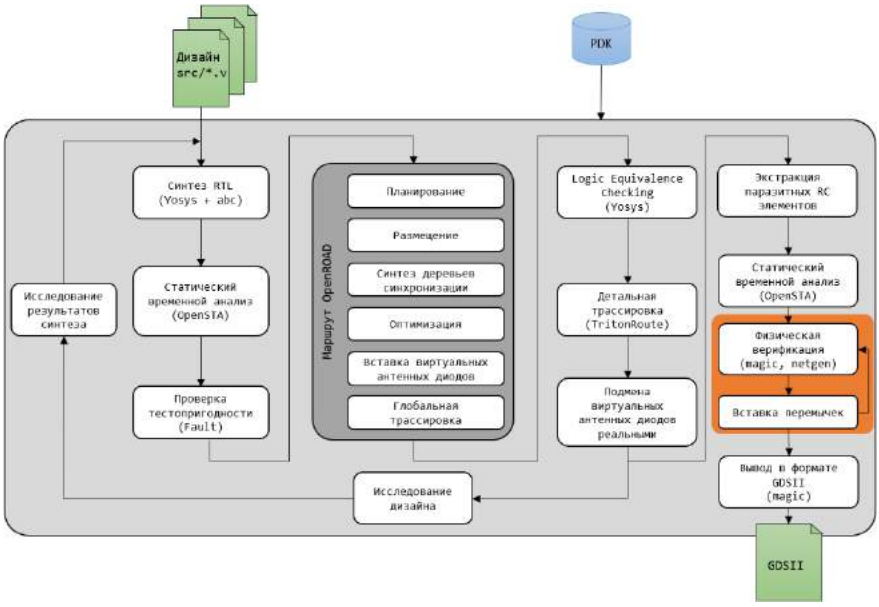


Рис. 8. Модифицированный маршрут OpenLane с вызовом программы вставки перемычек
Fig. 8. Modified flow OpenLane with jumper insertion software call

```
if {$step_name == "antenna_check"} {
  set repair_iteration 0
  set num_violations_prev -1
  set do_antenna_repair 1
  if {$num_violations_now != 0} {
    while {$do_antenna_repair} {
      exec ./JI2 --def $def --report $ant_log --out $def
      set num_violations_prev $num_violations_now
      set local_steps [dict create \
        "lvs" "run_lvs_step $LVS_ENABLED " \
        "drc" "run_drc_step $DRC_ENABLED " \
        "antenna_check" "run_antenna_check_step $ANTENNACHECK_ENABLED"
      ]
      dict for {step_name step_exe} $local_steps {
        set ::env(CURRENT_STEP) $step_name
        [lindex $step_exe 0] [lindex $step_exe 1] ;
      }
    }
  }
  ...
}
```

Листинг 2. Часть кода модифицированного скрипта flow.tcl
Listing 2. Example of the modified flow.tcl script source code

Список литературы / References

[1]. M. Shalan and T. Edwards, "Building OpenLANE: A 130nm OpenROAD-based Tapeout- Proven Flow : Invited Paper," 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA, 2020, pp. 1-6.

[2]. Зыков А.В., Ильясов Р.Ф. Анализ работы программных инструментов с открытым исходным кодом OpenLANE для разработки микросхем // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2022. Выпуск 4. С. 87-92. doi:10.31114/2078-7707-2022-4-87-92

[3]. T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo and B. Xu, "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project," Proc. ACM/IEEE Design Automation Conference, 2019, pp. 76:1-76:4. (Invited Paper)

[4]. S. Reda, "Overview of the OpenROAD Digital Design Flow from RTL to GDS," 2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 2020, pp. 1-1, doi: 10.1109/VLSI-DAT49148.2020.9196319.

[5]. Lin, Chung-Wei & Tsai, Ming-Chao & Lee, Kuang-Yao & Chen, Tai-Chen & Wang, Ting-Chi & Chang, Yao-Wen. (2007). Recent Research and Emerging Challenges in Physical Design for Manufacturability/Reliability. Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC. 238-243. 10.1109/ASPDAC.2007.357992.

[6]. Кравченко, В. САПР компании Synopsys. Основные средства и возможности / В. Кравченко, Д. Радченко // ЭЛЕКТРОНИКА: Наука, Технология, Бизнес. — Выпуск 5. — 2003. — С. 31-33.

[7]. K. Herman, M. Montanares and J. Marin, "Design and Implementation of Integrated Circuits Using Open Source Tools and SKY130 Free PDK," 2023 30th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Kraków, Poland, 2023, pp. 105-110, doi: 10.23919/MIXDES58562.2023.10203216.

[8]. Tsai, Chia-Chun & Hsu, Feng-Tzu & Kuo, Chung-Chieh & Wu, Jan-Ou & Lee, Trong-Yen. (2008). X-clock tree construction for antenna avoidance. International Conference on Solid-State and Integrated Circuits Technology Proceedings, ICSICT. 2248 - 2251. 10.1109/ICSICT.2008.4735038.

[9]. Cho, Minsik & Mitra, Joydeep & Pan, David. (2008). Manufacturability Aware Routing. 10.1201/9781420013481.ch38.

Информация об авторах / Information about authors

Дмитрий Александрович БУЛАХ – кандидат технических наук, доцент института интегральной электроники. Сфера научных интересов: алгоритмы моделирования, обработки и визуализации данных в САПР СБИС.

Dmitriy Aleksandrovich BULAKH – Candidate of Technical Sciences, Assistant Professor at the Institute of Integrated Electronics. Research interests: VLSI EDA data simulation, processing and visualization algorithms.

Андрей Владимирович КОРШУНОВ – кандидат технических наук, доцент института интегральной электроники, доцент. Сфера научных интересов: исследование и разработка методов проектирования энергоэффективных СБИС и систем на кристалле.

Andray Vladimirovich KORSHUNOV – Candidate of Technical Sciences, Assistant Professor at the Institute of Integrated Electronics. Research interests: research and development of design methods for energy-efficient VLSI and SoCs.

DOI: 10.15514/ISPRAS-2023-35(6)-13

Применение алгоритмов машинного обучения для предсказания турбулентной вязкости

[1, 2] *Д.И. Романова,* ORCID: 0000-0002-5771-4114 <*romanovadi@gmail.com*>
[1] *А.С. Епихин,* ORCID: 0000-0002-6249-4283 <*andreyepikhin@ispras.ru*>
[1] *Д.Ю. Ильина,* ORCID: 0009-0009-6321-7155 <*ilina.diu@ispras.ru*>

[1] *Институт системного программирования РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
[2] *Московский государственный университет имени М.В. Ломоносова,*
Россия, 119991, Москва, Ленинские горы, д. 1.

Аннотация. В работе проведено исследование алгоритмов машинного обучения для предсказания турбулентной вязкости на примере течения за обратным уступом. Данные для обучения получены с помощью расчёта с применением программного комплекса OpenFOAM и модели турбулентности $\boldsymbol{k-\varepsilon}$. Для предсказания турбулентной вязкости выполнен анализ значимости параметров течения, включающих пульсации скоростей, градиенты давления и скорости, инварианты тензора скоростей деформаций и их комбинации. Произведено сравнение различных алгоритмов машинного обучения и проанализирована значимость входных признаков. Получено, что наиболее оптимальный алгоритм для предсказания турбулентной вязкости в данной задаче является Decision Tree Regressor. С помощью выбранной модели выполнено предсказание распределения турбулентной вязкости в расчётной области для различных чисел Рейнольдса.

Ключевые слова: турбулентная вязкость; машинное обучение; обратный уступ; осреднённые по Рейнольдсу уравнения Навье-Стокса; OpenFOAM.

Для цитирования: Романова Д.И., Епихин А.С., Ильина Д.Ю. Применение алгоритмов машинного обучения для предсказания турбулентной вязкости. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 199–212. DOI: 10.15514/ISPRAS–2023–35(6)–13.

Благодарности: Исследование выполнено при поддержке Российского научного фонда (проект № 23-71-10091).

Application of Machine Learning Algorithms to Predict Turbulent Viscosity

[1,2] *D.I. Romanova*, ORCID: 0000-0002-5771-4114 <romanovadi@gmail.com>
[1] *A.S. Epikhin*, ORCID: 0000-0002-6249-4283 <andreyepikhin@ispras.ru>
[1] *D.Yu. Ilina*, ORCID: 0009-0009-6321-7155 <ilina.diu@ispras.ru>

[1] *Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia.*

Abstract. The article discusses machine learning algorithms for predicting turbulent viscosity using the case of flow past the backward-facing step. The training data is obtained by calculations using the OpenFOAM software package and a $\boldsymbol{k-\varepsilon}$ turbulence model. The significance of flow parameters, including velocity fluctuations, pressure and velocity gradients, strain rate tensor and their combinations and invariants are analyzed for predicting turbulent viscosity. Different machine learning algorithms are compared. It is found that the most optimal algorithm for predicting turbulent viscosity in this case is the Decision Tree Regressor. Using the chosen model, the distribution of turbulent viscosity in the computational domain is predicted for various Reynolds numbers.

Keywords: turbulent viscosity; machine learning; backward step; Reynolds-averaged Navier-Stokes equations; OpenFOAM.

For citation: Romanova D.I., Epikhin A.S., Ilina D.Yu. Application of machine learning algorithms to predict turbulent viscosity. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 199-212 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-13.

Acknowledgements. The research was supported by the Russian Science Foundation, grant No. 23-71-10091.

1. Введение

Алгоритмы машинного обучения (МО) стали мощным подходом для решения проблем во многих областях, таких как обработка и классификация изображений, поисковые и рекомендательные системы, распознавание речи, автопилот и здравоохранение, а также многие другие. Расширение области применения МО привело к созданию новых алгоритмов в гидро- и аэромеханике. Что крайне важно для области, так как позволяет получить более точное решение с использованием меньших вычислительных затрат. Например, в работе [1] используется технология SD (Shallow Decoder) для предсказания точной картины турбулентного течения на основе данных осреднённого моделирования на грубой сетке.

В последнее время физически обусловленные нейронные сети (Physics-Informed Neural Networks (PINN)) получили широкое распространение и показывают хороший результат при гладком решении [2]–[5]. Однако их производительность падает при наличии резких градиентов или разрывов. В работах [6]–[9] нейронные сети используются для предсказания турбулентных напряжений в различных вариантах. В частности, в работе [6] используются TBNN (Tensor Basis Neural Network) для предсказания анизотропного тензора напряжений Рейнольдса.

В работе [7] используются две нейронных сети, первая из которых предсказывает оптимальную турбулентную вязкость, а вторая для предсказания нелинейной части напряжений Рейнольдса. Для обучения нейронной сети используется DNS (Direct Numerical Simulation) расчёт задачи течения над периодическим холмом. Расчёт с помощью полученной модели по своему качеству превзошел RANS моделирование с использованием модели турбулентности.

Кластеризация данных используется в работе [8] для выявления областей для обучения напряжения, позволяя увеличить эффективность обучения нейронной сети, которая

предсказывает напряжения Рейнольдса. Такого рода подход позволяет отбросить часть данных, не представляющую интереса для обучения, которая составляет довольно большую часть датасета. Работа [9] посвящена модификациям стандартной нейронной сети для учёта граничных условий при вычислении анизотропной части рейнольдсовых напряжений. Модель также принимает на вход число Рейнольдса, что позволило получить улучшение решения в сравнении с работой [6].

Глубокая нейронная сеть, обученная на DNS данных, используется в работе [10] для предсказания напряжений Рейнольдса вместо гипотезы Буссинеска. В качестве входных признаков для нейронной сети авторы используют такие RANS данные, как турбулентная кинетическая энергия, диссипация турбулентной кинетической энергии, скорость, градиент скорости, градиент давления, тензор деформаций, и расстояние до стенки. Расчёт с использованием предложенной модели значительно превосходит точностью классический RANS расчёт с использованием модели турбулентности.

В перечисленных работах представлены возможности большого класса моделей машинного обучения, однако прямое использование их затруднено в силу отсутствия реализации описанных алгоритмов в открытом доступе. Таким образом, актуальность настоящей работы заключается в исследовании алгоритмов машинного обучения и выборе оптимальной модели для предсказания турбулентной вязкости, что в дальнейшем позволит предложить новую модель турбулентности. Преимуществом такого рода модели является их универсальность, так как большинство существующих RANS моделей турбулентности имеют достаточно ограниченную область применимости и часто требуют калибровки коэффициентов для точного расчёта той или иной задачи [11]–[16].

2. Обзор форматирования

Рассматривается классическая задача течения за обратным уступом, которая подробно описана в работе [17]. Расчёт потока проводится с использованием осреднённых по Рейнольдсу уравнений Навье–Стокса:

$$\begin{cases} \nabla \cdot U = 0, \\ \frac{\partial U}{\partial t} + (U \cdot \nabla)U = \frac{-1}{\rho}\nabla p + (\nu_m + \nu_t)\nabla^2 U. \end{cases} \quad (1)$$

Здесь U — средняя по Рейнольдсу скорость, p — давление с учётом турбулентной кинетической энергии, $\rho = \text{const}$ — плотность потока, ν_m — молекулярная кинематическая вязкость, ν_t — турбулентная вязкость, $s = 0.5(\nabla U + (\nabla U)^T)$ — осреднённый тензор скоростей деформаций.

2.1 Расчет турбулентной вязкости

Расчёт турбулентной вязкости проводится с использованием $k - \varepsilon$ модели турбулентности [18], [19]. В данной модели турбулентная кинематическая вязкость вычисляется по формуле (2).

$$\nu_t = \frac{C_\mu k^2}{\varepsilon}, \quad (2)$$

где k — турбулентная кинетическая энергия; ε — диссипация турбулентной кинетической энергии, C_μ — коэффициент турбулентной модели.

Для расчёта k и εсистема уравнений (1) дополняется двумя уравнениями (3).

$$\begin{cases} \frac{\partial k}{\partial t} + (U \cdot \nabla)k = \nabla \cdot (\nu_m \nabla k) - \nabla \cdot \left(\frac{\nu_t}{\sigma_k}\nabla k + P_k\right) - \varepsilon, \\ \frac{\partial \varepsilon}{\partial t} + (U \cdot \nabla)\varepsilon = C_{\varepsilon 1} P_k \frac{\varepsilon}{k} - C_{\varepsilon 2}\frac{\varepsilon^2}{k} + \nabla \cdot \left(\frac{\nu_t}{\sigma_\varepsilon}\nabla \varepsilon\right). \end{cases} \quad (3)$$

Здесь $P_k = \nu_t \nabla U \cdot [\nabla U + (\nabla U)^T]$ — скорость производства турбулентной кинетической энергии средним течением, $C_\mu = 0.09$, $C_{\varepsilon 1} = 1.44$, $C_{\varepsilon 2} = 1.92$, $\sigma_k = 1.0$, $\sigma_\varepsilon = 1.3$ — коэффициенты модели турбулентности.

2.2 Предсказание турбулентной вязкости при помощи алгоритмов машинного обучения

Предсказание турбулентной вязкости с помощью алгоритмов машинного обучения происходит на основе обезразмеренных средних параметров течения, и их комбинаций, перечисленных в следующем разделе, по формуле (4).

$$\nu_t = g\left(f_i(\vec{U}, p)\right), \tag{4}$$

где g — некоторая функция, определяемая используемым алгоритмом машинного обучения. Обезразмеривание проводится с помощью параметров U_0, ν_m и H — характерная длина (в задаче с обратным уступом это высота уступа).

2.3 Начальные и граничные условия

Для замыкания математической модели, приведённой в предыдущем разделе необходимо определить начальные и граничные условия в задаче течения над обратным уступом, показанной на рис. 1. Высота ступеньки $H = 0.025$ м.

В начальный момент времени среда в расчётной области покоится, молекулярная кинематическая вязкость $\nu_m = 1.5114 \cdot 10^{-5}$ м^2/с. На входе в расчётную область (слева на рис. 1) поток имеет постоянную скорость $U_0 = 13.3$ м/с. Тогда число Рейнольдса $\mathrm{Re} = U_0 H/\nu_m = 22000$. На выходе расчётной области (справа на рис. 1) задано условие нулевого градиента. На верхней и нижней границах канала стоит условие прилипания. Задача рассматривается в двумерной постановке, на передней и задней гранях граничные условия не задаются. Давление во всей расчётной области в начальный момент времени задано $p_0 = 0$ м^2/с^2. Расчётная область содержит 2991 ячеек.

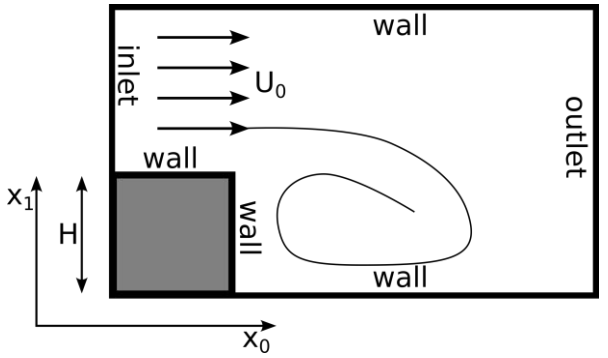


Рис. 1. Задача течения над обратным уступом
Fig. 1. Flow past backward facing step

Турбулентная кинетическая энергия на входе задаётся значением $k = 0.2388$ м^2/с^2, которое получается по формуле $k = 3/2 \cdot (U_0 \cdot I)^2$, где $I = 0.03$ — интенсивность турбулентности. На стенках используются пристеночные функции, на выходе — условие нулевого градиента. Диссипация турбулентной кинетической энергии на входе задаётся значением $\varepsilon = 15.3401$ м^2/с^3, которое вычисляется по формуле $\varepsilon = C_\mu^{3/4} \cdot k^{3/2}/l$, где $l = 0.05 \cdot h$. Так же, как и в случае турбулентной кинетической энергии на стенках используются пристеночные функции, на выходе – условие нулевого градиента (табл. 1).

Для валидации решения использовались расчёты при других числах Рейнольдса – $\mathrm{Re} = 20000{,}15000$, которые достигались увеличением молекулярной кинематической вязкости в расчёте до $\nu_m = 1.6625 \cdot 10^{-5}$ и $2.21667 \cdot 10^{-5}$ м^2/с соответственно.

Табл.1. Граничные условия в пакете OpenFOAM
Table 1. Boundary conditions in OpenFOAM package

	inlet	**outlet**	**wall**
U	13.3 м/с	zeroGradient	noSlip
p	zeroGradient	0 м^2/с^2	zeroGradient
k	0.2388 м^2/с^2	zeroGradient	kqRWallFunction
ε	15.3401 м^2/с^3	zeroGradient	epsilonWallFunction

3 Обучающая выборка

3.1 Начальные и граничные условия

С применением пакета OpenFOAM проведен расчет течения за обратным уступом и получена обучающая выборка в соответствии с математической моделью, приведённой в разделе 2. На рис. 1 приведено сравнение численного моделирования с экспериментом [17].

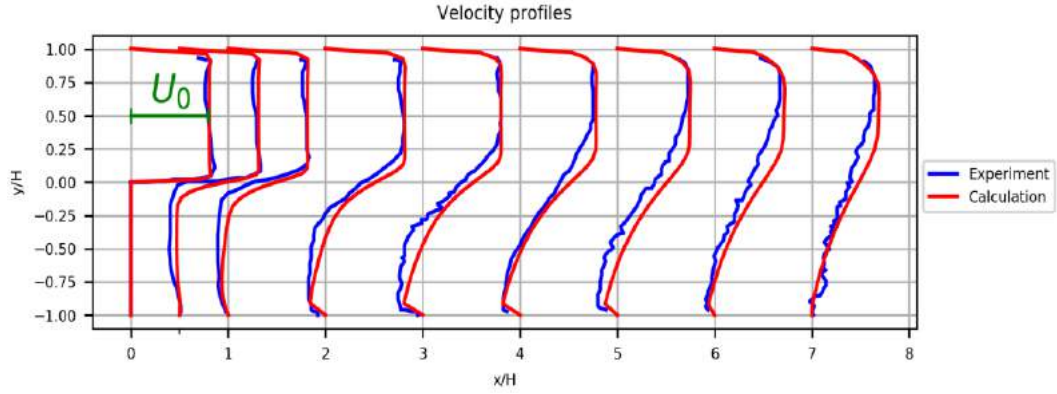


Рис. 2. Профили скорости в различных сечениях за обратным уступом
Fig. 2. Velocity profiles in various sections in flow past backward facing step

Среднеквадратичное отклонение расчётных профилей от экспериментальных, вычисляемое по 270 точкам (по 30 точек в каждом сечении) составило 0.7%. Исходя из этого значения, можно считать полученный расчёт достаточно точным. В настоящей работе полученные расчётные данные будут использованы для обучения моделей, предсказывающих турбулентную вязкость.

В качестве обучающей выборки берутся данные с временных срезов, взятых каждые 0.01 с с нулевого момента времени, по момент времени 0.3 с, когда происходит выход на стационар. Итого получается 30 временных срезов, 29 из которых берутся для обучения и 1 временной срез в момент 0.23 с — для тестирования. Каждый из срезов по времени содержит записи в 2991 пространственной точке, каждая из которых состоит из 130 признаков (всего датасет содержит 89730 строк).

В тестовый момент времени 0.23 с турбулентная вязкость имеет следующее распределение в расчётной области, которое показано на рис. 2.

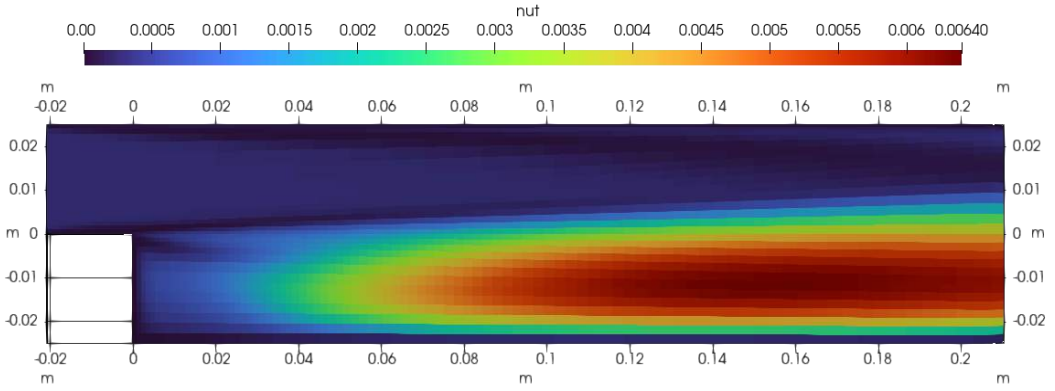


Рис. 3. Распределение турбулентной вязкости в расчётной области в момент времени 0.23 с, полученное с помощью $k-\varepsilon$ модели турбулентности
Fig. 3. Distribution of turbulent viscosity in the computational domain at time 0.23 s, obtained using the $k-\varepsilon$ turbulence model

3.2 Исследование параметров течения, используемых в качестве входных признаков

В настоящем разделе будут перечисленные всевозможные параметры течения и обоснования значимости их использования для предсказания турбулентной вязкости. В следующем разделе будет проведено подробное исследование вклада каждого из параметров в результат предсказания, и часть параметров будет исключена. Параметры течения и их комбинации, которые возможно использовать алгоритмами МО в качестве входных признаков можно разделить на три логические группы.

Первая группа содержит основные параметры течения — это давление p, компоненты скорости U_i, магнитуда скорости $|U|$, среднее по Рейнольдсу значение скорости $\overline{U}$ и квадрат пульсаций скорости $\overline{U'}^2$. Здесь и ниже используются обезразмеренные параметры, но знак волны над ними опускается.

Следующая группа входных признаков обусловлена общим видом зависимости турбулентной вязкости от параметров течения наиболее распространённым в настоящее время. Исходя из гипотезы Буссинеска о линейной зависимости турбулентной вязкости от тензора скоростей деформаций s, компоненты последнего используются в качестве входных признаков. Также существует и обобщенная гипотеза, предложенная Поупом [20], которая включает в себя зависимость не только от тензора скоростей деформаций s, но и от тензора завихренности $r = 0.5(\nabla U - (\nabla U)^T)$ и других параметров, приведённых в (5) и (6). Так же к этой группе признаков относится параметр y — расстояние до стенки.

Разрабатываемые алгоритмы имеют своей целью предсказание турбулентной вязкости в каждой из ячеек расчётной сетки, без ограничений на геометрию задачи и её расчётной сетки. В связи с этим появляется необходимость передать информацию о течении в окрестности точки предсказания. Для этого используется третья группа входных признаков, которая содержит градиенты основных параметров и их инварианты: $\nabla U, I_1(\nabla U), I_2(\nabla U), \nabla p, |\nabla p|$. В этой группе также представлен параметр V — объём ячеек расчётной ячейки.

$$\lambda_1 = Tr(s^2), \lambda_2 = Tr(r^2), \lambda_3 = Tr(s^3),$$
$$\lambda_4 = Tr(r^2 s), \lambda_5 = Tr(r^2 s^2). \quad (5)$$

$$
\begin{aligned}
T^{(1)} &= s, & T^{(6)} &= r^2 s + sr^2 - \frac{2}{3} I \cdot Tr(sr^2),\\
T^{(2)} &= sr - rs, & T^{(7)} &= rsr^2 - r^2 sr,\\
T^{(3)} &= s^2 - \frac{1}{3} I \cdot Tr(s^2), & T^{(8)} &= srs^2 - s^2 rs,\\
T^{(4)} &= r^2 - \frac{1}{3} I \cdot Tr(r^2), & T^{(9)} &= r^2 s^2 + s^2 r^2 - \frac{2}{3} I \cdot Tr(s^2 r^2),\\
T^{(5)} &= rs^2 - s^2 r, & T^{(10)} &= rs^2 r^2 - r^2 s^2 r,
\end{aligned}
\qquad (6)
$$

Выше был перечислен полный список используемых входных признаков, который в дальнейшем будет сокращён.

4 Анализ входных признаков

Всего среди входных параметров получилось 130 признаков. Данная цифра включает в себя компоненты векторов и тензоров по отдельности. Необходимо произвести оценку значимости каждого из входных признаков для оптимизации работы алгоритмов МО.

Из приведённых признаков лишь 58 являются линейно независимыми (у которых корреляция отлична от ±1), что связано с двумерной постановкой рассматриваемой задачи. Данные признаки будут представлять полную группу уникальных признаков. Среди линейно независимых параметров присутствуют компоненты следующих параметров: $|U|$, U, V, $\overline{U}$, p, y, $|\nabla p|$, ∇p, ∇U, λ_1, $\overline{U'}^2$, λ_5, $T^{(6)}$, $T^{(2)}$, $T^{(7)}$, λ_3, $T^{(10)}$, $T^{(9)}$, $I_2(\nabla U)$, $T^{(5)}$, $I_1(\nabla U)$, $T^{(1)}$. Признаки перечислены в порядке уменьшения корреляции с целевым параметром ν_t. Эти входные признаки будут использоваться далее в исследовании. Стоит также отметить, что данные признаки являются значимыми в двумерной задаче, в случае перехода на трёхмерный вариант, значимые признаки и их корреляции могут отличаться.

Объём ячеек расчётной сетки V хорошо коррелирует с целевым параметром турбулентной вязкости, что связано с принципом построения сетки — измельчение её в местах сложного течения. Таким образом при равномерном построении расчётной сетки алгоритмы плохо предсказывают целевой параметр, так как подгоняются по данному параметру. В следствии чего данный параметр был исключён из входных признаков.

В табл. 2 приведены 9 входных признаков и значение их корреляции с целевой переменной ν_t.

Табл.2. Корреляция входных признаков с ν_t
Table 2. Correlation of input features with ν_t

Признак	$\|U\|$	U_0	$\bar{U}_0$	p	U_1	y	$\bar{U}_1$	$\|\nabla p\|$
Корреляция с ν_t	0.51	0.48	0.40	0.37	0.34	0.29	0.28	0.26

Следующие параметры течения имеют корреляцию с целевым параметром выше 0.05: $|U|$, U, $\overline{U}$, p, y, $|\nabla p|$, ∇p, ∇U, λ_1, $\overline{U'}^2$, λ_5. Вышеперечисленные признаки выделяются в группу признаков высокой корреляции с целевым параметром.

Визуализация набора данных для 4 входных признаков: $|U|$, p, U_0, U_1, имеющих значительную корреляцию с целевым параметром ν_t приведена на рис. 4.

Различные признаки имеют сильно отличающиеся распределения, так, например, распределение параметров на рис. 4 значительно более равномерное, чем распределение параметров на рис. 5.

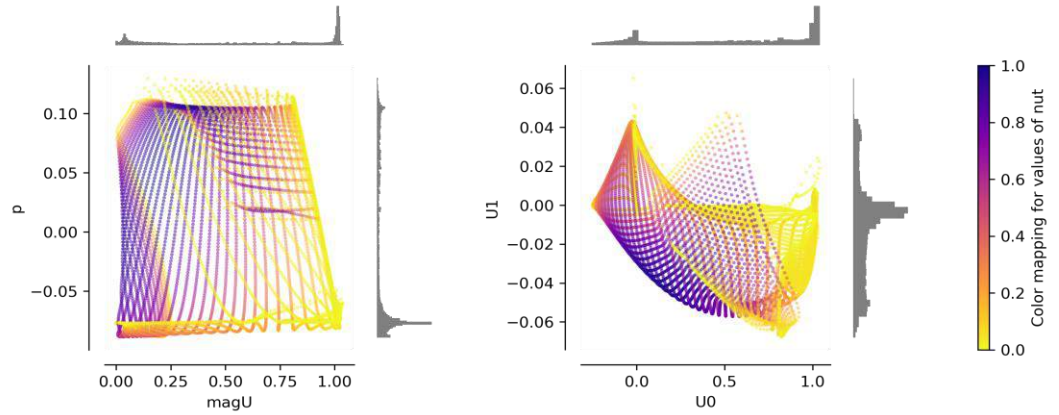


Рис. 4. Зависимость целевой переменной от входных признаков $|U|$, p, U_0, U_1, имеющих значительную корреляцию с целевым параметром

Fig. 4. Dependence of the target variable on input features $|U|$, p, U_0, U_1, having a significant correlation with the target parameter

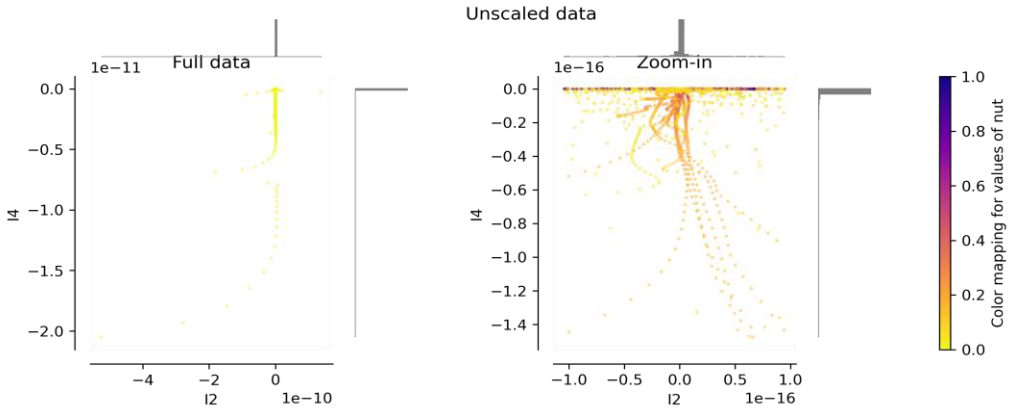


Рис. 5. Зависимость целевой переменной от λ_3 (I2) и λ_5 (I4) входных признаков с плохим распределением

Fig. 5. Dependence of the target variable on λ_3 (I2) and λ_5 (I4) input features with poor distribution

Большое количество выбросов, наблюдаемое на рис. 5 имеет сильное влияние на качество обучения и предсказания моделей. Для действенного уменьшения разброса данных подходят преобразователи данных, основанные на вероятностных характеристиках, такие, как, например, QuantileTransformer и PowerTransformer библиотеки scikit-learn. Однако данные преобразователи не сохраняют расстояний между точками, и в процессе использования будут существенно искажать данные при обобщении решения на другие задачи, в особенности, на задачи с другой геометрией расчётной сетки. В связи с этим было проведено исследование значимости входных параметров, которое показало, что сильно неравномерное распределение параметров плохо сказывается на процесс обучения. Тогда была выделена третья группа входных признаков, которая выбиралась по результатам сравнения распределения конкретного признака с равномерным распределением с помощью теста Колмогорова-Смирнова по двум выборкам. Распределение входного параметра считалось удовлетворительным, если p-value превышало 10^{-6}, либо параметр statistic отличен от единицы. Данная входная группа представлена следующими признаками: U, $\overline{U}$, $\overline{U'}^2$, $|U|$, $I_2(\nabla U)$, p.

Перечисленные три группы входных признаков приведены в табл. 3.

Табл.3. Три группы входных признаков, используемые для предсказания турбулентной вязкости
Table 3. Three groups of input features used for turbulent viscosity prediction

Все линейно-независимые признаки (Unique)	Признаки, обладающие высокой корреляцией с целевым параметром (Correlating)	Признаки с наиболее равномерным распределением (Uniform)
$\|\boldsymbol{U}\|, \boldsymbol{U}, \overline{\boldsymbol{U}}, \boldsymbol{p}, \boldsymbol{y}, \|\nabla \boldsymbol{p}\|, \nabla \boldsymbol{p}, \nabla \boldsymbol{U}, \boldsymbol{\lambda_1}, \overline{\boldsymbol{U'}}^{\boldsymbol{2}}, \boldsymbol{\lambda_5}, \boldsymbol{T^{(6)}}, \boldsymbol{T^{(2)}}, \boldsymbol{T^{(7)}}, \boldsymbol{\lambda_3}, \boldsymbol{T^{(10)}}, \boldsymbol{T^{(9)}}, \boldsymbol{I_2(\nabla U)}, \boldsymbol{T^{(5)}}, \boldsymbol{I_1(\nabla U)}, \boldsymbol{T^{(1)}}$	$\|\boldsymbol{U}\|, \boldsymbol{U}, \overline{\boldsymbol{U}}, \boldsymbol{p}, \boldsymbol{y}, \|\nabla \boldsymbol{p}\|, \nabla \boldsymbol{p}, \nabla \boldsymbol{U}, \boldsymbol{\lambda_1}, \overline{\boldsymbol{U'}}^{\boldsymbol{2}}, \boldsymbol{\lambda_5}$	$\boldsymbol{U}, \overline{\boldsymbol{U}}, \overline{\boldsymbol{U'}}^{\boldsymbol{2}}, \|\boldsymbol{U}\|, \boldsymbol{I_2(\nabla U)}, \boldsymbol{p}$

5 Алгоритмы машинного обучения

Для предсказания турбулентной вязкости будут использованы следующие алгоритмы машинного обучения:

- Decision Tree Regressor (DTR),
- Random Forest Regressor (RFR),
- K-Neighbors Regressor (KNR),
- Support Vector Regressor (SVR).

Модель регрессии, основанная на дереве решений (Decision Tree Regressor) имеет древовидную структуру. В процессе обучения она разбивает набор данных на все меньшие и меньшие подмножества, таким образом постепенно разрабатывается соответствующее дерево решений. Модель имеет один параметр — max_depth, оптимальное значение которого для рассматриваемой задачи составило 20.

Модель регрессии, основанная на алгоритме случайного леса (Random Forest Regressor) — это метод ансамблевого обучения, который объединяет прогнозы нескольких деревьев решений для получения более точного и стабильного прогноза. Модель имеет один параметр — max_depth, оптимальное значение которого для рассматриваемой задачи составило 35.

Модель регрессии, основанная на алгоритме k ближайших соседей (K-Neighbors Regressor) предсказывает значение целевого параметра методом интерполяции значений в k соседних точках обучающей выборки. Модель имеет один параметр — n_neighbors, оптимальное значение которого для рассматриваемой задачи составило 2.

Модель регрессии, основанная на методе опорных векторов (Support Vector Regressor) разделяет данные гиперповерхностями. Модель обладает двумя параметрами, требующими оптимизации: C = 2000 и g = 5.

6 Результаты

Выполнены расчёты турбулентной вязкости с использованием различных алгоритмов машинного обучения. В табл. 4 представлено время обучения моделей, время предсказания, ошибка предсказания на тесте и ошибка предсказания на валидационной выборке для случая, вычисленные по формуле (7) Расчёты выполнялись для различных наборов данных: все линейно-независимые признаки (Unique), признаки, обладающие высокой корреляцией с целевым параметром (Correlating), признаки с наиболее равномерным распределением

(Uniform). Подробное описание наборов данных приведено в табл. 3. Расчёты проводятся с использованием различных алгоритмов МО, описанных в разделе 5.

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{target_i - predict_i}{target_i} \cdot 100\% \quad (7)$$

Табл.4. Результаты различных алгоритмов МО
Table 4. Results of different ML algorithms

Данные	Алгоритм	Время обучения (с)	Время предсказания (с)	Ошибка на тесте (%)	Ошибка на валидации (%)
Unique	DTR	0.373000	0.006295	0.1077	12.5783
	RFR	24.900586	0.173247	1.1171	12.4196
	KNR	0.010028	0.370409	180.8233	360.5632
	SVR	9.517584	4.939535	358.1833	506.1422
Correlating	DTR	0.251941	0.011947	0.0468	12.3927
	RFR	14.159169	0.121596	0.9104	13.1626
	KNR	0.003095	0.085218	0.0054	16.1173
	SVR	15.379704	1.817574	0.5049	29.0284
Uniform	DTR	0.174901	0.006432	0.1331	12.7837
	RFR	11.445493	0.158551	1.2617	12.1861
	KNR	0.013292	0.028157	180.8184	360.4825
	SVR	55.467758	2.267595	382.2995	528.7027

Из табл. 4 можно видеть, что алгоритмы KNR и SVR справились с задачей лишь на хорошо коррелирующем с целевым признаком наборе данных, на котором алгоритм KNR имел минимальное время обучения. В среднем минимальное время обучения и предсказания у алгоритма DTR, имеющим, в то же время, минимальную ошибку. Наилучшее качество предсказания алгоритм DTR выдаёт на хорошо коррелирующем с целевым признаком наборе данных.

Сравнение предсказании турбулентной вязкости различными алгоритмами в характерных сечениях для хорошо коррелирующего с целевым признаком набора данных показано на рис. 6. Видно, что для этого набора данных предсказания хорошо совпадают с целевым признаком.

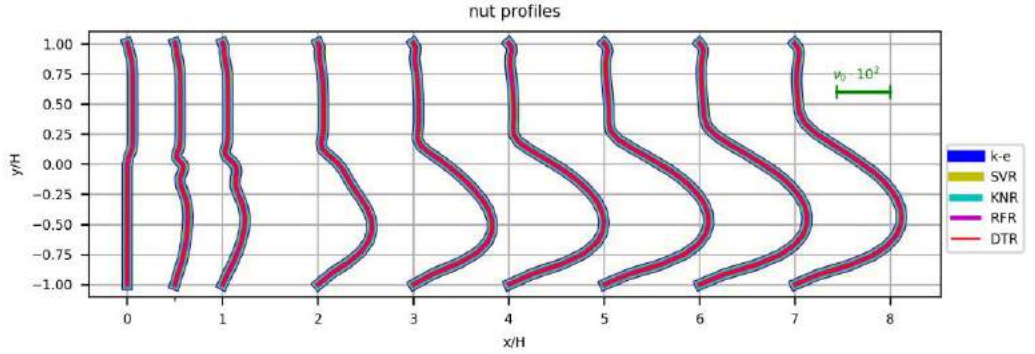


Рис. 6. Предсказания турбулентной вязкости различными алгоритмами на разных срезах для хорошо коррелирующего с целевым признаком набора данных
Fig. 6. Predictions of turbulent viscosity by various algorithms in different sections for a data set that is well correlated with the target feature

Для выбранной задачи и предложенного выше алгоритма машинного обучения проведено предсказание турбулентной вязкости для чисел Рейнольдса 20000 и 15000. Результаты сравнения с расчетом с применением модели турбулентности $k-\varepsilon$ показаны на рис. 7. Видно, что алгоритм DTR удовлетворительно воспроизводит турбулентную вязкость, погрешность в сечениях составляет 1.6% и 2.8%, соответственно.

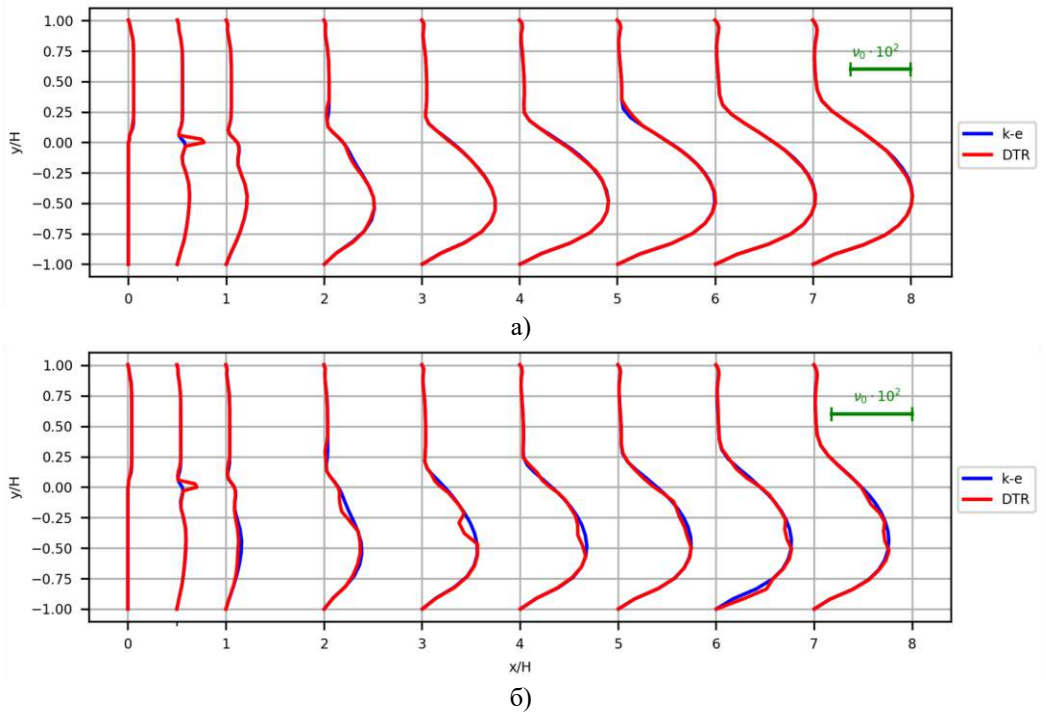


Рис. 7. Сравнение предсказаний турбулентной вязкости с помощью $k-\varepsilon$ модели турбулентности и алгоритма DTR при: а) $Re = 20000$ и б) $Re = 15000$

Fig. 7. Comparison of turbulent viscosity predictions using the $k-\varepsilon$ turbulence model and the DTR algorithm under: a) $Re = 20000$ and b) $Re = 15000$

7 Выводы

Выполнено исследование методов предсказания турбулентной вязкости с помощью различных алгоритмов машинного обучения. На примере течения за обратным уступом проведён анализ гидродинамических данных с целью получения наилучшего качества предсказания при минимизации требуемых параметров течения. Предложен наиболее быстрый и устойчивый алгоритм — Decision Tree Regressor. В результате анализа значимости входных признаков показано, что наибольшее влияние на качество предсказания оказывает корреляция с целевым параметром. Получено, что алгоритмы, использующие данные, хорошо коррелирующие с целевыми признаками $|U|$, U, $\overline{U}$, p, y, $|\nabla p|$, ∇p, ∇U, λ_1, $\overline{U'}^2$, λ_5, показали наилучшее предсказание целевого параметра. Относительная погрешность между алгоритмом DTR и расчетом с применением модели турбулентности $k-\varepsilon$ составила порядка 0.04% для случая на котором выполнялось обучения модели, для других чисел Рейнольдса ошибка в характерных сечениях не превышала 3%, что говорит о принципиальной возможности использования предложенного алгоритма для предсказания турбулентной вязкости.

Также получено, что ряд признаков мешает обучению моделей. Одним из таких признаков является объём ячейки расчетной сетки. Стоит отметить, что данный параметр подбирается пользователем с учетом общих соображений о характере течения рассматриваемой задачи и не зависит от мгновенной структуры потока. Таким образом данный параметр не является репрезентативным.

Список литературы / References

[1]. N. B. Erichson, L. Mathelin, Z. Yao, S. L. Brunton, M. W. Mahoney, and J. N. Kutz, “Shallow neural networks for fluid flow reconstruction with limited sensors,” Proceedings of the Royal Society A, vol. 476, no. 2238, p. 20200097, 2020, doi: 10.1098/rspa.2020.0097.

[2]. E. Haghighat, A. C. Bekar, E. Madenci, and R. Juanes, “A nonlocal physics-informed deep learning framework using the peridynamic differential operator,” Computer Methods in Applied Mechanics and Engineering, vol. 385, p. 114012, 2021, doi: 10.1016/j.cma.2021.114012.

[3]. S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, “Scientific machine learning through physics–informed neural networks: Where we are and what’s next,” Journal of Scientific Computing, vol. 92, no. 3, p. 88, Jul. 2022, doi: 10.1007/s10915-022-01939-z.

[4]. X. Jin, S. Cai, H. Li, and G. E. Karniadakis, “NSFnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations,” Journal of Computational Physics, vol. 426, p. 109951, 2021, doi: https://doi.org/10.1016/j.jcp.2020.109951.

[5]. L. Lu, X. Meng, Z. Mao, and G. Karniadakis, “DeepXDE: A deep learning library for solving differential equations,” SIAM Review, vol. 63, pp. 208–228, Feb. 2021, doi: 10.1137/19M1274067.

[6]. J. Ling, R. Jones, and J. Templeton, “Machine learning strategies for systems with invariance properties,” Journal of Computational Physics, vol. 318, pp. 22–35, 2016, doi: 10.1016/j.jcp.2016.05.003.

[7]. R. McConkey, E. Yee, and F.-S. Lien, “Deep learning-based turbulence closure with improved optimal eddy viscosity prediction,” Jul. 2021.

[8]. Y. Frey Marioni, E. A. de Toledo Ortiz, A. Cassinelli, F. Montomoli, P. Adami, and R. Vazquez, “A machine learning approach to improve turbulence modelling from dns data using neural networks,” International Journal of Turbomachinery, Propulsion and Power, vol. 6, no. 2, 2021, doi: 10.3390/ijtpp6020017.

[9]. R. Fang, D. Sondak, P. Protopapas, and S. Succi, “Neural network models for the anisotropic reynolds stress tensor in turbulent channel flow,” Journal of Turbulence, vol. 21, nos. 9-10, pp. 525–543, 2020, doi: 10.1080/14685248.2019.1706742.

[10]. H. D. Pasinato, F. D. Gerosa, and E. A. Krumrick, “Reynolds stresses prediction using deep neural networks,” Computational Mechanics, vol. XXXVIII, pp. 905–914, 2021.

[11]. G. Kalitzin, G. Medic, and G. Xia, “Improvements to sst turbulence model for free shear layers, turbulent separation and stagnation point anomaly,” in 54th aiaa aerospace sciences meeting, 2016, p. 1601.

[12]. P. C. Rocha, H. B. Rocha, F. M. Carneiro, M. V. da Silva, and C. F. de Andrade, “A case study on the calibration of the k– sst (shear stress transport) turbulence model for small scale wind turbines designed with cambered and symmetrical airfoils,” Energy, vol. 97, pp. 144–150, 2016.

[13]. P. Rocha, H. Rocha, F. Carneiro, M. Silva, and A. Bueno, “K- sst (shear stress transport) turbulence model calibration: A case study on a small scale horizontal axis wind turbine,” Energy, vol. 65, Jan. 2013, doi: 10.1016/j.energy.2013.11.050.

[14]. D. Romanova et al., “Calibration of the k- sst turbulence model for free surface flows on mountain slopes using an experiment,” Fluids, vol. 7, no. 3, 2022, doi: 10.3390/fluids7030111.

[15]. K. Barkalov, I. Lebedev, M. Usova, D. Romanova, D. Ryazanov, and S. Strijhak, “Optimization of turbulence model parameters using the global search method combined with machine learning,” Mathematics, vol. 10, no. 15, 2022, doi: 10.3390/math10152708.

[16]. Д. И. Романова, “Калибровка k- модели турбулентности в пакете openfoam с помощью методов машинного обучения для моделирования потоков на склонах гор на основе эксперимента,” Труды Института системного программирования РАН, vol. 33, no. 4, pp. 227–240, 2021, doi: 10.15514/ispras-2021-33(4)-16.

[17]. R. Pitz and J. Daily, “Experimental study of combustion in a turbulent free shear layer formed at a rearward facing step,” 19th Aerospace Sciences Meeting, 1981, doi: 10.2514/6.1981-106.

[18]. B. Launder and D. Spalding, “The numerical computation of turbulent flows,” Comput. Methods Appl. Mech. Eng., vol. 103, pp. 456–460, Jan. 1974.

[19]. B. Launder, A. Morse, W. Rodi, and D. Spaldiug, “Spaldiug, the prediction of free shear flows - a comparison of the performance of six turbulence models,” Proceedings of NASA Conference on Free Shear Flows, 1972.

[20]. S. B. Pope, “A more general effective-viscosity hypothesis,” Journal of Fluid Mechanics, vol. 72, no. 2, pp. 331–340, 1975, doi: 10.1017/S0022112075003382.

Информация об авторах / Information about authors

Дарья Игоревна РОМАНОВА – кандидат физико-математических наук, младший научный сотрудник Института системного программирования РАН и Московского государственного университета имени М.В. Ломоносова. Сфера научных интересов: вычислительная аэро- и гидромеханика, турбулентные течения, склоновые течения, машинное обучение.

Daria Igorevna ROMANOVA – Cand. Sci. (Phys.-Math.), junior researcher at the Institute of System Programming and Lomonosov Moscow State University. Research interests: computational aero and fluid mechanics, turbulent flows, slope flows, machine learning.

Андрей Сергеевич ЕПИХИН – кандидат технических наук, зав. лаб. Института системного программирования РАН. Сфера научных интересов: вычислительная аэро- и гидромеханика, турбулентные течения, струйные течения и аэроакустика, разработка программного обеспечения.

Andrey Sergeevich EPIKHIN – Cand. Sci. (Tech.), head of the laboratory at the Institute of System Programming of the RAS. Research interests: computational fluid dynamics, turbulent and jet flows, aeroacoustics, software development.

Дарья Юрьевна ИЛЬИНА – старший лаборант Института системного программирования РАН, студентка 1-го курса магистратуры ФПМИ МФТИ, кафедра Системного Программирования.

Daria Yurevna ILINA – laboratory assistant at the Institute of System Programming of the RAS, 1st year master's degree student, Faculty of Applied Mathematics and Informatics (MIPT).

DOI: 10.15514/ISPRAS-2023-35(6)-14

Эффективная реализация быстрого метода мультиполей для взаимодействия частиц с ньютоновским потенциалом

В.М. Аушев, ORCID: 0009-0009-3571-5448 <*aushevvm@gmail.com*>

Московский государственный технический университет имени Н.Э. Баумана, 105005, Россия, г. Москва, ул. 2-я Бауманская, д. 5, к. 1

Аннотация. В работе рассматривается быстрый метод мультиполей с использованием матриц поворота для операторов трансляции для расчета взаимодействия частиц с нютоновским потенциалом, а также его приложение для случая, когда взаимодействие между частицами описывается законом Био—Савара. В работе приведены формулы, необходимые для реализации алгоритма, а также такие редко затрагиваемые моменты, как нормировка сферических гармоник и связанная с ней нормировка матриц Вигнера. Основное внимание в работе уделено описанию деталей программной реализации, которые позволяют существенно ускорить работу кода, как для CPU, так и для GPU (с использованием технологии CUDA). Приведено подробное изложение предлагаемых методик, а также иллюстрирующие их листинги кода. С их использованием был реализован программный комплекс на C++ и проведено сравнение с открытыми программными реализациями быстрого метода мультиполей. Данное сравнение подтверждает высокую эффективность предложенной реализации.

Ключевые слова: быстрый метод мультиполей; матрицы Вигнера; мортоновское дерево; SIMD; CUDA; открытые библиотеки

Для цитирования: Аушев В.М. Эффективная реализация быстрого метода мультиполей для взаимодействия частиц с ньютоновским потенциалом. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 213–234. DOI: 10.15514/ISPRAS–2023–35(6)–14

Effective Implementation of the Fast Multipole Method for Particle Interaction with Newtonian Potential

V.M. Aushev, ORCID: 0009-0009-3571-5448 <*aushevvm@gmail.com*>

Bauman Moscow State Technical University,
105005, Moscow, 2-nd Baumanskaya st., 5.

Abstract. We consider rotation-based fast multipole method for the Laplace equation and its application in cases where particle interactions are governed by the Biot—Savart law. The paper presents the necessary formulas for algorithm implementation and addresses less frequently discussed topics, such as the normalization of spherical harmonics and the associated Wigner matrices normalization. The main focus of the paper is devoted to describing the details of the software implementation that significantly accelerate the performance of the code both on CPU and GPU (using CUDA technology). We provide a comprehensive explanation of proposed techniques and include code examples. We have implemented a C++ program using these methods and conducted a comparative analysis with open-source implementations of the fast multipole method, confirming the high efficiency of our approach.

Keywords: fast multipole method; Wigner matrices; Morton tree; SIMD; CUDA; open source libraries

For citation: Aushev V.M. Effective implementation of the fast multipole method for particle interaction with Newtonian potential. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 213-234 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-14

1. Введение

Данная работа посвящена практической реализации быстрого метода мультиполей (БММ) для N тел, взаимодействующих по закону Ньютона в трехмерном случае (сила обратно пропорциональна квадрату расстояния), а также между телами, сила взаимодействия для которых описывается законом Био—Савара. Быстрый метод мультиполей позволяет решать описанную задачу N тел с линейной сложностью, в то время как прямой метод решения имеет сложность $O(N^2)$ — таким образом, прямым методом крайне затруднительно решать задачи с большим числом тел. Быстрый метод мультиполей является развитием другого метода, который нельзя не упомянуть в контексте быстрых методов. Речь идет о методе Барнса—Хата [1], который имеет сложность $O(N \log N)$.

Задача N тел возникает во многих областях математики и физики; самый простой пример — расчет траектории небесных тел под действием гравитации. В математике задача N тел возникает как следствие дискретизации интегрального уравнения для решения задач физики, например, в электродинамике [2-3], в гидродинамике и теории упругости [4-5]. Теоретическому описанию быстрого метода мультиполей посвящено множество работ, например, [6-8]. Отдельно выделим работу [9], в которой авторы не только подробно описывают теорию, но и объясняют, за счет чего достигается быстродействие алгоритма на простых примерах.

Изначальная формулировка быстрого метода мультиполей для трехмерных задач, хоть и имеет линейную асимптотику, имеет сложность $O(p^4)$, где p — количество мультиполей. В связи с этим практическое использование метода для реального количества тел оказывается медленнее, чем уже упомянутый метод Барнса—Хата. Для исправления этой ситуации были созданы модификации метода, использующие вращения (сложность $O(p^3)$) и диагональные формы операторов трансляции (сложность $O(p^2)$) [7]. Существуют также и другие подходы, позволяющие понизить сложность алгоритма, основанные на сочетании с методом частиц [10-11]. Использование идей метода частиц позволяет также ускорить метод за счет использования быстрого преобразования Фурье для операторов трансляции [12]. Также

возможна реализация быстрого метода мультиполей “без мультиполей”, как описано в статье [13].

Отдельно упомянем существующие открытые программные реализации быстрого метода мультиполей [14-18]. В данной работе мы проведем сравнение с двумя реализациями [14-18].

Дальнейшее изложение устроено следующим образом. Во втором разделе мы приведем краткое теоретические описание используемых формул, а также подробно рассмотрим нормировку сферических гармоник и следующую из нее нормировку матриц Вигнера; кроме этого, мы представим выражения для вычисления сил с использованием мультипольных разложений. Во втором разделе мы перейдем к подробному описанию деталей программной реализации, включая рекуррентные формулы для присоединенных полиномов Лежандра и матриц Вигнера, а также различные техники, позволяющие в разы ускорить код на C++ в случае CPU версии кода и CUDA C++ для GPU версии. В третьем разделе приведены таблицы, иллюстрирующие скорость работы кода для разных версий, включая распараллеливание для систем с общей и распределенной памятью. В четвертом разделе приведено сравнение реализованного программного комплекса[1] с уже упомянутыми выше реализациями.

2. Теоретические основы быстрого метода мультиполей

2.1 Общая схема работы быстрого метода мультиполей

Пусть имеется N частиц с координатами $\mathbf{x} \in \mathrm{R}^n$, и для каждой частицы необходимо вычислить выражение вида

$$u(\mathbf{x}_i) = \sum_{i \neq j} f(\mathbf{x}_i, \mathbf{y}_j) = g(\mathbf{x}_i - \mathbf{y}_j) = g(\mathbf{r}_{ij}).$$

Задачи такого вида возникают во многих областях науки, например, при расчете движения тел под действием гравитационных сил, в этом случае $g(\mathbf{r}_{ij}) = Gm_i m_j \dfrac{\mathbf{x}_j - \mathbf{y}_i}{\mathrm{P}\mathbf{x}_i - \mathbf{y}_j \mathrm{P}}$. Очевидно, прямой метод подсчета сил для всех тех имеет сложность $O(N^2)$, поэтому представляют интерес методы, решающие данную задачу с меньшей сложностью; такие методы называются быстрыми. Основная идея быстрых методов состоит в том, что функцию $g(\mathbf{x}-\mathbf{y})$ при достаточно удаленных точках $\mathbf{x}$ и $\mathbf{y}$ можно приближенно заменить функцией $h(\mathbf{x},\mathbf{y})$, которая факторизуется, т.е.

$$h(\mathbf{x},\mathbf{y}) = \sum_{i=0}^{p} a_i(\mathbf{x}) b_i(\mathbf{y}).$$

Смысл этого приближения виден на простом примере, когда

$$g(\mathbf{x},\mathbf{y}) = A(\mathbf{x})B(\mathbf{y}),$$

в этом случае мы можем решить задачу N тел с линейной сложностью: на первом шаге находим величину $Sy = \sum_{i=1}^{N} B(\mathbf{y}_i)$, на втором шаге находим влияние всех тел на частицу как $u(\mathbf{x}_i) = A(\mathbf{x}_i)Sy, \ \ i = \overline{1, N}.$

Одним из представителей быстрых методов является быстрый метод мультиполей, имеющий сложность $O(N)$. Описание работы алгоритма приведено во многих статьях, например, в [9]

[1] https://github.com/ViktorAushev/laplace-fmm/

представлено очень подробное изложение идеи метода, а также приведены конкретные формулы. В этой связи мы ограничимся кратким описанием алгоритма.

1) На первом шаге для всех частиц строится квад- или октодерево в зависимости от размерности задачи (рис. 1, а-в). Данная процедура очень важна для эффективной численной реализации алгоритма, поэтому в дальнейшем мы подробно рассмотрим этот момент. Под деревом на примере двумерной области мы понимаем следующую структуру: на самом верхнем уровня дерева имеем корень — квадрат, содержащий все частицы, уровнем ниже имеем четыре квадратных ячейки, получающиеся дроблением исходного квадрата, то есть родительской ячейки; для корня дерева полученные четыре ячейки — дочерние (потомки). Рекурсивно разбивая ячейки, получаем структуру дерева, на самом нижнем уровне имеем ячейки без детей — то есть листья. Под обходом дерева “вверх” будем понимать обход уровней дерева от листьев к корню, под движением “вниз” — путь от корня к листьям. Отметим, что в общем случае дерево можно строить так, что листья находятся на разных уровнях, но мы этот случай не будем рассматривать. Также считаем, что в дереве нет пустых ячеек, то есть которые не содержат частиц — таким образом имеем адаптивную структуру дерева (при описании реализации дерева в дальнейшем повествовании приведен пример такого дерева).
2) Нахождение мультипольных разложений в листьях дерева (назовем этот шаг P2M, рис. 1, г). Данные разложения представляют собой аппроксимацию функции $g(\mathbf{x},\mathbf{y})$ вдали от ячейки, то есть находим коэффициенты $b_i(\mathbf{y})$ для функции $h(\mathbf{x},\mathbf{y})$.
3) Поднимаясь вверх по уровням дерева, для каждой ячейки находим ее мультипольные разложения, используя разложения дочерних ячеек. Для этого необходимо выполнять трансляцию мультипольных разложений из дочерних ячеек в родительские (операция M2M, рис. 1, г).
4) Спускаемся от корня дерева вниз, для каждой ячейки находим аппроксимацию влияния дальних соседей на частицы внутри ячейки, которое представляется в виде локального разложения. Для нахождения локальных разложений необходимо выполнять трансляцию мультипольных разложений в локальные (операция M2L, рис. 1, д).
5) Спускаемся от корня дерева вниз, для каждой ячейки находим локальное разложение, связанное с влиянием дальних соседей на родительскую ячейку — для этого выполняется трансляция локального разложения родительской ячейки в дочернюю (операция L2L, рис. 1, е).
6) Для всех частиц в листьях находим потенциал (силу): влияние от дальних ячеек учитываем, используя локальные разложения (операция L2P, рис. 1, е), а взаимодействие между частицами из ближайших соседей листа находим по прямой формуле (т.е. используя функцию $g(\mathbf{x},\mathbf{y})$ — операция P2P, рис. 1, ж).

В алгоритме мы упомянули ближних и дальних соседей. Ближние соседи ячейки — ячейки того же уровня дерева, соприкасающиеся с ячейкой. Для нахождения дальних соседей мы берем родительские ячейки ближних соседей, рассматриваем все их дочерние ячейки; среди них дальними соседями являются все, которые не являются ближними.

Таким образом, для реализации алгоритма быстрого метода мультиполей нам необходимо выполнять операции P2M, M2M, M2L, L2L и L2P. Далее приведено краткое теоретическое описание данных операций применительно к ньютоновскому потенциалу и силе.

Все этапы работы алгоритма приведены на единой схеме на рис. 1.

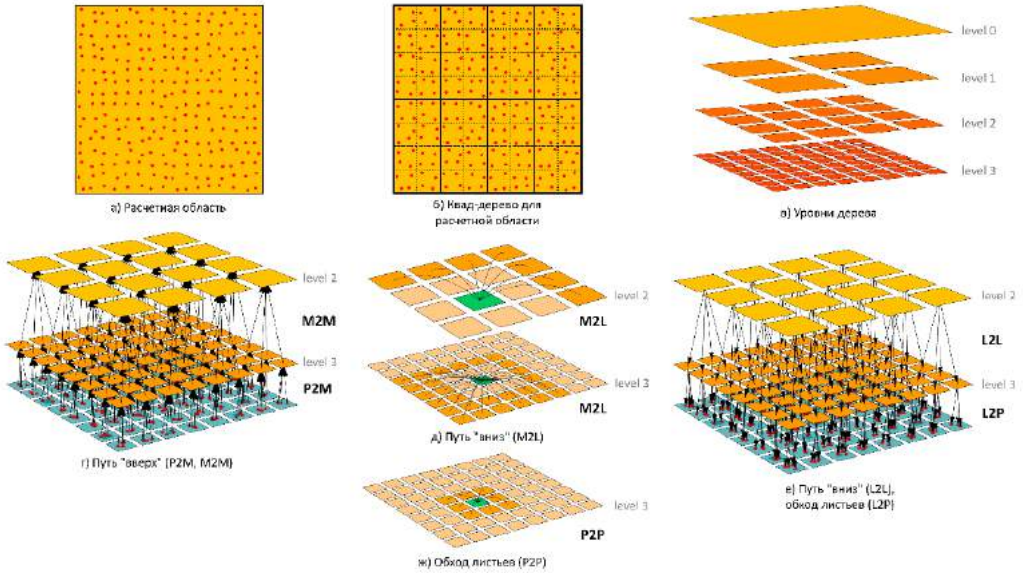


Рис. 1. Этапы алгоритма быстрого метода мультиполе
Fig. 1. Stages of the fast multipole method

2.2 Теория для ньютоновского потенциала и силы

Будем считать, что потенциал заряда q, расположенного в точке $\mathbf{y}$, имеет вид

$$\Phi(\mathbf{x},\mathbf{y}) = \frac{q}{\mathrm{P}\mathbf{x}-\mathbf{y}\mathrm{P}}, \qquad \mathbf{x},\mathbf{y} \in \mathrm{R}^3,$$

а напряженность поля —

$$\mathbf{E}(\mathbf{x},\mathbf{y}) = q\frac{\mathbf{x}-\mathbf{y}}{\mathrm{P}\mathbf{x}-\mathbf{y}\mathrm{P}^3}, \qquad \mathbf{x},\mathbf{y} \in \mathrm{R}^3.$$

Изложение теории начнем с введения сферических функций Бесселя [7]:

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} \cdot P_n^{|m|}(\cos\theta)e^{im\phi}, \tag{1}$$

где $P_n^m(x)$ — присоединенный полином Лежандра, определяемый формулой Родрига:

$$P_n^m(x) = (-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}P_n(x),$$

а $P_n(x)$ — полином Лежандра степени n. Отметим тот момент, что введенные в (1) и используемые в быстром методе мультиполей для трехмерного уравнения Лапласа [7-8] сферические гармоники отличаются от тех, которые получаются в физике при выводе собственных функций оператора момента импульса в квантовой механике [19-20]:

$$\tilde{Y}_n^m(\theta,\phi) = (-1)^m\sqrt{\frac{2n+1}{4\pi}}\sqrt{\frac{(n-m)!}{(n+m)!}} \cdot \tilde{P}_n^m(\cos\theta)e^{im\phi}, \tag{2}$$

где

$$\tilde{P}_n^m(x) = (1-x^2)^{m/2}\frac{d^m}{dx^m}P_n(x),$$

то есть для присоединенных полиномов Лежандра отличие в множителе $(-1)^m$. Таким образом,

$$\tilde{Y}_n^m(\theta,\phi) = \sqrt{\frac{2n+1}{4\pi}}\sqrt{\frac{(n-m)!}{(n+m)!}}\cdot P_n^m(\cos\theta)e^{im\phi}.$$

В итоге, имеем следующее соответствие между сферическими функциями с разными нормировками:

$$\varepsilon_m\sqrt{\frac{4\pi}{2n+1}}Y_n^m(\theta,\phi) = \tilde{Y}_n^m(\theta,\phi), \tag{3}$$

где

$$\varepsilon_m = (-1)^{(|m|-m)/2} = \begin{cases}(-1)^m, & m<0,\\ 1, & m,, 0.\end{cases}$$

Соотношение (3) пригодится в дальнейшем изложении при определении матриц Вигнера. Подробно про различные варианты нормировок сферических гармоник и их обоснование написано в [19].

Перейдем к изложению основных формул. Подробную формулировку и доказательство используемых теорем можно найти в [7-8].

Теорема 1. (Мультипольное разложение, P2M)

$$\Phi(\mathbf{x}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{M_n^m}{r^{n+1}}\cdot Y_n^m(\theta,\phi),$$

где

$$M_n^m = \sum_{i=1}^{N} q_i\cdot\rho_i^n\cdot Y_n^{-m}(\alpha_i,\beta_i).$$

Теорема 2. (Локальное разложение, L2P)

$$\Phi(\mathbf{x}) = \sum_{j=0}^{\infty}\sum_{k=-j}^{j}L_j^k\cdot Y_j^k(\theta,\phi)\cdot r^j, \tag{4}$$

где

$$L_j^k = \sum_{i=1}^{N} q_i\cdot\frac{Y_j^{-k}(\alpha_i,\beta_i)}{\rho_i^{j+1}}.$$

Если требуется найти не потенциал, а силу, поступим следующим образом. Найдем градиент разложения (4):

$$-\mathbf{E}(\mathbf{x}) = \nabla_{(r,\theta,\phi)}\Phi(\mathbf{x}) = \begin{pmatrix}\sum_{n=1}^{\infty}\sum_{m=-n}^{n}L_n^m\cdot Y_n^m(\theta,\phi)\cdot n\cdot r^{n-1}\\ \sum_{n=0}^{\infty}\sum_{m=-n}^{n}L_n^m\cdot\frac{\partial Y_n^m(\theta,\phi)}{\partial\theta}\cdot r^n\\ \sum_{n=0}^{\infty}\sum_{m=-n}^{n}L_n^m\cdot\frac{\partial Y_n^m(\theta,\phi)}{\partial\phi}\cdot r^n\end{pmatrix}.$$

Получили выражение в сферических координатах. Для перехода в декартовы координаты нам понадобится обратная матрица Якоби:

$$(J^{-1})^T = \begin{pmatrix} \sin\theta\cos\phi & \dfrac{\cos\theta\cos\phi}{r} & -\dfrac{\csc\theta\sin\phi}{r} \\ \sin\theta\sin\phi & \dfrac{\cos\theta\sin\phi}{r} & \dfrac{\csc\theta\cos\phi}{r} \\ \cos\theta & -\dfrac{\sin\theta}{r} & 0 \end{pmatrix}.$$

В итоге получаем

$$\nabla_{(x,y,z)}\Phi(\mathbf{x}) = (J^{-1})^T \nabla_{(r,\theta,\phi)}\Phi(\mathbf{x}) = \begin{pmatrix} A\sin\theta\cos\phi + B\cos\theta\cos\phi - C\csc\theta\sin\phi \\ A\sin\theta\sin\phi + B\cos\theta\sin\phi + C\csc\theta\cos\phi \\ A\cos\theta - B\sin\theta \end{pmatrix},$$

где

$$\begin{pmatrix} A \\ B \\ C \end{pmatrix} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} L_n^m r^{n-1} \begin{pmatrix} n\cdot Y_n^m(\theta,\phi) \\ \dfrac{\partial Y_n^m(\theta,\phi)}{\partial\theta} \\ \dfrac{\partial Y_n^m(\theta,\phi)}{\partial\phi} \end{pmatrix}. \qquad (5)$$

Распишем производные [21]:

$$\frac{\partial Y_n^m(\theta,\phi)}{\partial\phi} = i\cdot m\cdot Y_n^m(\theta,\phi),$$

$$\frac{\partial Y_n^m(\theta,\phi)}{\partial\theta} = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}}\cdot\frac{(n+1-|m|)P_{n+1}^{|m|}(\theta) - (n+1)\cos(\theta)P_n^{|m|}(\theta)}{\sin(\theta)}\cdot e^{im\phi}.$$

Заметим, что при $n = m = 0$ производные обращаются в ноль, в итоге (5) примет вид

$$\begin{pmatrix} A \\ B \\ C \end{pmatrix} = \sum_{n=1}^{\infty}\sum_{m=-n}^{n} L_n^m r^{n-1} \begin{pmatrix} n\cdot Y_n^m(\theta,\phi) \\ \sqrt{\dfrac{(n-|m|)!}{(n+|m|)!}}\cdot\dfrac{(n+1-|m|)P_{n+1}^{|m|}(\theta) - (n+1)\cos(\theta)P_n^{|m|}(\theta)}{\sin(\theta)}\cdot e^{im\phi} \\ i\cdot m\cdot Y_n^m(\theta,\phi) \end{pmatrix}.$$

Замечание: при $\theta = 0$ имеем неопределенность в выражении для $\dfrac{\partial Y_n^m(\theta,\phi)}{\partial\theta}$; в этом случае выражение следует понимать в смысле предела при $\theta \to 0$. Однако при численной реализации из-за погрешности вычислений и случайного расположения частиц θ не равняется нулю, и все вычисления проходят корректно.

Теорема 3. (Трансляция мультипольного разложения, M2M)

$$M_j^k = \sum_{n=0}^{j}\sum_{m=-n}^{n} \frac{O_{j-n}^{k-m}\cdot i^{|k|-|m|-|k-m|}\cdot A_n^m\cdot A_{j-n}^{k-m}\cdot\rho^n\cdot Y_n^{-m}(\alpha,\beta)}{A_j^k}.$$

Теорема 4. (Трансляция мультипольного разложения в локальное, M2L)

$$L_j^k = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} \frac{O_n^m i^{|k-m|-|k|-|m|}\cdot A_n^m\cdot A_j^k\cdot Y_{j+n}^{m-k}(\alpha,\beta)}{(-1)^n A_{j+n}^{m-k}\cdot\rho^{j+n+1}}.$$

Теорема 5. (Трансляция локального разложения, L2L)

$$L_j^k = \sum_{n=j}^{p}\sum_{m=-n}^{n} \frac{O_n^m \cdot i^{|m|-|m-k|-|k|} \cdot A_{n-j}^{m-k} \cdot A_j^k \cdot Y_{n-j}^{m-k}(\alpha,\beta)\cdot \rho^{n-j}}{(-1)^{n+j}\cdot A_n^m}.$$

Коэффициенты A_n^m имеют вид

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!\cdot(n+m)!}}. \tag{6}$$

Заметим, что выполнение операций M2M, M2L и L2L имеет сложность $O(p^4)$. Улучшить ситуацию можно двумя способами: используя вращения или диагональные формы операторов [7-8]. В первом случае сложность составляет $O(p^3)$, во втором — $O(p^2)$. В данной работе мы рассмотрим первый вариант. Для начала нам понадобятся две леммы [22].

Лемма 1. *Пусть гармоническая функция* $\Phi(\mathbf{x})$ *имеет вид*

$$\Phi(\mathbf{x}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(L_n^m r^n + \frac{M_n^m}{r^{n+1}}\right)Y_n^m(\theta,\phi),$$

где (r,θ,ϕ) *— сферические координаты точки* $\mathbf{x}$. *При повороте системы координат на угол* β *вокруг оси* z *в положительном направлении функция* $\Phi(\mathbf{x})$ *примет вид*

$$\Phi(\mathbf{x}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(\tilde{L}_n^m r^n + \frac{\tilde{M}_n^m}{r^{n+1}}\right)Y_n^m(\theta,\phi'),$$

где (r,θ,ϕ') *— новые координаты точки* $\mathbf{x}$, *при этом*

$$\tilde{L}_n^m = L_n^m e^{im\beta}, \quad \tilde{M}_n^m = M_n^m e^{im\beta}.$$

Лемма 2. *Пусть гармоническая функция* $\Phi(\mathbf{x})$ *имеет вид*

$$\Phi(\mathbf{x}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(L_n^m r^n + \frac{M_n^m}{r^{n+1}}\right)Y_n^m(\theta,\phi),$$

где (r,θ,ϕ) *— сферические координаты точки* $\mathbf{x}$. *При повороте системы координат на угол* α *вокруг оси* y *в положительном направлении функция* $\Phi(\mathbf{x})$ *примет вид*

$$\Phi(\mathbf{x}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(\tilde{L}_n^m r^n + \frac{\tilde{M}_n^m}{r^{n+1}}\right)Y_n^m(\theta',\phi),$$

где (r,θ',ϕ) *— новые координаты точки* $\mathbf{x}$, *при этом*

$$\tilde{L}_n^m = \sum_{k=-n}^{n}\ddot{d}_{mk}^n(\alpha)L_n^k, \quad \tilde{M}_n^m = \sum_{k=-n}^{n}\ddot{d}_{mk}^n(\alpha)M_n^k.$$

Здесь $\tilde{d}_{mk}^n$ — компоненты матрицы Вигнера [23, 19, 20]. Матрицы Вигнера используются для пересчета сферических гармоник при повороте системы координат [23, 19]:

$$\ddot{\tilde{Y}}_n^m(\theta',\phi') = \sum_{k=-n}^{n}\ddot{\tilde{D}}_{km}^n(\alpha,\beta,\gamma)\bar{\tilde{Y}}_n^k(\theta,\phi), \tag{7}$$

где (α,β,γ) — эйлеровы углы, $\tilde{Y}_n^m$ — сферические гармоники, определенные в (2), $\tilde{D}_{km}^n$ — матрица Вигнера, определенная как [23]

$$\tilde{D}_{mk}^n(\alpha,\beta,\gamma) = e^{im\alpha}\tilde{d}_{mk}^n(\beta)e^{ik\gamma},$$

$$\tilde{d}^n_{mk}(\beta) = \sqrt{(n+k)!(n-k)!(n+m)!(n-m)!} \times \sum_{s=\max(0,k-m)}^{\min(n-m,n+k)} \frac{(-1)^s \left(\sin\left(\frac{\theta}{2}\right)\right)^{2s-k+m} \left(\cos\left(\frac{\theta}{2}\right)\right)^{k-m+2(n-s)}}{s!(s+m-k)!(k+n-s)!(n-m-s)!}.$$

Подставим в (7) связь сферических гармоник с разной нормировкой (3), получим

$$\varepsilon_m Y^m_n(\theta',\phi') = \sum_{k=-n}^{n} \tilde{D}^n_{km}(\alpha,\beta,\gamma) Y^k_n(\theta,\phi)\varepsilon_k.$$

Так как $1/\varepsilon_m = \varepsilon_m$, получаем

$$Y^m_n(\theta',\phi') = \sum_{k=-n}^{n} D^n_{km}(\alpha,\beta,\gamma) Y^k_n(\theta,\phi),$$

где

$$D^n_{mk}(\alpha,\beta,\gamma) = \varepsilon_m \varepsilon_k \tilde{D}^n_{mk}(\alpha,\beta,\gamma). \tag{8}$$

Соотношения (3) и (8) важно учитывать при написании кода, например, в Wolfram Mathematica, где под сферическими функциями Бесселя и матрицами Вигнера понимаются как раз $\tilde{Y}^m_n$ и $\tilde{D}^n_{mk}$.

Обозначим за T оператор трансляции (M2M, M2L или L2L), $R_y(\theta)$ — оператор поворота вокруг оси y, $R_z(\phi)$ — оператор поворота вокруг оси z. Тогда оператор T записывается в виде

$$T = R_z(-\phi) R_y(-\theta) T_z R_y(\theta) R_z(\phi),$$

где T_z — оператор трансляции вдоль оси z, при этом в T_z входят сферические гармоники, вычисленные для нулевых углов, что позволяет существенно упростить запись операторов трансляции, так как $Y^m_n(0,0) = \delta_{m0}$.

Теорема 6. (Операторы трансляции вдоль оси z). *При трансляции в положительном направлении вдоль оси z верны следующие формулы пересчета коэффициентов:*

$$\text{M2M:} \qquad M^k_j = \sum_{n=0}^{j-|k|} \frac{O^k_{j-n} \cdot A^0_n \cdot A^k_{j-n} \cdot \rho^n}{A^k_j},$$

$$\text{M2L:} \qquad L^k_j = \sum_{n=k}^{\infty} \frac{O^k_n \cdot A^k_n \cdot A^k_j}{(-1)^{n+k} A^0_{j+n} \cdot \rho^{j+n+1}},$$

$$\text{L2L:} \qquad L^k_j = \sum_{n=j}^{p} \frac{O^k_n \cdot A^0_{n-j} \cdot A^k_j \cdot \rho^{n-j}}{(-1)^{n+j} \cdot A^k_n}.$$

Обратим внимание на то, как поменялись индексы суммирования. Так как для коэффициента A^m_n из условия неотрицательности выражения под знаком факториала имеем $n > m$ и $n+m > 0$, а в M2M имеется A^k_{j-n}, то получаем ограничение на индекс n: $n < j-|k|$. Аналогично в M2L, благодаря коэффициенту A^k_n имеем $n > k$, из-за чего меняется нижний индекс суммирования, а в показателе (-1) добавляется k.

Как видно из приведенных выше формул, сложность оператора поворота $R_z(\phi)$ составляет $O(p^2)$, оператора $R_y(\theta)$ — $O(p^3)$ и оператора T_z также $O(p^3)$, в итоге с помощью вращений можно выполнять трансляции T со сложностью $O(p^3)$.

2.3 Вычисление взаимодействий по закону Био—Савара

Быстрый метод мультиполей для ядра трехмерного уравнения Лапласа можно применить к частицам, взаимодействующим по закону Био—Савара. Пусть частице с координатой $\mathbf{y}$ поставлен в соответствие ток $\mathbf{j}$, и нас интересует вычисление выражения

$$\mathbf{B}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{j}\times(\mathbf{x}-\mathbf{y})}{\mathrm{P}\mathbf{x}-\mathbf{y}\,\mathrm{P}^3}. \tag{9}$$

Распишем данное выражение покомпонентно, обозначив $\mathbf{r} = \mathbf{x}-\mathbf{y}$:

$$\begin{aligned} B_x &= \frac{j_y r_z - j_z r_y}{\mathrm{P}\mathbf{r}\,\mathrm{P}^3}, \\ B_y &= \frac{j_z r_x - j_x r_z}{\mathrm{P}\mathbf{r}\,\mathrm{P}^3}, \\ B_z &= \frac{j_x r_y - j_y r_x}{\mathrm{P}\mathbf{r}\,\mathrm{P}^3}. \end{aligned} \tag{10}$$

Обозначим

$$\mathbf{F}^\alpha = j_\alpha \frac{\mathbf{r}}{\mathrm{P}\mathbf{r}\,\mathrm{P}^3}, \quad F_\beta^\alpha = j_\alpha \frac{r_\beta}{\mathrm{P}\mathbf{r}\,\mathrm{P}^3}, \qquad \alpha,\beta = x,y,z,$$

тогда (10) можно переписать следующим образом:

$$\begin{aligned} B_x &= F_z^y - F_y^z, \\ B_y &= F_x^z - F_z^x, \\ B_z &= F_y^x - F_x^y. \end{aligned} \tag{11}$$

Для вычисления $\mathbf{F}^\alpha$ можно использовать уже описанный быстрый метод мультиполей для ядра трехмерного уравнения Лапласа, после чего можно найти все компоненты вектора $\mathbf{B}$, используя формулы (11).

3. Детали программной реализации

3.1 Вычисление специальных функций

Непосредственная программная реализация описанных формул быстрого метода мультиполей сопряжена с некоторыми трудностями. Во-первых, для трехмерного уравнения Лапласа необходимо вычислять сферические гармоники и компоненты матриц Вигнера. Для первых все сводится к вычислению присоединенных полиномов Лежандра; для этого мы воспользуемся рекуррентными соотношениями [2]:

$$(n-m)P_n^m(cos\theta) = (2n-1)\cos\theta\cdot P_{n-1}^m(\cos\theta) - (n+m-1)P_{n-2}^m(\cos\theta),\ 0,,\ m,,\ n-2,$$

$$P_m^m(\cos\theta) = \frac{(2m)!}{2^m m!}(-\sin\theta)^m,\ m,,\ 0$$

$$P_{m+1}^m(\cos\theta) = (2m+1)\cos\theta\cdot P_m^m(\cos\theta),\ m,,\ 0.$$

Для вычисления компонент матриц Вигнера мы также будем использовать рекуррентные соотношения [24]:

$$\tilde{d}^n_{mn} = (-1)^{n+m} g_{nm} (1+\cos\theta)^m (\sin\theta)^{n-m}, \quad n,, 0,0,, m,, n,$$

$$\tilde{d}^n_{n,k-1} = \frac{n+k}{\sqrt{n(n+1)-k(k-1)}} \frac{\sin\theta}{1+\cos\theta} \tilde{d}^n_{nk}, \quad n>0,-n<k,, n$$

$$\tilde{d}^n_{m,k-1} = \sqrt{\frac{n(n+1)-m(m+1)}{n(n+1)-k(k-1)}} \tilde{d}^n_{m+1,k} + \tag{12}$$

$$+\frac{m+k}{\sqrt{n(n+1)-k(k-1)}} \frac{\sin\theta}{1+\cos\theta} \tilde{d}^n_{mk}, \quad n>0,0,, m<n,-n<k,, n,$$

где

$$g_{nm} = \frac{1}{2^n}\sqrt{\frac{(2n)!}{(n-m)!(n+m)!}},$$

причем коэффициенты g_{nm} можно вычислить рекурсивно:

$$g_{0,0}=1, \quad g_{n,0} = \sqrt{\frac{2n-1}{2n}} g_{n-1,0}, \quad g_{n,m} = \sqrt{\frac{n-m+1}{n+m}} g_{n,m-1}.$$

Данные формулы подходят для случая, когда аргумент $\theta \in [0;\frac{\pi}{2}]$. Для остальных случаев можно использовать следующие соотношения:

$$\tilde{d}^n_{m,k}(\pi-\theta) = (-1)^{n+m} \tilde{d}^n_{m,-k}(\theta),$$

$$\tilde{d}^n_{m,k}(-\theta) = (-1)^{m-k} \tilde{d}^n_{m,k}(\theta).$$

Для нахождения матриц Вигнера применительно к быстрому методу мультиполей необходимо использовать уже упомянутое соотношение (8).

Формулы (12) дают приемлемую погрешность при небольшом числе мультиполей [24], например, при $N=20$ максимальная относительная ошибка составляет порядка 10^{-10}; впрочем, на практике такое число мультиполей зачастую даже избыточно, так как при $N=20$ ошибка L_2 подсчета силы и потенциала составляет не больше 10^{-7}. Если же есть необходимость использования большого числа мультиполей, то целесообразно использовать псевдоспектральный метод поворота мультипольных коэффициентов [25], при использовании которого не нужно знать матрицы Вигнера и который обеспечивает достаточную точность даже для $N=1000$, однако этот метод работает примерно в 2 раза дольше, чем при использовании матриц Вигнера.

3.2 Построение дерева

Перейдем к обсуждению деталей, позволяющих ускорить работу кода. Одной из ключевых вещей в алгоритме является построение дерева, и здесь существует два подхода:

- рекурсивное построение дерева;
- построение дерева на основе фрактальной кривой.

Рекурсивное построение дерева реализуется очень просто, однако имеет несколько существенных недостатков:

1) данные хранятся в разных участках памяти, из-за этого не используется кэш при обращении к данным;
2) из-за наличия рекурсий трудно эффективно распараллелить код на CPU;
3) так как дерево строится рекурсивно, реализовать его построение на CUDA становится практически невозможно, так как в kernel-функциях невозможно

динамически выделять память. Из-за этого приходится строить дерево на CPU, затем копировать его на GPU, что влечет за собой использование сложных структур данных, снижает производительность и читаемость кода.

В этой связи наиболее целесообразным представляется построение дерева с помощью фрактальной кривой; в нашей реализации используется мортоновская кривая. Реализовать такое построение несколько сложнее, чем рекурсивное, однако у него имеются следующие достоинства:

1) данные хранятся последовательно в виде одномерных массивов, что позволяет оптимизировать обращения к памяти, так как используется кэш;
2) последовательное хранение данных позволяет эффективно распараллелить не только построение дерева, но и все операции с мультипольными разложениями;
3) код для GPU получается аналогичным коду для CPU, новые структуры данных практически не нужно использовать, в итоге улучшается читаемость кода и сам код эффективно распараллеливается;
4) построение дерева с использованием фрактальной кривой работает быстрее, чем рекурсивное построение.

Суть построения дерева на основе мортоновской кривой состоит в следующем (рассмотрим для наглядности двумерную область, в трехмерной все аналогично).

1) Пусть все частицы расположены в квадрате $[0;1)\times[0;1)$. Если это не так, все координаты можно смаштабировать, чтобы это условие выполнялось.
2) Выбирается глубина дерева, например, $\text{depth} = 3$. В этом случае в дереве будет три уровня, нулевой — корень с одной ячейкой $[0;1)\times[0;1)$, на первом уровне — максимум 4 ячейки ($[0;0.5)\times[0;0.5)$, $[0.5;1)\times[0;0.5)$, $[0;0.5)\times[0.5;1)$, $[0.5;1)\times[0.5;1)$), на втором — максимумум 16 ячеек, получающиеся дроблением на 4 ячеек второго уровня.
3) Для каждой частицы на основе ее координат (x, y) присваивается мортоновский индекс следующим образом. Пусть координаты (x, y) в двоичной записи имеют вид $0, x_1x_2x_3...$ и $0, y_1y_2y_3...$ соответственно, тогда для уровня дерева на глубине d мортоновский индекс

$$\text{idx}_d(x, y) = x_1y_1x_2y_2...x_dy_d.$$

Например, если у нас имеется только нулевой уровень дерева, то есть корень, то считаем, что индекс равен нулю. Для первого уровня возможно четыре варианта: $00_2 = 0_{10}$, $01_2 = 1_{10}$, $10_2 = 2_{10}$, $11_2 = 3_{10}$.

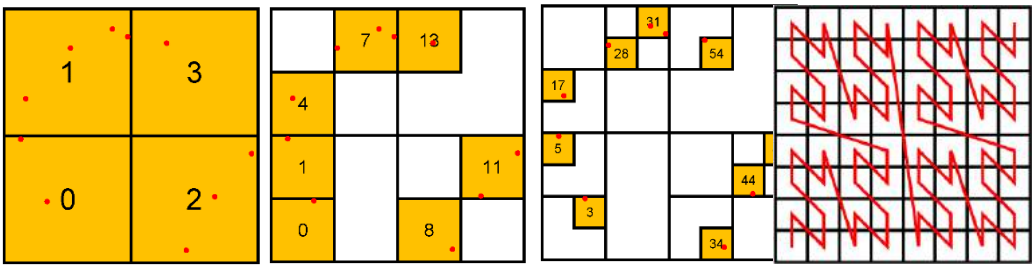
4) Находим мортоновские индексы частиц, считая, что они находятся на уровне $d = \text{depth} - 1$, то есть в листьях дерева. Координаты центров листьев при этом можно пересчитать из индексов по формуле

$$x = \frac{1}{2^{d+1}} + \sum_{i=1}^{d} \frac{1}{2^i} x_i, \qquad y = \frac{1}{2^{d+1}} + \sum_{i=1}^{d} \frac{1}{2^i} y_i.$$

5) Для определения индекса верхнего уровня достаточно выполнить битовый сдвиг индекса: $\text{idx}_{d-1} = \text{idx}_d >> 2$. При этом мы получим родительскую ячейку, а ее потомки — все ячейки нижнего уровня, у кого имеется одинаковый сдвинутый индекс. Данную процедуру повторяем для всех уровней, поднимаясь “вверх”, таким образом получая структуру дерева. Отметим, что для быстрого определения потомков ячеек целесообразно отсортировать частицы по их мортоновским

индексам, тогда частицы, принадлежащие одной ячейке, будут идти в массиве последовательно.

Пример получившихся уровней для некоторого распределения частиц для $\mathrm{depth} = 4$, а также визуализация обхода ячеек мортоновской кривой приведены на рис. 2.



а) *б)* *в)* *г)*

Рис. 2. а)—в): ячейки на первом, втором и третьем уровне дерева соответственно с подписанными мортоновскими индексами ячеек; г): обход всех ячеек дерева третьего уровня мортоновской кривой.
Fig. 2. а)—в): cells at the first, second, and third levels of the tree, respectively, with assigned Morton indices of cells; г): traversing all cells at the third level of the Morton curve.

Основное время построения дерева занимает сортировка частиц по их мортоновским кодам, поэтому важно сделать эту сортировку оптимальной. Для сортировки можно использовать различные библиотечные функции; также была реализована сортировка подсчетом. Сравнение всех вариантов представлено в табл. 1.

Табл. 1. Сортировка 50,000,000 частиц на CPU с помощью различных функций. В скобках указано ускорение относительно одного потока.
Table 1. Sorting 50,000,000 particles on a CPU using different functions, with the acceleration relative to a single thread shown in parentheses.

Число потоков	`std::sort`	`tbb:: parallel_sort`	`boost:: block_indirect_sort`	`counting sort`
1	7.82	5.79	5.68	4.52
2	4.32	3.18	3.04	3.05
4	2.82	2.11	1.78	1.93
8	1.58	1.26	1.25	1.08
16	1.31 (x5.97)	1.06 (x5.46)	0.78 (x7.28)	0.61 (x7.4)

В табл. 2 приведено аналогичное сравнение с доступными библиотечными функциями на GPU, в том числе с `cub::SortPairs`, которая реализует эффективную для сортировки целых чисел Radix Sort.

Табл. 2. Сортировка 50,000,000 частиц на GPU с помощью различных функций.
Table 2. Sorting 50,000,000 particles on a CPU using different functions.

`thrust::sort`	`cub::SortPairs`	`counting sort`
0.193	0.154	0.130

Исходя из приведенных таблиц, можно сделать вывод, что оптимальнее всего использовать сортировку подсчетом: она не только эффективно распараллеливается, но также легко реализуется.

3.3 Ускорение кода на CPU и GPU

При реализации формул мультипольных разложений необходимо постоянно возводить числа в целую степень, но делать это с помощью встроенной функции `pow`, которая учитывает

возможность возведения в нецелую степень, будет неэффективно. Вместо этого следует использовать “быстрое” возведение в целую неотрицательную степень, код для которого выглядит следующим образом:

```
template <typename T>
__device__ __host__ constexpr T Pow (T base, unsigned int exp) {
       T res = 1;
       while (exp) {
       if (exp & 1) {
       res *= base;
       }
       exp >>= 1;
       base *= base ;
       }
       return res ;
}
```

Листинг 1. Реализация функции возведения в целую неотрицательную степень
Listing 1. Implementation of the power function for non-negative integer exponents

В формулах для мультипольных разложений также присутствуют различные постоянные коэффициенты, как, например, биномиальные коэффициенты, которые можно просчитать заранее один раз и сохранить в массив. Аналогично можно один раз просчитать матрицы Вигнера: в силу того, что ячейки дерева всегда имеют форму квадрата, все возможные углы поворота заранее известны, поэтому можно в начале работы программы просчитать все матрицы. Все возможные углы можно найти, рассмотрев куб, состоящий из $7\times7\times7$ кубиков, и посчитав углы между центральной и всеми остальными ячейками, не включая ближних соседей.

Заметим, что сферические гармоники обладают следующим свойством:

$$Y_n^m(\theta,\phi) = \overline{Y_n^{-m}(\theta,\phi)}, \tag{13}$$

аналогично $M_n^m = \overline{M_n^{-m}}$, $L_n^m = \overline{L_n^{-m}}$, где черта означает комплексное сопряжение. Из этого следует, что можно хранить только половину коэффициентов и выполнять операции только над этой половиной, что в два раза сокращает количество вычислений.

После применения данных простых оптимизаций мы перейдем к обсуждению оптимизаций самых трудозатратных алгоритмов БММ, а именно M2L и P2P. В случае M2L можно соптимизировать вращения коэффициентов: при повороте вокруг оси z необходимо вычислять выражение вида

$$d_{m,k}^n M_n^k + d_{m,-k}^n M_n^{-k}.$$

Используя соотношение (13) для сферических функций, получаем

$$d_{m,k}^n M_n^k + d_{m,-k}^n M_n^{-k} = \mathrm{Re}\, M_n^k (d_{m,k}^n + d_{m,-k}^n) + i\,\mathrm{Im}\, M_n^k (d_{m,k}^n - d_{m,-k}^n),$$

то есть имеем два умножения вместо четырех.

Перейдем к оптимизации P2P. Заметим, что при расчете влияния двух частиц достаточно посчитать расстояние один раз, а затем можем получить потенциал или силу для обеих частиц; таким образом, мы переходим от кода

```
for (int i = 0; i < N; ++i)
       for (int j = 0; j < N; ++j) {
       // считаем расстояние между точками r(i,j)
       p[i] += q[j] / r(i,j);
       }
```

к коду

```
for (int i = 0; i < N; ++i)
       for (int j = i; j < N; ++j) {
       // считаем расстояние между точками r(i,j)
              p[i] += q[j] / r(i,j);
              p[j] += q[i] / r(i,j);
       }
```

В зависимости от типа взаимодействия (сила или потенциал), ускорение может составить от : 1.3 до : 1.9 раз.

Следующая оптимизация, позволяющая еще более существенно ускорить P2P — это векторизация, то есть использование SIMD инструкций. Для этого необходимо хранить в виде отдельных массивов все x, y и z координаты частиц, а также их заряды. В результате при обработке листьев можно с помощью одной инструкции процессора находить разность координат для 2, 4 или 8 частиц (SSE, AVX и AVX512 инструкции), а также их произведение, сумму, корень (для нахождения расстояния) и т.д. В случае использования AVX512 инструкций ускорение P2P может составлять около 3 раз.

С учетом всего вышесказанного, можно реализовать не только эффективный последовательный код, но и параллельный. Для CPU версии был реализован параллелизм для систем с общей и распределенной памятью с помощью библиотек TBB и MPI. Реализация параллельных расчетов на CPU не составляет большой сложности при готовом последовательном коде, в то время как реализация для GPU имеет ряд особенностей, которые необходимо учесть для получения максимального ускорения:

1) Использование константной памяти. Для реализации формул мультипольных разложений необходимо знать биномиальные коэффициенты, коэффициенты трансляции M2M, M2L, L2L и матрицы Вигнера. Если их передавать в качестве внешней переменной в kernel-функцию, то будет происходить обращение к глобальной памяти, которое занимает много времени. В то же время перечисленные коэффициенты (за исключением матриц Вигнера, они занимают слишком много места, а константная память ограничена 64 Кб) можно хранить в константной памяти, доступ к которой лишь немногим медленнее, чем к регистровой.
2) Использование регистровой и локальной памяти. Если массив приходит на вход kernel-функции и обращение к его элементам происходит чаще одного раза, то можно завести локальный статический массив и скопировать в него элементы внешнего массива, таким образом существенно экономя на обращениях к памяти. Данный статический массив будет находится в локальной памяти, доступ к которой несколько медленее, чем к регистровой, но быстрее, чем к глобальной. Если же нужно часто обращаться к одному элементу массива, то его можно скопировать в локальную переменную, в этом случае будет обращение к регистровой памяти.
3) Использование shared memory и исполнение нитями в блоке одинакового кода. Нагляднее всего это можно продемонстрировать при обходе листьев для подсчета потенциала (силы) из локальных разложений. Можно реализовать параллелизм по частицам, тогда у всех нитей в блоке в общем случае будут различные локальные разложения и будет исполняться различный код. В то же время, параллелизм можно осуществить по листьям, тогда у всех нитей в блоке будет одинаковое локальное разложение, при этом коэффициенты локального разложения можно скопировать в shared memory, оптимизировав обращения к памяти. В этом случае упомянутое выше копирование в локальный массив не даст выигрыша, так как для подсчета потенциала (силы) каждая нить только один раз обращается к коэффициентам разложения; при использовании же shared memory каждой нити достаточно скопировать какую-то часть массива локальных коэффициентов в общий массив, а затем в цикле обращаться уже к локальному массиву – таким образом оптимизируются обращения

к памяти.

4. Результаты расчетов

В данном разделе мы представим результаты расчетов для взаимодействия частиц по закону Ньютона и по закону Био—Савара. Для тестов бралось равномерное распределение частиц в кубе $[0;1]^3$. Погрешность алгоритма будем считать в пространстве L_2, т.е.

$$\varepsilon_{L2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{\mathrm{P}\mathbf{f}_i^{\text{approx}} - \mathbf{f}_i^{\text{exact}}\,\mathrm{P}^2}{\mathrm{P}\mathbf{f}_i^{\text{exact}}\,\mathrm{P}^2}}.$$

Так как код был распараллелен для систем с общей и распределенной памятью, а также с использованием CUDA, будут приведены таблицы, отражающие эффективность распараллеливания. Модель CPU — Intel Core i9-10980XE, GPU — NVIDIA Titan V.

4.1 Параллелизм для систем с общей памятью

В табл. 3-4 представлено время работы (в секундах) различных этапов алгоритма при разном числе потоков (то есть тестируется система с общей памятью).

Табл. 3. Время работы различных этапов БММ для ньютоновского потенциала. Число частиц — 5,000,000, погрешность $\varepsilon_{L2} \approx 10^{-7}$ (в скобках указано ускорение относительно одного потока)

Table 3. The time taken by different stages of the FMM for the Newtonian potential. The particle count is 5,000,000, with the error $\varepsilon_{L2} \approx 10^{-7}$ (the acceleration relative to a single thread is provided in parentheses)

Число потоков	Tree	P2M	M2M	M2L+L2L	P2P+L2P	Total	Direct
1	9.49	3.31	0.36	18.5	42.7	75.2	: 2 мес.
2	6.54	1.65	0.18	9.23	23.25	41.35	: 4 нед.
4	3.05	0.85	0.09	4.61	11.95	21.05	: 2 нед.
8	2.05	0.40	0.042	2.26	6.05	11.26	: 8 дн.
16	1.57 (x6.0)	0.22 (x15.0)	0.025 (x14.4)	1.17 (x15.8)	3.22 (x13.3)	6.66 (x11.3)	: 4 дн.
GPU	0.59 (x16)	0.089 (x37)	0.018 (x20)	0.44 (x42)	0.41 (x104)	1.55 (x49)	: 8 ч.

Табл. 4. Время работы различных этапов БММ для закона Био—Савара. Число частиц — 5,000,000, погрешность $\varepsilon_{L2} \approx 10^{-7}$ (в скобках указано ускорение относительно одного потока)

Table 4. The time taken by different stages of the FMM for the Biot—Savart law. The particle count is 5,000,000, with the error $\varepsilon_{L2} \approx 10^{-7}$ (ehe acceleration relative to a single thread is provided in parentheses)

Число потоков	Tree	P2M	M2M	M2L+L2L	P2P+L2P	Total	Direct
1	0.69	5.44	0.40	69.4	30.6	106.7	: 36 ч
2	0.46	2.92	0.22	36.7	16.2	56.8	: 18 ч
4	0.30	1.48	0.11	18.9	8.55	29.6	: 9 ч
8	0.20	0.89	0.07	11.4	5.35	18.2	: 4.5 ч
16	0.154 (x4.5)	0.46 (x11.8)	0.034 (x11.8)	5.77 (x12.0)	3.17 (x9.7)	9.85 (x10.8)	: 1.2 ч
GPU	0.082 (x8.1)	0.26 (x21)	0.071 (x5.6)	5.78 (x12)	0.40 (x77)	6.61 (x26)	: 250 с

В случае видеокарты видно, что эффективнее всего параллелится обработка листьев и M2L — это самые трудоемкие операции во всем алгоритме, поэтому суммарно получается хорошее ускорение, несмотря на незначительное ускорение остальных процедур.

Отметим, что алгоритм для закона Био—Савара можно реализовать, выполнив три раза БММ для ньютоновского потенциала, однако эффективнее будет реализовать его отдельно, считая заряды частиц векторами, в результате все мультипольные коэффициенты становятся трехмерными векторами из комплексных чисел вместо обычных комплексных чисел. Данный подход позволяет эффективно векторизовать код, и в результате для расчета сил по закону Био—Савара на CPU требуется всего на $20-30\%$ больше времени по сравнению с обычным БММ; наивная реализация потребовала бы, очевидно, в три раза больше времени. Для видеокарты разница между двумя алгоритмами составляет больше двух раз в связи с тем, что больше всего после векторизации зарядов замедляется операция M2L, а ее эффективность распараллеливания не такая высокая, как у обработки листьев, которая не сильно меняется по времени при изменении способа расчеты силы. Связано это прежде всего с тем, что среди операций P2P и L2P большую часть времени занимает прямой расчет P2P, а подсчеты ньютоновской силы и силы по закону Био—Савара по прямой формуле почти не отличаются по времени.

4.2 Параллелизм для систем с распределенной памятью

В табл. 5-6 представлено время работы (в секундах) БММ в зависимости от различного числа MPI процессов и некоторого числа потоков на каждом процессе (то есть тестируется система с общей и распределенной памятью).

В дальнейшем мы приведем сравнение с открытой реализацией БММ, где будет показано, что полученное ускорение как для систем с общей памятью, так и для систем с распределенной памятью является весьма хорошим.

Табл. 5. Время работы БММ для ньютоновского потенциала при разном числе MPI процессов (M) и потоков (N). Число частиц — 5,000,000, погрешность $\varepsilon_{L2} \approx 10^{-7}$ (в скобках указано ускорение относительно последовательной версии)

Table 5. Performance of the FMM for the Newtonian potential with different number of MPI processes (M) and threads (N). The system involves 5,000,000 particles, with an error $\varepsilon_{L2} \approx 10^{-7}$ (acceleration relative to the sequential version is provided in parentheses)

N нитей / *M* проц.	1	2	4	8	16	GPU
1 (без MPI)	78.9	41.1 (x1.92)	24.7 (x3.19)	12.8 (x6.16)	7.15 (x11.0)	2.70 (x29)
2	84.1 (x0.94)	48.8 (x1.62)	24.6 (x3.21)	13.0 (x6.07)	7.3 (x10.8)	—
3	48.7 (x1.62)	25.2 (x3.13)	13.1 (x6.02)	7.2 (x11.0)	4.6 (x17.2)	—
4	21.1 (x3.74)	12.3 (x6.41)	7.02 (x11.2)	4.29 (x18.4)	3.06 (x25.8)	—

Табл. 6. Время работы БММ для закона Био—Савара при разном числе MPI процессов (M) и потоков (N). Число частиц — 5,000,000, погрешность $\varepsilon_{L2} \approx 10^{-7}$ (в скобках указано ускорение относительно последовательной версии)

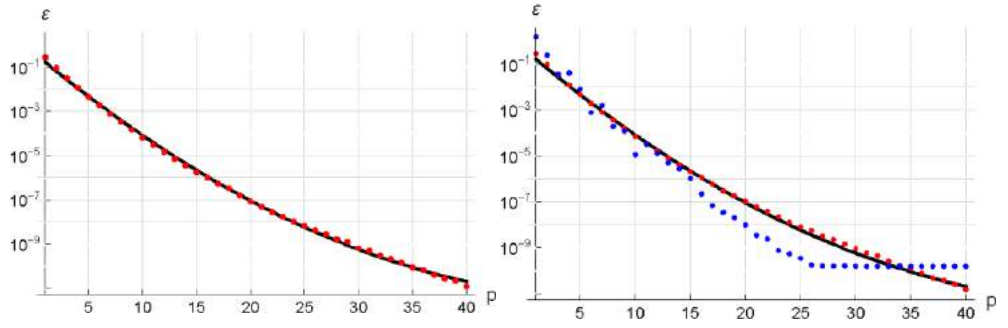
Table 6. Performance of the FMM for the Biot—Savart law with different number of MPI processes (M) and threads (N). The system involves 5,000,000 particles, with an error $\varepsilon_{L2} \approx 10^{-7}$ (acceleration relative to the sequential version is provided in parentheses)

N нитей / *M* проц.	1	2	4	8	16	GPU
1 (без MPI)	106.7	56.8 (x1.88)	29.6 (x3.60)	18.2 (x5.86)	9.85 (x10.8)	6.61 (x16)
2	107 (x0.99)	57.9 (x1.84)	36.0 (x2.96)	18.7 (x5.71)	10.2 (x10.5)	—

3	55.1 (x1.94)	30.5 (x3.50)	19.0 (x5.6)	10.0 (x10.7)	6.3 (x16.9)	—
4	28.8 (x3.70)	15.8 (x6.75)	8.84 (x12.1)	5.97 (x17.9)	4.0 (x26.7)	—

4.3 Определение количества мультиполей и глубины дерева

В быстром методе мультиполей имеются две величины, которые необходимо определить перед выполнением программы. Первая — это число мультиполей, которое влияет на точность расчета. Для определения зависимости погрешности от числа мультиполей мы рассмотрим задачу со $100\,000$ частиц (были рассмотрены варианты и с другим числом частиц, но результаты везде оказались схожие). Результаты изображены на рис. 3.



а) закон Ньютона *б) закон Био—Савара*

Рис. 3. Ошибка ε_{L2} для подсчета силы (красный цвет) и потенциала (синий цвет) в зависимости от числа мультиполей. Черным цветом изображена аппроксимация зависимости $\varepsilon(p)$ для силы.

Fig. 3. The error ε_{L2} in calculating force (red) and potential (blue) as a function of the number of multipoles. The approximation of the force error $\varepsilon(p)$ is represented in black.

В результате получилось, что погрешности для ньютоновской силы и для силы по закону Био—Савара имеют схожие величины. Интересно поведение ошибки вычисления потенциала — при небольшом числе мультиполей она убывает быстрее, чем для силы, а затем выходит на плато.

Погрешность для силы была аппроксимирована с помощью метода наименьших квадратов, в результате получилось

$$\varepsilon(p) \approx \exp(0.00868\,p^2 - 0.94\,p - 0.88), \quad p(\varepsilon) \approx 54.2 - 10.73\sqrt{\ln(2.4\varepsilon) + 25.5}.$$

Данные формулы удобно использовать в алгоритме для выбора числа мультиполей в зависимости от требуемой точности.

Другой важный параметр — глубина дерева. Глубина дерева влияет на время расчета, и для каждой конкретной задачи существует оптимальная глубина. Однако эта величина зависит от многих факторов: число мультиполей, оптимальность реализации различных этапов алгоритма (в первую очередь, M2L и P2P — они занимают наибольшее время), а также само расположение частиц: при равномерном распределении и при сильно неоднородном оптимальная глубина дерева может сильно отличаться. В этой связи мы не можем привести конкретные формулы для определения оптимальной глубины дерева.

4.4 Сравнение с другими реализациями

Будем проводить сравнение для задачи взаимодействия частиц по закону Ньютона. На рис. 4 представлено сравнение с реализацией [14], где есть однопоточный CPU вариант и GPU вариант.

В случае кода для CPU текущая реализация работает быстрее примерно в 3 раза; в случае же видеокарты ускорение куда более существенное и достигает 10 раз.

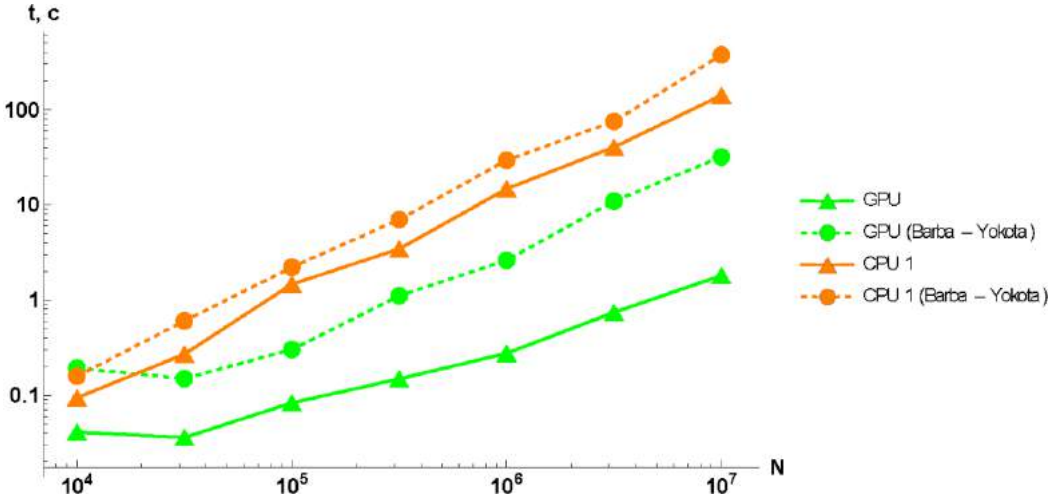


Рис. 4. Сравнение текущей реализации БММ с реализацией R. Yokota и L.A. Barba на видеокарте (GPU) и на одном потоке процессора (CPU 1).
Fig. 4. The comparison between the proposed FMM implementation and R. Yokota and L.A. Barba's GPU implementation and sequential CPU version (CPU 1).

Подробный анализ качества реализации CPU версии можно провести, сравнивая с реализацией [18], которая включает распараллеливание для систем с общей и распределенной памятью, а также, что очень важно, векторизацию P2P (рис. 5). Результаты на одном потоке получились практически идентичные; на 18 потоках текущая реализация несколько лучше параллелится, в то время как с добавлением MPI на очень большом числе частиц ($>10^7$) возможно небольшое отставание; в целом же, временные результаты получились схожими.

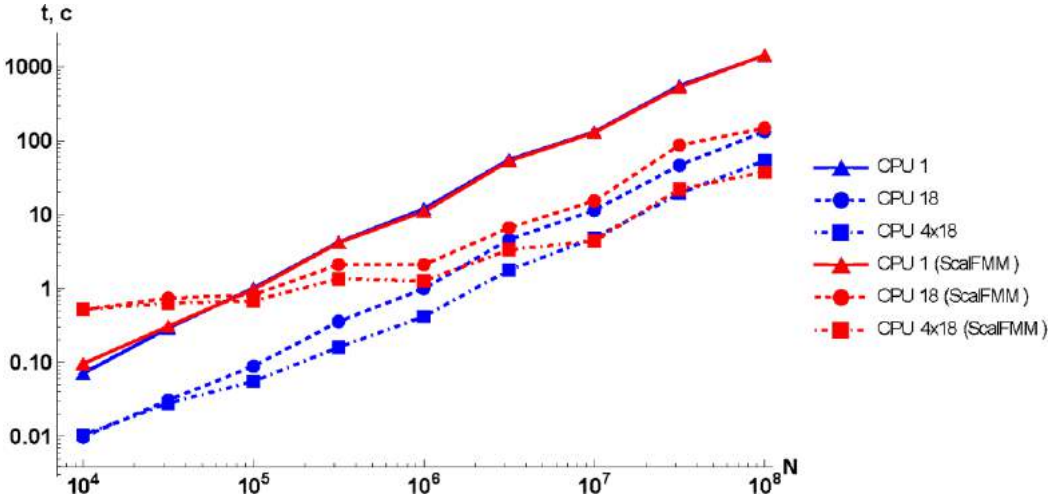


Рис. 5. Сравнение текущей реализации трехмерного БММ со ScalFMM на одном потоке (CPU 1), на 18 потоках (CPU 18) и на 4 узлах с 18 потоками (CPU 4x18).
Fig. 5. Comparison of the proposed implementation of the FMM with ScalFMM on a single thread (CPU 1), on 18 threads (CPU 18), and on 4 nodes with 18 threads each (CPU 4x18).

Данные сравнения подтверждают эффективность текущей реализации трехмерного быстрого метода мультиполей.

5. Заключение

В работе представлено описание быстрого метода мультиполей для частиц, взаимодействующих по закону Ньютона и по закону Био—Савара. Представлены все необходимые для расчета формулы, а также упомянута нормировка сферических гармоник и матриц Вигнера – тематика, редко освещаемая в других работах. Кроме теоретичеких формул, приведены рекуррентные соотношения, позволяющие запрограммировать алгоритм; кроме этого, подробно описаны техники, позволяющие существенно ускорить работу кода, в том числе подробно изложено построение дерева с помощью кривой Мортона, что позволяет эффективно распараллелить код на CPU и GPU. С использованием данных техник был реализован программный комплекс на языке C++, распараллеленный для систем с общей памятью (с помощью библиотеки TBB) и для систем с распределенной памятью (с использованием MPI), а также для видеокарт с использованием технологии CUDA. Для анализа эффективности написанной программы было проведено сравнение с двумя реализациями, которые также включают различные способы распараллеливания, и наглядно подтверждена высокая скорость работы кода.

Список литературы / References

[1]. J. Barnes, P. Hut. A hierarchical O(NlogN) force-calculation algorithm. // Nature. 1986. Vol. 324. P. 446–449. DOI: 10.1038/324446a0.

[2]. K. Nabors, J. White. FastCap: A multipole-accelerated 3-D capacitance extraction program // IEEE Transactions on Computer-Aided Design. 1991. Vol. 10. P. 1447–1459.

[3]. M. Kamon, M. J. Tsuk, J. K. White. FASTHENRY: a multipole-accelerated 3-D inductance extraction program. // IEEE Transactions on Microwave Theory and Techniques. 1994. Vol. 42, № 9. P. 1750–1758.

[4]. L. Greengard, M.C. Kropinski, A. Mayo. Integral equation methods for stokes flow and isotropic elasticity in the plane. // J. Comput. Physics. 1996. Vol. 125, № 2. P. 403–414. DOI: 10.1006/jcph.1996.0102

[5]. A.-K. Tornberg, L. Greengard. A fast multipole method for the three-dimensional Stokes equations. // Journal of Computational Physics. 2008. Vol 227, № 3. P. 1613–1619. DOI: 10.1016/j.jcp.2007.06.029

[6]. L. Greengard, V. Rokhlin. A fast algorithm for particle simulations. // J. Comput. Phys. 1987. Vol. 73, № 2. P. 325–348. DOI: 10.1016/0021-9991(87)90140-9

[7]. H. Cheng, L. Greengard, V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. // J. Comput. Phys. 1999. Vol. 155. P. 468–498. DOI: 10.1006/jcph.1999.6355

[8]. M. A. Epton, B. Dembart. Multipole translation theory for the three-dimensional Laplace and Helmholtz equations. // SIAM J. on Scientific Computing. 1995. Vol. 16, №4. P. 865–897. DOI: 10.1137/0916051

[9]. R. Beatson, L. Greengard. A short course on fast multipole methods. // Wavelets, Multilevel methods and Elliptic PDEs. 1997. P. 1–37.

[10]. J. Makino. Yet another fast multipole method without multipoles — pseudoparticle multipole method. // J. Comput. Phys. 1999. Vol. 151, № 2. P. 910–920.

[11]. A. Kawai, J. Makino. A simple formulation of the fast multipole method: pseudo-particle multipole method. // SIAM Conference on Parallel Processing for Scientific Computing. 1998. DOI: 10.48550/arXiv.astro-ph/9812431

[12]. N.Y. Gnedin. Hierarchical Particle Mesh: An FFT-accelerated fast multipole method. // The Astrophysical Journal Supplement Series. 2019. Vol. 243, № 2. 19 p. DOI: 10.3847/1538-4365/ab2d24

[13]. C.R. Anderson. An implementation of the fast multipole method without multipoles. // SIAM J. on Scientific and Statistical Computing. 1992. Vol. 13. P. 923–947. DOI: 10.1137/0913055

[14]. R. Yokota, L.A. Barba. Treecode and fast multipole method for N-Body simulation with CUDA. // GPU Computing Gems. Emerald Edition. Boston: Elsevier and Morgan Kaufmann, 2011. P. 113–132. DOI: 10.1016/B978-0-12-384988-5.00009-7

[15]. B. Bramas. (2020). TBFMM: A C++ generic and parallel fast multipole method library. // Journal of Open Source Software. 2020. Vol. 5, № 56. P. 2444–2453. DOI: 10.21105/joss.02444

[16]. Z. Gimbutas, L. Greengard. Computational Software: Simple FMM Libraries for Electrostatics, Slow Viscous Flow, and Frequency-Domain Wave Propagation. // Communications in Computational Physics. 2015. Vol. 18, № 2. 516–528. DOI: 10.4208/cicp.150215.260615sw

[17]. T. Wang, R. Yokota, L.A. Barba. ExaFMM: a high-performance fast multipole method library with C++ and Python interfaces. // Journal of Open Source Software. 2021. Vol. 6, № 61. P. 3145–3149. DOI: 10.21105/joss.03145
[18]. P. Blanchard, B. Bramas, O. Coulaud, et al. ScalFMM: A Generic Parallel Fast Multipole Library. // SIAM Conference on Computational Science and Engineering. 2015.
[19]. E.O. Steinborn, K. Ruedenberg. Rotation and translation of regular and irregular solid spherical harmonics. // Advances in Quantum Chemistry. 1973. Vol. 7. P. 1–81.
[20]. L. Biedenharn, J. Louck, P. Carruthers. Angular momentum in quantum physics: Theory and Application. Cambridge: Cambridge University Press, 1984. 716 p.
[21]. Abramowitz M., Stegun I.A. Handbook of mathematical functions. New York: Dover, 1965. 1046 p.
[22]. L. Greengard, V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. // Acta Numerica. 1997. Vol. 6. P. 229–269. DOI: 10.1017/S0962492900002725
[23]. E. Wigner. Group theory and its application to the quantum mechanics of atomic spectra. New York: Academic Press, 1959. 384 p. DOI: 10.1017/S0013091500025220
[24]. H. Dachsel. Fast and accurate determination of the Wigner rotation matrices in the fast multipole method. // J. Chem. Phys. 2006. Vol. 124. P. 144115–144121. DOI: 10.1063/1.2194548
[25]. Z. Gimbutas, L. Greengard. A fast and stable method for rotating spherical harmonic expansions. J. Comput. Phys. 2009. Vol. 228, № 16. P. 5621-5627. DOI: 10.1016/j.jcp.2009.05.014

Информация об авторах / Information about authors

Виктор Михайлович АУШЕВ – студент магистратуры кафедры “Прикладная математика”. Сфера научных интересов: быстрые методы, высокопроизводительные вычисления, численные методы электродинамики, метод конечных элементов, численные методы линейной алгебры.

Viktor Mikhailovich AUSHEV – master's student of Applied Mathematics department. Research interests: fast methods, high performance computing, numerical methods of electrodynamics, finite element method, numerical methods of linear algebra.

DOI: 10.15514/ISPRAS-2023-35(6)-15

Численное и экспериментальное исследование гидродинамики теплообменного аппарата

Е.С. Байметова, ORCID: 0000-0002-4534-0936 <*baimetova.e.s@gmail.com*>
Е.А. Митрюкова, ORCID: 0000-0002-5350-5366 <*mit_e_a@mail.ru*>

Ижевский государственный технический университет имени М.Т. Калашникова,
426069, Россия, Ижевск, ул. Студенческая, д. 7.

Аннотация. В работе проведен сравнительный анализ результатов численного моделирования и экспериментального исследования течения жидкости в теплообменном аппарате (ТА). В качестве инструмента для численного моделирования использовался пакет для решения задач МСС OpenFOAM. Экспериментальные исследования проводились на гидравлическом стенде ТМЖ-2М. Теплообменный аппарат подключался напрямую к модулям гидравлического стенда. В качестве рабочей жидкости была использована дистиллированная вода комнатной температуры. Верификация экспериментальных результатов проводилась на основе численного моделирования в пакете OpenFOAM с применением тех же характеристик сред, используемых при натурном эксперименте. Анализ полученных данных показал хорошую сходимость результатов численного и экспериментального исследования многосекционного ТА.

Ключевые слова: численное моделирование; экспериментальное исследование; теплообменный аппарат; гидродинамика; сравнительный анализ.

Для цитирования: Байметова Е.С., Митрюкова Е.А. Численное и экспериментальное исследование гидродинамики теплообменного аппарата. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 235–246. DOI: 10.15514/ISPRAS–2023–35(6)–15**.**

Numerical and Experimental Study of the Hydrodynamics of a Heat Exchanger

E.S. Baymetova, ORCID: 0000-0002-4534-0936 <baimetova.e.s@gmail.com>
E.A. Mitryukova, ORCID: 0000-0002-5350-5366 <*mit_e_a@mail.ru*>

Kalashnikov Izhevsk State Technical University
7, Studencheskaya st., Izhevsk, 426069, Russia.

Abstract. The work provides a comparative analysis of the results of numerical modeling and experimental studies of fluid flow in a heat exchanger (HE). The OpenFOAM package for solving CFD problems was used as a tool for numerical modeling. Experimental studies were carried out on the TMJ-2M hydraulic stand. The heat exchanger was connected directly to the modules of the hydraulic stand. Distilled water at room temperature was used as the working fluid. Verification of the experimental results was carried out on the basis of numerical modeling in the OpenFOAM package using the same characteristics of the media used in the field experiment. Analysis of the obtained data showed good convergence of the results of the numerical and experimental study of the multi-section HE.

Keywords: numerical simulation; experimental research; heat exchanger; hydrodynamics; comparative analysis.

For citation: Baymetova E.S., Mitryukova E.A. Numerical and experimental study of the hydrodynamics of a heat exchanger. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 235-246 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-15**.**

1. Введение

Теплообменные аппараты являются неотъемлемой частью многих энергетических систем. Высокие требования к современному энергетическому оборудованию приводят к постоянному совершенствованию теплообменных устройств и появлению нестандартных конструкторских исполнений ТА, направленных на повышение их ресурсоемкости и энергоэффективности. Одним из таких решений является распределение потока рабочей жидкости по параллельным каналам малого сечения с внутренним оребрением. Охлаждение формирующихся при этом микротечений происходит более эффективно, чем в каналах стандартной конфигурации. Такой подход требует применения раздающих и собирающих коллекторных систем сложной формы. В [1, 2] рассмотрена гидродинамика типовых раздающих коллекторных систем. Вопросы гидродинамики широкого ряда коллекторных систем наиболее полно исследуются в справочнике по гидравлическим сопротивлениям И.Е. Идельчика [3]. Однако, для современных конструкций данные методики не применимы ввиду их нестандартной конфигурации. В результате, оценка гидравлических характеристик возможна либо с помощью эксперимента [4], либо численного моделирования [5-9]. Комплексный подход, заключающийся в использовании математической гидравлической модели ТА, построенной на результатах численного моделирования гидродинамики ТА, верифицированных с помощью продувочных экспериментов на воде, представляется наиболее эффективным с точки зрения временных и экономических затрат.

Исследуемый в работе многосекционный ТА включает в себя раздающий и принимающий коллектора (рис. 1a), которые соединены между собой набором из 11 параллельно расположенных каналу подвода и отвода рабочей жидкости секций (рис. 1b). Каждая секция содержит по 6 микроканалов, внутреннее оребрение которых выполнено в виде симметричных трапеций с узкой верхней кромкой (рис. 1c).

a) *b)* *c)*

Рис.1. Многосекционный теплообменный аппарат:
a) впускной и выпускной каналы, b) общий вид многоканального ТА,
c) внутренняя геометрия микроканалов
Fig.1. Multi-section heat exchanger:
a) inlet and outlet channels, b) general view of multi-channel HE, c) internal geometry of microchannels

Процесс течения рабочей жидкости в данном ТА достаточно сложный, ввиду геометрических особенностей данного вида конструкции. Вопросы моделирования гидродинамики данной

коллекторной системы рассматриваются в работах [7, 9]. В [7] рассмотрена упрощенная схема исследуемого ТА, без учета течения рабочей жидкости в микроканалах.

В данной работе проводится экспериментальное и численное исследование гидродинамики многосекционного ТА, с учетом внутренних оребренных микроканалов.

2. Экспериментальные исследования гидродинамических процессов в многосекционном ТА

Объектом исследования является многосекционный ТА, по которому прогоняется рабочая жидкость (рис. 1). Канал подвода и отвода рабочей жидкости выполнен в виде трубы круглого сечения с диаметром $d = 0{,}021$ м. Каналы отвода сгруппированы в 11 секций, каждая из которых содержит по шесть микроканалов с внутренним оребрением (рис. 1c).

Экспериментальное исследование проводилось на гидравлическом стенде ТМЖ-2М, представленном на рис. 2. Теплообменный аппарат подключался напрямую к модулям стенда. В качестве рабочей жидкости была использована дистиллированная вода комнатной температуры.

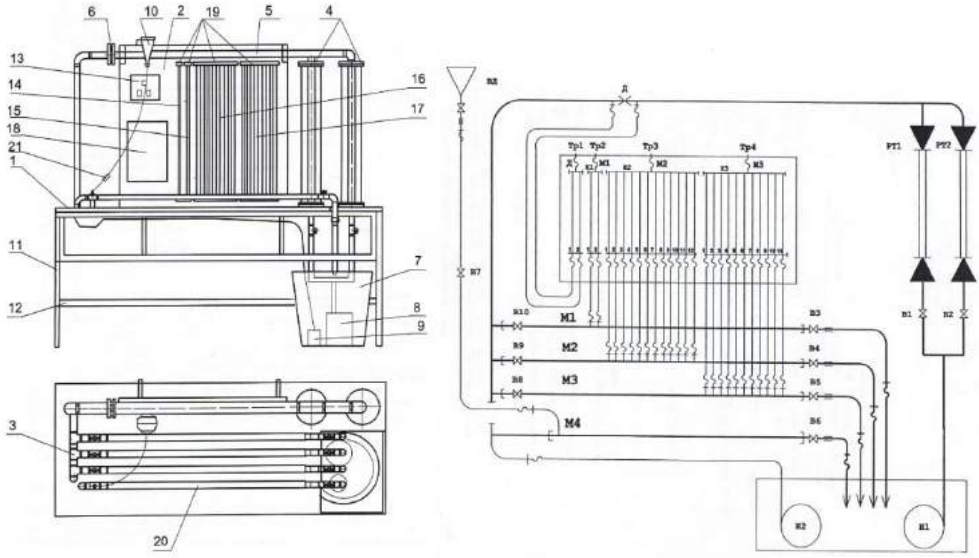


Рис.2. Внешний вид гидравлического стенда ТМЖ-2М
Fig.2. Appearance of the TMJ-2M hydraulic stand

Стенд изготовлен в напольном исполнении. В состав стенда входят: стол 1, щит пьезометров 2, впускной коллектор 3, ротаметры 4, напорная магистраль 5 со встроенной диафрагмой 6, бак 7 с насосом 8 и помпой 9, делительная воронка 10, комплект исследуемых модулей.

На поверхности стола 1 закреплены два ротаметра 4 (Р1 и Р2), верхние фланцы которых с помощью трубопроводов подведены к напорной магистрали 5. Нижние фланцы ротаметров 4 через трубопроводную арматуру (вентили В1 и В2) соединены с насосом 8 (Н1). В напорную магистраль 5 вмонтирована мерная диафрагма 6, контрольные точки которой с помощью гибких трубок соединены с пьезометрическими трубками щита пьезометров 2. Напорная магистраль 9 подведена к коллектору 3. Щит пьезометров 2 установлен вертикально на задних стойках стола 1. На щите пьезометров 2 расположены панель управления 13, четыре группы пьезометров 14 – 17, штатив с делительной воронкой 10 и панель для информации 18. На панели управления 13 размещены клавиши включения сети, насоса Н1 и помпы Н2. Каждая из четырех групп пьезометров 14-17 состоит из прозрачных пьезометрических трубок, верхние концы которых объединены между собой общими

коллекторами 19. В коллекторах 19 выведены гибкие сливные трубки с зажимами для выравнивания давлений в пьезометрах. Кроме того, имеется модуль 20 для демонстрации режимов течения жидкости с ее подкрашиванием через гибкий шланг 21.

Схема подключения ТА к гидравлическому стенду представлена на рис. 3. ТА крепился к расчетным модулям при помощи гибких шлангов, верхний шланг был подключен к входному модулю М1, на который подавалась вода в различном скоростном диапазоне, нижний шланг был подключен к выходному модулю В3. К входному и выходному отверстиям ТА был подключен тройник, соединяющий ТА с панелью пьезометров 16 и модулем подвода и отвода рабочей жидкости.

Рис.3. Схема подключения ТА к гидравлическому стенду ТМЖ-2М
Fig.3. Connection diagram of the HE to the TMJ-2M hydraulic stand

В рамках экспериментальных исследований была проведена серия испытний, показания пьезометров при которых снимались при подаче рабочей жидкости через ротаметры Р1 и Р2. Максимальный расход подаваемый ротаметрами Р1 и Р2 составляет $Q = 0{,}0004\ м^3/с - 0{,}0007 м^3/с$. По показаниям пьезометров рассчитывался перепад давлений в раздающем и собирающем коллекторах, подключенных к модулям М1 и В3, производился расчет гидравлических сопротивлений коллекторной системы в целом по формуле:

$$\zeta = \frac{2\Delta p}{\rho U_{вх}^2},$$

где $\rho = 998\ кг/м^3$ – плотность воды, $U_{вх}$ – средняя скорость потока на входе в коллектор. Зависимость коэффициента сопротивления коллектора с микроканалами от числа Рейнольдса приведена на рис. 4. Данные показаны при полной подаче расхода через ротаметры Р1 и Р2 в диапазоне 10-100%. Наблюдается снижение коэффициента сопротивления с ростом числа Рейнольдса, связанное с уменьшением трения при увеличении скорости жидкости. Наиболее интенсивное снижение происходит в диапазоне чисел Рейнольдса $Re = 5000 \div 15000$.

При численном моделировании течения жидкости в многосекционном ТА с учетом гидродинамического подобия, исследовался диапазон чисел Рейнольдса $Re = 1044 \div 3281$.

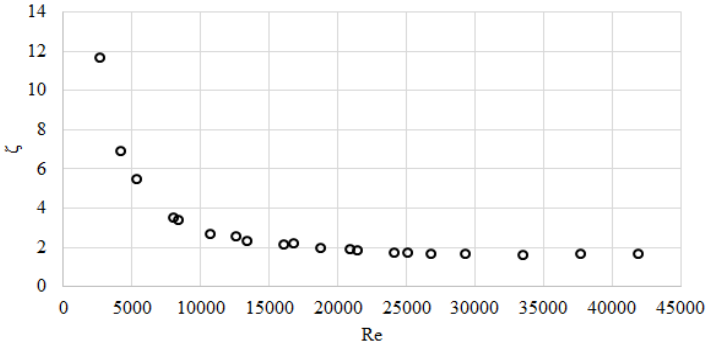


Рис.4. Экспериментальное представление распределения зависимости коэффициента гидравлических сопротивлений ζ от числа Рейнольдса Re
Fig.4. Experimental representation of the distribution of the dependence of hydraulic resistances coefficient ζ on the Reynolds number Re

3. Численные исследования гидродинамических процессов в многосекционном ТА

На рис. 5 изображена расчетная схема ТА. Сетка строилась с использованием модуля для построения сеток Mesh в пакете Salome и импортировалась в пакет OpenFOAM для дальнейшего численного моделирования. После проведения анализа сеточной сходимости на 6 расчетных сетках по перепаду давления в системе была выбрана сетка, содержащая 41434117 млн. тетраэдральных элементов для 11 секционного ТА (рис. 5b). В области сшивки микроканалов с раздающей и собирающей частями коллекторной системы сетка содержала элементы с показаниями не ортогональности больше 70 градусов. Это учитывалось при настройке расчетных схем в файле fvSchemes. Визуализация данных проводилась в пакете ParaView.

Граничные условия, используемые при численном моделировании в пакете OpenFOAM:

- На границе «inlet» задавалась средняя скорость течения $U_{вх}$.
- На границе «outlet» задается избыточное давление 0 Па.
- На границах «walls» условие прилипания.

Численное моделирование проводилось на стационарном ПК, трудоемкость расчетов зависела от скорости подачи рабочей жидкости, чем больше скорость, тем дольше производился расчет до момента установления, один расчет производился 168-252 часа.

В качестве исследуемой рабочей жидкости использовалась дистиллированная вода комнатной температуры.

Математическая модель основывается на системе уравнений Навье-Стокса. При построении математической модели использовались следующие допущения: течение стационарное, жидкость несжимаемая, вязкость постоянная, шероховатость материала не учитывается.

Реальная рабочая жидкость в теплообменном аппарате – гидравлическое масло с максимальной скоростью подачи в ТА 5 м/с.

На первом этапе был проведен анализ гидродинамического подобия по числу Рейнольдса при максимальной гидродинамической нагрузке ТА, рабочий диапазон которого варьируется от 1 м/с до 5 м/с (кинематическая вязкость воды $\nu_в = 10^{-6}м^2/с$, гидравлического масла $\nu_м = 32 \cdot 10^{-6}м^2/с$). Течение жидкости исследуется в низкорейнольдсовом диапазоне скоростей, подача жидкости регулируется ротаметром Р1, максимальный расход которого ограничивается вентилем В1 до значения расхода, не превышающего $Q = 0{,}00004\ м^3/с$.

$$d = 0.021\text{м}, \qquad U_{\text{м(max)}} = 5\text{м/с}, \qquad (U_{\text{м}} = 1 \div 5\text{м/с}),$$

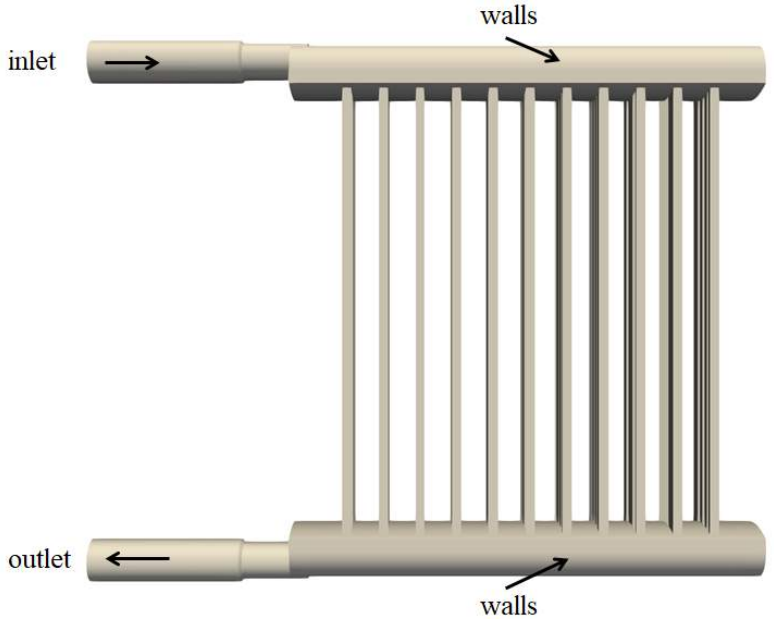
$$Re_{\text{м}} = Re_{\text{в}},$$

$$\frac{U_{\text{м}} \cdot d}{\nu_{\text{м}}} = \frac{U_{\text{в}} \cdot d}{\nu_{\text{в}}},$$

$$Re_{\text{м}} = \frac{U_{\text{м}} \cdot d}{\nu_{\text{м}}} = \frac{5 \cdot 0.021}{32 \cdot 10^{-6}} = 3281,$$

$$U_{\text{в}} = \frac{\nu_{\text{в}}}{\nu_{\text{м}}} \cdot U_{\text{м}} = \frac{10^{-6}}{32 \cdot 10^{-6}} \cdot 5 = 0.15\text{м/с},$$

где d – диаметр входного/выходного отверстия, $U_{\text{м}}, U_{\text{в}}$ – скорость подачи масла/воды, $Re_{\text{м}}$, $Re_{\text{в}}$ – число Рейнольдса для масла/воды.



a)

b)

Рис.5. Расчетная схема многосекционного теплообменного аппарата: a) схема подвода и отвода рабочей жидкости, b) расчетная сетка ТА
Fig.5. Calculation diagram of a multi-sectional heat exchanger a) diagram of supply and removal of working fluid, b) HE calculation grid

После подбора скоростей путем гидродинамического подобия вода/масло было проведено численное моделирование при разной скорости подачи рабочей жидкости, значения которых соответствуют значениям рабочего диапазона ТА в условиях эксплуатации на гидравлическом масле.

Соблюдение гидродинамического подобия по числу Рейнольдса требует проведения экспериментальных и численных исследований со скоростями воды до 0.15 м/с. Максимальное число Рейнольдса течения равно $Re = 3281$. Геометрическая сложность коллекторной системы приводит к турбулизации течения даже при низких скоростях на входе в систему, поэтому для расчета подключалась модель турбулентности Спаларта-Аллмареса (SA) [10]. Несмотря на то, что данная модель разрабатывалась для задач внешней аэродинамики, область ее применения оказалась намного шире и она с успехом применяется для расчета внутренних турбулентных течений.

Таким образом, математическая постановка задачи описывается следующими уравнениями:

$$\frac{\partial U_i}{\partial x_i} = 0, \tag{1}$$

$$U_j \frac{\partial U_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial {x^2}_j} - \frac{\partial(\overline{u_i u_j})}{\partial x_j}, \tag{2}$$

где U – вектор осредненной скорости, p – избыточное давление, отнесенное к плотности жидкости, ν - кинематический коэффициент вязкости, i, j – индексы, принимающие значения 1, 2, 3, $\overline{u_i u_j}$ – тензор напряжений Рейнольдса, который определяется как:

$$-\left(\overline{u_i u_j}\right) = \nu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3} k \delta_{ij}, \tag{3}$$

где ν_t – турбулентная вязкость, k – турбулентная кинетическая энергия.

Модель турбулентности SA основана на уравнении:

$$U_j \frac{\partial \tilde{\nu}}{\partial x_j} = \mathrm{c}_{b1}(1 - f_{t2})\tilde{S}\tilde{\nu} - \left[c_{w1} f_w - \frac{\mathrm{c}_{b1}}{k^2} f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{1}{\sigma}\frac{\partial}{\partial x_j}\left[(\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_j}\right] + \frac{\mathrm{c}_{b2}}{\sigma}\frac{\partial \tilde{\nu}}{\partial x_j}\frac{\partial \tilde{\nu}}{\partial x_i}, \tag{4}$$

где $\tilde{\nu}$ – модифицированная турбулентная вязкость, определяемая как:

$$\tilde{\nu} = \nu_t / f_{v1},$$

где f_{v1} – пристеночная функция затухания, со следующими замыкающими отношениями:

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \qquad \chi = \frac{\tilde{\nu}}{\nu}, \qquad f_{t2} = c_{t3} e^{-c_{t4}\chi^2},$$

$$\tilde{S} = S + \frac{\tilde{\nu}}{k^2 d^2} f_{v2}, \qquad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}},$$

$$f_w = g\left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right]^{\frac{1}{6}}, \qquad g = r + c_{w2}(r^6 - r), \qquad r = \frac{\tilde{\nu}}{\tilde{S} k^2 d^2}.$$

Коэффициенты замыкания:

$$\mathrm{c}_{b1} = 0.1355, \qquad \mathrm{c}_{b2} = 0.622, \qquad \mathrm{c}_{v1} = 7.1, \qquad \sigma = \frac{2}{3},$$

$$c_{w1} = \frac{\mathrm{c}_{b1}}{k^2} + \frac{(1 + \mathrm{c}_{b2})}{\sigma}, \qquad c_{w2} = 0.3, \qquad c_{w3} = 2,$$

$$k = 0.41, \;\; c_{t3} = 1.2, \;\; c_{t4} = 0.5\,.$$

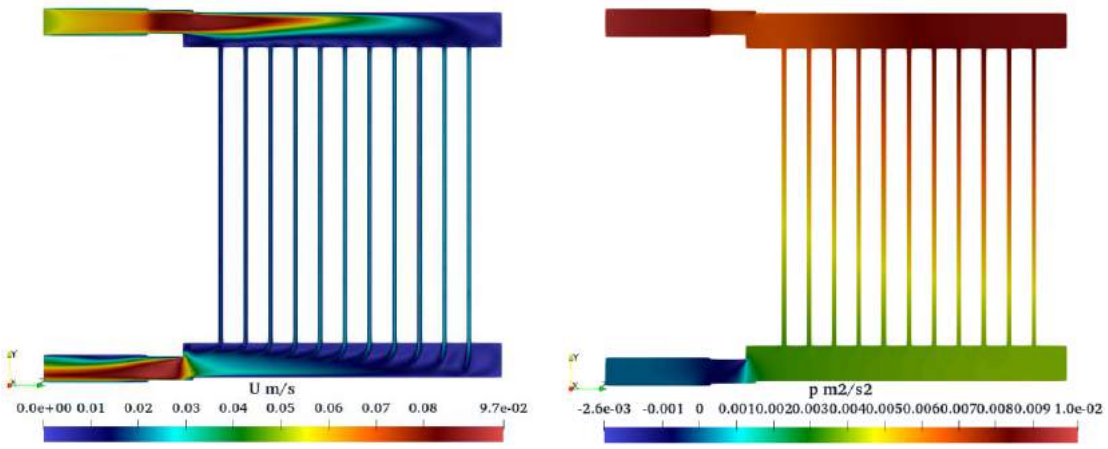
Граничные условия на стенках для турбулентных переменных рассчитываются следующим образом:

$$\tilde{\nu} = 0, \;\; \nu_t = 0.$$

Построенная таким образом на основе уравнений сохранения текучих сред система уравнений (1) – (4) решается приближенно на основе метода конечных объемов в рамках стационарной постановки.

Решение проводилось на основе численного моделирования в пакете OpenFOAM, с применением решателя simpleFoam и использованием тех же геометрических и гидродинамических характеристик что и в натурном эксперименте.

На рис. 6 представлено распределение скоростного поля и давления при скорости подачи рабочей жидкости при числе $Re = 1044$. Конструкция коллектора способствует формированию возвратного течения за уступом при входе потока в раздающий коллектор (рис. 6a). В результате, основной поток оттесняется к верхней стенке и снижается подача жидкости в первые секции коллекторной системы. Зона застойного течения формируется также в тупиковом конце конструкции. В результате, гидравлическая развертка системы характеризуется максимальной нагрузкой на 6-11 секции ТА (рис. 7).



a) *b)*

Рис.6. Распределение скоростного поля a) и перепада давления b) в ТА
Fig.6. Distribution of the velocity field a) and pressure drop b) in the HE

Неравномерность гидравлической нагрузки также видна по распределению скоростей в собирающем коллекторе. Здесь дополнительное сопротивление обусловлено сужением канала и формированием зоны возвратного течения перед уступом.

На рис. 6b показана картина распределения статического давления в системе, по которой можно качественно оценить уровень давления в микроканалах, подтверждающих характер гидравлической нагрузки, описанной выше.

По полученным распределениям давления на входе и выходе из коллекторной системы был рассчитан перепад давлений и коэффициент гидравлических потерь. Для данного числа Рейнольдса коэффициент сопротивления равен 28.5, что соответствует экспериментальному значению с отклонением в 2.5%.

Для детализации потока в пакете ParaView был применен фильтр SurfaceLIC, на основе которого хорошо отслеживаются вихревые структуры за уступом и в тупиковом конце конструкции (рис. 8a). На рис. 8b показана структура потока без подключения модели турбулентности при той же скорости подачи рабочей жидкости. Дальнейшее увеличение скорости подачи воды не позволяет получить устойчивое решение без применения модели турбулентности.

Сравнение результатов численного и экспериментального исследований показано на рис. 9. Сравнивались коэффициенты гидравлического сопротивления ТА в полученном рабочем диапазоне скоростей.

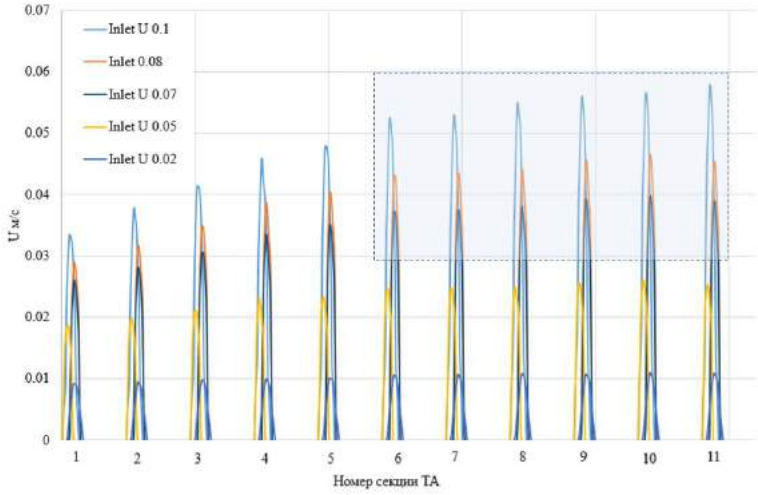
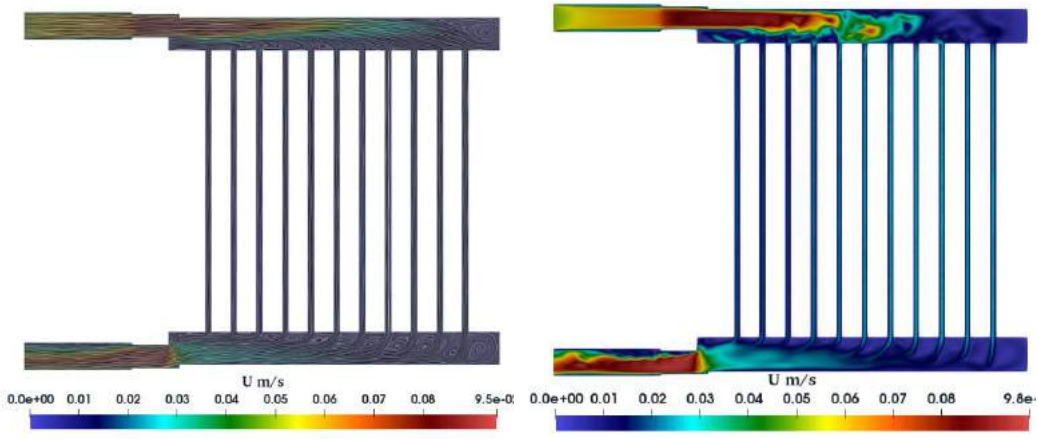


Рис.7. Гидродинамическая нагрузка секций ТА
Fig.7. Hydrodynamic load of HE sections



a) *b)*

Рис.8. a) Структура течения с моделью турбулентности (SA) 0.05м/с,
b) структура течения без модели турбулентности 0.05м/с
Fig.8. a) Flow structure with turbulence model (SA) 0.05m/s,
b) flow structure without turbulence model 0.05m/s

Наименьшее отклонение расчетного коэффициента сопротивления от экспериментального наблюдается при числе Рейнольдса $Re = 1044$. При больших числах Рейнольдса расхождение увеличивается, что обусловлено гидравлической разверткой системы и не учетом шероховатости внутренней поверхности ТА.

4. Заключение

В работе проведено численное и экспериментальное исследование гидродинамических процессов, протекающих в ТА сложной геометрии. В ходе экспериментальных исследований

была проведена серия из 200 экспериментов. Получена зависимость гидравлического сопротивления ТА в зависимости от числа Рейнольдса. Показано, что экспериментальные значения коэффициента гидравлического сопротивления снижаются с ростом числа Рейнольдса и значительно превышают коэффициенты, полученные по методике И.Е. Идельчика [9].

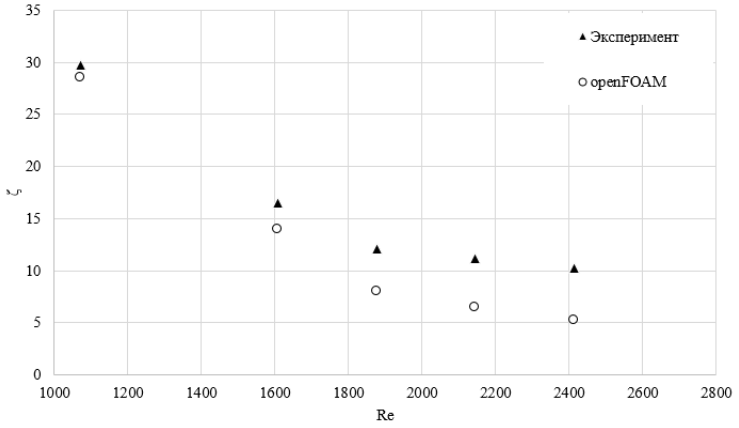


Рис.9. Сравнение результатов численного и экспериментального исследований
Fig.9. Comparison of numerical and experimental results

Путем гидродинамического подобия был подобран ряд скоростей, соответствующих рабочему диапазону ТА при эксплуатации на гидравлическом масле.
Данные полученные путем численного моделирования в пакете OpenFOAM согласуются с данными экспериментальных исследований, что говорит о применимости пакета OpenFOAM для анализа гидродинамических параметров, протекающих в ТА.

Список литературы / References

[1]. Дельнов В.Н. Гидродинамика типичных раздающих коллекторных систем. ЯЭУ: современные представления и перспективы исследования. АО «Государственный научный центр Российской Федерации – Физико-энергетический институт имени А.И. Лейпунского». Обнинск, Россия. 2020. С. 116-128. / Del'nov V.N. Hydrodynamics of typical distributing collector systems. JSC State Scientific Center of the Russian Federation, Institute of Physics and Energy named after A.I. Leypunsky, Obninsk, Russia, 2020. pp. 116-128 (in Russian).

[2]. Дубоносов, А.Ю. Гидродинамика входных цилиндрических коллекторов теплообменных аппаратов теплоэнергетических установок. Краснодар: Изд. Дом – Юг, 2013. – 124 с. / Dubonosov, A.Yu. Hydrodynamics of input cylindrical collectors of heat exchangers of thermal power plants. Krasnodar, Izd. Dom – Yug, 2013, 124 p. (in Russian).

[3]. Идельчик И.Е. Справочник по гидравлическим сопротивлениям. Москва. Машиностроение. 1992. С. 671. / Idel'chik I.E. Manual of hydraulic resistances. Moscow, Mechanical engineering, 1992, 671 p. (in Russian).

[4]. Лунина С.В., Дельнов В.Н. Тестовые расчеты гидродинамики раздающих коллекторных систем теплообменников и реакторов. ЯЭУ АО «Государственный научный центр Российской Федерации – Физико-энергетический институт имени А.И. Лейпунского». 2020. Обнинск, Россия. С. 129-137. / Lunina S.V., Del'nov V.N. Test calculations of hydrodynamics of distribution collector systems of heat exchangers and reactors. NPP JSC State Scientific Center of the Russian Federation, Institute of Physics and Energy named after A.I. Leypunsky. 2020, Obninsk, Russia, pp. 129-137 (in Russian).

[5]. Быстров П.И., Михайлов В.С. Гидродинамика коллекторных теплообменных аппаратов. М.: Энергоиздат, 1982. 224 С. / Bystrov P.I., Mihajlov V.S. Hydrodynamics of collector heat exchangers. Moscow, Energoizdat, 1982, 224 p. (in Russian).

[6]. Филиппов Г.Ф., Меламед Л.Э., Тропкина А.И. Иерархия моделей анализа коллекторных систем и макромасштабный анализ. Проблемы энергетики. 2010. № 5-6. С. 3-17. / Filippov G.F., Melamed

L.E., Tropkina A.I. Hierarchy of models for analysis of reservoir systems and macroscale analysis. Energy problems, 2010, no. 5-6, pp. 3-17 (in Russian).

[7]. Королева М.Р., Терентьев А.Н., Чернова А.А. Гидродинамика коллектора сложной формы. Вестник Рыбинской государственной авиационной технологической академии им. П.А. Соловьева. 2021. №3 (58). С.50-55. / Koroleva M.R., Terent'ev A.N., Chernova A.A. Hydrodynamics of a complex-shaped reservoir. Bulletin of the Rybinsk State Aviation Technological Academy named after. P.A. Solovyova, 2021, no. 3 (58), pp. 50-55.

[8]. Baimetova E.S., Koroleva M.R. Research of conjugate heat transfer in a collector of a complex shape of an external fins. В книге: XXI International Conference on the Methods of Aerophysical Research (ICMAR 2022). Abstracts. Novosibirsk, 2022. С. 13-14.

[9]. Байметова Е.С. Численное моделирование гидродинамических процессов в многоканальном коллекторе. Труды МАИ. 2023. № 130. / Baimetova E.S. Numerical simulation of hydrodynamic processes in a multichannel collector. Trudy MAI, 2023, no. 130 (in Russian).

[10]. Hoem M.E. Wind Turbine Simulations with OpenFOAM. In book: Progress in Turbulence VIII. Norwegian University of Science and Technology, 2019, pp.305-310.

Информация об авторах/ Information about authors

Елена Сергеевна БАЙМЕТОВА – старший преподаватель кафедры Тепловые двигатели и установки Ижевского государственного технического университета имени М.Т. Калашникова. Сфера научных интересов: вычислительная гидрогазодинамика и интенсификация процессов теплообмена.

Elena Sergeevna BAYMETOVA is a senior lecturer at the Department of Heat Engines and Plants, Izhevsk State Technical University named after M.T. Kalashnikov. Research interests: computational fluid dynamics and intensification of heat exchange processes.

Екатерина Александровна МИТРЮКОВА – аспирант кафедры Тепловые двигатели и установки Ижевского государственного технического университета имени М.Т. Калашникова. Сфера научных интересов: вычислительная гидрогазодинамика и интенсификация процессов теплообмена.

Ekaterina Aleksandrovna MITRYUKOVA is a postgraduate student at the Department of Heat Engines and Installations of the Izhevsk State Technical University named after M.T. Kalashnikov. Research interests: computational hydro-gas dynamics and intensification of heat transfer processes.

DOI: 10.15514/ISPRAS-2023-35(6)-16

Использование переноса стиля как способ улучшения обобщающей способности нейросети в задаче детекции объектов

[1,2] *Д.К. Карачёв,* ORCID: 0000-0002-1008-2535 <*denis.karachev@ocrv.ru*>
[1] *С.Е. Штехин,* ORCID: 0000-0003-2866-4864 <*sergei.shtekhin@ocrv.ru*>
[2] *В.С. Тарасян,* ORCID: 0000-0002-5288-5221 <*vtarasyan@gmail.com*>
[1] *И.Ю. Смолин,* ORCID: 0000-0002-7272-1232 <*ilya.smolin@ocrv.ru*>
[1] *М.В. Исаков,* ORCID: 0000-0002-8677-8862 <*maksim.isakov@ocrv.ru*>

[1] *Филиал №11 ООО «ОЦРВ» Сириус,*
354340, Россия, п.г.т. Сириус, ул. Международная 2/1.
[2] *Уральский государственный университет путей сообщения,*
620034, Россия, г. Екатеринбург, ул. Колмогорова, 66.

Аннотация. В данной работе предлагается реализация подхода обучения нейронной сети для детекции объектов с помощью аугментации - переноса стиля. Данный метод улучшает обобщающую способность нейросети для определения местоположения объектов на изображении за счет улучшения взаимодействия с низкоуровневыми признаками, такими как текстуры, цвета и небольшие изменения форм. Экспериментально доказана эффективность метода и продемонстрированы числовые значения метрик детекции на нескольких наборах данных с различными классами. Применение аугментации предлагается с помощью ранее не использованной архитектуры нейросети, способной переносить произвольное количество стилей. Особенностью подхода также является то, что веса нейросети для стилизации замораживаются и она добавляется в граф детекционной сети, что позволяет увеличить скорость аугментации.

Ключевые слова: нейронные сети; компьютерное зрение; перенос стиля; машинное зрение; машинное обучение; детектирование объектов.

Для цитирования: Карачев Д.К., Штехин С.Е., Тарасян В.С., Смолин И.Ю., Исаков М.В. Использование переноса стиля как способ улучшения обобщающей способности нейросети в задаче детекции объектов. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 247–264. DOI: 10.15514/ISPRAS–2023–35(6)–16.

Благодарности: Авторы выражают благодарность ОАО РЖД.

Style Transfer as a Way to Improve the Generalization Ability of a Neural Network in an Object Detection Task

[1,2] *D.K. Karachev*, ORCID: 0000-0002-1008-2535 <*denis.karachev@ocrv.ru*>
[1] *S.E. Shtekhin*, ORCID: 0000-0003-2866-4864 <*sergei.shtekhin@ocrv.ru*>
[2] *V.S. Tarasyan*, ORCID: 0000-0002-5288-5221 <*vladimir@usurt.ru*>
[1] *I.U. Smolin*, ORCID: 0000-0002-7272-1232 <*ilya.smolin@ocrv.ru*>
[1] *M.V. Isakov*, ORCID: 0000-0002-8677-8862 <*maksim.isakov@ocrv.ru*>

[1] *Industry Center for Information Systems' Development and Deployment, 2/1, Mejdunarodnaya, Sochi, 354340, Russia.*
[2] *Ural State University of Railway Transport, 66, Kolmogorova, Ekaterinburg, 620034, Russia.*

Abstract. This paper proposes the implementation of a neural network training approach for object detection, using augmentation - style transfer. This method improves the generalization ability of the neural network to determine the location of objects in the image by improving the interaction with low-level features such as textures, different colors and small changes in shapes. The effectiveness of the method is experimentally proved and the numerical values of the object detection metrics are demonstrated on several datasets with different classes. The application of augmentation is proposed using an unused before neural network architecture capable of carrying an arbitrary number of styles. The peculiarity of the approach is also that the weights of the neural network for styling are frozen and it is added to the graph of the detection network, which allows augmentation speed.

Keywords: neural networks; computer vision; style transfer; machine learning; object detection.

For citation: Karachev D.K., Shtekhin S.E., Tarasyan V.S., Smolin I.U., Isakov M.V. Style transfer as a way to improve the generalization ability of a neural network in an object detection task. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 247-264 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-16.

Acknowledgements. The authors are grateful to the Russian Railway Company.

1. Введение

С 2012 года конволюционные нейронные сети начали широко применяться в области компьютерного зрения. По качеству предсказаний они превосходят классические алгоритмы компьютерного зрения. По скорости работы и объему потребляемой памяти выигрывают у полносвязных нейронных сетей. Однако в большинстве случаев в процессе обучения нейронной сети форма объектов не запоминается, и она предпочитает более лёгкий путь для достижения цели – переобучаться на текстуры [1].

Для решения этой проблемы, в задаче классификации изображений, были предложены различные методы переноса стиля [2]. Данные методы представлены в разделе 2.

Целью данной работы является применение метода переноса стиля изображений для повышения точности в задаче детектирования объектов в реальных условиях. Предлагаемый метод является универсальным для применения в различных областях, где решается задача детекции объектов. Для детектирования объектов использовалась архитектура нейронной сети Yolo v5 [3], в основе которой лежит конволюционная сеть. Метод был проверен на задаче детектирования инструментов на наборе данных с ограниченным доступом. При обучении нейронной сети выяснилось, что модель ошибается в распознавании некоторых инструментов, переобучаясь на похожие текстуры и цвета инструментов. Также метод был проверен на задаче детектирования кистей рук в перчатках разного цвета. Обучение такой модели проводилось на открытых наборах данных (рассмотрен в разделе 3) [4-7].

Разработка и применение аугментации переноса стиля для нейросети детектирования объектов позволили исключить переобучение нейросети на объекты с похожими текстурами

и разной формой. Применение аугментации предлагается реализовать с помощью ранее неиспользуемой архитектуры нейросети, способной переносить произвольное количество стилей. Особенностью подхода также является то, что веса нейросети для стилизации замораживаются, и нейронная сеть добавляется в граф детекционной сети, что позволяет ускорить скорость аугментации. Подробное описание метода представлено в разделе 3. Тестирование метода было проведено на данных, которые рассмотрены в разделе 4.

Применение аугментации переноса стиля позволило улучшить метрики детектирования объектов в среднем на 7%. Результаты экспериментов изложены в разделе 5. Обзор существующих методов представлен в следующем разделе.

2. Обзор существующих работ

Задача переноса стиля берет свое начало в рендеринге [8] и тесно связана с синтезом и передачей текстуры [9-11]. Некоторые ранние подходы включают сопоставление гистограмм по откликам линейных фильтров [12] и непараметрическую выборку [13, 14]. Эти методы обычно полагаются на низкоуровневую статистику и не способны уловить семантические структуры. Авторы работы [15] впервые продемонстрировали впечатляющие результаты переноса стиля путем сопоставления статистики признаков в конволюционных слоях глубокой нейронной сети. Недавно было предложено несколько усовершенствований. В работе [16] авторы представили схему, основанную на случайных марковских полях в пространстве признаков для обеспечения соблюдения локальных закономерностей. Также в [17] предложили способы управления изменением цвета, пространственным расположением и масштабом передачи стиля. В работе [18] улучшили качество передачи стиля видео, наложив временные ограничения.

Подход авторов [15] основан на медленном процессе оптимизации, который итеративно обновляет изображение, чтобы минимизировать функцию потерь для стиля и контента. Процесс переноса стиля данным методом занимает продолжительное время даже на современных графических процессорах. Решение данной проблемы заключается в замене процесса оптимизации на нейронную сеть, которая обучена оптимизировать ту же самую функцию [19-21]. Эти методы передачи информации стиля примерно на три порядка быстрее, чем альтернатива, основанная на оптимизации изображения, что открывает возможности для применения в реальном времени. В работе [22] предлагается алгоритм для улучшения гранулярности. Авторы в [23] предложили способы улучшения качества и разнообразия генерируемых образцов. Однако, вышеуказанные методы передачи стиля ограничены в том смысле, что каждая сеть привязана к фиксированному стилю. Для решения этой проблемы авторы работы [24] представили одну сеть, которая способна кодировать 32 стиля и их интерполяции. В работе [25] предложили прямолинейную архитектуру, которая может синтезировать до 300 текстур и перенос 16 стилей. Тем не менее, два вышеуказанных метода не могут адаптироваться к произвольным стилям, которые не наблюдаются во время обучения.

Авторы работы [26] представили метод передачи стиля, который может передавать произвольные стили благодаря специальному слою подмены. Учитывая активации признаков изображений контента и стиля, слой изменения стилей корректирует признаки контента на наиболее близкие по значению признаки стиля в режиме скользящего окна. Тем не менее, их слой замены стиля создает новое узкое место в вычислениях: более 95% вычислений тратится на корректировку стиля для входных изображений размера 512 × 512.

Работа [15] потенциально может быть использована для обучения на изображениях. Из плюсов данного подхода можно отметить то, что он позволяет регулировать силу применения стиля за счёт изменения количества итераций. Однако это не позволяет быстро изменять стиль изображения в связи с идеей самого подхода. Он итеративно изменяет тензор и на каждой итерации пропускает картинки со стилем и контентом через нейросеть, что требует

значительного количества времени на обработку даже одного изображения. Поэтому данный вариант разумно использовать только когда имеется значительное количество памяти под набор данных, и есть возможность его сгенерировать заранее. Время, затраченное на стилизацию, будет прямо пропорционально зависеть от размера набора данных.

Стоит подробнее рассказать про работу [19], в которой автор предложил использовать идею из предыдущего исследования [15]. Автор предлагает новую функцию потерь под названием «потеря восприятия» (perseptual loss), с помощью которой решает задачи переноса стиля и повышение разрешения. Этот подход позволяет обучать отдельную нейросеть под каждый стиль. Следовательно, есть возможность на выходе получить небольшую нейросеть, которая имеет мало параметров и требует немного времени на исполнение. Недостаток подхода в том, что для того, чтобы иметь возможность применения большого количества стилей, необходимо большое количество нейросетей, что ограничивает вычислительные возможности и увеличивает время обучения сети. Также нужно тщательно выбирать стили, так как изначально нет информации о том, какой стиль приносит наибольшее влияние.

В работе [27] автор предлагает новый универсальный подход, который заключается в том, чтобы один раз обучить нейросеть, которая затем позволит перенести стиль из любого изображения. Аналогично авторам вышеупомянутых работ, предлагается использовать предобученную нейросеть типа VGG [28] для извлечения карт признаков изображений стиля и контента. Особенностью работы является блок под названием AdaIn (Adaptive Instance Normalization), благодаря которому происходит смешение стиля и контента в один набор карт признаков, который потом используется декодером, для получения итогового результата. Данный подход можно считать лучшим по сравнению с предыдущими, т.к. имеется возможность применять стилизацию в реальном времени для любого количества стилей без дополнительного дообучения нейросетей. Схема применения стилизации представлена на рис. 1.

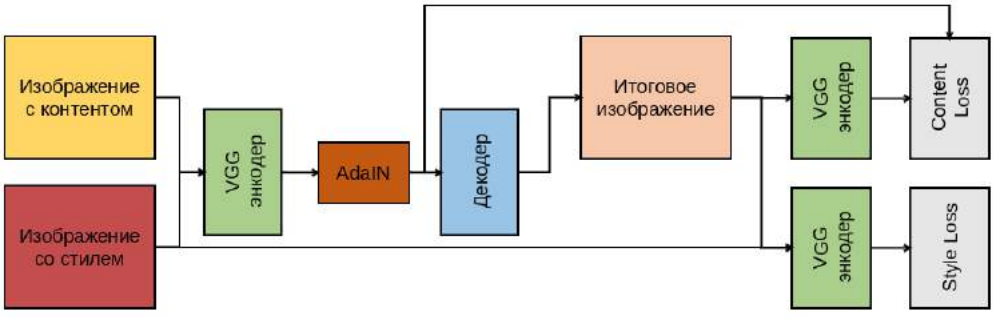


Рис. 1. Схема AdaIn подхода стилизации
Fig. 1. AdaIn's diagram of the stylization approach

Другой проблемой при передаче стиля является выбор функции потерь. Подход [15] обеспечивает соответствие стилей путем сопоставления статистики второго порядка между активациями признаков, отражаемой матрицей Грамма. Были предложены и другие эффективные функции потерь, такие «adversarial loss» [21], «histogram loss» [29], «CORAL loss» [30] и расстояние между средним и дисперсией канала [31]. Стоит отметить, что все вышеперечисленные функции потерь направлены на согласование некоторых статистических характеристик между образцом и синтезированным изображением.

Существует несколько альтернативных подходов генерации изображений, включая вариационные автокодировщики [32], авторегрессионные модели [33] и генеративно-состязательные сети (GAN) [34]. Примечательно, что GAN достигли наиболее

впечатляющего визуального качества. GAN также применялись для передачи стиля [14] и генерации изображений для кросс доменных преобразований [35-37].

В работе [2] преобразования стиля используются для аугментации при обучении нейронной сети для задачи классификации. Авторы используют несколько разновидностей стиля и производят их сравнения. В результате они получают значительное преимущество при использовании совместно с классическими аугментациями.

Работа [38] подчёркивает проблематику детекционных нейросетей с объектами, которые не находились в домене обучающей выборки. Авторы используют технику переноса стиля для обучения нейросети, которая должна детектировать людей на картинах различных художников. В работе применяется классический AdaIn [27] подход для генерации искусственного набора данных styleCOCO, из части открытого набора данных Microsoft's COCO [41], на котором происходит дообучение предобученной модели Faster-RCNN и достигается лучший результат на тестовом наборе данных People-Art [39]. Данная работа подтверждает актуальность использования подходов изменения стиля изображений для обучения нейросетей. Стоит отметить, что авторы не ставили целью улучшить детекцию на реальных объектах и ограничились только исследованиями в области художественных картин.

Вышеприведенные работы использовали стилизацию в основном для задач классификации за исключением [38], в которой создается styleCOCO набор данных для улучшения работы детектора на картинах художников. Стоит отметить, что как модель стилизации изображений для создания обучающего датасета, так и целевые доменные области работы детектора отличаются в сравнении с текущей работой. В статье [38] использовалась стилизация, чтобы реальные люди на фотографиях, в оригинальном датасете COCO, стали похожи на людей на картинах, таким образом получается, что стилизованный датасет styleCOCO - это своеобразная вариация датасета People Art. За счет применения стилизации эти датасеты становятся визуально похожи. В данной работе, стилизация применяется для обучения нейросети на объектах, которые не похожи на тот набор объектов, которые нейросеть должна определять. Обученная модель должна детектировать реальные инструменты и перчатки, которые не похожи визуально на стилизованные.

В данной работе представлены результаты исследований применительно к задаче детекции для нескольких наборах данных, собранных в реальных условиях, которые, как специализированные, так и находящиеся в открытом доступе. Подробное описание предложенного метода, представлено далее.

3. Предлагаемый метод

В предыдущем разделе работы были рассмотрены существующие методы борьбы с переобучением и повышения устойчивости работы нейросетей со своими преимуществами и недостатками. Здесь предлагается использовать перенос стиля не только как метод для борьбы с переобучением, но еще и как метод для улучшения обобщающей способности нейросети в случае, когда объекты могут отличаться, например, по цветам. В качестве наглядного примера можно привести детекцию кистей рук. Вариант, когда в обучающей выборке все изображения, на которых руки без перчаток, но тестирование происходит на картинках, на которых перчатки присутствуют. Изменение стиля позволит получить вариативность цветовой окраски объектов на изображении и обеспечит незначительные изменения в формах объектов, что искусственно расширит обучающий набор данных и позволит в дальнейшем лучше детектировать кисти рук в перчатках.

На текущем этапе работы в качестве детектора используется нейросеть типа Yolov5 [3]. Она была выбрана, та как на открытых данных имеет хорошие метрики скорости и точности, однако в наших тестовых данных имеются существенные недочеты и ошибки в предсказаниях. Основными проблемами является то, что модель путает классы между собой,

а также не детектирует один и тот же класс на изображении, при изменении цвета детектируемого объекта. Для устранения этих ошибок было предложено использовать стилизованную аугментацию.

В настоящее время существует несколько алгоритмов переноса стиля, каждый из них имеет свои плюсы и минусы. В зависимости от метода стилизации изменяется итоговый алгоритм обучения модели детекции.

Для применения стилизации был выбран подход, аналогичный работе [28], т.к. он имеет ряд преимуществ перед другими подходами. Однако была сделана небольшая модификация. Энкодер нейросети VGG дополнительно связывается с декодером с помощью дополнительных конкатенаций между слоями одного уровня по типу реализации U-Net [39] сети, но с дополнительными AdaIn блоками между ними. Упрощенная схема модифицированной сети представлена на рис. 2.

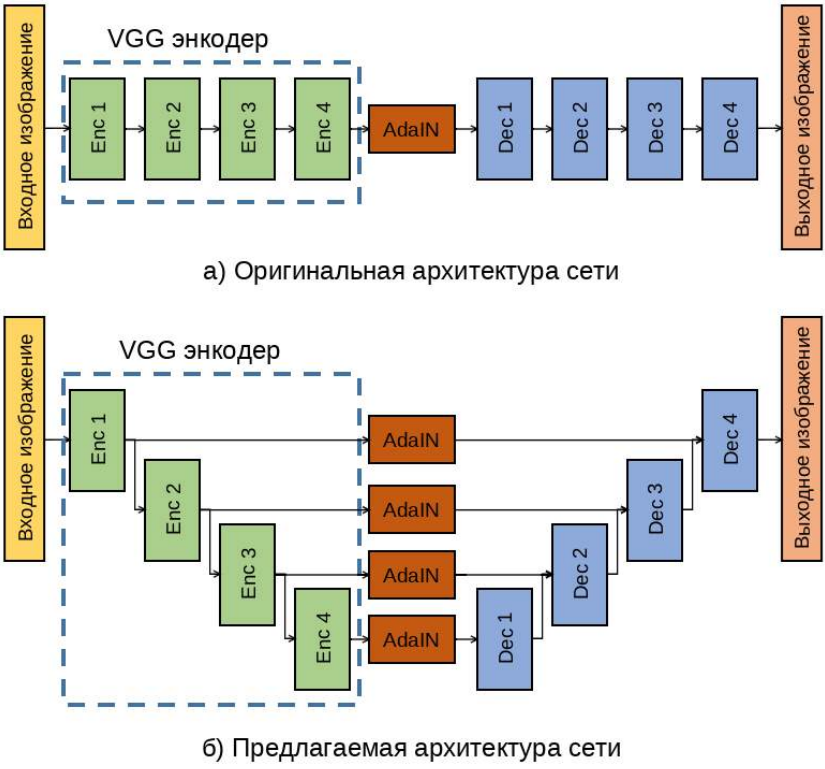


Рис. 2. Оригинальная и предлагаемая архитектура нейросети для стилизации изображений
Fig. 2. Original and modified neural network architecture for image styling

Обычно процесс аугментации изображения происходит с использованием нескольких потоков во время подготовки пакета данных, перед его подачей в нейросеть. Однако данный подход не является подходящим для предлагаемой аугментации. Стилизованные изображения для обучения нейросети можно получить несколькими способами:

- сгенерировать набор данных заранее;
- изменять изображения в процессе формирования пакета данных.

Генерация набора данных заранее является плохой практикой. Во-первых, для корректной работы аугментации, необходимо для каждой обучающей эпохи сгенерировать набор изображений, который по количеству изображений будет равен исходной тренировочной

выборке. Это значит, что память, занимаемая изображениями, на жестком диске увеличится в N раз, где N – количество обучающих эпох. Также необходимо изменять конвейер (pipeline) обучения нейросети, для того чтобы правильно подгружать необходимые картинки для каждой эпохи. Поэтому было решено, что данный подход не подходит для текущей реализации.

Классическим подходом для реализации аугментаций является изменение изображений в процессе подготовки пакета данных. Во время процесса обучения нейросети происходит сбор изображений в режиме многопоточности, что позволяет быстро собирать наборы картинок и подавать их в нейросеть. Базовые аугментации, осуществляющие преобразования поворота и масштабирования, применяются во время этого этапа предобработки изображений, требуют мало вычислительных ресурсов, быстро работают и не требуют много дополнительной памяти. Если использовать стилизацию аналогичным способом, то необходимо создавать экземпляр нейросети для переноса стиля в каждом потоке, что займет дополнительную память. Также произойдет замедление работы за счет того, что операции по сборке пакета данных происходят с использованием центрального процессора. Использование графических ускорителей для аугментации при сборке пакета данных изображений нецелесообразно. На графических картах происходит обучение детекционной нейросети и требуется вся видеопамять для нейросети и пакета данных. Чем больше потоков для сбора пакета данных используется, тем больше видеопамяти необходимо для аугментации.

Исходя из вышесказанного, можно сделать вывод, что использовать нейронную сеть для стилизации в процессе сборки пакета данных нецелесообразно. В данной работе предлагается объединить детекционную нейросеть и нейросеть для стилизации в один граф, заморозив при этом веса нейросети для стилизации. Стилизация будет применяться с определенной вероятностью для всего пакета данных во время обучения нейросети для детекции. Таким образом, решается проблема распараллеливания на нескольких графических устройствах, т.к. количество экземпляров нейросетей для стилизации равно количеству нейросетей для детекции, и они считаются за одни экземпляр нейросети. Вследствие этого происходит экономия видеопамяти. Отсутствует привязка к количеству потоков при сборе пакета данных. Также скорость работы аугментации значительно увеличивается за счёт графических ускорителей. Схема реализованного подхода аугментации данных представлена на рис. 3. Наборы данных, на которых тестировался реализованный подход, описаны в следующем разделе.

4. Наборы данных

Основной набор данных для задачи детектирования объектов представляет собой набор из 200 аннотированных видеофайлов с изображениями 36 специализированными инструментами. Средняя продолжительность видео от 8 до 10 минут, частота кадров – 25, разрешение видео 1280x720, 1080x1920.

В данной работе, для проверки гипотезы об улучшении качества обучения детекционной сети с помощью аугментации из основного набора данных были выбраны три контрольные подвыборки (табл. 1), содержащие изображения с:

- двумя инструментами (визуально схожими);
- шестью инструментами (неуверенно распознаваемыми);
- шестью инструментами, из которых три неуверенно и три хорошо распознаваемые.

Первая подвыборка содержит два инструмента и является базовой для подбора параметров аугментации. Вторая и третья подвыборка содержат по шесть инструментов и являются контрольными для проверки качества ошибочно детектируемых инструментов.

Рис. 3. Схема реализованного подхода применения аугментации
Fig. 3. Scheme of the implemented augmentation approach

Табл. 1. Названия и соответствующие им индексы инструментов
Table 1. Names and corresponding tool indices

Набор инструментов	Индекс	Название рус.	Название англ.
1	*0*	ключ торцевой	socket wrench
	1	метла	broom
2	*0*	ключ торцевой	socket wrench
	1	метла	broom
	2	молоток костыльный	spike maul
	3	кувалда	sledge hammer
	4	лопата	shovel
	5	трамбовка ручная	wooden rammer
3	*0*	ключ торцевой	socket wrench
	1	метла	broom
	2	домкрат гидравлический	hydraulic jack
	3	шаблон путевой	track template
	4	станок рельсошлифовальный	rail grinder small
	5	кувалда	sledge hammer

Обусловлено это тем, что из-за большого количества данных обучение на всех инструментах занимает большое количество времени, что делает невозможным подбор всех параметров для обучения за разумное время. Выбор инструментов происходил таким образом, чтобы

проверить улучшение детекции ошибочно классифицируемых инструментов (например, лопата и торцевой ключ), при этом не ухудшив качество хорошо детектируемых инструментов.

Баланс классов и информация об ограничивающих прямоугольниках представлены на рис. 4 и рис. 5 соответственно.

На каждый класс приходится от 8 до 11 тыс. объектов, преимущественно они сосредоточены в центре изображения, но есть и в других произвольных местах. Размер ограничивающих прямоугольников в основном составляет 10-40% от размеров всего изображения, но также есть экземпляры, занимающие большую часть изображения.

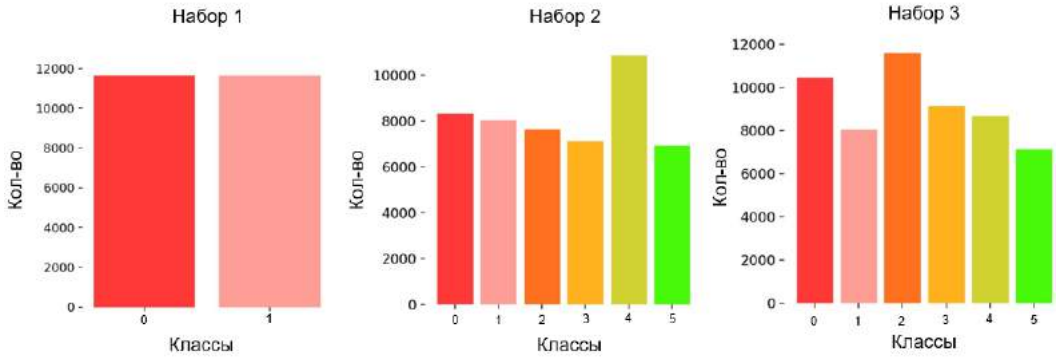


Рис. 4. Баланс классов для трёх наборов данных
Fig. 4. Class balance for three datasets

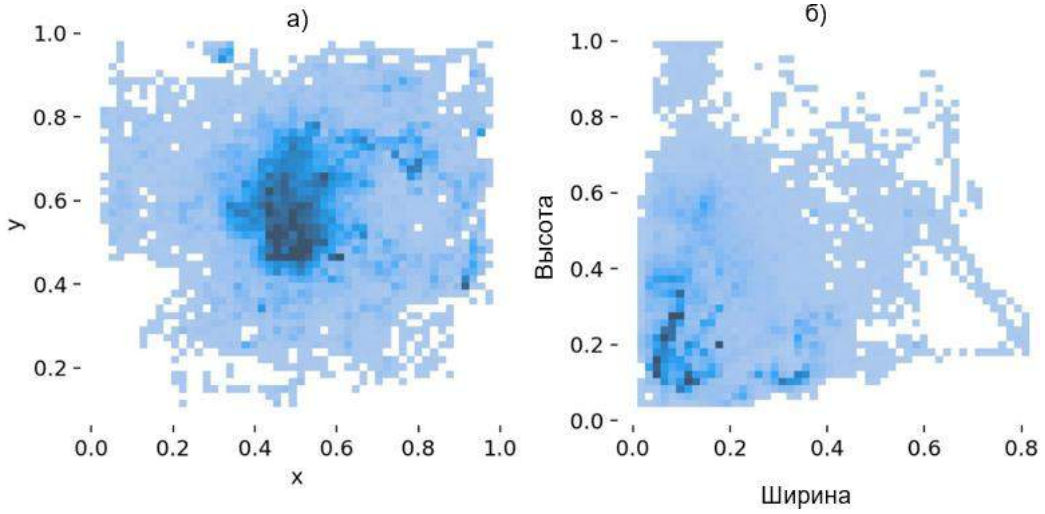


Рис. 5. Координаты центра ограничивающих прямоугольников (а) и их размеры относительно размеров изображения (б)
Fig. 5. The coordinates of the center of the bounding boxes (a) and their dimensions relative to the dimensions of the image (b)

Дополнительно, обучение моделей для детектирования кистей рук человека проводилось на наборах данных, представленных в свободном доступе:

- Human-Parts [4], представляет собой набор данных с тремя классами: человек, лицо человека, кисть руки человека.

- HaGRID (HAnd Gesture Recognition Image Dataset) [5], набор данных для распознавания жестов, снятый на веб-камеру и камеру телефона.
- COCO-hands [6], 26 тыс. изображений, содержащих кисть руки человека из набора данных Microsoft's COCO [41].
- TV-hands, набор изображений извлеченных из набора данных ActionThread [42], который содержит около 4500 видеороликов. Из каждого ролика было выбрано 2-3 кадра.
- YouTube-hands [7], нарезка кадров видео с платформы YouTube. Представлены различные действия: приготовление еды, танцы, спортзал, игра на музыкальных инструментах, работа с инструментами и т.д. Для объективной оценки качества детекции, видео, на которых руки были в перчатках исключены.

Обучение нейросети для переноса стиля происходило на наборе данных MS-COCO [41], который использовался как изображения контента, изображения для стилизации были взяты из набора данных WikiArt [43]. Параметры экспериментов, которые были проведены на описанных наборах данных и полученные результаты экспериментов, подробно представлены в следующем разделе.

5. Эксперименты и сравнительный анализ полученных результатов

Обучение производилось на одной видеокарте Nvidia A100 40 GB. Следующие гиперпараметры были выбраны для обучения: размер пакета данных (batch_size) 32, количество потоков (num_workers) 16, коэффициент скорости обучения (learning_rate) 0.0001, коэффициент стиля (style_weight) 5.0, коэффициент контента (content_weight) 1.0. В качестве оценки точности была выбрана метрика mean Average Precision (mAP) [44] с использованием порогового значения Intersection over Union 0.5 (mAP 0.5) и среднее значение mAP 0.5..0.95 по разным порогам Intersection over Union, от 0.5 до 0.95 [45].

Результат применения нейросети для стилизации изображений представлен на рис. 6.

Обучение детекционной сети на инструментах происходит на наборе данных с ограниченным доступом, подробная информация о котором представлена выше. Всего для обучения доступны 36 инструментов. Прежде чем проверять гипотезу с использованием аугментации, модель детекции была обучена на всех инструментах.

Детекционная нейросеть обучалась со следующими параметрами:

1. epoch – 96;
2. batch_size – 12;
3. img_size – 1024x1024;
4. final learning rate – 0.001;
5. optimizer – SGD;
6. средние аугментации семейства Yolo.

Из табл. 2, где представлены метрика качества mAP для сравнения подходов, видно, что использование стилизации как аугментации улучшает метрику детекции от 0.004 до 0.05 mAP, что свидетельствует о значительном положительном эффекте использования стилизации.

Рис. 6. Пример оригинальных и стилизованных изображений из нашего набора данных с измененным инструментом
Fig. 6. Example of original and stylized images from our dataset with a modified tool

Табл. 2. Результаты обучения детекционной нейросети на инструментах с использованием аугментации для изменения стиля и без нее
Table 2. Training results of a detection neural network on instruments with/without style augmentation

	Обычные аугментации		**Обычные аугментации + стилизация**	
	mAP 0.5	***mAP 0.5..0.95***	***mAP 0.5***	***mAP 0.5..0.95***
1 набор (2 инструмента)	0.878	0.486	**0.925**	**0.547**
2 набор (6 инструментов)	0.838	0.568	**0.876**	**0.588**
3 набор (6 инструментов)	0.851	0.576	**0.855**	**0.577**

Также, после проведения экспериментов было выявлено, что стилизация позволяет избежать ошибок предсказаний и значительно повышает качество детекции. При этом качество

предсказания на других инструментах не ухудшилось. На рис. 7а), 7в) представлены примеры работы модели детектирования инструментов, обученной с применением алгоритма стилизации, на рис. 7б), 7г) – результаты работы модели детектирования инструментов, обученной без использования алгоритма стилизации. Как видно из рисунков, модель, обученная с применением аугментации, детектирует инструменты, которые ранее не обнаруживались.

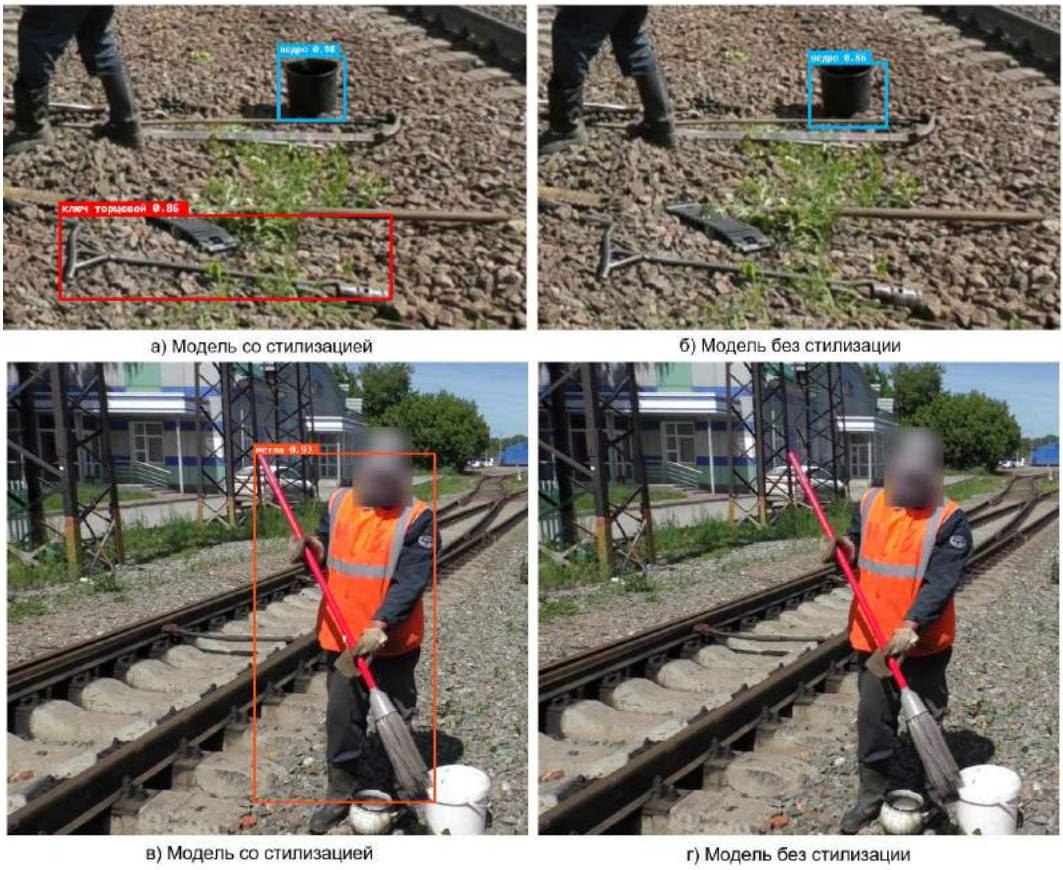


Рис. 7. Пример детекции инструментов
Fig. 7. Tool detection example

Аугментация с изменением стиля также была протестирована в задаче детекции кистей рук, так как было замечено, что модель плохо отрабатывает в случаях, когда человек в перчатках. Причиной такой работы нейросети является то, что открытые данные, на которых она обучалась, не содержат изображения с кистями рук в перчатках. Изображения в перчатках изменяют не только цвет рук, но и объем. Использование стилизации в данном случае является подходящим решением, так как она изменяет цвет текстуры, а также производит небольшую деформацию, в зависимости от применяемого стиля. Пример изменения стиля представлен на рис. 8.

Детекционная модель обучалась на нескольких наборах данных [4-7], которые находится в открытом доступе. Параметры обучения сети аналогичны эксперименту с инструментами, описанному выше.

Результаты обучения нейросети на кисти рук приведены в табл. 3.

По табл. 3 видны значительные улучшения метрики качества детекции от 0.01 до 0.19 mAP, что подтверждает эффективность использования аугментации по изменению стиля. На рис. 9а), 9в) представлены примеры работы модели детектирования кистей рук, обученной с применением алгоритма стилизации, на рис. 9б), 9г) – результаты работы модели детектирования кистей рук, обученной без использования алгоритма стилизации. Как видно из рисунков, модель, обученная с применением аугментации, детектирует кисти рук, которые ранее не обнаруживались.

Визуальный пример сравнительного предсказания моделей представлен на рис. 9.

Табл. 3. Результаты обучения детекционной нейросети на кисти рук с использованием аугментации для изменения стиля и без нее
Table 3. Training results of a detection neural network on hands with/without style augmentation

Набор данных	**Обычные аугментации**		**Обычные аугментации + стилизация**	
	mAP 0.5	***mAP 0.5..0.95***	***mAP 0.5***	***mAP 0.5..0.95***
Human-Parts [4]	0.81	0.50	**0.82**	0.50
HaGRID [5]	0.53	0.27	**0.54**	**0.28**
COCO-hand [6]	0.77	0.37	**0.83**	**0.38**
TV-hand [40]	0.57	0.24	**0.67**	**0.26**
YouTube-hand [7]	0.70	0.42	**0.89**	**0.57**
Все наборы данных	0.84	0.49	**0.9**	**0.51**

6. Заключение

В данной работе предложен подход по улучшению обобщающей способности нейросети для задачи детекции объектов (Object Detection) с помощью метода переноса стиля изображений. Данный метод улучшает обобщающую способность нейросети для определения местоположения объектов на изображении путём уменьшения переобучения нейросети на низкоуровневые признаки, такие как текстуры, цвет и небольшие отклонения форм. Предложена ранее не использованная модификация архитектуры нейросети для стилизации изображений, которая является существенным вкладом авторов в данную проблематику. Разработана программная реализация предлагаемого метода для использования в качестве аугментации, способной переносить произвольное количество стилей на изображения.

Экспериментально подтверждено улучшение обобщающей способности нейросети на нескольких наборах данных из области детекции частей тела (Human-Parts, HaGRID, COCO-hand и др.) Полученные результаты демонстрируют универсальность предлагаемого метода и устойчивость работы в задачах различного прикладного назначения. На наборе данных с ограниченным доступом для детектирования железнодорожных инструментов было достигнуто улучшение метрики mAP@0.5 на 4.5%. В среднем было достигнуто улучшение метрики mAP@0.5 на 7% как для наборов данных с ограниченным доступом с инструментами, так и для открытых наборов данных для распознавания кистей рук.

Рис. 8. Пример изображения обычной руки и стилизованной
Fig. 8. Example with a normal hand and a stylized hand

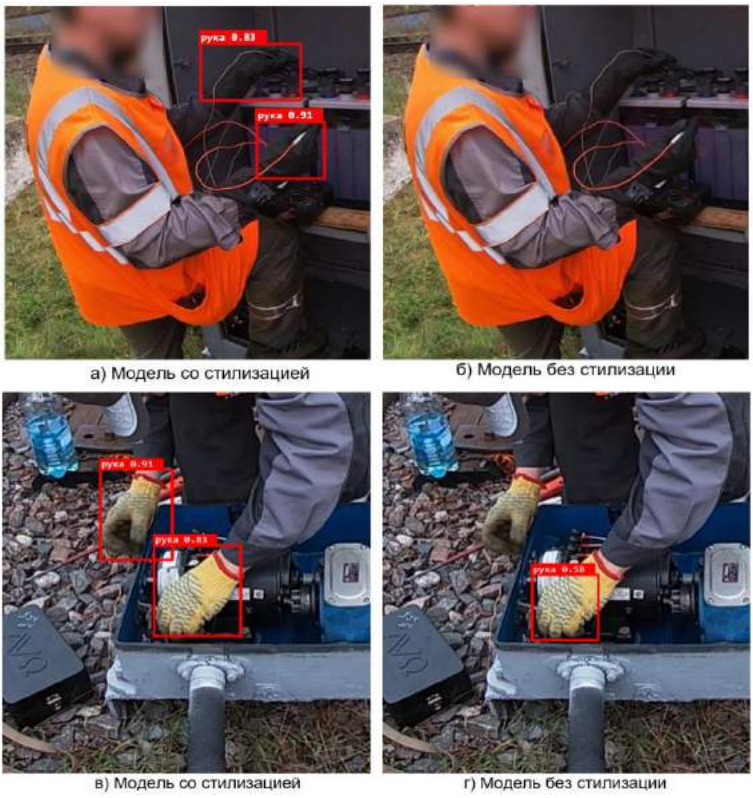


Рис. 9. Пример детекции кистей рук
Fig. 9. Hand detection example

Список литературы / References

[1]. Гримов Р. Нейронные сети предпочитают текстуры и как с этим бороться. Хабр. https://habr.com/ru/company/ods/blog/453788/.

[2]. Xu Zheng, Tejo Chalasani, Koustav Ghosal, Sebastian Lutz, and Aljosa Smolic. STaDA: Style Transfer as Data Augmentation. In Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. Vol. 4 VISAPP: VISAPP, 107-114, 2019, Prague, Czech Republic, https://www.scitepress.org/Link.aspx?doi=10.5220/0007353401070114.

[3]. Jocher, G., Stoken, A., Borovec, J., NanoCode012, ChristopherSTAN, Changyu, L., … Rai, P. ultralytics/yolov5: v3.1 – Bug Fixes and Performance Improvements (v3.1) [Computer software]. Zenodo, 2020. https://doi.org/10.5281/zenodo.4154370.

[4]. Xiaojie Li., Lu Yang, Qing Song, Fuqiang Zhou. Detector-in-Detector: Multi-Level Analysis for Human-Parts, 2019. https://arxiv.org/abs/1902.07017.

[5]. Kapitanov A., Makhlyarchuk A., Kvanchiani K. HaGRID. Hand Gesture Recognition Image Dataset. SberDevices, Russia, 2022, arXiv:2206.08219v1.

[6]. Narasimhaswamy S., Wei Z., Wang Y., Zhang J., Hoai M., Contextual Attention for Hand Detection in the Wild. IEEE/CVF International Conference on Computer Vision (ICCV), 2019. doi: 10.1109/ICCV.2019.00966.

[7]. Huang, M., Narasimhaswamy, S., Vazir, S., Ling, H., Hoai, M. Forward Propagation, Backward Regression and Pose Association for Hand Tracking in the Wild. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 6396-6406, doi: 10.1109/CVPR52688.2022.00630.

[8]. Kyprianidis, J. E., Collomosse, J., Wang, T., & Isenberg, T. State of the 'Art': A Taxonomy of Artistic Stylization Techniques for Images and Video. IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 5, pp. 866-885, May 2013. doi: 10.1109/TVCG.2012.160.

[9]. Efros A. A. and Leung T. K. Texture synthesis by non-parametric sampling. Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 1033-1038 vol.2, doi: 10.1109/ICCV.1999.790383.

[10]. Lara Raad, and Bruno Galerne, Efros and Freeman Image Quilting Algorithm for Texture Synthesis, Image Processing On Line, 7 (2017), pp. 1–22. doi:10.5201/ipol.2017.171.

[11]. Elad M. and Milanfar P. Style-transfer via texture-synthesis. arXiv. doi:10.48550/arXiv.1609.03057.

[12]. Heeger D. J., Bergen J. R. Pyramid-Based Texture Analysis/Synthesis. Proceedings of the International Conference on Image Processing (SIGGRAPH), 1995, vol. 3, pp, 648-651, https://api.semanticscholar.org/CorpusID:47266338.

[13]. Bousmalis K., Silberman N., Dohan D., Erhan D., Krishnan D. Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. arXiv:1612.05424v1, 2016.

[14]. Collobert R., Kavukcuoglu K., Farabet C. Torch7: A Matlab-like Environment for Machine Learning. In: BigLearn, NIPS Workshop, 2011.

[15]. Gatys L. A., Ecker A. S., Bethge M. Image style transfer using convolutional neural networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.

[16]. Li C., Wand M. Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 2479-2486, https://doi.org/10.1109/CVPR.2016.272.

[17]. Gatys L. A., Ecker A. S., Bethge M., Hertzmann A., Shechtman E. Controlling Perceptual Factors in Neural Style Transfer. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 3730-3738, doi: 10.1109/CVPR.2017.397.

[18]. Ruder M., Dosovitskiy A., Brox T. Artistic Style Transfer for Videos. In: Rosenhahn B., Andres B., (eds). Pattern Recognition. GCPR 2016. Lecture Notes in Computer Science(), vol 9796. Springer, Cham. https://doi.org/10.1007/978-3-319-45886-1_3.

[19]. Johnson J., Alahi A., Fei-Fei L. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9906. Springer, Cham. https://doi.org/10.1007/978-3-319-46475-6_43.

[20]. Ulyanov D., Lebedev V., Vedaldi A., Lempitsky V. Texture networks: Feed-forward Synthesis of Textures and Stylized Images. Proceedings of the 33-rd International Conference on Machine Learning (ICML), New York, NY, USA, 2016. JMLR: W&CP vol. 48, pp. 2027-2041, 2016.

[21]. Li C., Wand M. Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9907. Springer, Cham. https://doi.org/10.1007/978-3-319-46487-9_43.

[22]. Wang X., Oxholm G., Zhang D., Wang Y.-F. Multimodal Transfer: A Hierarchical Deep Convolutional Neural Network for Fast Artistic Style Transfer. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 7178-7186, doi: 10.1109/CVPR.2017.759.

[23]. Ulyanov D., Vedaldi A., Lempitsky V. Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 4105-4113, doi: 10.1109/CVPR.2017.437.

[24]. Dumoulin V., Shlens J., Kudlur M. A Learned Representation for Artistic Style., In Proceedings of the International Conference on Learning Representations (ICLR), 2017, https://openreview.net/forum?id=BJO-BuT1g, arXiv preprint arXiv:1610.07629, 2016.

[25]. Li Y., Fang C., Yang J., Wang Z., Lu X., Yang M.-H. Diversified Texture Synthesis with Feed-Forward Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 266-274, doi: 10.1109/CVPR.2017.36.

[26]. Chen T. Q., Schmidt M. Fast Patch-based Style Transfer of Arbitrary Style. arXiv preprint arXiv:1612.04337, 2016.

[27]. Huang, X., Belongie, S. Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization, 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 1510-1519, doi: 10.1109/ICCV.2017.167.

[28]. Simonyan, K., Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2014. arXiv. doi: 10.48550/arXiv.1409.1556

[29]. Wilmot P., Risser E., Barnes C. Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses. arXiv. preprint arXiv:1701.08893, 2017.

[30]. Peng X., Saenko K. Synthetic to Real Adaptation with Deep Generative Correlation Alignment Networks. 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 2018, pp. 1982-1991, doi: 10.1109/WACV.2018.00219.

[31]. Li Y., Wang N., Liu J., Hou X. Demystifying Neural Style Transfer. arXiv. 2017. doi: 10.48550/arXiv.1701.01036.

[32]. Kingma D. P., Welling M. Auto-encoding variational bayes. In ICLR, 2014, doi: 10.48550/arXiv.1312.6114.

[33]. Oord A. v. d., Kalchbrenner N., Kavukcuoglu K. Pixel recurrent neural networks. In ICML, 2016, doi: 10.48550/arXiv.1601.06759.

[34]. Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative adversarial nets. NIPS 2014, 2672; http://papers.nips.cc/paper/5423-generative-adversarial-nets.

[35]. Taigman Y., Polyak A., Wolf L. Unsupervised cross- domain image generation. In ICLR, 2017, doi: 10.48550/arXiv.1611.02200.

[36]. Liu M.-Y., Tuzel O. Coupled generative adversarial networks. In NIPS, 2016, https://proceedings.neurips.cc/paper/2016/file/502e4a16930e414107ee22b6198c578f-Paper.pdf.

[37]. Kim T., Cha M., Kim H., Lee J., Kim J. Learning to discover cross-domain relations with generative adversarial networks. arXiv, 2017, doi: 10.48550/arXiv.1703.05192.

[38]. Kadish D., Risi S., Løvlie A. S. Improving object detection in art images using only style transfer //2021 International Joint Conference on Neural Networks (IJCNN). – IEEE, 2021. – С. 1-8.

[39]. Cai H. et al. The cross-depiction problem: Computer vision algorithms for recognising objects in artwork and in photographs //arXiv preprint arXiv:1505.00110. – 2015.

[40]. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv 2015. doi: 10.48550/arXiv.1505.04597.

[41]. Lin T.-Y., Maire M., Belongie S., Bourdev L., Girshick R., Hays J., Perona P., Ramanan D., Zitnick C. L., Dollár P. (2014). Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B.,

Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol. 8693. Springer, Cham. doi: 10.1007/978-3-319-10602-1_48.

[42]. Wang Y., Hoai, M. Improving Human Action Recognition by Non-action Classification. arXiv 2016. doi: 10.48550/arXiv.1604.06397.

[43]. Tan W. R., Chan C. S., Aguirre H., Tanaka K. Improved ArtGAN for Conditional Synthesis of Natural Image and Artwork. IEEE Transactions on Image Processing, 28(1), pp. 394–409, 2019. doi: 10.1109/TIP.2018.2866698.

[44]. Beitzel S.M., Jensen E.C., Frieder O. MAP. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA, 2009. doi: 10.1007/978-0-387-39940-9_492.

[45]. Rezatofighi H., Tsoi N., Gwak J., Sadeghian A., Reid I., Savarese S. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. arXiv. 2019. doi: 10.48550/arXiv.1902.09630.

Информация об авторах / Information about authors

Денис Константинович КАРАЧЕВ – ведущий специалист по анализу данных. Сфера научных интересов: компьютерное зрение.

Denis Konstantinovich KARACHEV – Senior Data Scientist. Research interests: computer vision.

Сергей Евгеньевич ШТЕХИН – Руководитель группы компьютерного зрения лаборатории ИИ и нейронных сетей. Сфера научных интересов: компьютерное зрение.

Sergey Evgenievich SHTEKHIN – Team Lead Data Scientist. Research interests: computer vision.

Владимир Сергеевич ТАРАСЯН – кандидат физ.-мат. наук, доцент Уральского государственного университета путей сообщения, заведующий кафедрой “Мехатроника”. Сфера научных интересов: компьютерное зрение, интеллектуальный анализ данных, интеллектуальные системы управления.

Vladimir Sergeevich TARASYAN – Cand. Sci. (Phys.-Math.), Associate Professor at Ural state University of railway transport, Head of Mechatronic Department. Research interests: computer vision, intelligence data analysis, intelligence control systems.

Илья Юрьевич СМОЛИН – старший специалист по анализу данных. Сфера научных интересов: компьютерное зрение.

Ilya Urevich SMOLIN – Middle Data Scientist. Research interests: computer vision.

Максим Владимирович ИСАКОВ – специалист по анализу данных. Сфера научных интересов: компьютерное зрение.

Maksim Vladimirovich ISAKOV – Junior Data Scientist. Research interests: computer vision.

DOI: 10.15514/ISPRAS-2023-35(6)-17

G.F. Miller's Dictionary "Description of Pagan Peoples Living in Kazan Province, such as Cheremis, Chuvash and Votyaks..." as a Valuable Source for Studying the Tatar Language of the 18th Century (based on LingvoDoc material)

F.Sh. Nurieva, ORCID: 0000-0001-*9957*-9734 <*fanuzanurieva@yandex.ru*>
G.R. Galiullina, ORCID: 0000-0001-6923-2190 <*caliullina@list.ru*>
A.F. Yusupov, ORCID: 0000-0003-0363-5303 <*faikovich@mail.ru*>

Kazan Federal University,
18, Kremlevskaya str., Kazan, 420008, Russia.

Abstract. This article presents the results of an analysis of three dictionaries by G.F. Miller, which were created in the 18th century and are hosted on the LingvoDoc platform (lingvodoc.ispras.ru). Miller's manuscript dictionaries from the collection of the Russian State Archive of Ancient Documents (RSAAD) are introduced into scholarly discourse for the first time. The analysis reveals: 1) in the 18th century, all consonant transitions in the Tatar language had already been completed; 2) at this time, some vowels (known as the Volga region vowels) had not yet completed their transition process, and for [*o] and [*u], this process had not even begun; 3) in the 18th century, dialectal zones were already identified in the Tatar language; 4) the 1791 period Cyrillic-script dictionary differs in innovations from modern Tatar language and Latin-script dictionaries; 5) Latin-script dictionaries are closer to the modern Tatar literary language.

Keywords: linguistic platform; LingvoDoc; G. F. Miller; phonetics; archive; dictionary; dialect; Tatar language

For citation: Nurieva F.Sh., Galiullina G.R., Yusupov A.F. G.F. Miller's Dictionary "Description of Pagan Peoples Living in Kazan Province, such as Cheremis, Chuvash and Votyaks..." as a Valuable Source for Studying the Tatar Language of the 18th Century (based on LingvoDoc material). Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 6, 2023. pp. 265-282. DOI: 10.15514/ISPRAS-2023-35(6)-17

Словарь Г.Ф. Миллера «Описание живущих в Казанской губернии языческих народов, яко то черемис, чуваш и вотяков...» как источник для изучения татарского языка XVIII века (на материале ЛингвоДок)

Ф.Ш. Нуриева, ORCID: 0000-0001-*9957*-9734 <*fanuzanurieva@yandex.ru*>
Г.Р. Галиуллина, ORCID: 0000-0001-6923-2190 <*caliullina@list.ru*>
А.Ф. Юсупов, ORCID: 0000-0003-0363-5303 <*faikovich@mail.ru*>

Казанский (Приволжский) федеральный университет
420008, Россия, г. Казань, ул. Кремлевская, д.18.

Аннотация. В статье представлены результаты анализа трех словарей Г.Ф. Миллера, созданных в XVIII веке и размещенных на платформе ЛингвоДок (lingvodoc.ispras.ru). Впервые в научный оборот вводятся рукописные словари Г.Ф. Миллера из фонда РГАДА. В процессе анализа выявлено: 1) все процессы

перехода в области согласных в татарском языке в XVIII веке уже завершились; 2) перелом поволжских гласных к этому периоду еще для всех гласных не завершился, а для [*o] и [*u] данный процесс даже не начался; 3) в XVIII веке в татарском языке были уже выделены диалектные зоны; 4) словарь 1791 года на кириллической графике отличается инновациями от современного татарского языка и словарей на латинской графике; 5) словари на латинской графике более близки к современному татарскому литературному языку.

Ключевые слова: лингвистическая платформа; ЛингвоДок; Г.Ф. Миллер; фонетика; архив; словарь; диалект; татарский язык.

Для цитирования: Нуриева Ф.Ш., Галиуллина Г.Р., Юсупов А.Ф. Словарь Г.Ф. Миллера «Описание живущих в Казанской губернии языческих народов, яко то черемис, чуваш и вотяков...» как источник для изучения татарского языка XVIII века (на материале ЛингвоДок). Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 265-282 (на английском языке).. DOI: 10.15514/ISPRAS-2023-35(6)-17.

1. Introduction

During the mid-18th century, scientists began to document and gather information on the dialects of the Tatar language. Concurrently, historians such as G.F. Miller, I. Georgi, F.-I. Stralenberg, P. Pallas and others were attempting to identify cultural and material elements and collect lexical material in distinct regions where Tatars resided. The trilingual dictionary of Miller, "Description of the pagan peoples living in Kazan province, such as Cheremis, Chuvash and Votyaks: showing their residence, political institutions..." (St. Petersburg, 1791) is of great scientific importance in this aspect. However, it has not yet received adequate attention in Tatar linguistics.

Gerhard Friedrich (known as Fyodor Ivanovich in Russia) Miller (1705-1783) was born in Westphalian town of Herford, Germany. After graduating from school, he studied humanities at Rinteln University. He focused on history, philosophy as well as classical languages at university.

Miller travels to Leipzig, a cultural center in Germany. It was there that he became interested in the idea of traveling to a distant and intriguing country. Many former students, such as the theology student A.I. Osterman, who had dropped out of his studies in Jena, had already made successful careers there [1: 190].

In 1725, Miller arrived in St. Petersburg and became forever linked with Russia. He accepted Russian citizenship in 1748. Miller left memories of the beginning of his service in Russia, stating that "I especially taught instructions in Latin, history, and geography to the upper class of the gymnasium..." [2: 147].

Miller's work began during the period when Peter the Great's ideas for organizing academic expeditions to comprehensively study the territories and languages of the Russian state, particularly its eastern regions, were being implemented. The academic team was tasked with studying the nature and natural resources of the interior regions of Siberia as well as its history and the ethnography of indigenous peoples. Miller, a professor at the St. Petersburg Academy of Sciences, was a member and leader of the academic team during the academicians' journey through Siberia (1733-1743) as part of the Great Northern Expedition.

The scientific organization of the expedition should be noted. It involved students, freelance translators, copyists, and guides who were responsible for searching and processing research material.

On October 18, 1733, Miller arrived in the Kazan province on his way to an expedition to Siberia. He was tasked by the St. Petersburg Academy of Sciences to study the Volga peoples and ancient monuments. Miller collected information about the cultural and daily life of the peoples in the Volga region and the Urals. The author's stay in the province resulted in notes that served as the basis for more serious academic works. One such work is the "Description of three pagan peoples in Kazan province, namely Cheremis, Chuvash and Votyak" [3]. The work was first published in "Monthly

Works...” in 1756, 23 years after the author’s visit to the Ural-Volga region and its inhabitants (see Fig. 1).

In the introduction, the author describes his acquaintance with the inhabitants of the Volga region: “In Kazan Province, in addition to Russian peasants who settled there, six other peoples have lived there since ancient times and mostly still reside in the same places. These peoples are referred to by the following titles in Russia: 1) the Tatars reside in both the city of Kazan and the surrounding villages, with a significant population in the Kungur district (now the Perm region), which also includes the Bashkirs living in the Ufa province; 2) the Cheremis, and 3) Chuvash people live on both banks of the Volga River, both above and below the city of Kazan; 4) the Votyaks own most of the land between the Volga and Kama rivers; 5) the Mordva people reside in Nizhny Novgorod province, with some also living in Penza province, which is part of Kazan province; 6) the Permiaks and Permichis lived between Sol-Kamsk and Cherdyn, and are considered one people along with the Zyrians of Ustyug, Sol-Vychegotsky and Yarensky” [4].

Structurally, the study consists of eight chapters: “About their Dwellings and Civil Routines”, “About their Physical and Mental Qualities”, “About their Clothing”, “About their Food, Industry, and Trades”, “About their Languages, Arts, and Sciences”, “About their Natural Law and Concept of God and Divine Affairs”, “About their Pagan Law and Rituals”, “About their Secular Customs” The test appears to be a comparative and ethnographic description executed in the spirit of European literature about exotic corners of the ecumene which was widely spread during the Age of Enlightenment [4].

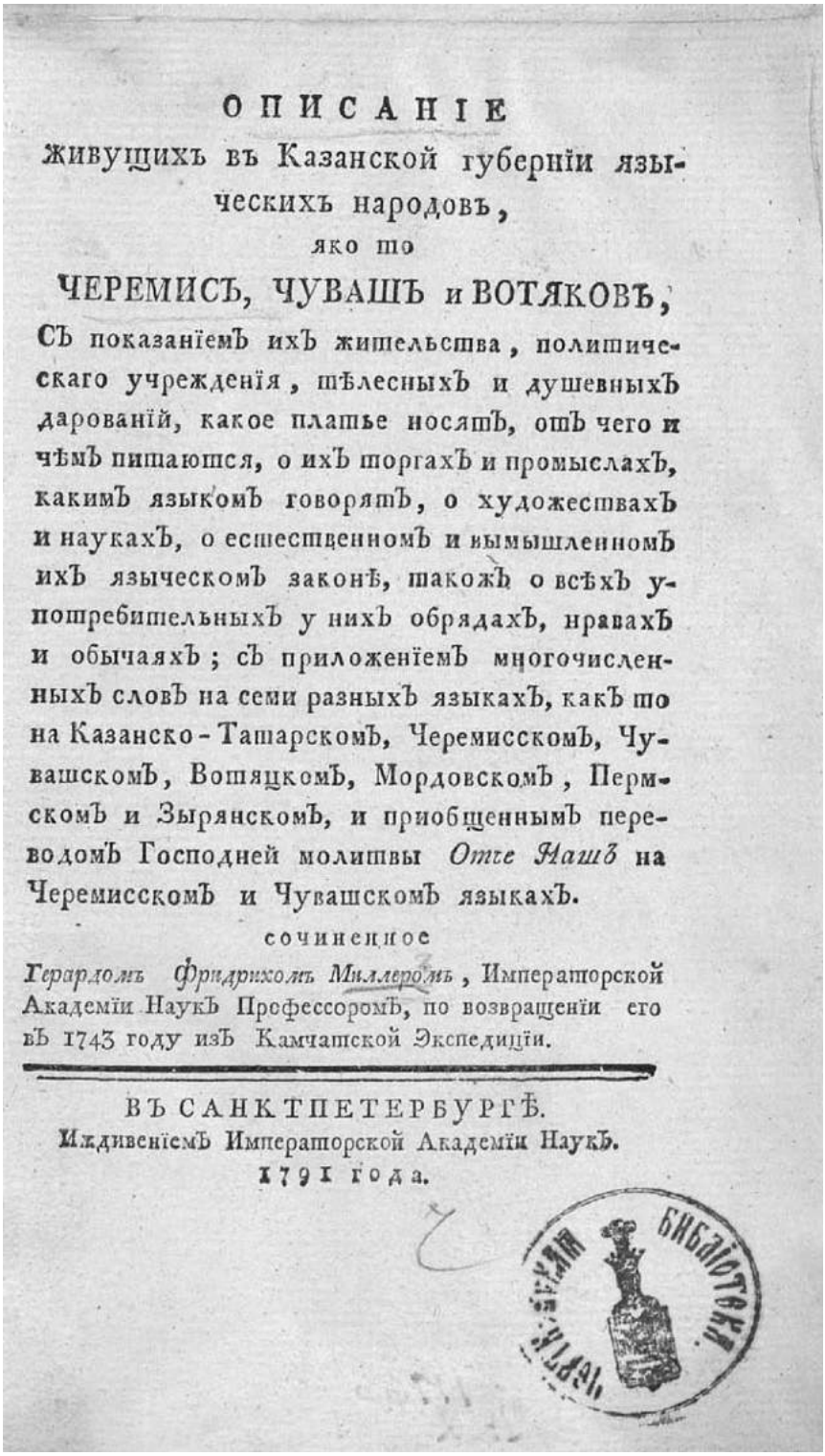
For our study we used materials from Fund No. 1125 of the Russian State Archives of Ancient Documents (RSAAD), specifically Miller’s Portfolio 199 containing files 513 g1 and 513 g2 (refer to Fig. 2, 3). Additionally, we consulted the Cyrillic dictionary presented as a separate appendix in the 1791 edition [4: 81-99] (see Fig. 4, 5).

Miller’s dictionaries 513 g1 and 513 g2 are written in Latin script. Although the dictionaries share the same lexical material, they differ in their principles of dictionary construction.

Miller’s dictionary of 1791 was published in Cyrillic and differs from the other two dictionaries in its graphics, phonetic features, and structure. It presents the material of seven languages spoken by the Volga peoples with translations into Russian. The dictionary section of the work is presented in pages 81-99 of the 1791 edition. “Dictionary in the Tatar, Cheremis, Chuvash, Voshltsky, Mordovian, Perm and Zyryansk languages, with Russian translation” [4: 81] contains over 2650 word-forms in eight languages spoken by the people of the Volga region. The Tatar section alone includes more than 325 word-forms, presented on pages 82, 84, 86, 88, 90, 90, 92, 92, 94, 96, 98 of the dictionary.

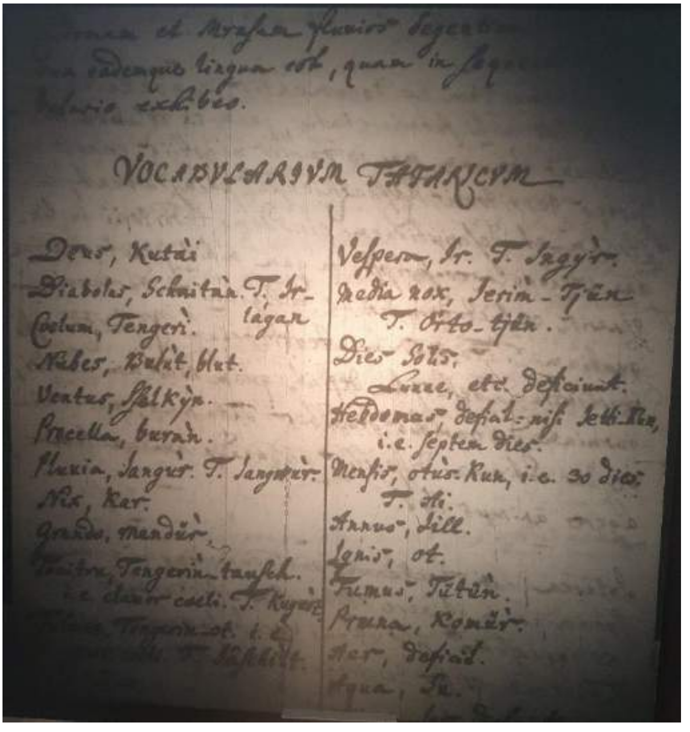
The three Miller’s Dictionaries were added to the Corpus of Written Monuments of the Tatar Language, available at https://lingvodoc.ispras.ru/. This source was used to analyze the material. So far, Miller’s Dictionaries have not been researched to study the Tatar language. It is worth noting, that the LingvoDoc platform contains manuscript dictionaries that have not been previously published and explored [5].

The **study’s significance** lies in Miller’s dictionary material, which serves as a crucial source of information on the history and historical dialectology of the Tatar language. The material is also valuable because Miller documented the expedition’s locations of stay: “both in the city of Kazan and the surrounding villages, there should be a significant population in the Kungur district (now the Perm region), which also includes the Bashkirs living in the Ufa province” [4]. By comparing historical materials with modern data, we can identify both archaic and innovative phenomena in the Tatar language.



ОПИСАНІЕ
живущихъ въ Казанской губерніи языческихъ народовъ,
яко то
ЧЕРЕМИСЪ, ЧУВАШЪ и ВОТЯКОВЪ,
Съ показаніемъ ихъ жительства, политическаго учрежденія, щѣлесныхъ и душевныхъ дарованій, какое платье носятъ, отъ чего и чѣмъ питаются, о ихъ торгахъ и промыслахъ, какимъ языкомъ говорятъ, о художествахъ и наукахъ, о естественномъ и вымышленномъ ихъ языческомъ законѣ, такожъ о всѣхъ употребительныхъ у нихъ обрядахъ, нравахъ и обычаяхъ; съ приложеніемъ многочисленныхъ словъ на семи разныхъ языкахъ, какъ то на Казанско-Татарскомъ, Черемисскомъ, Чувашскомъ, Вотяцкомъ, Мордовскомъ, Пермскомъ и Зырянскомъ, и приобщеннымъ переводомъ Господней молитвы *Отче Нашъ* на Черемисскомъ и Чувашскомъ языкахъ.
сочиненное
Герардомъ Фридрихомъ Миллеромъ, Императорской Академіи Наукъ Профессоромъ, по возвращеніи его въ 1743 году изъ Камчатской Экспедиціи.
ВЪ САНКТПЕТЕРБУРГѢ.
Иждивеніемъ Императорской Академіи Наукъ.
1791 года.

Fig. 1. Miller's Dictionary, 1791



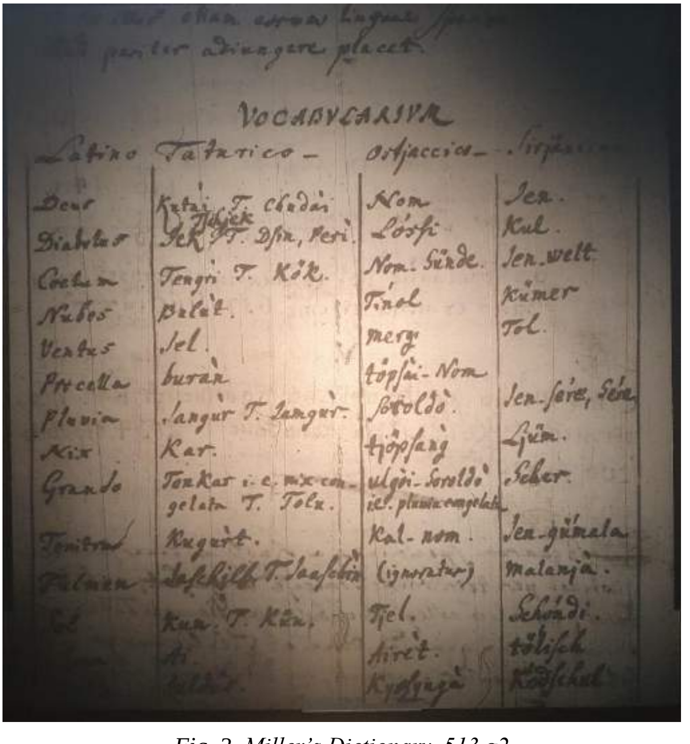
...nam et Wrasa fluvios ...
... eademque lingua est, quam in seq...
tabula exhibeo.

VOCABULARIUM TATARICUM

Deus, Kutai
Diabolus, Schaitan. T. Ir-lagan
Coelum, Tengeri.
Nubes, Bulut, blut.
Ventus, Jelkyn.
Procella, buran.
Pluvia, Jangur. T. Jangmur.
Nix, Kar.
Grando, mandur.
Tonitru, Tengerin tausch. i.e. clamor coeli. T. Kugurt
Fulmen, Tengerin ot. i.e. ... coeli. T. Jaschilt

Vesper, Ir. T. Jngyr.
Media nox, Jerim-Tjün T. Orto-tjün.
Dies, Son.
Lunae, etc. deficiunt.
Hebdomas, deficit, nisi Jetti Kun, i.e. septem dies.
Mensis, otus Kun, i.e. 30 dies. T. Ai.
Annus, Jill.
Ignis, ot.
Fumus, Tütün.
Pruna, Komür.
Aer, deficit.
Aqua, Su.

Fig. 2. Miller's Dictionary, 513 g1

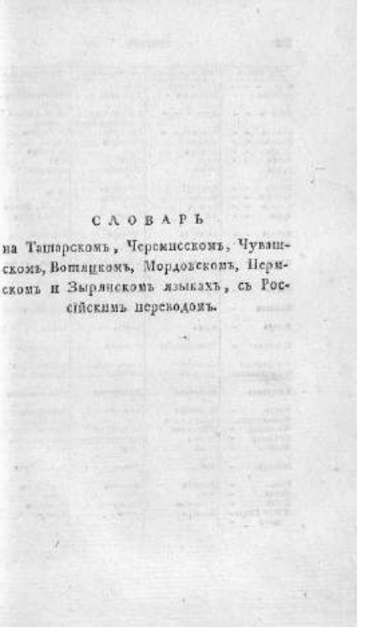


... etiam earum linguae ...
... pariter adiungere placet.

VOCABULARIUM

Latino	Tatarico –	Ostjaccico –	Sirjanico
Deus	Kutai T. Chudai	Nom	Jen
Diabolus	Jek T. Djin, Peri	Lörsi	Kul
Coelum	Tengri T. Kök	Nom-Sünde	Jen-welt
Nubes	Bulut	Tinol	Kümer
Ventus	Jel	merg	Tol
Procella	buran	töpsai-Nom	
Pluvia	Jangur T. Jangur	sorolde	Jen-sere, Sera
Nix	Kar	tjöpsang	Lüm
Grando	Tonkar i.e. nix congelata T. Tolu	ulgri-soroldo i.e. pluvia congelata	Seber
Tonitru	Kugurt	Kal-nom	Jen-gümala
Fulmen	Jaschilb T. Jaschin	(ignoratur)	malanja
Sol	Kun T. Kün	Tjel	Schondi
	Ai	Airet	tölisch
	Jultus	Kyoljnga	Kodschul

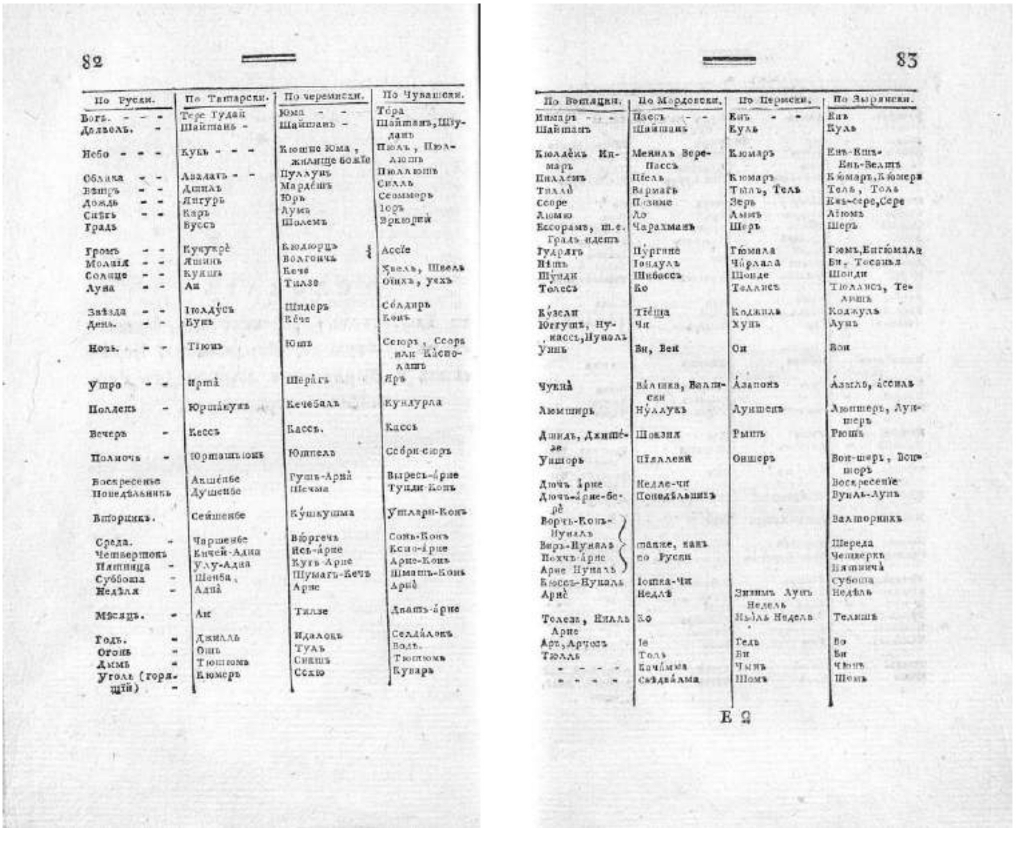
Fig. 3. Miller's Dictionary, 513 g2



СЛОВАРЬ

на Татарскомъ, Черемисскомъ, Чувашскомъ, Вотяцкомъ, Мордовскомъ, Пермскомъ и Зырянскомъ языкахъ, съ Россійскимъ переводомъ.

Fig. 4. Miller's Dictionary, 1791



82

По Русски.	По Татарски.	По черемисски.	По Чувашски.
Богъ.	Тере Тудай	Юма	Тора
Дьяволъ.	Шайтанъ	Шайтанъ	Шайтанъ, Шудамъ
Небо	Кукъ	Кюшне Юма, жилище божіе	Пюлъ, Пюлюмъ
Облака	Авалатъ	Пуллунъ	Пюллюмъ
Вѣтръ	Дшилъ	Мардешъ	Силлъ
Дождь	Янгуръ	Юръ	Сеомморъ
Снѣгъ	Каръ	Луме	Юръ
Градъ	Буссъ	Шолемъ	Эркюрий
Громъ	Кукукрѣ	Кюдюрцъ	Асеіе
Молнія	Яшинъ	Волгончъ	
Солнце	Кукшъ	Кече	Хвелъ, Швелъ
Луна	Ай	Тылзе	Ойхъ, уехъ
Звѣзда	Іюлдусъ	Шидеръ	Солдиръ
День.	Кунъ	Кече	Конъ
Ночь.	Тюнъ	Юшъ	Сеоръ, Сеоръ или Кясполашъ
Утро	Иртѣ	Шерягъ	Яръ
Полдень	Юршкунъ	Кечебалъ	Кундурла
Вечеръ	Кессъ	Кассъ.	Кассъ
Полночь	Юрташтюнъ	Юшпелъ	Сеоръ-сюръ
Воскресенье	Акшенбе	Гушъ-Арня	Вырес-арне
Понедѣльникъ	Дюшенбе	Шечма	Тундик-Конъ
Вторникъ.	Сейшенбе	Кушкушма	Утлари-Конъ
Среда.	Чаршенбе	Вюргечъ	Сонъ-Конъ
Четвертокъ	Кичей-Адна	Ись-арне	Ксио-арне
Пятница	Улу-Адна	Кугъ-Арне	Арне-Конъ
Суббота	Шенба,	Шумагъ-Кечъ	Шмашъ-Конъ
Недѣля	Аднѣ	Арне	Арнѣ
Мѣсяцъ.	Ай	Тилзе	Дналъ-арне
Годъ.	Джиллъ	Идалокъ	Селдилокъ
Огонь	Отъ	Туль	Водъ.
Дымъ	Тютюмъ	Сикшъ	Тютюмъ
Уголь (горящій)	Кюмеръ	Сескю	Куваръ

83

По Вотяцки.	По Мордовски.	По Пермски.	По Зырянски.
Инмаръ	Пасъ	Енъ	Енъ
Шайтанъ	Шайтанъ	Куль	Куль
Кюлдейн Инмаръ	Менилъ Вере-Пассъ	Кюмаръ	Енъ-Кышъ-Енъ-Велшъ
Пиллемъ	Пелль	Кюмаръ	Кюмаръ, Кюмеръ
Тилль	Вармагь	Тылъ, Тель	Толь, Толь
Сооре	Пизмне	Зеръ	Ежъ-сере, Сере
Люмъ	Ло	Лымъ	Аюмъ
Ессорамъ, т.е. Градъ идетъ	Чарахманъ	Шеръ	Шеръ
Гудрагъ	Пургине	Гюмала	Гюмъ, Енгюмала
Бiтъ	Іондулъ	Чарлала	Би, Тосвыл
Шунди	Шибассъ	Шонде	Шонди
Толесъ	Ко	Толлисъ	Тюллисъ, Телишъ
Кузели	Тѣшта	Коджилъ	Коджулъ
Юкгунъ, Нукассъ, Нунолъ	Чи	Хунъ	Лунъ
Уинъ	Ви, Вей	Ой	Вой
Чукна	Валгшка, Валисей	Азапонъ	Азыль, ассилъ
Лымширъ	Нуллукъ	Лунщенъ	Лунщеръ, Луншеръ
Дшинъ, Джинде	Шовзия	Рынъ	Рюшъ
Уишоръ	Піллевий	Оншеръ	Вой-шеръ, Войшеръ
Дючъ ірне	Недле-чи		Воскресеніе
Дючъ-арне-беръ	Понедѣльникъ		Вуналь-Лунъ
Ворчъ-Конъ-Нуналъ			Вторникъ
Верь-Нуналъ	также, какъ по Русски		Шереда
Покчъ-арне			Чепшеркъ
Арне Нуналъ			Пятничѣ
Кюсенъ-Нуналъ	Іошка-Чи		Субота
Арнѣ	Недлѣ	Зизимъ Лунъ Недель	Недѣль
Толезь, Нилль Арне	Ко	Нылъ Недель	Телишъ
Артъ, Арчосъ	Іе	Гедъ	Во
Тюлль	Толь	Ви	Би
- - - -	Качамма	Чынъ	Чынъ
- - - -	Сайдвалма	Шомъ	Шомъ

Е 2

Fig. 5. Miller's Dictionary, 1791, pages 82 and 83

The purpose of this article is to present the results of an analysis of Tatar dictionary material, identifying the dialectal features recorded by Miller in the 18th century in various Tatar territories. The material consists of three of Miller's dictionaries, which have not been the subject of special research until now and are now available on the LingvoDoc platform.

To achieve our goal, we completed the following tasks:

- prepared a comprehensive description of three Miller's dictionaries;
- transcribed them into modern Tatar language and created a glossary;
- translated the dictionary entries into Russian;
- added three editions of the Dictionary (513 g1, 513 g2, 1791) to the LingvoDoc platform's database;
- searched for and created etymological links between words in the three dictionaries, including modern Tatar literary language and dialects;
- conducted phonetic analysis using LingvoDoc tools to identify commonalities and differences in the dictionaries using a data processing and analysis program;
- analyzed cognates of the dictionaries using the LingvoDoc program to establish the nature of these similarities and differences.

The study examines the grapho-phonetic features of the Tatar material found in Miller's dictionaries.

2. Discussion

Using the the LingvoDoc platform's program for phonetic data processing and analysis, we obtained a comprehensive description of Miller's Dictionaries phonetic system. The program used Latin transcription. The following results were obtained from the processed dictionary material:

1) phonological systems of vowel and consonant sounds (refer to Fig. 6, 7, 8);
2) phoneme combination tables (refer to Fig. 9, 10, 11).

Результаты анализа (444 проанализированные текстовые объекты):

ФОНЕТИЧЕСКИЙ АНАЛИЗ

Словарь татарского языка Г.Ф. Миллера 1756 – CSV (Excel) данные

Согласные звуки:

	m		n					
p	b	t	d		c	k	g	q
				ʧ				
		s	z	ʃ				
						x		h
	w				j			
			r					
			l					

Гласные звуки:

i	y	ɨ	u
e	ø		o
a	ä		

Fig. 6. Miller's Dictionary, 513 g1

Результаты анализа (353 проанализированные текстовые объекты):

ФОНЕТИЧЕСКИЙ АНАЛИЗ

Словарь татарского языка Г.Ф. Миллера 1759 – CSV (Excel) данные

Согласные звуки:

	m		n					
p	b	t	d	c	k	g	q	
			ʧ					
		s	z	ʃ				
	f				x		h	
				j				
			r					
			l					

Гласные звуки:

i	y	ɨ	u
e	ø		o
a	ä		

Fig. 7. Miller's Dictionary, 513 g2

Результаты анализа (306 проанализированные текстовые объекты):

ФОНЕТИЧЕСКИЙ АНАЛИЗ

Словарь татарского языка Г.Ф. Миллера 1791 – CSV (Excel) данные

Согласные звуки:

	m	mʲ			n	nʲ								
p	b	bʲʲ		t	tʲʲ		d	dʲ			k	kʲ	g	gʲ
							ʧ							
			s	sʲ	z	zʲ	ʃ	ʒ						
		f	v						x					
									j	jʲ				
					r	rʲ								
					l	lʲ								

Гласные звуки:

i	ɨ	u
e		o
a		

Fig. 8. Miller's Dictionary, 1791

The inventory of phonemes and allophones reveals a general pattern in the transmission of consonant sounds. Differences are only noticeable in the graphic design of certain consonants such as *ng* [*ng* – н], *tsch* [*ch* – ч], and *sch* [*sh* – ш] in the Latin versions of the dictionary.

The vowels present a complex picture, with [ö], [ü], and [ɨ] missing in the Cyrillic variant. However, all vowels of the Tatar language are included in the Latin-script dictionaries.

After analyzing the materials of three dictionaries using the LingvoDoc tools, a number of features were identified.

2.1 General features of the Dictionaries:

2.1.1 Archaic features in the field of vowels, including the preservation of the archaic, proto-Turkic reflex [*o] and [*u] (Tables 1 and 2).

Miller's dictionaries record the regular use of the graphemes o and u in accordance with the literary graphemes u and o. This is consistent with the data presented in L.T. Makhmutova's monograph, which indicates that the Kypchak process of vowel movement *u > o, *o > u, *ü > ö, *ö > ü has not been completed in Mishar dialects [6: 34].

2.1.2 In terms of consonants, Miller's Dictionaries are generally the same (Tables 3-5).

Reliable rows:

[b]—[b]—[b]—[b]
[t]—[t]—[t]—[t]
[k]—[k]—[k]—[k]
[m]—[m]—[m]—[m]
[ʃ]—[ʃ]—[ʃ]—[ʃ]
[l]—[l]—[l]—[l]

[s]—[s]—[s]—[s]
[j]—[j]—[j]—[j]
[j]—[j]—[j]—[dʒ]
[n]—[n]—[n]—[n]
[r]—[r]—[r]—[r]

Распределение гласных:

	i	y	ɨ	u	e	ø	o	a	ä
#____:	+	+		+	+		+	+	+
+ГУБ_:	+	+	+	+	+	+	+	+	+
+ЗУБ_:	+	+	+	+	+	+	+	+	+
+ПАЛ_:	+	+		+	+		+	+	+
+ЗЯЗ_:	+		+	+	+	+	+	+	+
+ЛАР_:	+								
_+ГУБ:	+	+	+	+	+		+	+	+
_+ЗУБ:	+	+	+	+	+	+	+	+	+
_+ПАЛ:	+		+	+				+	+
_+ЗЯЗ:	+	+	+	+	+	+	+	+	+
_+ЛАР:								+	+

Распределение согласных:

	m	n	p	b	t	d	c	k	g	q	ʧ	s	z	ʃ	x	h	w	j	r	l
#____:	+		+	+	+	+		+	+	+	+	+		+				+		
+ПЕР_:	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+
+ЦНТ_:	+	+	+		+			+			+	+	+	+				+	+	+
+ЗАД_:	+	+	+	+	+	+		+	+		+	+	+	+	+			+	+	+
+ОГУ_:	+	+	+	+	+	+		+	+		+	+	+	+	+			+	+	+
_+ПЕР:	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+
_+ЦНТ:	+			+	+	+		+	+			+	+	+					+	+
_+ЗАД:	+		+	+	+	+		+	+	+	+	+	+	+			+	+	+	+
_+ОГУ:	+		+	+	+	+		+	+	+	+	+	+	+			+	+	+	+

Fig. 9. Miller's Dictionary, 513 g1

2.2 Differences in G.F. Miller's Dictionaries

2.2.1 Correspondences by initial vowel.

The lexemes in the 513 g2 dictionary retain the archaic Old Turkic [*e]. In contrast, the 513 g1 and 1791 dictionaries exhibit is a Volga region vowel change, corresponding to the vowel [i] found in modern Tatar language (Table 6).

2.2.2 The 513 g2 and 1791 dictionaries retain the Old Turkic reflex [u], while the 513 g1 dictionary coincides with Modern Tatar on this feature (Table 7).

2.2.3 All three dictionaries coincide in terms of initial vowels of the second syllable. The analysis based on the LingvoDoc tool produces the following reliable rows:

[e]–[e]–[e]–[e]	[ɨ]–[u]–[u]–[u]	[a]–[a]–[a]–[a]
[e]–[i]–[i]–[e]	[ɨ]–[ɨ]–[u]–[a]	[e]–[u]–[u]–[e]

2.2.4 Only dictionary 513 g1 records a feature on [ɨ] ~ [u] that is not identified in the other two dictionaries (Table 8).

The analysis shows that the program combines different linguistic phenomena into one table. This includes: a) a graphic reflection of archaic labial vowel harmony, such as *олтурдым* (olturdym) meaning 'I sat down', *одун* (odun) meaning 'firewood' and b) a dialectal feature in the transmission of the lexeme *сару* (saru) meaning 'yellow' and *йылу* (yylu) meaning 'warm'. This innovation is noted in the works of Tatar dialectologists with the observation that "the correspondence of y ~ u is found only in the Vyatsky dialects (in Bisermen, the subdialect of Kryashen). The Zakazan dialect exhibits a dialectal peculiarity in the limited words *сару – сары, кыру көн – кory көн, тулумады – тулмады* (saru – sary, kyru kөn – kory kөn, tulumady – tulmady [7: 146].

Распределение гласных:

	i	y	ɨ	u	e	ø	o	a	ä
#____:	+	+	+	+	+		+	+	+
+ГУБ_:	+	+	+	+	+		+	+	+
+ЗУБ_:	+	+	+	+	+	+	+	+	+
+ПАЛ_:	+	+		+	+	+	+	+	+
+ЗЯЗ_:	+	+	+	+	+	+	+	+	+
+ЛАР_:	+			+					
_+ГУБ:	+	+	+	+	+		+	+	+
_+ЗУБ:	+	+	+	+	+	+	+	+	+
_+ПАЛ:	+			+				+	+
_+ЗЯЗ:	+	+	+	+	+	+	+	+	+
_+ЛАР:					+			+	+

Распределение согласных:

	m	n	p	b	t	d	c	k	g	q	ʧ	s	z	ʃ	f	x	h	j	r	l
#____:	+		+	+	+	+	+	+			+	+	+	+	+	+		+		
+ПЕР_:	+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+
+ЦНТ_:	+	+			+			+	+			+	+	+		+			+	+
+ЗАД_:	+	+	+	+	+	+		+	+		+	+	+	+		+		+	+	+
+ОГУ_:	+	+	+	+	+	+		+	+		+	+	+	+		+		+	+	+
_+ПЕР:	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
_+ЦНТ:	+			+	+	+		+	+			+	+	+		+			+	+
_+ЗАД:	+			+	+	+		+	+		+	+	+			+	+	+	+	+
_+ОГУ:	+			+	+	+	+	+	+		+	+	+			+	+	+	+	+

Fig. 10. Miller's Dictionary, 513 g2

Распределение гласных:

	i	ɨ	u	e	o	a
#____:	+		+		+	+
+ГУБ_:	+	+	+	+	+	+
+ЗУБ_:	+	+	+	+	+	+
+ПАЛ_:	+		+	+		+
+ЗЯЗ_:	+		+	+	+	+
+ЛАР_:						
_+ГУБ:	+	+	+	+	+	+
_+ЗУБ:	+	+	+	+	+	+
_+ПАЛ:	+		+	+		+
_+ЗЯЗ:	+	+	+	+	+	+
_+ЛАР:						

Распределение согласных:

	m	mʲ	n	nʲ	p	b	bʲʲ	t	tʲʲ	d	dʲ	k	kʲ	g	gʲ	ʯ	s	sʲ	z	zʲ	ʃ	ʒ	f	v	x	j	jʲ	r	rʲ	l	lʲ
#____:	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+			+		+		+	+					
+ПЕР_:	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+			+	+	+	+	+	+	+	+
+ЦНТ_:						+	+					+	+								+										
+ЗАД_:	+	+	+	+				+	+	+	+	+	+	+		+	+	+	+	+		+	+		+	+	+	+	+	+	+
+ОГУ_:	+	+	+	+				+	+	+	+	+	+	+		+	+	+	+	+		+	+		+	+	+	+	+	+	+
_+ПЕР:	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
_+ЦНТ:	+	+																								+	+	+	+		
_+ЗАД:	+	+	+	+		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		
_+ОГУ:	+	+	+	+		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		

Fig. 11. Miller's Dictionary, 1791

Table 1. The correspondence [o] ~ [u]

[u]		[u]	
унбер	'одиннадцать'	унбер	'одиннадцать'
унике	'двенадцать'	унике	'двенадцать'
утыз	'тридцать'	утыз	'тридцать'
		утын | утын	'дрова | дрова'
утын	'бревно'		
утырдым	'сел'	утырдым	'сел'
		утыз көн	'тридцать дней'
		утыргыч	'стул'
		утырыр	'сядет'
[o]		[o]	
on-bir	'одиннадцать'	on-bir	'одиннадцать'
on-iky | on-ike	'двенадцать | двенадцать'	on-eki	'двенадцать'
otos | otus	'тридцать | тридцать'	otus	'тридцать'
		odun | otun	'дрова | дрова'
otun	'бревно'		
[utyrdym]	'сел'	olturadym	'сел'
		otus kun	'тридцать дней'
		oturgusch	'стул'
		olturar	'сядет'
[u]		[o]	
унбер	'одиннадцать'	Онъ Беръ	'одиннадцать'
унике	'двенадцать'	Онъ Ика	'двенадцать'
утыз	'тридцать'	Отъ-усъ	'тридцать'
утын	'дрова'	Отанъ	'дрова'

Table 2. The correspondence [u] ~ [o]

[o]		[u]	
ком	'песок'	kumak | kum	'песок | песок'
болан	'лось'	bulan	'лось'
бодай	'пшеница'	buktai	'пшеница'
тоз	'соль'	tust	'соль'
болак	'источник'	bulak	'источник'
коры	'сухой'	kulak | kuru	'ухо | сухой'
колын | колынчак	'жеребёнок'	kulun	'жеребёнок'
солы	'овёс'	sulu	'овёс'
[u]		[u]	
kumak | kum	'песок | песок'	Кумъ	'песок'
bulan	'лось'	Буланъ	'лось'
budai	'пшеница'	Будай	'пшеница'
tuz	'соль'	Тусъ	'соль'
bulak	'источник'		
kuru | kulak	'сухой | ухо'		
uluntschak | kulu	'жеребёнок'		
sula	'овёс'		

Table 3. Reliable range [b]—[b]—[b]—[b]

[b]		[b]	
баш | башмак	'голова | башмаки'	besch | bashmak | basch	'пять | башмаки |
бозау	'телёнок'	busau | busuu	'телёнок|телёнок'
болан	'лось'	bulan	'лось'
балык	'рыба'	balyk	'рыба'
бодай	'пшеница'	buktai	'пшеница'
бүген	'сегодня'	bugun	'сегодня'
[b]		[b]	
basch | besch	'голова|пять'	Бешъ | Башъ | Башмакъ	'пять | голова |
bozou	'телёнок'	Бузау	'телёнок'
bulan	'лось'	Буланъ	'лось'
balÿk	'рыба'	Балыкъ	'рыба'
budai	'пшеница'	Будай	'пшеница'
bugun	'сегодня'	Бугунъ	'сегодня'

Table 4. Reliable range [t]—[t]—[t]—[t]

[t]		[t]	
таш	'камень'	tasch	'камень'
тимер	'железо'	temir	'железо'
тел	'язык'	til	'язык'
тиен	'белка'	tiin | tijin	'белка | белка'
тоз	'соль'	tust	'соль'
тук	'сытый'	tok	'сытый'
[t]		[t]	
tasch	'камень'	Ташъ | Тушакъ	'камень|постель'
temür	'железо'	Темуръ	'железо'
til	'язык'	Телль	'язык'
tijin	'белка'	Танде | Тонъ | Тїенъ	'утром | шуба | белка'
tuz	'соль'	Тусъ	'соль'
		Токъ | Таукъ | Тагакъ	'сытый | курица |

2.2.5 There is a specific difference in the correspondence [s] ~ [z] with only the 513 g2 dictionary retaining [s] (Table 9).

The consonantal [s] ~ [z] deafening noted by Miller is still observed in the dialects of the Zakazan Kryashens and Lower Kama region [6: 68].

2.2.6 The dictionaries under consideration show a general sequence of consonantal transfer, but the Cyrillic dictionary (1791) presents innovations that appear in many positions. Let us look at the most significant of them.

2.2.6.1 The correspondence between [j] ~ [dʒ]. Miller's dictionary records the variation in transmission between [j] and [dʒ] (Table 10).

In Tatar Literary Language, the [j] sound is preserved before the ancient back vowels. The use of the [zh] sound is a phonetic identification feature of the Middle dialect of the Tatar language. It is observed in subdialects such as Menzelinsky, Kasimov, Zakazan, Nagorny, and in the Tatar subdialects of the Urals. The use of the [zh] sound in the Tatar language is an ancient phenomenon [8: 62; 7: 30, 43, 70].

Table 5. Reliable range [k]—[k]—[k]—[k]

[k]		[k]	
кояш	‘солнце’	kün | kun	‘солнце | солнце’
ком	‘песок’	kumak | kum	‘песок | песок’
кургаш | каракургаш	‘свинец | свинец’	korgaschin	‘свинец’
куык	‘пузырь’	kok	‘пузырь’
кутак, ир-атның…	‘мужской детородный…	kutak	‘мужской половой о…’
казан	‘котёл’	kazan	‘котёл’
[k]		[k]	
kun	‘солнце’	Куяшъ	‘солнце’
kumak | kum	‘песок | песок’	Кумъ	‘песок’
kara-korgoldschin	‘свинец’	Кара-Коргашъ	‘свинец’
kuuk	‘пузырь’	Куокъ	‘пузырь’
kodok | kutak	‘мужской половой о…’	Кутакъ	‘мужской детородны…
kazan	‘котёл’	Казанъ	‘котёл’

Table 6. Correspondence [e]-[i]

[i]		[e]	
иртән	‘утром’	erten | erte | ertiä	‘утро | утро |
ишек	‘дверь’	eshik	‘дверь’
		(itmiak)	‘хлеб’
исәйдем	‘насытился’		
исерде	‘напился’		
[i]		[i]	
irte | erten	‘утром’	Ирта	‘утро’
ischik	‘дверь’	Ишекъ	‘дверь’
		Итмякъ	‘хлеб’
isÿedm	‘насытился’		
itschedm | itscher | i…	‘напился |		

Table 7. Preservation of the archaic [u]-[o]

[o]		[u]	
борын	‘нос’	burun | murun	‘нос | нос’
бозау	‘телёнок’	busau | busuu	‘телёнок |
[u]		[u]	
mordu	‘нос’	Буронь	‘нос’
bozou	‘телёнок’	Бузау	‘телёнок’

Table 8. The correspondence [ɨ] ~ [u]

[ɨ]		[ɨ]	
(беркесен | берсе көн	‘послезавтра’	birsy kün | basa kun	‘послезавтра’
		fary-balyk	‘осётр’
сары туфрак | сары	‘жёлтая почва’	sary	‘жёлтый’
утырдым	‘сел’	utyrdym	‘сел’
сарык	‘овца’		
		jily	‘тёплый’
[u]		[a]	
biürsugun | birsi-kun	‘послезавтра’	Бирзакунъ	‘послезавтра’
saru-balÿk	‘осётр’	Бекра	‘осётр’
saru dobrjak | saru	‘жёлтая почва’		
olturadym	‘сел’		
odun | otun	‘дрова | дрова’	Отанъ	‘дрова’
		Саракъ	‘овца’
jilu | julu	‘тёплый | тёплый’		

Table 9. The correspondence [s] ~ [z]

[z]		[s]	
бозау	'телёнок'	busau | busuu	'телёнок'
кузы	'ягнёнок'	koosy | koosu	'ягнёнок'
кызыл	'красный'	kysyl	'красный'
[z]		[z]	
bozou	'телёнок'	(Бузау)	'телёнок'
koozy	'ягнёнок'		
kyzyll	'красный'		

Table. 10. The correspondence [j] ~ [dʒ]

[j]		[j]	
янгыр	'дождь'	jangur | jamgyr	'дождь | дождь'
яшен	'молния'	jaschilb | jaaschin	'молния'
йомырка	'яйцо'	jomurtka	'яйцо'
яшел	'зелёный'	jaschil	'зелёный'
ярканат	'летучая мышь'	jarganat	'летучая мышь'
якты	'светлый'		
		jily	'тёплый'
ярты кен | ярты тен	'полдень|полночь'	jarymkun	'полдень'
якшәмбе	'воскресенье'	jek-schambe	'воскресенье'
еш	'лес'	jitsch	'лес'
йерер	'ходит'	juerer	'ходит'
[j]		[j]	
janqur | janqwur	'дождь | дождь'	Янгуръ	'дождь'
jaschilt	'молния'	Яшинъ	'молния'
jamurtka	'яйцо'	Іимурткагъ	'яйцо'
jaschil	'зелёный'	Яшелъ	'зелёный'
jarganat	'летучая мышь'		
jeryk | jaryk	'светлый'		
jilu | julu	'тёплый'		
[j]		[j]	
ел	'год'	jill	'год'
юл	'дорога'	jel | jol	'ветер | дорога'
		jetmysch | jette | jet…	'семьдесят|семь'
		jurjak	'сердце'
елап торам | елыйм	'плакал | плачу'		
[j]		[ʤ]	
jill	'год'	Джиллъ	'год'
jol | sjel kyn	'дорога|ветренный	Джоль | Джасишъ | Джи…	'дорога|светлый'
jetton | jetim |	'семьдесят'	Джидмышь | Джиде	'семьдесят|семь'
		Джуракъ	'сердце'
		Джиллименъ	'плачу'

This peculiarity is clearly presented in the Atlas of Tatar folk subdialects (https://atlas.antat.ru/atlas/maps.html?mapnom=33) [9] (refer to Fig. 12).

2.2.6.2 The next feature requiring attention is the correspondence [s] ~ [ch] (Table 11). This phonetic phenomenon, recorded in Miller's dictionary, can be correlated with the subdialects of the Urals of the Middle dialect of the Tatar language.

This peculiarity is clearly presented in the Atlas of Tatar folk subdialects (https://atlas.antat.ru/atlas/maps.html?mapnom=41) [9] (refer to Fig. 13).

Dialectologists note that the correspondence [s] ~ [ch] is particularly evident in the Tatar subdialects of the Urals. For instance, in the Krasnoufimsky subdialect себен – чебен (seben – cheben) 'fly', *сәй – чәй* (səy – chəy) 'tea', салгы – чалгы (salgy – chalgy) 'scythe', *сабата – чабата* (sabata – chabata) 'lapti', *бесән – печән* (besən – pechən) 'hay', *быскы – пычкы* (bysky – pychky) 'saw', *сыбырткы – чыбырткы* (sybyrtky – chybyrtky) 'whip', *баргас – баргач* (bargas – bargach) 'coming' [7: 304]. In Safakulsky sudialects sound [s] is systematically used instead of literary [ch]: *сәй – чәй* (səy – chəy) 'tea', *салгы – чалгы* (salgy – chalgy) 'scythe', *сикертkә – чикертkә* (sikertkə – chikertkə) 'grasshopper', систа – чиста (sista – chista) 'clean', килгәс – килгәч (kilgəs – kilgəch) 'coming' [7: 384].

In the Mamadysh, Laishev, Kasimov, and Zlatoust subdialects, as well as in the subdialects of Zakazan Kryashens, there have been isolated cases of [s] ~ [ch] correspondence recorded: *калас – калач* (kalas – kalach) ‘kalach’, *күмәс – күмәч* (kouməs – koumәch) ‘bread’, *кескенә – кечкенә* (keskenə – kechkenə) ‘small’, *эссә – эчсә* (essə – echsə) ‘if he/she drinks’, *умас – умач* (umas – umach) ‘dumplings’, *кесерәк – кечерәк* (keserək – kecherək) ‘smaller’ [7: 30, 44, 88].

Table. 11. The correspondence [s] ~ [ch]

[ch]		[ʧ]	
(ач | ачыкты)	‘голодный’	atsch | asch	‘голодный’
		tschatsch	‘волосы’
		sitschen	‘сено’
(атау)	‘холм’	kitschi tau	‘холм’
[ʧ]		[ss]	
(aschtap | aschtady)	‘голодный’	Ассъ	‘голодный’
		Чассъ	
bitschen	‘сено’	(Псіенъ)	‘сено’
kitschu-tau	‘холм’		

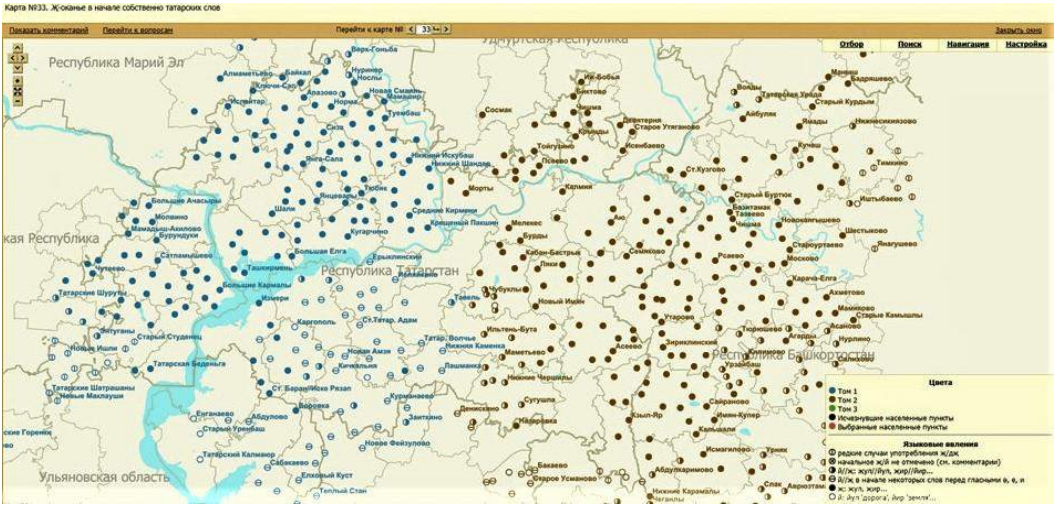


Fig. 12. Atlas of Tatar folk subdialects

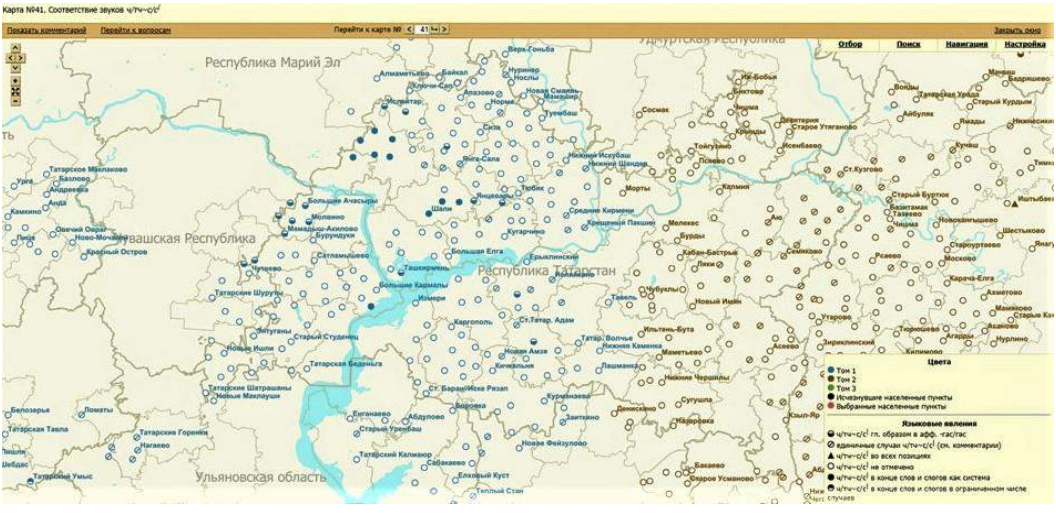


Fig. 13. Atlas of Tatar folk subdialects

2.2.6.3 Violation of vowel harmony in the second syllable: *катинъ* | *катинъ – куль* (katin'| katin') – (kul) 'wife|servant'; балчикъ (balchik) 'clay'; итакь (itak) 'boots'; аигеръ (aiger) 'stallion'; *алтинъ* (altin) 'gold'; *джидмышъ* | *джиде* (dzhidmysh| dzhide) 'seventy|seven'.

According to Tatar dialectologists, the feature is widespread "in the Perm subdialect of the Middle Dialect in the Kungu, Shaubi subgroup. For example, *акрин – акрын* (akrin – akryn) 'slow', *ишану – ышану* (ishanu – yshanu) 'to believe', *мастимал – тастымал* (tastimal – tastymal) 'hand towel', *утлик – утлык* (utlik – utlik) 'firebox', and *башлик – башлык* (bashlik – bashlik) 'headwear' [7: 183].

The Sergach and Chistopolsky subdialects of the Tatar language provide isolated examples of the correspondence of back vowels to front vowels. For instance, *абзий – абзый* (abziy – abziy) 'older person', *кина – кыйна* (kina – kyina) 'beat' [6: 51], in the Vyatka subdialect: *китыр – китер* (kityr – kiter) 'give me', *тәмны – тәмле* (tamny – tamle) 'delicious', *мескенкаем – мескенкәем* (meskenkaem – meskenkəem) 'poor one' [7: 148].

3. Results of the Study

Electronic corpora make it possible to fully reflect the language processes in different subsystems of the language. Thus, we will be able to predict in time the further development of the structure and possible changes in the norms of the codified Tatar language.

After analyzing phonetic and phonological transcriptions and calculating regular word correspondences in three etymologically related dictionaries, we discovered the etymological-phonetic closeness of lexemes.

Using a special comparative-historical program based on phonetic and etymological criteria, we determined that the dictionaries have varying degrees of distance from modern Tatar language (refer to Fig. 14).

Fig. 14 shows that according to phonetic-etymological criteria the distance from the manuscript dictionaries in the Latin script (513 g1, 513 g2) to the modern Tatar language is closer than between the 1791 Cyrillic dictionary.

4. Conclusion

The analysis of Miller's dictionaries on the LingvoDoc platform revealed the following:

1) The transition processes in consonant area of the Tatar language were completed by the 18th century.
2) The vowel change in the 18th century has not yet been completed for all vowels, and for [o] and [u], this process had not yet begun.
3) In the 18th century, dialectal zones were already identified in the Tatar language. Miller's dictionary of 1791 already represents the dialectal features of the Ural subdialects, including the correspondence [s] ~ [ch] and the correspondence [j] ~ [dʒ]. Additionally, there were unusual violations of row harmony.
4) Based on the obtained data, certain phonetic processes can be dated.
5) The analysis revealed that the Cyrillic version differs from modern Tatar language and the Latin versions of the dictionaries. It underwent innovative changes that brought the dictionaries closer to the Tatar subdialects of the Urals, especially the Perm subdialect. The Latin versions of the manuscript dictionaries show similarity to modern Tatar language in terms of the graphical and phonetic features presented.

The LingvoDoc platform provides access to corpora and concordances of Miller's dictionaries, enabling users to verify the findings using complete materials that are often unavailable in library collections.

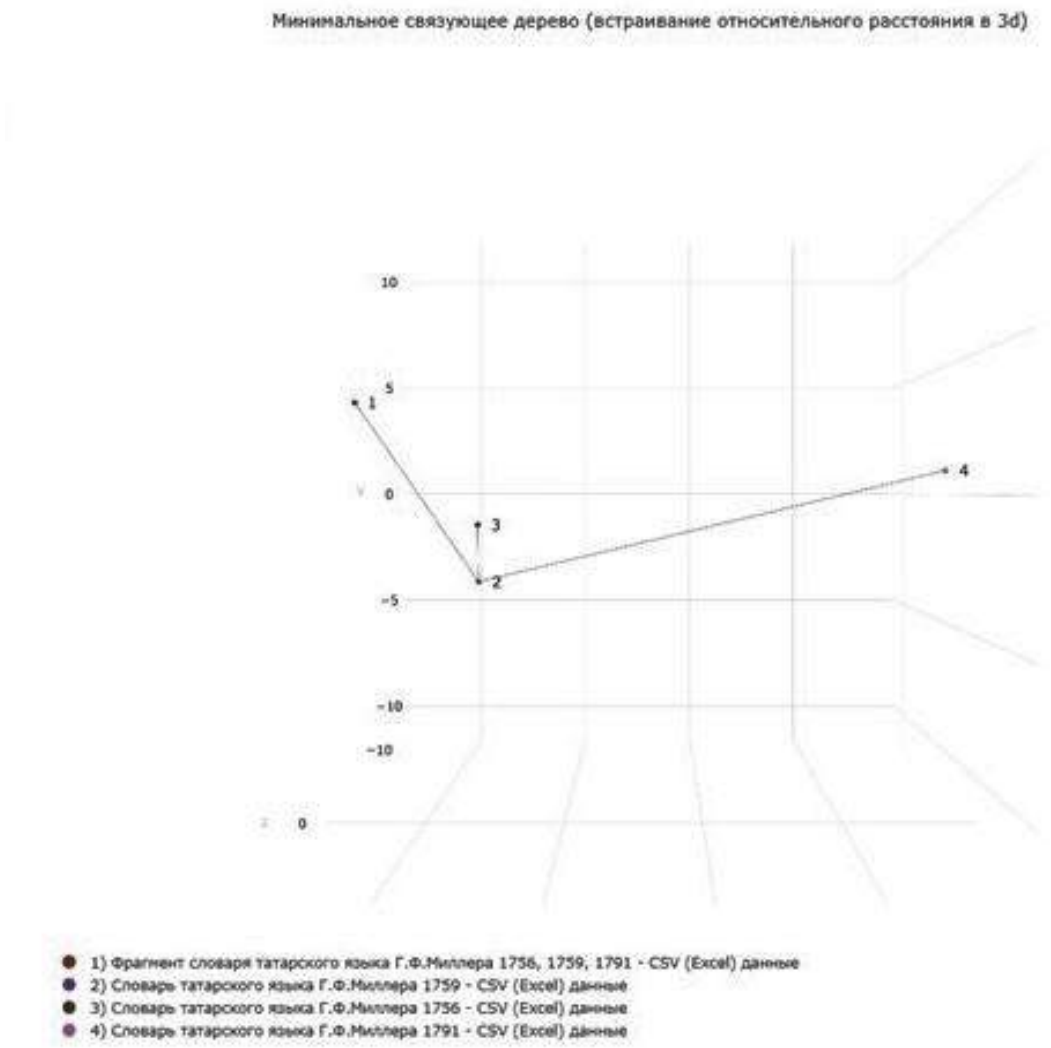


Fig. 14. Etymological Distance Tree

Список литературы / References

Шапиро А.Л. Русская историография с древнейших времен до 1917 г.: учебное пособие. 2-е изд., испр. и доп. М., Ассоц. «Россия»: Культура, 1993, 761 с. / Shapiro A.L. Russian Historiography from Ancient Times to 1917: textbook. 2nd edition, revised and supplemented. Moscow, Assoc. Russia: Kultura, 1993, 761 p. (in Russian).

Миллер Г.Ф. История Сибири. М.-Л.: Изд-во Акад. наук СССР, 1937-1941, Т. 1. 1937, 608 с. / Miller G.F. History of Siberia. Moscow-Leningrad, Publishing House of Academy of Sciences of the USSR, 1937-1941, vol. 1, 1937, 608 p. (in Russian).

Ежемесячные сочинения к пользе и увеселению служащих, при Императорской Академии наук. СПб., 1756, (июль), С. 33-64., (август) С. 119-145. / Monthly essays for the benefit and amusement of those serving at the Imperial Academy of Sciences. St. Petersburg, 1756, (July), pp. 33-64., (August) pp. 119-145 (in Russian).

Миллер Г.Ф. Описание живущих в Казанской губернии языческих народов, яко то черемис, чуваш и вотяков: с показанием их жительства, политического учреждения .... СПб., Имп. Акад. наук, 1791, 99, [2] с., [8] л. ил. / Miller G.F. Description of the Pagan Peoples Living in Kazan Province, such as Cheremis, Chuvash and Votyak: Showing their Residence, Political Institutions ... St. Petersburg, Imperial Academy of Sciences, 1791, 99, [2] p., [8] l. ill (in Russian).

Норманская Ю.В., Кошелюк Н.А. Неопубликованный мансийский словарь П.С. Палласа – ранее неизвестный мансийский диалект? Урало-алтайские исследования. 2020, № 1 (36). С. 92-100 / Normanskaya Y.V., Koshelyuk N.A. Unpublished Mansi dictionary of P.S. Pallas: a previously unknown Mansi dialect? Ural-Altaic Studies, vol. 1 (36), 2020, pp. 92-100 (in Russian).

Махмутова Л.Т. Опыт исследования тюркских диалектов (мишарский диалект татарского языка). М.: Наука, 1978. 272 с. / Makhmutova L.T. Experience in the Study of Turkic dialects (Mishar dialect of the Tatar language). Moscow, Nauka, 1978, 272 p. (in Russian).

Татар халык сөйләшләре: ике китапта / [авторлар коллективы: Ф.С. Баязитова и др.]. Казан: Мәгариф, 2008. Кит. 1: Беренче китап. 2008. 461 б. / Tatar Subdialects: two editions / [author's team: F.S. Bayazitova and others]. Kazan, Magarif, 2008. vol. 1. 2008. 461 p. (in Tatar).

Баязитова Ф.С. Говоры татар-крящен в сравнительном освещении. М.: Наука, 1986, 247 с. / Bayazitova F.S. Tatar-Kryashchen Subdialects in Comparative Research. Moscow, Nauka, 1986, 247 p. (in Russian).

Атлас татарских народных говоров, https://atlas.antat.ru/ (дата обращения: 10.12.2023) / The Atlas of Tatar Folk Subdialects, Available at: https://atlas.antat.ru, accessed 10.12.2023. (in Russian).

Информация об авторах / Information about authors

Фануза Шакуровна НУРИЕВА – доктор филологических наук, профессор. Научные интересы: татарский язык, тюркология, диалектология, сравнительно-историческое языкознание.

Fanuza Shakurovna NURIEVA – Dr. Sci. (Philology), Professor. Research interests: Tatar language, turkology, dialectology, comparative and historical linguistics.

Гульшат Раисовна ГАЛИУЛЛИНА – доктор филологических наук, профессор, заведующий кафедрой. Область научных интересов: татарский язык, языки народов РФ, лексикология, семасиология, ономастика, лингвокультурология, социолингвистика, этническая культура.

Gulshat Raisovna GALIULLINA – Dr. Sci. (Philology), Professor, Head of the Department. Research interests: Tatar language, languages of the peoples of the Russian Federation, lexicology, semasiology, onomastics, linguoculturology, sociolinguistics, ethnic culture.

Айрат Фаикович ЮСУПОВ – доктор филологических наук, доцент. Сфера научных интересов: татарский язык, сравнительно-историческое языкознание, текстология, семиотика, исламоведение.

Airat Faikovich YUSUPOV – Dr. Sci. (Philology), Associate Professor. Research interests: Tatar language, comparative and historical linguistics, textual criticism, semiotics, Islamic studies.

DOI: 10.15514/ISPRAS-2023-35(6)-18

Studying the Structure of the Sound of Russian Speech by Chinese and Ghanaian Students in a Digital Format: a Linguodidactic Aspect

S.A. Deryabina, ORCID: 0000-0002-5415-5433 <*deryabina.sa@gmail.com*>
Ren Wanying, ORCID: 0000-0002-2811-4582 <*1042215040@pfur.ru*>
Nketiah Eugenia, ORCID: 0000-0001-8149-8477 <*eugenianketiah@gmail.com* >
Patrice Lumumba Peoples' Friendship University of Russia
6, Miklukho-Maklaya st., Moscow, 117198, Russia

Abstract. The purpose of the research is to study the peculiarities of the structure of the sound of Russian speech by Chinese and Ghanaian students at different levels of study, based on experimental phonetic programs. The subject of the study is the intonational structure of the main communicative intentions: greeting, address, completion, incompletion, a question without an interrogative word, a question with the Russian conjunction «а» ("and"), a positive evaluation and surprise. The study contributes to the creation of an ethno-oriented methodology for teaching Russian intonation. The scientific novelty is in the analysis of the sound of the Russian speech of foreign students at the syntagma level in comparison with that of a native Russian speaker using the Praat and Lingvodoc programs. The results of the study consist in the visual representation of the peculiarities of the structure of the main communicative intentions in the Russian speech of Chinese and Ghanaians to predict and correct errors at the level of intonation.

Keywords: Intonation; phonetic skills; Chinese students; Ghanaian students; experimental phonetic programs; Praat; Lingvodoc.

For citation: Deryabina S.A., Ren Wanying, Nketiah Eugenia. Studying the structure of the sound of Russian speech by Chinese and Ghanaian students in a digital format: a linguodidactic aspect. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 283-292. DOI: 10.15514/ISPRAS-2023-35(6)-18.

Изучение оформления русской звучащей речи китайскими и ганскими студентами в цифровом формате: лингводидактический аспект

С.А. Дерябина, ORCID: 0000-0002-5415-5433 <*deryabina.sa@gmail.com*>
Жэнь Ваньин <1042215040@pfur.ru>
Нкетия Юджения <eugenianketiah@gmail.com>
Российский университет дружбы народов имени Патриса Лумумбы
117198, г. Москва, ул. Миклухо-Маклая, 10/2

Аннотация. Цель работы заключается в исследовании особенностей оформления русской звучащей речи китайскими и ганскими студентами на разных уровнях обучения на основе экспериментально-фонетических программ. Предметом внимания послужило интонационное оформление основных коммуникативных интенций: приветствие, обращение, завершенность, незавершенность, вопрос без вопросительного слова, вопрос с союзом «а», положительная оценка и удивление. Исследование представляет интерес для создания этноориентированной методики обучения русской интонации. Научная новизна – анализ звучащей речи иностранных студентов на уровне синтагмы в сравнении с носителями русского языка с использованием программ Praat и Lingvodoc. Результаты исследования

состоят в наглядном представлении особенностей оформления основных коммуникативных намерений в русской речи китайцев и ганцев с целью предупреждения и коррекции ошибок на уровне интонации.

Ключевые слова: Intonation, phonetic skills, Chinese students, Ghanaian students, experimental phonetic programs, Praat, Lingvodoc.

Для цитирования: Дерябина С.А., Жэнь Ваньин, Нкетия Юджения. Изучение оформления русской звучащей речи китайскими и ганскими студентами в цифровом формате: лингводидактический аспект. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 283–292 (на английском языке).. DOI: 10.15514/ISPRAS–2023–35(6)–18.

1. Introduction

The relevance of the research lies in the fact that the study of the structure of the sound of Russian speech occupies an important place in the process of teaching foreign students, and often poses major challenges. Moreover, inaccurate intonation can lead to communication failures. According to the survey conducted by S.A. Deryabina in 2021, in which 118 undergraduate, graduate, and postgraduate students from different countries participated, more than 90% of foreigners are of the view that, working on improving auditory-pronunciation skills and on prolonging the foreign language accent is relevant [7].

A characteristic feature of the development of the modern era is the inclusion of information communication technologies in all spheres of life [6]. Today, modern technology is widely used in the educational space. When teaching Russian intonation to foreign students, it is possible to use multimedia learning aids such as «Буква. Слово. Диалог» (“Letter. Word. Dialogue”) [3], ВФГК «Вперед» (“Forward”) [4] among others. Due to the rapid development of modern technologies in teaching, it is necessary to take full advantage of the opportunities offered by new effective forms of teaching, thus experimental phonetic programs.

Objectives of the study:

- to describe the results of the analysis of the sound of foreign students' Russian speech in comparison with that of a native Russian speaker using the Praat and Lingvodoc programs;
- to determine the degree of pronunciation skills formation of Chinese and Ghanaian students at the level of intonation;
- to create nationally oriented visual materials for Chinese and Ghanaians on the intonational structure of the main communicative intentions of the sound of Russian speech, demonstrating typical deviations at different stages of learning.

The following research methods are used in the work: description, analysis, comparison, experiment, evaluation.

The theoretical basis of the study is the internationally recognized work of E.A. Bryzgnova. In her opinion, seven types of intonational constructions (IC) can be distinguished in the Russian language, which express different purposes of an utterance [5]. IC-1 expresses completion in narrative sentences, without oppositions and emotional meanings. The middle descending tone formalizes the stressed syllable in this type of IC. IC-2 is used in questions with interrogative words, where the descending center is pronounced with a bit of pitch enhancement. It is also used in an appeal, demand, exclamation, statement with opposition, emphasis. IC-3 is most evident in an interrogative sentence without an interrogative word. In addition, this type of IC also expresses incompletion, comparison, and enumeration in non-terminal syntagmas. It is characterized by a significant increase in the pitch of the stressed part. IC-4 expresses an incomplete question with comparison, a question with the connotation of a demand; it is also used together with IC-3 in a non-terminal syntagma, where it means incompletion, comparison, and enumeration. However, unlike IC-3, IC-4 in an incomplete syntagma makes the speech formal. IC-5 is realized when expressing emotional evaluation, desire, and regret in sentences with and without the pronouns (какой (which), как (how), сколько (how many) etc.). Unlike other types of IC, IC-5 has two centers, so this construction is

possible in a sentence with at least two syllables. IC-6 expresses qualitative and quantitative evaluation, repeated question (переспрос), bewilderment, surprise, along with IC-3 and IC-4 in an incomplete syntagma, it formalizes the incompletion of an utterance. In the implementation of IC-6, the pitch level at the center of the vowel is raised, and the backward part is pronounced with a high middle tone. IC-7 serves as a means of expressing disagreement and negation in sentences with pronoun words, and in expressive utterances it intensifies affirmation, negation, and evaluation. In the practice of teaching Russian as a foreign language, generally, students are familiarized with the first five ICs, without the meaning of incompletion. Usually, future philologists and linguists get acquainted with IC-6 and IC-7, which have the meaning of incompletion.

The practical significance of the study consists in finding out the difficulties that Chinese and Ghanaian students face in learning Russian intonation based on experimental phonetic programs to create an ethno-oriented method of teaching Russian intonation to Chinese and Ghanaians.

2. Discussion and results

For the foreign students in this paper, Chinese and Ghanaian students, mastering Russian intonation is an integral and a difficult part of the learning process. They are faced with various difficulties due to the interference of their native languages with the target language. To determine the degree of formation of pronunciation skills at the level of intonation and the typical mistakes that the representatives of the two countries make, an experimental study was conducted.

The materials for the analysis are the speech recordings of the students (1 Russian, 4 Chinese, 4 Ghanaians), obtained from the reading of an excerpt from V.M. Shukshin's story “The Sun, the Old Man and the Girl” («Солнце, старик и девушка») (293 words); the results of the assessment of the students' Russian language proficiency level based on the Lingvodoc 3.0 platform; pitch contour illustrations of the read phrases (Praat).

The Audacity application [2] was used to cut the students' recordings into phrases in Wav format for further analysis. The Praat application was then used to convert the phrases into TextGrid format. Praat was also used for highlighting and naming phrases in Latin vowels for analysis as well as collecting pitch indicator data. The content of the work in the Lingvodoc application included creating a dictionary, uploading audio files in Wav and TextGrid formats and determining the level of foreigners’ Russian proficiency.

Researchers point out the possibilities and prospects of these experimental phonetic programs, Audacity and Praat, in the modern educational space for the purpose of teaching non-native languages to students, for example, Russian [8], Chinese [1], English [9], German [11], etc. Based on the software processing of the sounding speech, the works investigate the phonological features that occur in the pronunciation of native speakers, as well as compare the phonetic characteristics of the native and the studied languages.

The Lingvodoc program [12] developed at ISP RAS allows creating multilayer dictionaries and corpora with the obtained sound and text data. With the help of the program, experimental phonetic [10], etymological and morphological work is effectively reproduced.

Students with different levels of phonetic skills formation acted as foreign informants. The results of determining the level of Russian language proficiency on the Lingvodoc learning platform showed that 3rd year undergraduate students from China made errors at the B1 level, and 1st year master's students from China made errors at the C1 level. As for the Ghanaian informants, a 3rd year postgraduate student made errors at the C1 level, a 1st year master's student made errors at the A1 level, a PhD candidate made errors at the B1 level and a first-degree graduate from the University of Ghana made errors at the B1 level.

Below are the phrases used in the analysis:

- Здравствуйте, дедушка! (Hello, Grandpa!) – Greeting. Address
- Девушка поднялась и пошла в деревню. (The lady got up and went to the village.) –

Incompletion and completion

- Вы завтра придете сюда, дедушка? (Will you come here tomorrow, Grandpa?) – Question without an interrogative word
- А тебе? (And you?) – Question with the Russian conjunction «а» (“and”)
- Солнце-то какое! (The sun is so bright!) – Exclamation with a positive evaluation
- Ого! (Wow!) – Surprise

Let us consider the features of the intonational structure of these phrases with the help of experimental phonetic programs and in the tradition of the methodology of teaching Russian as a foreign language.

In this phrase, IC-2 is used in the meaning of greeting and address, where the center is pronounced with a bit of pitch enhancement. This is well illustrated in the intonational contour of the native speaker of Russian and the Chinese informants, who have advanced level of Russian phonetic skills (fig. 1). However, for the students with a B1 Russian language proficiency, pitch enhancement at the intonational center is not observed.

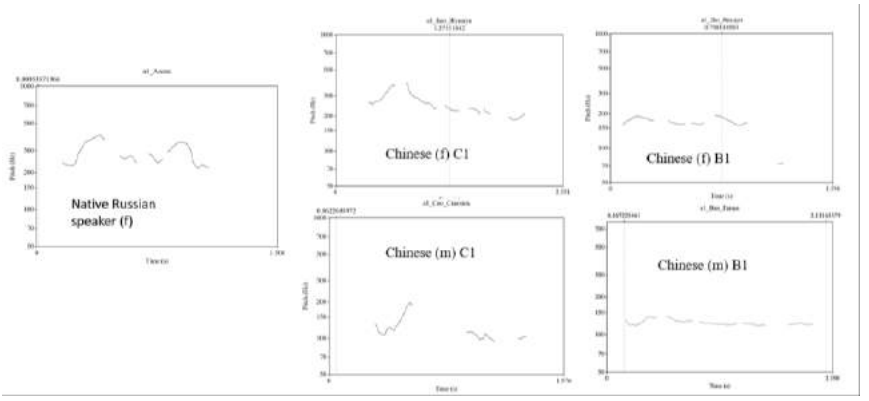


Fig.1. Intonational contour of the phrase «Здравствуйте, дедушка!» (“Hello, Grandpa!”) in the speech of Russian and Chinese informants

With regards to the Ghanaian participants, we see that the 3rd year graduate student with a C1 Russian language proficiency pronounces the phrase «Здравствуйте, дедушка!» (“Hello, Grandpa!”), with a slight increase at the center like that of the native speaker. The other participants (apart from the student with an A1 level of Russian language) also raise the pitch at the center of the phrase but at a very lower extent as compared to that of the native speaker (fig. 2).

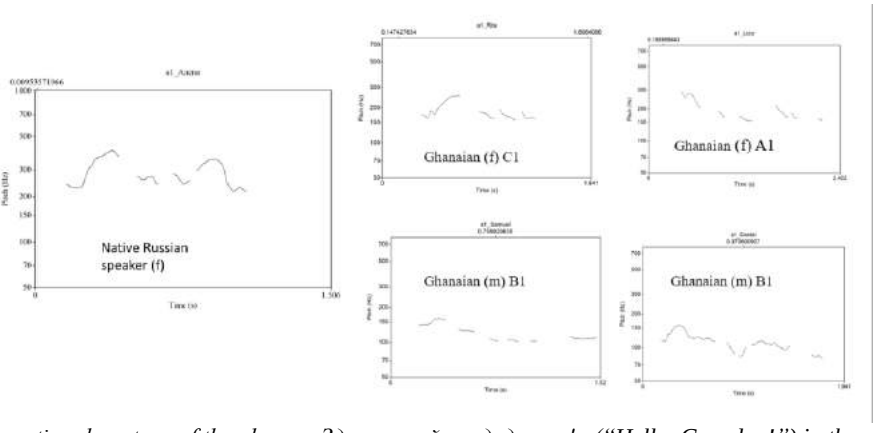


Fig.2. Intonational contour of the phrase «Здравствуйте, дедушка!» (“Hello, Grandpa!”) in the speech of Russian and Ghanaian informants

The category of completion/incompletion is usually considered as a relative semantic completion/incompletion of a linguistic unit, a criterion which shows the possibility or impossibility of its "communicative automatization" [16]. The phrase «Девушка поднялась и пошла в деревню» ("The lady got up and went to the village") consists of two syntagmas: «Девушка поднялась» ("The lady got up") and «и пошла в деревню» ("and went to the village"). The first syntagma is characterized by the intonation of incompletion (IC-3), the second by completion (IC-1).

As compared to the native speaker's intonation contour, most of the Chinese informants (except for the male Chinese speaker with advanced level of Russian language proficiency) have difficulty in expressing incompletion with a rising pitch. However, expressing completion with the help of IC-1 did not cause much difficulty for all the Chinese informants. Since in Chinese, the incompletion of an utterance is mainly characterized by a pause, it is challenging for Chinese learners to master the rising pitch for an unfinished syntagma (fig. 3).

In addition, it can be seen from the illustration that, the speech of the Chinese students with a B1 Russian language proficiency is interrupted. This shows the problem of pronouncing words within the same syntagma together.

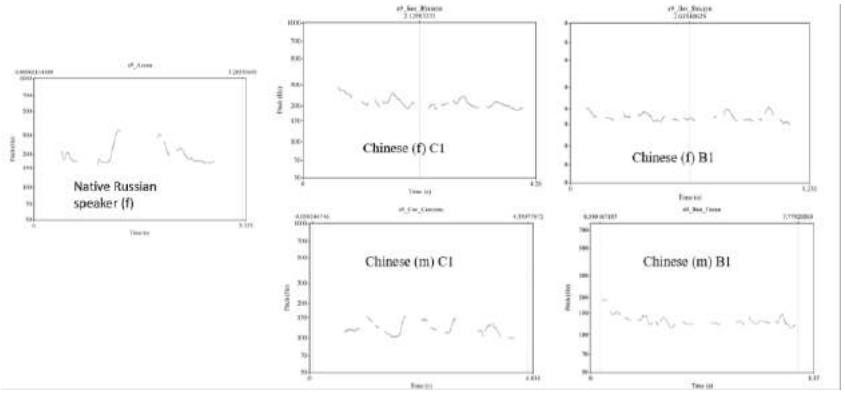


Fig.3. Intonational contour of the phrase «Девушка поднялась и пошла в деревню» ("The lady got up and went to the village") in the speech of Russian and Chinese informants

Looking at the pitch contours of the Ghanaian students in the pronunciation of the phrase above, it is seen that most of the participants (apart from the PhD candidate) had difficulty in expressing the first syntagma to show incompletion (IC-3). The second syntagma «и пошла в деревню» ("and went to the village") has the meaning of completion (IC-1), which is formed by lowering the intonation at the center. IC-1 in the meaning of completion is analogous in English language and the language Akan, hence the participants had no difficulty in expressing this phrase (fig. 4).

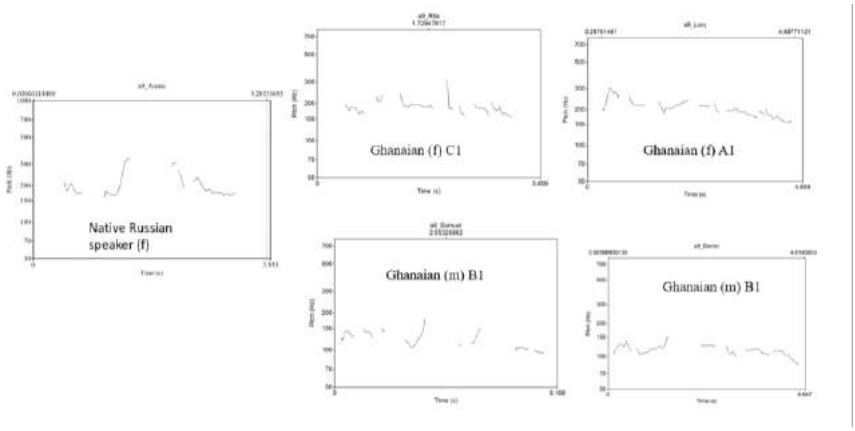


Fig.4. Intonational contour of the phrase «Девушка поднялась и пошла в деревню» ("The lady got up and went to the village") in the speech of Russian and Ghanaian informants

A question with an interrogative word is easily perceived by the ear due to its lexical composition, whereas the understanding of a question without an interrogative word relies more on the intonation of the phrase. In the case of a question without an interrogative word, the intonational center in IC-3 is on the word that carries the main semantic load. With the help of the context («- Вы завтра придете сюда, дедушка? … - Приду…») ("Will you come here tomorrow, Grandpa? ... - I will come...") it can be understood that the intonational center in this question should be on the word «придете» ("I will come"). In the intonational contour of the two undergraduate students, no clear intonational center is observed, but it is present in the master students' contour. This indicates the undergraduate students' inability to express IC with a rising pitch. It is also worth noting that the Chinese male informant with an advanced level of Russian incorrectly identified the intonational center by highlighting the word «завтра» ("tomorrow"), which is evident of his misunderstanding of the dialogue (fig. 5).

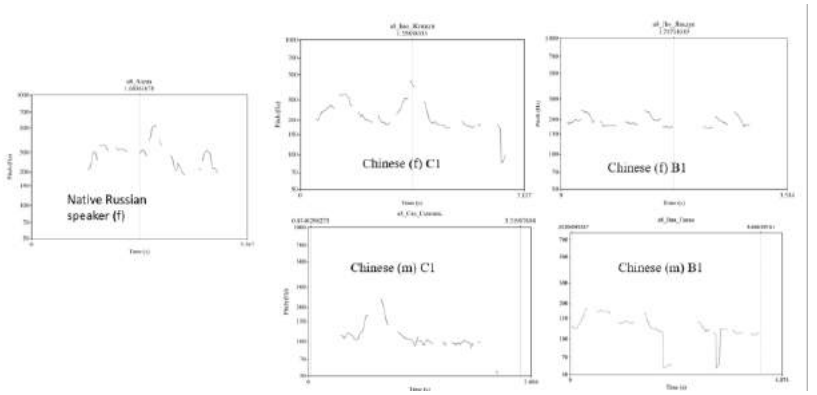


Fig.5. Intonational contour of the phrase «Вы завтра придете сюда, дедушка? » ("Will you come here tomorrow, Grandpa?") in the speech of Russian and Chinese informants

Concerning the question above, the Ghanaian informants with a B1 Russian language proficiency placed the intonational center on the word «завтра» ("tomorrow") instead of «придете» ("will come"), while the female participant with A1 level of Russian language placed the intonational center on the word «сюда» ("here"). The female Ghanaian participant with a C1 level of Russian language placed the intonational center on the word «придете» ("will come") like the native speaker. In this regard, we can say that most of the participants have difficulty in using the right intonation (IC-3) to ask a question without an interrogative word.

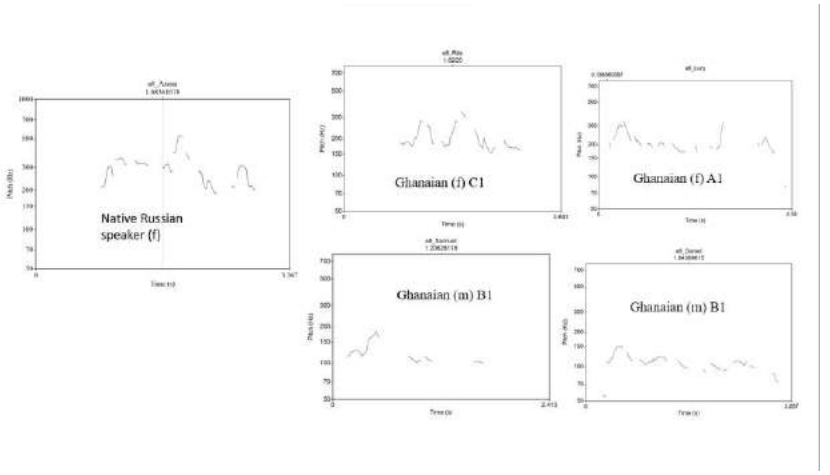


Fig.6. Intonational contour of the phrase «Вы завтра придете сюда, дедушка? » ("Will you come here tomorrow, Grandpa?") in the speech of Russian and Ghanaian informants

In this phrase IC-4 expresses an incomplete question with comparison. It can be seen that most of the informants use IC-4 correctly in this sense (fig. 7).

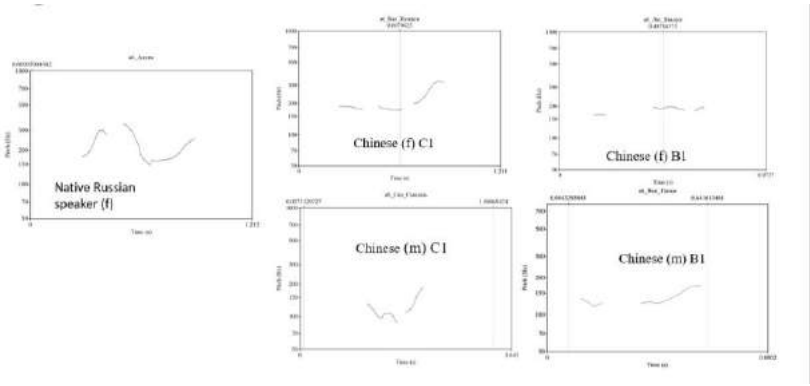


Fig.7. Intonational contour of the phrase «А тебе? » (“And you?”) in the speech of Russian and Chinese informants

Most of the Ghanaian participants also rightly used IC-4 to express an incomplete question (fig. 8) using the conjunction «а» (“and”) in the phrase «А тебе?» (“And you?”).

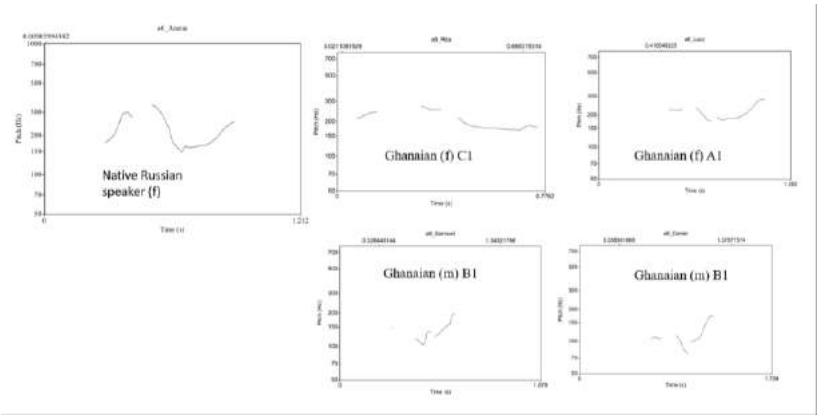


Fig 8. Intonational contour of the phrase «А тебе?» (“And you?”) in the speech of Russian and Ghanaian informants

A striking feature of IC-5 is the presence of two centers. This is clearly visible in the intonational contour of the native speaker, but the illustrations of the Chinese informants are more similar to a question with an interrogative word or a statement, thus, exclamation was not expressed (fig. 9).

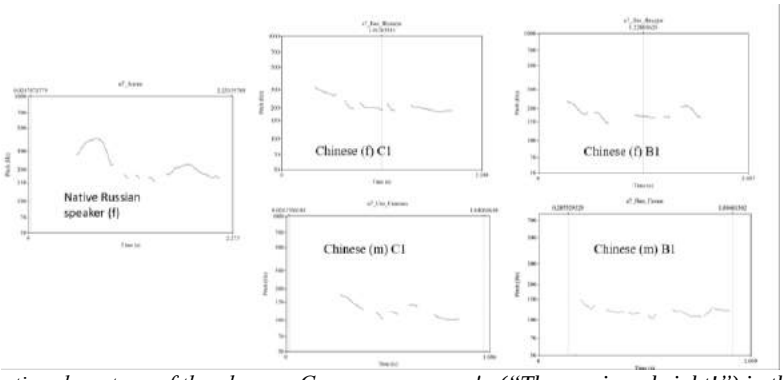


Fig.9. Intonational contour of the phrase «Солнце-то какое!» (“The sun is so bright!”) in the speech of Russian and Chinese informants

The intonational contours of the Ghanaian informants in the pronunciation of the phrase «Солнце-то какое! » ("The sun is so bright!") as compared to that of the native speaker do not clearly show two centers which is characteristic of IC-5 (fig. 10).

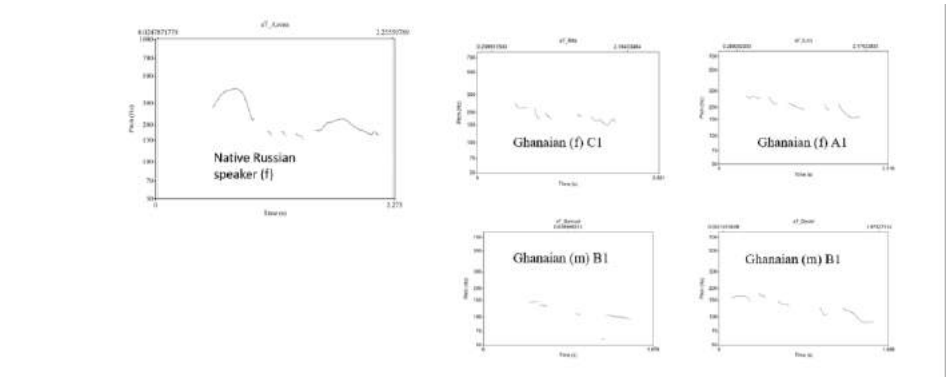


Fig.10. Intonational contour of the phrase «Солнце-то какое!» ("The sun is so bright!") in the speech of Russian and Ghanaian informants

IC-6 expresses surprise in this phrase. It is easy to notice that the Chinese informants find it difficult to render emotional expressions in their Russian speech (fig. 11).

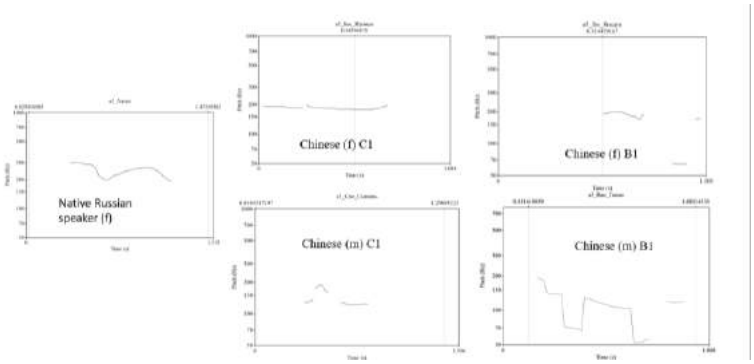


Fig.11. Intonational contour of the phrase «Ого! » ("Wow!") in the speech of Russian and Chinese informants

The female Ghanaian informants have similar intonational contours (fig. 12) like that of the native speaker in the phrase «Ого!» ("Wow!"), however the native speaker expressed more emotions in this phrase. The male Ghanaian participants had difficulty in using IC-6 to express surprise.

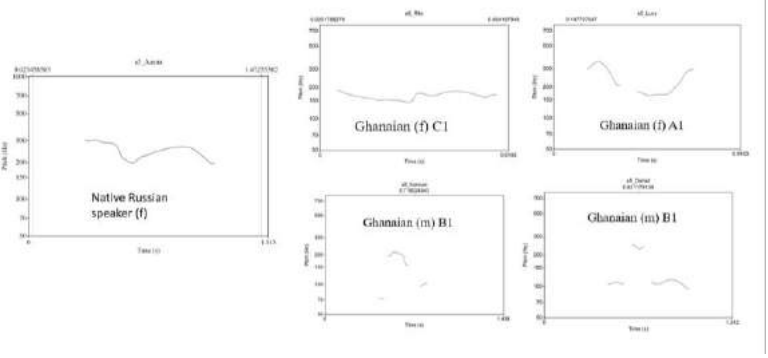


Fig.12. Intonational contour of the phrase «Ого!» ("Wow!") in the speech of Russian and Ghanaian informants

3. Conclusion

The results of the study led to the following conclusions:

1. The data of the analysis shows that Chinese and Ghanaian students have difficulties in assimilating Russian intonation. Inadequate pronunciation skills at the level of intonation hamper the formation of communicative competence. As a result, foreign students experience communicative failures in the process of communication.
2. This study shows the necessity of paying special attention to the study of Russian language intonation in the Chinese and Ghanaian classrooms, as it is observed that participants at the beginning, intermediate and advanced levels of Russian language face difficulties in using intonational constructions.
3. Most of the Chinese respondents had difficulties in expressing IC-5 and IC-6, and most of the Ghanaian respondents had difficulties in expressing IC-3 and IC-5.
4. Pauses within syntagmas are clearly observed in the speech of Chinese and Ghanaian students, which indicate the need to work on teaching how words within the same syntagma are pronounced together.
5. The existing methodology is not quite justified regarding teaching intonation to Chinese and Ghanaian students.
6. It is relevant to create an ethno-oriented method of teaching Russian intonation to Chinese and Ghanaians.
7. Experimental phonetic programs make it possible to create nationally oriented visual materials for teaching the intonational structure of the sound of Russian speech, observing how accurate or inaccurate foreign students articulate Russian phrases.

References

[1]. Antonova, A. S. Using Praat speech analysis software and Audacity audio editor in teaching pronunciation in Chinese / A. S. Antonova // Modern Oriental Studies. - 2019. - T. 1, № 1. – C. 5-9.
[2]. Audacity. Audio editor. https://www.audacityteam.org/, accessed Dec. 15, 2023).
[3]. Balykhina, T.M. et al. Letter. Word. Dialogue: Multimedia program for Russian language. Available at: http://rusist24.rudn.ru/index.php/obuchenie/katalog-uchebnykh-programm/bukva-slovo-dialog, accessed Dec. 15, 2023).
[4]. Balykhina, T.M. et al. VFGC “Forward”: Multimedia manual. Available at: http://rusist24.rudn.ru/index.php/obuchenie/katalog-uchebnykh-programm/vpered, accessed Dec. 15, 2023).
[5]. Bryzgunova E.A. Intonation // Russian grammar: in 2 vol. T. 1 / ed. by N.Y. Shvedova. M., 1980. C. 96-122.
[6]. Claro M., Salinas A., Cabello-Hutt T., San Martin E., Preiss D.D., Valenzuela S., Jara I. Teaching in a Digital Environment (TIDE): Defining and measuring teachers' capacity to develop students' digital information and communication skills. Computers & Education. 2018. Vol. 121. DOI 10.1016/j.compedu.2018.03.001.
[7]. Deryabina, S.A. Phoneticisation of teaching Russian as a foreign language in digital humanitarian knowledge: theoretical aspect / S.A. Deryabina, N.A. Lyubimova // Rusistika. - 2021. - T. 19, № 3. - C. 298-312. - DOI 10.22363/2618-8163-2021-19-3-3-298-312.
[8]. Garkusha, O.S. Revealing the relevant characteristics of sounding speech for native and non-native speakers (the Russian language) on the examples of audio signals studies / O.S. Garkusha, E.A. Maklakova // Science transforms reality: Proceedings of the International Interdisciplinary Scientific and Practical Student Conference, Voronezh, 27-31 March 2023 / Editor-in-Chief E.A. Maklakova. – Voronezh: Voronezh State Forest Engineering University named after G.F. Morozov. G.F. Morozov, 2023. - P. 48-57. - DOI 10.58168/STR2023_48-57.
[9]. Kolesnichenko, M.A. Using the possibilities of the PRAAT program in teaching English phonetics / M.A. Kolesnichenko // Asia-Pacific Studies: society, culture, politics: collection of materials, Vladivostok, 06 March 2019. - Vladivostok: Far Eastern Federal University, 2019. - C. 194-195.

[10]. Koshelyuk, N. A. Option "Phonology" of the LingvoDoc platform as a way of verification (on the material of the Sosvin dialect of the Mansi language) / N. A. Koshelyuk // Proceedings of the Institute of System Programming of the Russian Academy of Sciences. - 2022. - T. 34, № 6. - C. 165-172. - DOI 10.15514/ISPRAS-2022-34(6)-12.

[11]. Kuzmicheva, A. A. Complex analysis of speech utterance with the help of PRAAT software (on the material of the German language) / A. A. Kuzmicheva // Scientific discussion: issues of philology and methods of teaching foreign languages: a collection of articles on the materials of the International scientific-practical conference, Nizhny Novgorod, 04-05 April 2019 / Nizhny Novgorod State Pedagogical University named after Kozma Minin. – Nizhny Novgorod: federal state budgetary educational institution of higher professional education "Nizhny Novgorod State Pedagogical University named after Kozma Minin", 2019. - C. 65-67.

[12]. Lingvodoc. Linguistic platform. http://lingvodoc.ispras.ru/ , accessed Dec. 15, 2023.

[13]. Praat. Linguistic tool. https://www.fon.hum.uva.nl/praat/, accessed Dec. 15, 2023.

[14]. Sorokoletova N.Y. Intonation of incompleteness: universal or proper linguistic // Interaction of languages and cultures: studies of graduates and potential participants of Fulbright programs: Proceedings of the IV International Scientific Conference, Cherepovets, 15-16 April 2015 / edited by G.N. Chirsheva; co-editor: G.N. Chirsheva; V.P. Korovushkin. - Cherepovets: Cherepovets State University, 2015. - C. 118-121.

Information about authors

Светлана Александровна ДЕРЯБИНА - кандидат педагогических наук, доцент, доцент кафедры русского языка и методики его преподавания Российского университета дружбы народов имени Патриса Лумумбы. Сфера научных интересов: практическая фонетика, методика обучения русскому языку как иностранному, электронная лингводидактика.

Svetlana Aleksandrovna DERYABINA - Cand. Sci. (Ped.), Associate Professor, Associate Professor of the Department of the Russian Language and Its Teaching Methods, Peoples' Friendship University of Russia named after Patrice Lumumba. Research interests: practical phonetics, methods of teaching Russian as a foreign language, electronic linguodidactics.

Ваньин ЖЭНЬ - аспирант кафедры русского языка и методов его преподавания Российского университета дружбы народов имени Патриса Лумумбы. Сфера научных интересов: методика преподавания русского языка как иностранного, фонетика, овладение вторым языком, образовательные технологии, смешанное обучение.

Wanying REN - is a postgraduate student of the Department of the Russian Language and Its Teaching Methods of the Peoples' Friendship University of Russia named after Patrice Lumumba. Research interest: Teaching Russian as a foreign language, Phonetics, Second Language Acquisition, Educational Technology, Blended learning.

Юджения НКЕТИЯ - аспирант кафедры русского языка и методов его преподавания Российского университета дружбы народов имени Патриса Лумумбы. Сфера научных интересов: фонетика, овладение вторым языком, методика преподавания русского языка как иностранного, методика преподавания английского языка как иностранного, технологии обучения.

Eugenia NKETIAH - is a postgraduate student of the Department of the Russian Language and Its Teaching Methods of the Peoples' Friendship University of Russia named after Patrice Lumumba. Research interest: Phonetics, Second Language Acquisition, Teaching Russian as a foreign language, Teaching English as a foreign language, Educational Technology.

DOI: 10.15514/ISPRAS-2023-35(6)-19

Использование цифровых методов для выделения структурно-территориальных единиц центрально-южного диалекта удмуртского языка на основе анализа полевых записей

А.Ф. Уткина, ORCID: 0000-0002-1783-0380 <*sandra199608@rambler.ru*>

Удмуртский институт истории, языка и литературы УдмФИЦ УрО РАН, Россия, 426004, Удмуртская Республика, Ижевск, улица Ломоносова, 4.

Аннотация. В статье приведены результаты компьютерного фонетико-этимологического и глоттохронологического анализа данных говора д. Бобья-Уча Малопургинского района Удмуртской Республики. Насколько нам известно, диалектные особенности исследуемого говора ранее не привлекали внимание исследователей. В ходе работы по гранту РНФ были собраны аудиозаписи двух словарей разговорного языка, которые до сих пор используются в повседневной жизни. В настоящее время аудиоматериалы словарей затранскрибированы и размещены на платформе ЛингвоДок (lingvodoc.ispras.ru), что позволило установить этимологические связи с базовыми словарями удмуртских диалектов. Затем, с помощью специальных программ, было проанализировано расхождение говора д. Бобья-Уча от удмуртского литературного языка. Выявлены фонетические особенности исследуемого говора и уточнена его диалектная принадлежность. Сравнительно-исторический анализ фонетических особенностей показал, что согласные фонемы исследуемого говора не подверглись каким-либо изменениям и не имеют диалектных особенностей. Однако в говоре д. Бобья-Уча существует специфический звук ’, исследование которого требует дальнейших научных изысканий. Вокалическая система изучаемого говора определена несколькими изменениями акустико-артикуляционных свойств отдельных фонем: **і* > *ə*, **ɐ* > *ə*, **ɞ* > *з*. Таким образом, исходя из этимолого-фонетического и глоттохронологического анализа (совпадение 96%), можно видеть, что говор д. Бобья-Уча принадлежит к центрально-южному диалекту удмуртского языка.

Ключевые слова: удмуртский язык; диалекты; глоттохронология; фонетика; Лингводок.

Для цитирования: Уткина А.Ф. Использование цифровых методов для выделения структурно-территориальных единиц центрально-южного диалекта удмуртского языка на основе анализа полевых записей. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 293–310. DOI: 10.15514/ISPRAS–2023–35(6)–19.

Благодарности: Исследование выполнено при поддержке гранта Российского научного фонда (проект No 20-18-00403). Особая благодарность Ю. В. Норманской, д. ф. н., заведующей лабораторией «Лингвистические платформы» ИСП РАН, сотруднику отдела «Урало-алтайских языков» Института языкознания РАН, главному редактору Лингвистической платформы ЛингвоДок, за помощь и содействие в исследовании.

The use of digital methods to identify the structural and territorial units of the Central-southern dialect of the Udmurt language based on the analysis of field records

A.F. Utkina, ORCID: 0000-0002-1783-0380 <*sandra199608@rambler.ru*>

Udmurt Institute of History, Language and Literature UdmFRC UB RAS,
4, Lomonosov st., Izhevsk, Udmurt Republic, 426004, Russia.

Abstract. The article presents the results of computer phonetic-etymological and glottochronological analysis of data from the dialect of the village of Bobya-Ucha, Malopurginsky District, Udmurt Republic. As far as we know, the dialectal peculiarities of the studied material have not previously attracted the attention of researchers. In the course of work under the RSF grant, audio recordings of two dictionaries of the colloquial language were collected. The vocabulary of which is still used in everyday life. At the present days, the audio materials of the dictionaries were transcribed and placed on the LingvoDoc platform (lingvodoc.ispras.ru), which made it possible to establish etymological links with the basic dictionaries of Udmurt dialects. Then, with the help of special programs, the divergence of the Bobya-Ucha dialect from the Udmurt literary language was analyzed. The phonetic peculiarities of the studied dialect were revealed and its dialectal affiliation was clarified. The comparative-historical analysis of phonetic peculiarities has shown that the consonant phonemes of the studied dialect have not undergone any changes and have no dialectal peculiarities. However, there is a specific sound ’ in the Bobya-Ucha dialect, the study of which requires further research. The vocalic system of the studied vernacular is determined by several changes in the acoustic and articulatory properties of individual phonemes: **ӥ > ə*, **ɐ > ə*, **ɑ > з*. Thus, based on the etymological-phonetic and glottochronological analysis (96% coincidence), we can see that the dialect of the village Bobya-Ucha belongs to the central-southern dialect of the Udmurt language.

Keywords: the Udmurt language; dialects; glottochronology; phonetics; LingvoDoc.

For citation: Utkina A. F. The use of digital methods to identify the structural and territorial units of the Central-southern dialect of the Udmurt language based on the analysis of field records. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 6, 2023. pp. 293-310 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-19.

Acknowledgements. The research was supported by a grant from the Russian Science Foundation (project No. 20-18-00403). Special thanks to Yu. V. Normanskaya, Dr. Sci. (Phylology), Principal Researcher, the Head of the Linguistic Platforms Laboratory of the ISP RAS, Leading Researcher of the Department of Ural-Altaic Languages of the Institute of Linguistics of the Russian Academy of Sciences, Chief Editor of the Linguistic Platform LingvoDoc, for help and assistance in the research.

1. Введение

Проблема достоверного членения диалекта на структурно-территориальные единицы возникает особенно остро в случаях, когда нет четких границ распространения того или иного говора на территории. Для определения границ необходимо комплексное описание диалектов. В нашем случае собственно южный диалект удмуртского языка остается одним из наименее изученных диалектов.

К числу совершенно не исследованных говоров относится и говор деревни Бобья-Уча, расположенной в Малопургинском районе Удмуртской Республики. Деревня расположена близ деревень Курегово, Гожня, в речи жителей которых, как отмечают удмуртские диалектологи, присутствуют черты срединных говоров, хотя они относятся к южному наречию. Севернее их находятся населенные пункты Норьинской сельской управы, речь жителей которых относят также к срединным говорам [1].

Удмуртские диалекты впервые были описаны зарубежными учеными. Ф. Й. Видеманн, эстонский лингвист, опубликовал статью «Zur Dialektenkunde der wotjakischen Sprache», в которой описал шесть удмуртских диалектов: казанский, глазовский, елабужский, малмыжский, сарапульский и оренбургский [2]. Финский исследователь Т. Г. Аминофф представил только два территориальных варианта: казанский и вятский (якшурский) [3-4].

Венгерский ученый Б. Мункачи в сборнике «Votják népköltészeti hagyományok» [5] писал о пяти территориальных вариантах, а в более позднем труде выделял уже восемь диалектов: глазовский, елабужский, казанский, пермский, сарапульский, слободской, самарский и уфимский [6]. Финский ученый Ю. Вихманн в одной из своих работ оперировал восемью диалектами: бесермянским, глазовским, елабужским, казанским, малмыжским, малмыжско-уржумским, сарапульским, слободским, уфимским [7].

Среди отечественных ученых первым, кто занялся исследованием удмуртских диалектов, стал Т. К. Борисов. Он выделил четыре наречия удмуртского языка: северное, срединное, южное и крайне-южное [8]. В 1935 году С. П. Жуйковым была составлена первая карта диалектного членения удмуртского языка, на которой были отмечены северный, срединный и южный диалекты [9]. Позднее Р. Ш. Насибуллин выпустил диалектологическую карту с изображением пяти наречий удмуртского языка: северного, бесермянского, среднего, южного и периферийно-южного [10]. Ученые Т. И. Тепляшина и В. И. Лыткин в своей карте разделяют удмуртский язык на два наречия (северное и южное) и две группы говоров (срединные и бесермянские) [11]. В современной удмуртской диалектологии можно выделить два основных подхода к дроблению диалектной речи. В. К. Кельмаков и С. А. Максимов выделяют три наречия: северное, южное и бесермянское, а также группу срединных говоров, сочетающих особенности северного и южного наречий [12-13]. Другой удмуртский исследователь М. Г. Атаманов предлагает свой подход к классификации удмуртских диалектов. Он считает необходимым учитывать исторический фактор и выделяет в удмуртском языке два наречия: северноудмуртское и южноудмуртское. Кроме двух наречий, ученый выделяет особый нижневятский диалект, срединные говоры и бесермянский язык [14].

Среднеюжные говоры, куда входит исследуемый говор, были в той или иной степени описаны Г. А. Архиповым [15-16]. Исследователь отмечает, что в состав гласных входят 7 фонем (*a, o, u, e, i, į, ę*), почти полностью совпадающих в произношении с литературными *a, о, у, э, и, ы, ӧ*. Кроме того, в системе гласных среднеюжного диалекта встречаются дифтонги *wa*, *aw* и *ow*. В системе консонантизма имеются 30 фонем, включая фонемы *χ* и *f* русского происхождения. Специфической фонемой (архаического происхождения) является смычно-проходной носовой *ŋ*. Самым распространенным явлением в области фонетики в данном диалекте является ассимиляция, притом как в области согласных, так и в области гласных, например, *gįl'l'anį* < *gįl'janį* < *gįljanį* – лит. удм. *гылляны* ‘прополоскать’, *bakoosįt* < *bakaosįd* – лит. удм. *бакаосыд* ‘лягушки (твои)’ [17]. Необходимо также отметить замену согласных звуков *g* и *k*, *d* и *t* специфическим гортанным звуком **’** (гамза) в позиции перед другими согласными, например, *ба’ча* // *бакча* – лит. удм. *бакча* ‘огород’, *бор’дор* // *борддор* – лит. удм. *борддор* ‘стена’ [15].

Однако диалектные особенности говора д. Бобья-Уча, насколько нам известно, ранее не привлекали внимание исследователей.

2. Материалы и методы

В настоящее время на платформе ЛингвоДок (www.lingvodoc.ispras.ru) любой пользователь после регистрации имеет возможность создавать свои собственные словари и корпуса, а также анализировать материалы, размещенные другими пользователями и находящиеся в открытом доступе. В разделе «Инструменты» можно выбрать опцию «Глоттохронологический анализ языков/диалектов». Эта опция применима к любому набору языков, в словарях которых есть более 50 слов из стословного списка М. Сводеша. Выбор 100-словного списка Сводеша – Старостина [18] обусловлен исключительной обоснованностью и разработкой строгих семантических спецификаций [19-20], что позволяет получить достаточно точные сравнительные данные для разных языков. С. А. Старостиным была создана формула скорости распада языков, основанная на идее о том, что

с течением времени языки, которые однажды были родственными, начинают различаться и эволюционировать. Она учитывает количество различий в базисной лексике и позволяет оценить степень родства между языками (рис. 1). На ее основе разработан алгоритм построения деревьев языкового родства, реализованный в СУБД Starling (www.starling.rinet.ru) (см. [21]).

$$t = \sqrt{\frac{\ln\left(\frac{Nn(t)}{N_o}\right)}{-n\lambda^n \sqrt{Nn(t)}}}$$

Рис. 1. Формула С. А. Старостина для обсчета близости языков и диалектов, встроенная в ЛингвоДок
Fig. 1. S. A. Starostin's formula for calculating the proximity of languages and dialects, built into the LingvoDoc

На сегодняшний день на платформе ЛингвоДок доступны около 2000 словарей и корпусов, созданных на основе аудиословарей, собранных в полевых условиях в формате .wav, и архивных записей по уральским и алтайским языкам. На данный момент на платформе размещен 31 словарь по удмуртским диалектам. Этот материал позволил начать разработку программ для анализа данных с целью уточнения транскрипций языковых данных, этимологического анализа и построения классификаций на основе анализа фонетических инноваций в близкородственных языках и диалектах.

Для определения степени близости диалектов с точки зрения фонетических инноваций была разработана программа «Анализ когнатов в разных диалектах одного языка / в разных языках». Эту программу можно найти во вкладке «Инструменты» любого словаря. На первом этапе алгоритм вычисляет соответствия каждого символа в транскрипции с другими диалектами этого же языка, связанными этимологическими связями с текущим словарем. Затем пользователь выбирает словари для анализа в появляющемся меню. Эта программа позволяет полуавтоматически анализировать большие массивы словарных данных (15–20 тыс. единиц) для выявления рядов соответствий и дополнительных распределений на основе фонетических словарей диалектов одного языка и языков близкородственных. Эта функция необходима в работе с диалектными материалами, и диалектологам обычно не хватает подобной функциональности для выяснения полного набора соответствий в однотипном материале.

С помощью этих программ, которые позволяют оценить диалектные особенности, мы проанализируем материал по среднеюжным говорам центрально-южного диалекта удмуртского языка, собранный в д. Бобья-Уча в рамках работы по гранту РНФ No 20-18-00403. Автор статьи является жителем этого поселения. На цифровой диктофон записаны по 294 лексемы от носителей языка О. В. Ивановой и Е. С. Яковлевой, которые являются старожилами деревни. Каждое слово носительницы произносили изолированно 3 раза. Далее звуковая запись опроса была сегментирована в программе Sound Forge на отдельные слова с переводом на русский. Из сегментированных аудиофайлов были составлены словари, в которые также были добавлены параллели из современного удмуртского языка. Данные словари представлены в открытом доступе на платформе Лингводок, см. (Словарь среднеюжных говоров центрально-южного диалекта удмуртского языка и Словарь центрально-южного диалекта удмуртского языка (говор д. Бобья-Уча Малопургинского района УР) 2023).

В следующем разделе мы привели результаты сравнения словарей говора д. Бобья-Уча с двадцати двумя словарями диалектов удмуртского языка, выполненные с помощью программы «Анализ близости диалектов». В этой программе были подсчитаны фонетические различия в этимологически родственных словах. Также был выполнен анализ различий в базисной лексике семнадцати удмуртских словарей по алгоритму, разработанному С. А. Старостиным. Данный алгоритм позволяет выявить различия в базовой лексике между различными языками или диалектами. Результаты этих анализов могут помочь в определении степени родства и сходства между говором д. Бобья-Уча и другими говорами удмуртского языка.

3. Исследование и результаты

3.1 Фонетико-этимологический анализ среднеюжных говоров центрально-южного диалекта удмуртского языка (говор д. Бобья-Уча Малопургинского района УР)

Говоря о системе консонантизма удмуртских диалектов, необходимо обратиться к труду В. К. Кельмакова «Формирование и развитие фонетики удмуртских диалектов». В своей работе ученый реконструировал праудмуртскую систему консонантизма, состоящей из 28 единиц (табл. 1). Он также утверждает, что данная система согласных сохранилась вплоть до наших дней в отдельных периферийных диалектах удмуртского языка, в частности, в говоре д. Варклед-Бодья Агрызского района Татарстана [22].

Табл. 1. Система консонантизма праудмуртского языка по В. К. Кельмакову [22]
Table 1. The Proto-Udmurt language's system of consonants by V. K. Kelmakov [22]

Способ образования			Место образования						
			Губно-губные	Губно-зубные	Зубные	Альвеолярные	Палатальные	Средненебные	Заднебные
Шумные	Смычные	Глухие	*p*		*t*		*t'*		*k*
		Звонкие	*b*		*d*		*d'*		*g*
	Щелевые	Глухие			*s*	*š*	*ś*		
		Звонкие		*v*	*z*	*ž*	*ź*		
	Аффрикаты	Глухие				*č*	*ć*		
		Звонкие				*ǯ*	*ǵ*		
Сонорные	Щелевые	Серединные	*u̯*					*j*	
		Боковые			*l*		*l'*		
	Смычно-проходные носовые		*m*		*n*		*ń*		*ŋ*
	Дрожащий				*r*				

В системе согласных говора д. Бобья-Уча, в котором присутствуют 26 единиц из 28, отсутствуют лишь «экзотические» фонемы *u̯* и *ŋ*, ныне утраченные абсолютным большинством удмуртских диалектов.

В результате обработки материала двадцати четырех словарей с помощью программы «Анализ когнатов в разных диалектах одного языка» были выявлены следующие ряды соответствий для согласных фонем (табл. 2).

Исходя из результатов можно увидеть, что большинство рефлексов анлаутных согласных совпадают во всех двадцати четырех словарях. Инлаутные согласные в д. Бобья-Уча также совпадают с другими диалектами удмуртского языка. Неотмеченные в табл. 1 фонемы *ź, t', d'* чаще всего встречаются в середине слова, например, южБУ. I *adʲami* – лит. удм. *адями* ‘человек’, южБУ. II *kəzpu* – лит. удм. *кызьпу* ‘береза’, южБУ. I *batʲpu* – лит. удм. *бадьпу* ‘ива’. Фонемы *r, ž* встречаются в начале слова, но система не выделила их в надежных рядах согласных, поскольку слов с данными рефлексами в словарях удмуртских диалектов практически не имеются.

Табл. 2. Рефлексы анлаутных согласных в удмуртских диалектах
Table 2. Reflexes of anlaut consonants in Udmurt dialects

ПУдм	Северное наречие				срединные говоры					южное наречие												бесермянское наречие			архШ.
										юж.						пю.									
	севУЛ.	севЗМ.	ЗБ.	счК.	средЗ.	Бук.	средБ.	средНК.	средВ.	южБУ. I	южБУ. II	южЮ.	южЛВ.	южСФ.	южВБ.	Тих.	пюСВ.	пюСЮ.	пюТих.	бавлПУ.	бесШ.	бесВ.	бесП.	
**m*[1]	m	m	m	m	m	m	m	m	m	**m**	**m**	m	m	m	m	m	m	m	m	m	m	m	m	m	m
**n*	n	n	n	n	n	n	n	n	n	**n**	**n**	n	n	n	n	n	n	n	n	n	n	n	n	n	n
**p*	p	p	p	p	p	p	p	p	p	**p**	**p**	p	p	p	p	p	p	p	p	p	p	p	p	p	p
**b*	b	b	b	b	b	b	b	b	b	**b**	**b**	b	b	b	b	b	b	b	b	b	b	b	b	b	b
**t*	t	t	t	t	t	t	t	t	t	**t**	**t**	t	t	t	t	t	t	t	t	t	t	t	t	t	t
**d*	d	d	d	d	d	d	d	d	d	**d**	**d**	d	d	d	d	d	d	d	d	d	d	d	d	d	d
**k*	k	k	k	k	k	k	k	k	k	**k**	**k**	k	k	k	k	k	k	k	k	k	k	k	k	k	k
**g*	g	g	g	g	g	g	g	g	g	**g**	**g**	g	g	g	g	g	g	g	g	g	g	g	g	g	g
**ǯ*	ʤ	ʤ	?[2]	ʤ	ʤ	?	?	ʤ	ʤ	**ʤ**	**ʤ**	ʤ	ʤ	?	ʤ	?	ʤ	ʤ	?	ʤ	ʤ	ʤ	?	?
**s*	s	s	s	s	s	s	s	s	s	**s**	**s**	s	s	s	s	s	s	s	s	s	s	s	s	s	s
**z*	z	z	z	z	z	z	?	z	z	**z**	**z**	z	z	z	z	z	z	z	z	z	z	z	z	z	z
**v*	v	v	v	v	v	v	v	v	v	**v**	**v**	v	v	v	v	v	v	v	v	v	v	v	v	v	v
**l*	l	l	l	l	l	l	l	l	l	**l**	**l**	l	l	l	l	l	l	l	l	l	l	l	l	l	l
**l'*	lʲ	lʲ	lʲ	lʲ	lʲ	?	0	lʲ	lʲ	**lʲ**	**lʲ**	lʲ	lʲ	?	lʲ	lʲ	lʲ	lʲ	lʲ	lʲ	lʲ	lʲ	?	?
** ń*	nʲ	nʲ	?	nʲ	nʲ	?	?	nʲ	nʲ	**nʲ**	**nʲ**	nʲ	nʲ	?	nʲ	0	nʲ	nʲ	nʲ	nʲ	nʲ	nʲ	n	nʲ
**č*	ʧ	ʧ	ʧ	ʧ	ʧ	?	ts	ʦ	?	**ʧ**	**ʧ**	ʧ	ʧ	ʦ	ʧ	ʧ	ʦ	ʦ	ʦ	ʦ	ʦ	ʧ	?	?

[1] Здесь и далее праудмуртские согласные даны по В. К. Кельмакову [22].

[2] Знак вопроса «?» здесь и далее подразумевает отсутствие или недостаточное количество слов для точного определения программой, начинающихся с той или иной фонемы.

*ǯ	ʤ	ʤ	ʤ	ʤ	ʤ	?	?	ʤ	ʤ	**ʤ**	**ʤ**	ʤ	ʤ	?	ʤ	ʒ	ʐ	ʐ	?	ʤ	ʥ	ʥ	?	?
*ć	ʨ	ʨ	ʨ	ʨ	ʨ	?	ts	ʨ	ʨ	**ʨ**	**ʨ**	ʨ	ʨ	ʨ	ʨ	ʧ	ʨ	ʨ	ʨ	ʨ	ʨ	ʨ	ts	?
*š	ʂ	ʂ	ʃ	ʂ	ʂ	ʃ	ʃ	ʂ	ʂ	**ʂ**	**ʂ**	ʂ	ʂ	ʃ	ʂ	ʃ	ʂ	ʂ	ʃ	ʂ	ʃ	ʂ	ʃ	ʃ
*ś	ɕ	s	s	ɕ	ɕ	?	ç	ɕ	s^{j}	**ɕ**	**ɕ**	?	ɕ	s^{j}	ɕ	ç	ɕ	ɕ	s^{j}	ɕ	ɕ	ɕ	s	s
*j	j	j	j	j	j	?	j	j	j	**j**	**j**	j	j	j	j	0	ʥ	d^{j}	d^{j}	d^{j}	j	j	j	j

Таким образом, можно утверждать, что согласные фонемы исследуемого говора не подверглись каким-либо изменениям и не имеют диалектных особенностей. Здесь необходимо отметить специфический звук **’**, который встречается в говоре д. Бобья-Уча, например, в таких словах, как южБУ. I *ba’zən* – лит. удм. *бадӟым* ‘большой’, южБУ. I *ko’ʥɛm* – лит. удм. *кудӟем* ‘пьяный’. Однако данное явление требует дальнейшего, более подробного изучения.

Система вокализма в литературном удмуртском языке представлена семью гласными фонемами: *о, у, и, ы, э, а, ӧ*. Профессор В. К. Кельмаков в своей работе реконструировал для праудмуртского языка вокалическую систему в девять единиц [22] (табл. 3).

Табл. 3. Система вокализма праудмуртского языка
Table 3. The Proto-Udmurt’s system of vowels

Ряд / Подъем	Передний	Средний	Средне-задний	Задний
Верхний	*i*	*i̇*	*i̯*	*u*
Средний	*e*	*ȯ*	*e̯*	*o*
Нижний		*a*		

Необходимо отметить, что существует более современное исследование по изучению праудмуртской системы гласных первого слога. Лингвист М. П. Безенова предприняла попытку реконструкции праудмуртского вокализма, которая была верифицирована на полном материале исконной лексики в пяти говорах удмуртского языка, принадлежащих к разным диалектным группам [23] (табл. 4).

Табл. 4. Система вокализма праудмуртского языка первого слога
Table 4. The Proto-Udmurt’s system of vowels of the first syllable

Ряд / Подъем	Передний	Средний	Задний
Верхний	**i*	**ï*, **ʉ*	**u*
Средне-верхний		**ɵ*	
Средне-нижний	**ɛ*	**ɞ*	**ɔ*
Нижний 1[3]		**ɐ*	
Нижний 2		**a*	

[3] Разбиение нижнего подъема на «Нижний 1» и «Нижний 2» объясняется тем, что реконструируемая гласная **ɐ* по характеру подъема находится между гласными средне-нижнего (**ɛ*, **ɞ*, **ɔ*) и собственно нижнего (**a*) подъема. В МФА отдельного подъема для этой гласной не выделено [23].

Однако в абсолютном большинстве удмуртских диалектов система вокализма подверглась изменениям. Например, бавлинский говор, описанный И. В. Тараканoвым [24], представлен девятью гласными рефлексами (см. табл. 5).

В исследуемом нами говоре выделяются семь гласных фонем (*i, a, ə, ɛ, ɔ, u, ɜ*), а также дифтонг *ua*, которые сопоставлены с другими рефлексами гласных в удмуртских диалектах (табл. 6 и 7).

Табл. 5. Система вокализма бавлинского говора удмуртского языка
Table 5. The system of vowels of the Bavly dialect of the Udmurt language

Ряд / Подъем	Лабиальность	Передний	Средний	Средне-задний	Задний
Верхний	лабиализованный		*ʉ̈*		*u*
	нелабиализованный	*i*			
Средний	лабиализованный		*ö̜*		*o*
	нелабиализованный	*e*		*ə̂*	
Нижний	лабиализованный				
	нелабиализованный	[*ä*]			*a*

Табл. 6. Рефлексы анлаутных гласных в удмуртских диалектах
Table 6. Reflexes of anlaut vowels in Udmurt dialects

ПУдм	Северное наречие				срединные говоры					южное наречие											бесермянское наречие			архШ.
										юж.						пю.								
	севУЛ.	севЗМ.	ЗБ.	счК.	средЗ.	Бук.	средБ.	средНК.	средВ.	южБУ. I	южБУ. II	южЮ.	южЛВ.	южСФ.	южВБ.	Тих.	пюСВ.	пюСЮ.	пюТих.	бавлПУ.	бесШ.	бесВ.	бесП.	
**i*[4]	?	i	?	i	i	?	i	i	?	i	i	?	i	i	i	i	i	i	i	i	i	?	?	?
**u*	?	u	u	u	u	?	u	u	u	u	u	?	u	u	u	u	u	u	u	?	u	u	?	?
**a*	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a
**ï*	?	ï	?	ï	ï	?	?	ï	ï	ə	ə	?	?	?	ɐ	?	ï	ɐ	?	ə	?	ɐ	?	?
**ɛ*	ɛ	ɛ	e	?	ɛ	?	?	ɛ	ɛ	ɛ	ɛ	?	ɛ	e	ɛ	e	ɛ	ɛ	e	ɛ	e	?	?	e
**ɔ*	ɔ	ɔ	o	ɔ	ɔ	?	o	ɔ	ɔ	ɔ	ɔ	ɔ	ɔ	o	ɔ	o	ɔ	ɔ	o	ɔ	o	ɔ	o	o

Этимологический анализ удмуртских словарей позволил выявить инновационные особенности гласных для говора д. Бобья-Уча. Праудмуртская **ï* реализуется в аллофоне *ə*. Это подтверждают следующие слова, использующиеся в речи носителя исследуемого говора, например: южБУ. *əmdor* – лит. удм. *ымдур* ‘губа’, южБУ. *əm* – лит. удм. *ым* ‘рот’, южБУ. *ərgɔn* – лит. удм. *ыргон* ‘медь’. Аналогичное изменение **ï* > *ə* происходит и в словаре говора с. Покровский Урустамак Бавлинского района РТ, принадлежащего к периферийно-южному диалекту удмуртского языка, например, бавлПУ. *əʂkɐnɐ* - лит. удм. *ышкыны* ‘дергать’, бавлПУ. *əʐ* – лит. удм. *ыж* ‘овца’.

Что касается инлаутных гласных, то в говоре д. Бобья-Уча выделяются следующие инновационные переходы:

[4] Здесь и далее праудмуртские гласные даны согласно М. П. Безеновой [23].

1) **ï > ə*: южБУ. *kəzənə* – лит. удм. *кызыны* ‘кашлять’, южБУ. *pədʲɛs* – лит. удм. *пыдес* ‘колено’, южБУ. *mənənə* – лит. удм. *мыныны* ‘идти’и др. Данный переход встречается в словаре говора с. Покровский Урустамак Бавлинского района РТ в таких словах как бавлПУ. *d'ərjɐnɐ́* – лит. удм *йырйыны* ‘грызть’, бавлПУ. *kərnəʐ* - лит. удм. *кырныж* ‘ворон’ и т.д.

2) **ɐ > ə*: южБУ. *kəlʲlʲənə* – лит. удм. *кылльыны* ‘лежать’, южБУ. *nərkənə* – лит. удм. *ныркыны* ‘месить тесто’, южБУ. *ʯən* – лит. удм. *ӵын* ‘дым’ и др. Аналогичная тенденция наблюдается также в словаре говора с. Покровский Урустамак Бавлинского района, например, бавлПУ. *təpɐ́* – лит. удм. *тыпы* ‘дуб’, бавлПУ. *ləmə́* – лит. удм. *лымы* ‘снег’ и т.д.

3) **ɜ > з*: южБУ. II *sɜrənə* – лит. удм. *сӧрыны* ‘разрушить’, южБУ. *kɜm* – лит. удм. *кӧм* ‘корка, кожура’. Этот переход также встречается в словарях говоров д. Зюзино Шарканского района и д. Новая Казмаска Завьяловского района УР, принадлежащих к срединным говорам (средЗ. *lɜdïnï* – лит. удм. *лӧдыны* ‘теребить шерсть для шерстобитки’, средНК. *sɜrïnï* – лит. удм. *сӧрыны* ‘разбить’), говора д. Лолошур-Возжи Граховского района УР центрально-южного диалекта удмуртского языка (южЛВ. *kɜm* – лит. удм. *кӧм* ‘кожура’), говора с. Старый Варяш Янаульского района РБ периферийно-южного диалекта удмуртского языка (пюСВ. *kɜ* – лит. удм. *кӧ* ‘жернов’), говоров д. Шамардан Юкаменского района УР и д. Ворца Ярского района УР, принадлежащих к бесермянскому наречию удмуртского языка (бесШ. *lɜʦɛʁt* – лит. удм. *лэчыт* ‘острый’, бесВ. *pɜɛ ɛgərjɔ́s* – лит. удм. *пӧсь эгыръёс* ‘раскаленные угли’).

Табл. 7. Рефлексы гласных первого слога (после согласного) в удмуртских диалектах
Table 7. Reflexes of vowels of the first syllable (after a consonant) in Udmurt dialects

ПУдм	Северное наречие				срединные говоры					южное наречие											бесермянское наречие			архШ.
										юж.						пю.								
	севУЛ.	сев3М.	ЗБ.	счК.	средЗ.	Бук.	средБ.	средНК.	средВ.	южБУ. I	южБУ. II	южЮ.	южЛВ.	южСФ.	южВБ.	Тих.	пюСВ.	пюСЮ.	пюТих.	бавлПУ.	бесШ.	бесВ.	бесП.	
**i*	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i
**u*	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u
**a*	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a
**ï*	ï	ï	ï	ï	ï	ɨ	ɨ	ï	ï	ə	ə	ɐ	ï	ï	ï	ɨ	ï	ɐ	ï	ə	ɐ	ï	ɨ	ï
**ɐ*	?	ï	ï	ï	ï	ɨ	ɨ	ï	ï	ə	ə	ɐ	ï	ï	ï	ɨ̆	ï	ɐ	ï	ə	ɐ	ɐ	ɨ	ï
**ʉ*	?	u	?	u	u	?	?	u	u	?	?	u	u	e	u	?	ï	ʉ	e	ʉ	ɐ	ɐ	?	?
**ɛ*	ɛ	ɛ	e	ɛ	ɛ	?	e	ɛ	ɛ	ɛ	ɛ	?	ɛ	e	ɛ	e	ɛ	ɛ	e	ɛ	e	ɛ	e	e
**ɜ*	з	ï	з	з	з	?	?	з	з	з	з	?	з	з	з	?	з	ɵ	з	ɵ	з	з	?	e
**ɵ*	?	ï	?	ɵ	з	i	?	з	ɵ	з	з	?	з	?	ɵ	?	з	ɵ	?	ɵ	з	з	?	?
**ɔ*	ɔ	ɔ	o	ɔ	ɔ	o	o	ɔ	ɔ	ɔ	ɔ	ɔ	ɔ	o	ɔ	o	ɔ	ɔ	o	ɔ	o	ɔ	o	o
**ua*	wa	wa	ua	wa	ua	?	?	wa	wa	ua	ua	ua	wa	?	wa	?	wa	wa	?	wa	wa	wa	ua	va

Употребление в словах южБУ. I *pïnə* – лит. удм. *пуны* ‘собака’, южБУ. I *tïʂmɔn* – лит. удм. *тушмон* ‘враг’ фонемы *ï* вместо *и* предполагает восхождение к праудмуртскому **и̇*.

Аналогичное развитие указанной фонемы (потеря огубленности) произошло и в некоторых других диалектах, например в закамских говорах, в бесермянском, отчасти в кукморском говоре [22]. Причина такого развития пока не нашла удовлетворительного толкования.

Изменение *u* > *o* (южБУ. *oʒ* – лит. удм. *уж* ‘работа’, южБУ. *ozə* – лит. удм. *узы* ‘земляника’, южБУ. *ozər* – лит. удм. *узыр* ‘богатый’, южБУ. *otɔj* – лит. удм. *умой* ‘хороший’, южБУ. *ʂor* – лит. удм. *шур* ‘река’, южБУ. *toj* – лит. удм. *туй* ‘береста’, южБУ. *vonɛtənə* – лит. удм. *вунэтыны* ‘забыть’) указывает на то, что *o* в международной транскрипции обозначает открытую фонему *u*, каким он и является в исследуемом говоре, т. е. данная инновация не является переходом. По поводу перехода огубленного **ɐ* в неогубленное *ɜ* следует обратиться к трудам В. К. Кельмакова, исследовавшего эти фонемы более подробно (см. [22, 25, 26]). Что касается инновационных переходов **ï* > *ə* и **ɐ* > *ə*, то здесь фонема *ə* в говоре д. Бобья-Уча является вариантом фонемы *ï*. На это указывают исследования удмуртских ученых С. А. Максимова, В. К. Кельмакова, Л. Л. Карповой [1, 22, 27]. Однако все эти указанные изменения требуют дальнейших подробных исследований в других научных изысканиях.

Таким образом, на графике этимологико-фонетической близости говор д. Бобья-Уча ближе к говору д. Юмьяшур Алнашского района УР, который принадлежит к центрально-южному диалекту удмуртского языка (рис. 2).

3.2 Глоттохронологический анализ

Получив баланс между фонетическими изоглоссами и глоттохронологическим анализом, можно оценить степень близости между языками. На платформе ЛингвоДок, в разделе «Инструменты», пользователь имеет доступ к функции «Глоттохронологический анализ диалектов». В рамках этой функции используется формула С. А. Старостина для анализа базисной лексики каждого созданного или получившего права пользователем словаря. Результаты анализа представлены в таблице, включающей результаты формулы и процент родственных слов в списке, а также график близости в 2D и 3D форматах.

На сегодняшний день на базе платформы ЛингвоДок имеются семнадцать 100-словных словарей по следующим диалектам удмуртского языка:

- *северное наречие*: 1. Словарь д. Удмурт Лем, северное наречие удмуртского языка, верхнечепецкий говор; 2. Словарь по памятнику письменности «Закон Божий: Книжка с картинками для маленьких детей. На вотском языке глазовского наречия», 1912; 3. Словарь среднечепецкого диалекта северного наречия удмуртского языка (говор д. Кабаково Глазовского района УР), 2013;
- *срединные говоры*: 4. Словарь срединного диалекта удмуртского языка д. Зюзино Шарканского района; 5. Словарь срединного говора удмуртского языка (говор д. Новая Казмаска Завьяловского района УР); 6. Словарь срединного говора удмуртского языка (говор д. Вылынгурт Сюмсинского района УР); 7. Словарь, собранный П.С. Палласом в XVIII в., в настоящее время хранящийся в архиве РАН СПб, в фонде А. Шёгрена;
- *южное наречие*:
  - *центрально-южное наречие*: 8. Словарь центрально-южного диалекта удмуртского языка (говор д. Бобья-Уча Малопургинского района УР, информант: Е. С. Яковлева); 9. Словарь среднеюжных говоров центрально-южного диалекта удмуртского языка (говор д. Бобья-Уча Малопургинского района УР, информант: О. В. Иванова); 10. Словарь центрально-южного диалекта говора д. Юмьяшур Алнашского района УР; 11. Словарь центрально-южного диалекта удмуртского языка (говор д. Лолошур-Возжи Граховского района УР); 12. Словарь центрально-южного диалекта удмуртского языка (говор с. Варклед-Бодья Агрызского района РТ), 2013;

- *периферийно-южное наречие*: 13. Словарь буйско-таныпского говора периферийно-южного диалекта удмуртского языка (говор с. Старый Варяш Янаульского района РБ); 14. Словарь кукморского говора периферийно-южного диалекта удмуртского языка (говор д. Старая Юмья Кукморского района РТ); 15. Словарь бавлинского говора периферийно-южного диалекта удмуртского языка (говор с. Покровский Урустамак Бавлинского района РТ);

- *бесермянское наречие*: 16. Словарь бесермянского диалекта удмуртского языка (говор д. Шамардан Юкаменского района УР), 2003–2005, 2009–2012, 2013; 17. Словарь бесермянского наречия удмуртского языка (говор д. Ворца Ярского района УР), 2013.

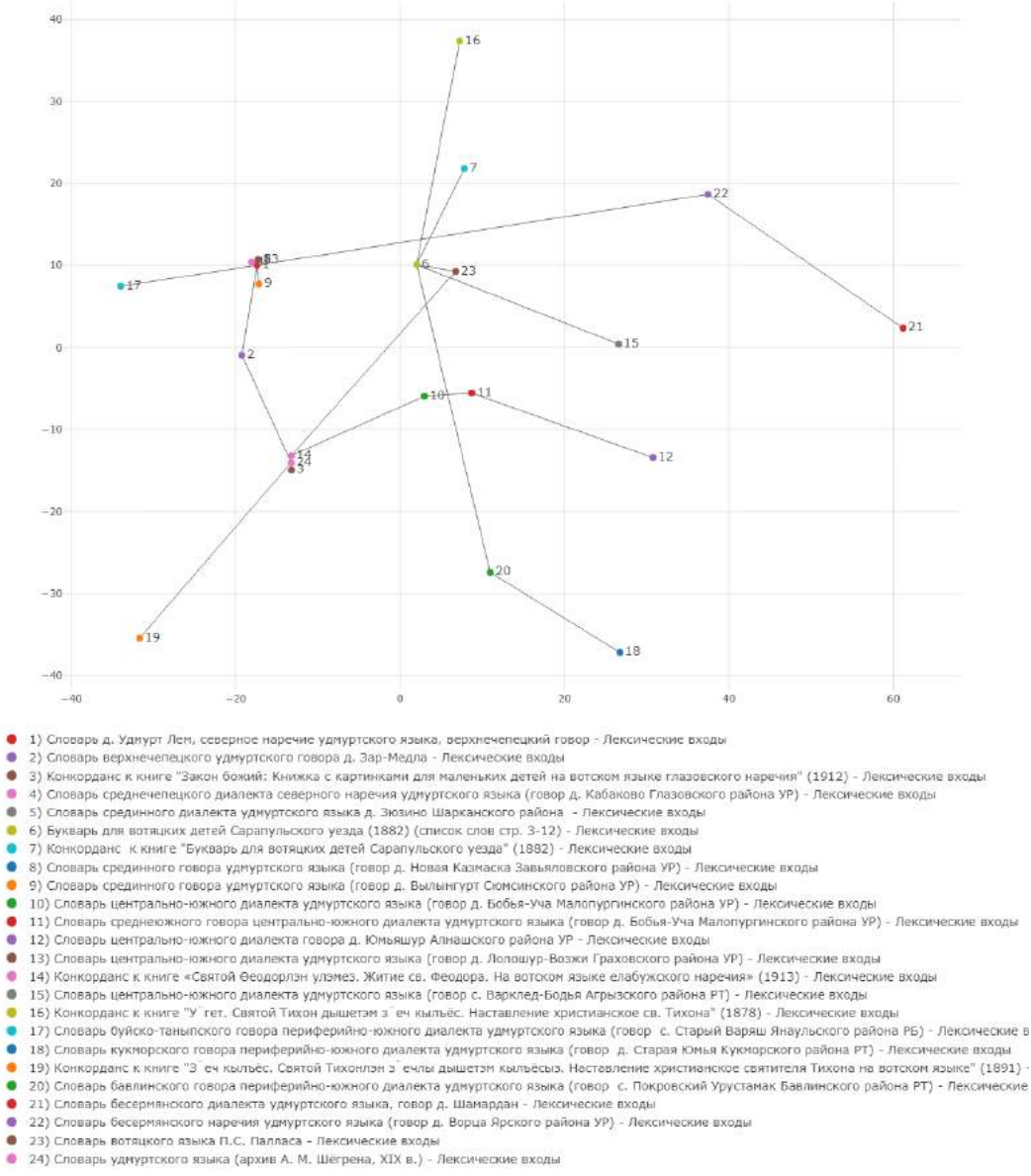


Рис. 2. График фонетико-этимологической близости диалектов удмуртского языка
Fig. 2. Graph of phonetic and etymological proximity of dialects of the Udmurt language

В результате обсчета был получен следующий результат близости рассмотренных идиом (табл. 8). График времени распада рассматриваемых идиомов показан на рис. 3.

Табл. 8. Время распада и процент общей лексики в диалектах удмуртского языка
Table 8. The time of decay and the percentage of common vocabulary in the dialects of the Udmurt language

	северное наречие		срединные говоры			южное наречие									Бесермянское наречие		17. архШ.
						юж.					пю.						
	1. ЗБ	2. счК.	3. средЗ.	4. средНК.	5. средВ.	6. южБУ. I	7. южБУ. II	8. южЮ.	9. южЛВ.	10. южВБ.	11. Тих.	12. пюСВ.	13. пюСЮ.	14. бавлПУ.	15. бесШ.	16. бесВ.	
1	n/a	0.67 (95%)	1.19 (87%)	0.94 (91%)	0.82 (93%)	1.21 (87%)	0.94 (91%)	1.08 (89%)	0.78 (94%)	0.83 (93%)	0.76 (94%)	1.00 (90%)	0.97 (91%)	1.10 (89%)	0.87 (93%)	0.83 (93%)	1.22 (87%)
2	0.67 (95%)	n/a	1.01 (90%)	0.41 (98%)	0.73 (95%)	1.03 (90%)	0.93 (92%)	0.92 (92%)	0.58 (96%)	0.63 (96%)	0.96 (91%)	0.73 (94%)	0.83 (93%)	0.82 (93%)	0.80 (94%)	0.62 (96%)	0.98 (91%)
3	1.19 (87%)	1.01 (90%)	n/a	0.41 (98%)	0.68 (95%)	1.26 (86%)	0.98 (91%)	0.81 (93%)	0.73 (95%)	0.90 (92%)	1.37 (84%)	0.99 (91%)	1.06 (89%)	1.15 (88%)	1.13 (88%)	0.90 (92%)	1.53 (80%)
4	0.94 (91%)	0.41 (98%)	0.41 (98%)	n/a	0.40 (98%)	0.84 (93%)	0.71 (95%)	0.74 (94%)	0.40 (98%)	0.58 (96%)	0.97 (91%)	0.86 (93%)	0.85 (93%)	0.83 (93%)	0.61 (96%)	0.40 (98%)	0.93 (92%)
5	0.82 (93%)	0.73 (95%)	0.68 (95%)	0.40 (98%)	n/a	0.85 (93%)	0.63 (95%)	0.56 (96%)	-0.00 (100%)	0.35 (98%)	1.21 (87%)	0.85 (93%)	0.71 (95%)	0.81 (93%)	0.88 (92%)	0.89 (92%)	1.75 (76%)
6	1.21 (87%)	1.03 (90%)	1.26 (86%)	0.84 (93%)	0.85 (93%)	n/a	0.80 (94%)	1.13 (88%)	0.73 (95%)	0.76 (94%)	1.22 (86%)	1.18 (87%)	0.94 (91%)	1.01 (90%)	0.97 (91%)	1.01 (90%)	1.50 (81%)
7	0.94 (91%)	0.93 (92%)	0.98 (91%)	0.71 (95%)	0.63 (96%)	0.80 (94%)	n/a	0.81 (93%)	0.58 (96%)	0.64 (96%)	1.43 (82%)	1.06 (89%)	0.82 (93%)	0.91 (92%)	1.11 (89%)	0.83 (93%)	1.58 (80%)
8	1.08 (89%)	0.92 (92%)	0.81 (93%)	0.74 (94%)	0.56 (96%)	1.13 (88%)	0.81 (93%)	n/a	0.42 (98%)	0.70 (95%)	1.50 (81%)	0.90 (92%)	0.86 (93%)	1.01 (90%)	0.95 (91%)	0.92 (92%)	1.41 (83%)
9	0.78 (94%)	0.58 (96%)	0.73 (95%)	0.40 (98%)	-0.00 (100%)	0.73 (95%)	0.58 (96%)	0.42 (98%)	n/a	-0.00 (100%)	0.55 (97%)	0.86 (93%)	0.72 (95%)	0.40 (98%)	0.60 (96%)	0.58 (96%)	1.08 (89%)
10	0.83 (93%)	0.63 (96%)	0.90 (92%)	0.58 (96%)	0.35 (98%)	0.76 (94%)	0.64 (96%)	0.70 (95%)	-0.00 (100%)	n/a	0.98 (91%)	0.85 (93%)	0.71 (95%)	0.50 (97%)	0.89 (92%)	0.82 (93%)	1.66 (78%)
11	0.76 (94%)	0.96 (91%)	1.37 (84%)	0.97 (91%)	1.21 (87%)	1.22 (86%)	1.43 (82%)	1.50 (81%)	0.55 (97%)	0.98 (91%)	n/a	1.19 (87%)	1.14 (88%)	0.83 (93%)	0.93 (92%)	1.08 (89%)	1.17 (88%)
12	1.00 (90%)	0.73 (94%)	0.99 (91%)	0.86 (93%)	0.85 (93%)	1.18 (87%)	1.06 (89%)	0.90 (92%)	0.86 (93%)	0.85 (93%)	1.19 (87%)	n/a	0.85 (93%)	0.85 (93%)	0.77 (94%)	0.85 (93%)	1.14 (88%)
13	0.97 (91%)	0.83 (93%)	1.06 (89%)	0.85 (93%)	0.71 (95%)	0.94 (91%)	0.82 (93%)	0.86 (93%)	0.72 (95%)	0.71 (95%)	1.14 (88%)	0.85 (93%)	n/a	0.70 (95%)	0.86 (93%)	0.93 (92%)	1.08 (89%)
14	1.10 (89%)	0.82 (93%)	1.15 (88%)	0.83 (93%)	0.81 (93%)	1.01 (90%)	0.91 (92%)	1.01 (90%)	0.40 (98%)	0.50 (97%)	0.83 (93%)	0.85 (93%)	0.70 (95%)	n/a	1.14 (88%)	0.97 (91%)	1.54 (80%)
15	0.87 (93%)	0.80 (94%)	1.13 (88%)	0.61 (96%)	0.88 (92%)	0.97 (91%)	1.11 (89%)	0.95 (91%)	0.60 (96%)	0.89 (92%)	0.93 (92%)	0.77 (94%)	0.86 (93%)	1.14 (88%)	n/a	0.78 (94%)	1.15 (88%)
16	0.83 (93%)	0.62 (96%)	0.90 (92%)	0.40 (98%)	0.89 (92%)	1.01 (90%)	0.83 (93%)	0.92 (92%)	0.58 (96%)	0.82 (93%)	1.08 (89%)	0.85 (93%)	0.93 (92%)	0.97 (91%)	0.78 (94%)	n/a	1.32 (85%)
17	1.22 (87%)	0.98 (91%)	1.53 (80%)	0.93 (92%)	1.75 (76%)	1.50 (81%)	1.58 (80%)	1.41 (83%)	1.08 (89%)	1.66 (78%)	1.17 (88%)	1.14 (88%)	1.08 (89%)	1.54 (80%)	1.15 (88%)	1.32 (85%)	n/a

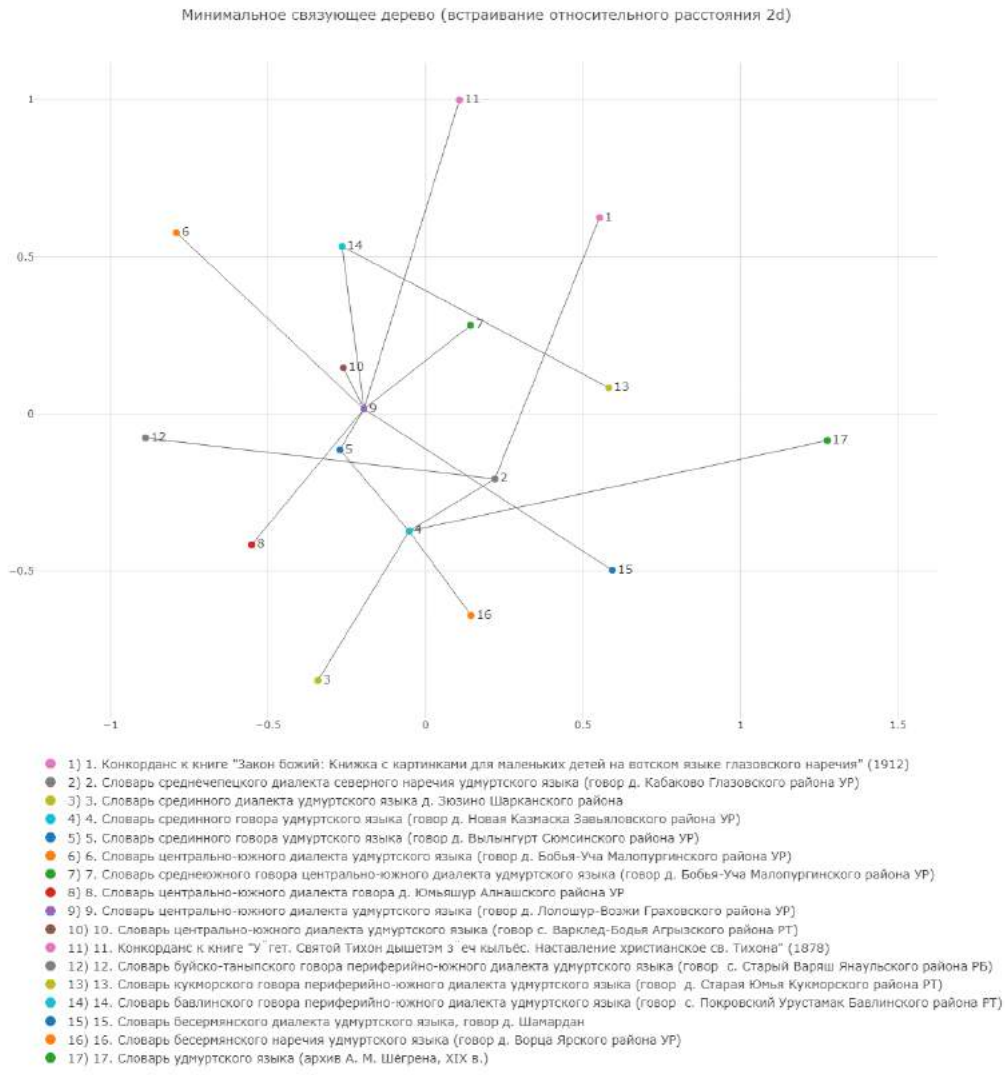


Рис. 3. График времени распада диалектов удмуртского языка
Fig. 3. Timeline of the collapse of dialects of the Udmurt language

4. Заключение

На сегодняшний день благодаря новым компьютерным технологиям и, в частности, уникальным программам на платформе ЛингвоДок появляются возможности быстрого воспроизведения экспериментально-фонетической, этимологической и морфологической работы исследователя.

Результаты фонетико-этимологического анализа говора д. Бобья-Уча Малопургинского района УР показали, что система консонантизма не подверглась каким-либо изменениям и состоит из 26 единиц. Однако в вокалической системе встречаются инновационные переходы как анлаутных – **ӥ* > *ə*, так и инлаутных гласных – **ӥ* > *ə*, **ʙ* > *ə*, **ɞ* > з. Таким образом, фонема *ə* в говоре д. Бобья-Уча является вариантом фонемы *ӥ*, а переход огубленного **ɞ* в неогубленное *з* требует более подробного исследования. Употребление фонемы *ӥ* вместо *и*

предполагает восхождение к праудмуртскому **и*. Исходя из графика и результатов анализа времени распада и процента общей лексики в диалектах удмуртского языка, мы видим, что списки базисной лексики исследуемого говора, говора д. Лолошур-Возжи Граховского района УР и говора с. Варклед-Бодья Агрызского района РТ совпадают на 96%, что по формуле С. А. Старостина указывает на время распада 580 и 640 лет назад соответственно. Однако вывод вызывает вопросы с исторической точки зрения, поскольку указанные населенные пункты появились не ранее 300–400 лет назад. По мнению М. Г. Атаманова д. Бобья-Уча была образована в 1650-е гг. [28], тем не менее, дата создания деревни не имеет прямого отношения к распаду двух говоров. В частности, потому, что носители могли проживать какое-то время в разных местах до прихода на территорию нынешнего проживания. Возможно, методика исследования требует усовершенствования, так как кроме собственно лингвистических процессов, необходимо принимать в расчет иные факторы, в частности, исторические. Мы можем утверждать, что полученные данные указывают лишь на принадлежность исследуемого говора к центрально-южному диалекту южного наречия. Исследование, проведенное в данной статье, требует дальнейшего продолжения с целью уточнения и дополнения результатов, поскольку выявлены не все случаи изменения звуков в речевом потоке, а также спорадические изменения.

Список использованных сокращений

архШ.	Словарь, собранный П.С. Палласом в XVIII в., в настоящее время хранящийся в архиве РАН СПб, в фонде А. Шёгрена
бавлПУ.	Словарь бавлинского говора периферийно-южного диалекта удмуртского языка (говор с. Покровский Урустамак Бавлинского района РТ), 2013
бесВ.	Словарь бесермянского наречия удмуртского языка (говор д. Ворца Ярского района УР), 2013
бесП.	Словарь вотяцкого языка П.С. Палласа
бесШ.	Словарь бесермянского диалекта удмуртского языка (говор д. Шамардан Юкаменского района УР), 2003–2005, 2009–2012, 2013
Бук.	Букварь для вотяцких детей Сарапульского уезда, 1882
ЗБ	Словарь по памятнику письменности «Закон Божий: Книжка с картинками для маленьких детей. На вотском языке глазовского наречия», 1912
лит. удм.	литературный удмуртский язык
ПУдм.	праудмуртский язык
пю.	периферийно-южный диалект
пюСВ.	Словарь буйско-таныпского говора периферийно-южного диалекта удмуртского языка (говор с. Старый Варяш Янаульского района РБ)
пюСЮ.	Словарь кукморского говора периферийно-южного диалекта удмуртского языка (говор д. Старая Юмья Кукморского района РТ)
пюТих.	Словарь по памятнику письменности «Ӟеч кылъёс. Святой Тихонлэн ӟечлы дышетэм кылъёсыз. Наставление христианское святителя Тихона на вотском языке», 1891
РБ	Республика Башкортостан
РТ	Республика Татарстан
севЗМ.	Словарь верхнечепецкого удмуртского говора д. Зар-Медла
севУЛ.	Словарь д. Удмурт Лем, северное наречие удмуртского языка, верхнечепецкий говор
средБ.	Словарь по памятнику письменности «Букварь для вотяцких детей Сарапульского уезда», 1882
средВ.	Словарь срединного говора срединных говоров удмуртского языка (говор д. Вылынгурт Сюмсинского района УР), 2013
средЗ.	Словарь срединного диалекта удмуртского языка д. Зюзино Шарканского района
средНК.	Словарь срединного говора удмуртского языка (говор д. Новая Казмаска Завьяловского района УР)
счК.	Словарь среднечепецкого диалекта северного наречия удмуртского языка (говор д. Кабаково Глазовского района УР), 2013
Тих.	Словарь по памятнику письменности «Наставление христианское святителя Тихона на

	вотском языке», 1891
УР	Удмуртская Республика
юж.	центрально-южный диалект
южБУ. I	Словарь центрально-южного диалекта удмуртского языка (говор д. Бобья-Уча Малопургинского района УР), 2023 (информант: Е. С. Яковлева)
южБУ. II	Словарь среднеюжных говоров центрально-южного диалекта удмуртского языка (говор д. Бобья-Уча Малопургинского района УР), 2023 (информант: О. В. Иванова)
южВБ.	Словарь центрально-южного диалекта удмуртского языка (говор с. Варклед-Бодья Агрызского района РТ), 2013
южЛВ.	Словарь центрально-южного диалекта удмуртского языка (говор д. Лолошур-Возжи Граховского района УР)
южСФ.	Словарь по памятнику письменности «Святой Ѳеодорлэн улэмез. Житие св. Феодора. На вотском языке елабужского наречия»
южЮ.	Словарь центрально-южного диалекта говора д. Юмьяшур Алнашского района УР

Список литературы / References

[1]. Насибуллин Р. Ш., Максимов С. А., Семёнов В. Г., Отставнова Г. В. Диалектологический атлас удмуртского языка. Карты и комментарий. Выпуск I. Научное издание. Ижевск: НИЦ «Регулярная и хаотическая динамика», 2009, с. 40. / Nasibullin R. Sh., Maksimov S. A., Semenov V. G., Otstavnova G. V. Dialectological atlas of the Udmurt language. Maps and commentary, issue I. Scientific publication. Izhevsk, SIC Regular and chaotic dynamics, 2009, p. 40. (in Russian).

[2]. Wiedemann F. J. Zur Dialektenkunde der wotjakischen Sprache. Bulletin de la classe historico-philologique de l'Académie Impériale des sciences de Saint-Petersbourg. 1858, vol. XV, pp. 240–256.

[3]. Aminoff T. G. Wotjakilaisia kielinäytteitä. JSFOu. 1886, 1, pp. 32–55.

[4]. Aminoff T. G. Votjakin äänne-ja muoto-opin luonnos. JSFOu. 1896, 2 (XIV), pp. 1–48.

[5]. Munkácsi B. Votják népköltészeti hagyományok. Budapest, 1887, 335 p.

[6]. Munkácsi B. Volksbräuche und Volksdichtung der Wotjaken. Herausgegeben von D.R. Fuchs. Helsinki, 1952, 715 p.

[7]. Wichmann Y. Wotjakischer Wortschatz / Aufgezeichnet Wichmann Y. Bearb. Uotila T. E., Korhonen M. Hrsg. Korhonen M. Helsinki, 1987, 421 p. (= Lexica Societatis Fenno-Ugricae. 1987, vol. 21.)

[8]. Борисов Т. К. Удмурт кыллюкам. Удмуртско-русский толковый словарь. Ижевск, 1932, 383 с. / Borisov T. K. Udmurt-Russian explanatory Dictionary. Izhevsk, 1932, 383 p. (in Russian and Udmurt).

[9]. Жуйков С. П. Удмурт АССР-ысь удмуртъёслэн интыен-интыен вераськемзэс возьматон карта (Диалектологическая карта удмуртского языка). Свердловск, 1935. / Zhuikov S. P. Dialectological map of the Udmurt language. Sverdlovsk, 1935. (in Udmurt).

[10]. Насибуллин Р. Ш. Из лексики удмуртских народных говоров. Вопросы удмуртского языкознания: Сборник статей. Вып. 3. Ижевск, 1975, с. 106–124. / Nasibullin R. Sh. From the vocabulary of Udmurt folk dialects. Questions of Udmurt linguistics: A collection of articles. Issue 3. Izhevsk, 1975, pp. 106–124. (in Russian).

[11]. Тепляшина Т. И., Лыткин В. И. Пермские языки. Основы финно-угорского языкознания: Марийский, пермские, угорские языки. М., 1976, с. 97–228. / Teplyashina T. I., Lytkin V. I. Permian languages. Fundamentals of Finno-Ugric linguistics: Mari, Permian, Ugric languages. Moscow, 1976, pp. 97–228. (in Russian).

[12]. Кельмаков В. К. К вопросу о диалектном членении удмуртского языка. Пермистика: Вопросы диалектологии и истории пермских языков: Сб. ст. Ижевск, 1987, с. 26–51. / Kel'makov V. K. On the question of dialect division of the Udmurt language. Permistics: Questions of dialectology and history of Permian languages: Collection of articles. Izhevsk, 1987, pp. 26–51. (in Russian).

[13]. Максимов С. А. Комментарий к картам «Диалектное членение удмуртского языка» и «Принадлежность опорных пунктов к территориальным диалектам». Диалектологический атлас удмуртского языка: Карты и комментарии. Вып. I. Ижевск, 2009, с. 42–48. / Maksimov S. A. Commentary on the maps “Dialect division of the Udmurt language” and “The affiliation of strongholds to territorial dialects”. Dialectological atlas of the Udmurt language: Maps and comments. Issue I. Izhevsk, 2009, pp. 42–48. (in Russian).

[14]. Атаманов-Эграпи М. Г. Происхождение удмуртского народа: Монография. Ижевск, 2010, с. 302–306. / Atamanov-Egrapi M. G. The origin of the Udmurt people: A monograph. Izhevsk, 2010, pp. 302–306. (in Russian).

[15]. Архипов Г. А. Морфологические особенности среднеюжного диалекта удмуртского языка I. О диалектах и говорах южноудмуртского наречия: Сб. статей и материалов / НИИ при Сов. Мин. УАССР. Ижевск, 1978, с. 3–46. / Arkhipov G. A. Morphological features of the Middle Southern dialect of the Udmurt language I. About dialects and dialects of the South Udmurt dialect: Collection of articles and materials / Research Institute at the Soviet Ministry of UASSR. Izhevsk, 1978, pp. 3–46. (in Russian).
[16]. Архипов Г. А. Среднеюринский говор I. Материалы по удмуртской диалектологии: Образцы речи / НИИ при Сов. Мин. УАССР. Ижевск, 1981, с. 5–44. / Arkhipov G. A. Sredneyurinsky dialect I. Materials on Udmurt dialectology: Speech samples / Research Institute at the Soviet Ministry of the UASSR. Izhevsk, 1981, pp. 5–44. (in Russian).
[17]. Архипов Г. А. Некоторые вопросы фонетики среднеюжного диалекта удмуртского языка. Ученые записки Тартуского государственного университета. Вып. 117: Труды по филологии 1. Тарту, 1962, с. 190–198. / Arkhipov G. A Some Phonetic Questions of the Middle-Southern Udmurt Dialect. Scientific Notes of the Tartu State University. Issue 117: Proceedings on Philology 1. Tartu, 1962, pp. 190–198. (in Russian).
[18]. Старостин С. А. Труды по языкознанию. Москва: Языки славянских культур, 2007, 924 с. / Starostin S. A. Works on linguistics. Moscow, Languages of Slavic Cultures, 2007, 924 p. (in Russian).
[19]. Starostin G. Preliminary lexicostatistics as a basis for language classification: A new approach. Journal of Language Relationship, 2010, 3, pp. 79–116.
[20]. Kassian A., Starostin G., Dybo A., Chernov V. The Swadesh wordlist. An attempt at semantic specification. Journal of Language Relationship, 2010, 4, pp. 46–89.
[21]. Старостин Г. С. Африканские языки: обзор лексико-статистической классификации. Том I: Методология. Койсанские языки. Москва: Языки славянских культур, 2013, 510 с. / Starostin G. S. Languages of Africa: an attempt at a lexicostatistical classification. Vol. I: Methodology. Khoisan languages. Moscow, Languages of Slavic Culture, 2013, 510 p. (in Russian).
[22]. Кельмаков В. К. Формирование и развитие фонетики удмуртских диалектов. Ижевск, 1993, 58 с. / Kel'makov V. K. Formation and development of phonetics of Udmurt dialects. Izhevsk, 1993, 58 p. (in Russian).
[23]. Безенова М. П. Удмуртский вокализм первого слога в историческом аспекте (на материале памятников письменности и современных диалектов). Дисс. … канд. филол. наук. М., 2014, 288 с. / Bezenova M. P. Udmurt vocalism of the first syllable in the historical aspect (based on the material of monuments of writing and modern dialects). Diss. ... Ph.D. at Philology. Moscow, 2014, 288 p. (in Russian).
[24]. Тараканов И. В. Фонетические особенности бавлинского диалекта удмуртского языка (в свете экспериментальных данных). Дисс. … канд. филол. наук. Тарту, 1958, 312 с. / Tarakanov I. V. Phonetic features of the Bavly dialect of the Udmurt language (in the light of experimental data). Diss. ... Candidate of Philological sciences. Tartu, 1958, 312 p. (in Russian).
[25]. Кельмаков В. К. Диалектная и историческая фонетика удмуртского языка. Ч. 1: учебное пособие для студентов педвузов, обучающихся по специальности «Родной язык и литература». Ижевск: Удмуртский университет, 2003. 276 с. / Kel'makov V. K. Dialect and historical phonetics of the Udmurt language. Part 1: textbook for students of pedagogical universities studying in the specialty “Native language and literature”. Izhevsk, Udmurt University, 2003, 276 p. (in Russian).
[26]. Кельмаков В. К. Диалектная и историческая фонетика удмуртского языка. Ч. 2. Ижевск: Удмурт. ун-т, 2004. 394 с. / Kel'makov V. K. Dialect and historical phonetics of the Udmurt language. Part 2. Izhevsk, Udmurt University, 2004, 394 p. (in Russian).
[27]. Карпова Л. Л. Среднечепецкий диалект удмуртского языка. Образцы речи. Ижевск, 2005. 581 с. / Karpova L. L. The Srednechepetsk dialect of the Udmurt language. Speech patterns. Izhevsk, 2005, 581 p. (in Russian).
[28]. Атаманов М. Г. По следам удмуртских воршудов. Ижевск: Удмуртия, 2001, с. 112. / Atamanov M. G. In the footsteps of the Udmurt vorshudov. Izhevsk, Udmurtia, 2001, p. 112. (in Russian).

Информация об авторах / Information about authors

Александра Филипповна УТКИНА – кандидат филологических наук, младший научный сотрудник отдела филологических исследований Удмуртского института истории, языка и литературы УдмФИЦ УрО РАН с 2019 года, старший научный сотрудник лаборатории

Лингвистических платформ с 2023 года. Сфера научных интересов: корпусная лингвистика, синтаксис, диалектология, лексикология.

Alexandra Filippovna UTKINA – Cand. Sci. (Philol.), Research Assistant of the Department of Philological Research of the Udmurt Institute of History, Language and Literature UdmFRC UB RAS since 2019, Senior Researcher of the Linguistic Platforms Laboratory since 2023. Research interests: corpus linguistics, syntax, dialectology, lexicology.

DOI: 10.15514/ISPRAS-2023-35(6)-20

Настоящее время на (–)ЧАР в младописьменном шорском языке

А.В. Есипова, ORCID: 0009-0002-5286-5842 <*aves7760@mail.ru*>

Институт языкознания РАН,

Россия, 125009, Москва, Большой Кисловский пер., д. 1 стр. 1

Аннотация. В статье дается анализ отглоссированного корпуса и конкорданса книг «Священная история …» (1883) и «Указание пути в Царствие Небесное …» (1884) на шорском наречии, которые находятся в открытом доступе на онлайн-платформе ЛингвоДок (lingvodoc.ispras.ru). Установлено, что книги являются памятниками слабо нормированного младописьменного шорского языка – письменного варианта кондомского (шорского) наречия. В корпусе выявлены систематические случаи отличия языка первых печатных изданий от современного языка, которые не были ранее описаны: мультивариативная форма настоящего времени, объединяющая аналитические, дефисные, переходные к синтетическим, и синтетические глагольные конструкции с одним и тем же показателем "(–)чар(ы)", а также элиминация аффикса третьего лица единственного числа "ы" в синтетических формах настоящего времени на "(–)чар(ы)" и процессы синтезации данной временной формы с неустоявшейся орфографией. Автором проведен анализ регулярности этих явлений, а также введены в научный оборот материалы мало изученного младописьменного шорского языка последней трети XIX в., сегодня находящегося под угрозой исчезновения, пополнен автоматический парсер шорского языка отсутствующими в нем аффиксами мультивариативных глагольных форм. Правило отпадения "(Ы)п" в основах, оканчивающихся на согласную, еще не нормировано и работает спорадически, не нормирована и орфография слитно-раздельно мультивариативной формы настоящего времени на "(–)чар". В статье разбирается материал, имеющий отношение к становлению грамматики и орфографии современного шорского языка.

Ключевые слова: тюркские языки; младописьменный шорский язык; первые книги; парсер; настоящее время.

Для цитирования: Есипова А.В. Настоящее время на (–)чар в младописьменном шорском языке. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 311–330. DOI: 10.15514/ISPRAS–2023–35(6)–20.

Благодарности: Работа выполнена при поддержке гранта Правительства РФ No 14.Y26.31.0014, проект «Языковое и этнокультурное разнообразие Южной Сибири в синхронии и диахронии: взаимодействие языков и культур» (сбор и обработка материала) и гранта РНФ № 18-18-00501 (проект «Создание электронного диалектологического атласа тюркских языков России»). Автор выражает глубокую благодарность этим фондам за возможность провести данные исследования. Автор также выражает признательность А. В. Дыбо и Ю. В. Норманской за ценные замечания, учтенные в ходе создания автоматического парсера Корпуса шорского языка и его описания.

Present Tense of (–)char in the Young Written Shor Language

A.V. Esipova, ORCID: 0009-0002-5286-5842 <*aves7760@mail.ru*>

Institute of Linguistics of the Russian Academy of Sciences,
b. 1, 1, Bolshoy Kislovsky lane, Moscow, 125009, Russia.

Abstract. The article analyzed the glossed corpus and concordance of the books "Sacred History ..." and Indication of the "Way to the Kingdom of Heaven ...", which are publicly available on the LingvoDoc platform located at lingvodoc.ispras.ru. The books are samples of a weakly normalized young–written Shor language - a written version of the Kondom (Shor) dialect. The corpus revealed cases of differences in the language of the first books from the modern language that had not been previously described: a multivariate form of the present tense, combining analytical (kölep char-ï 'bestows'), hyphenated, transitional to synthetic (pildir–char-ï 'means'), and synthetic (aitchar or aitchar-ï 'says') verbal constructions with the same indicator (–)char(ï); elimination of the third person singular affix -ï in synthetic forms of the present tense on (–)char(ï); the processes of synthesizing this time form with an unstable spelling. Our task was to analyze the regularity of these phenomena, as well as the introduction into scientific circulation of materials of the little-studied young-written Shor language, which is now in danger of extinction, as well as the replenishment of the automatic Parser of the Shor language with the missing affixes of multivariate verb forms. It was found that there is a single form of present, with the affix –char, as a more complete form of the present tense affix -cha, formed from the verb chat– 'to lie'; in analytical constructions, word forms with affix –ï predominate (83.3%), word forms without affix make up only 1.7%, in synthetic forms, the number of word forms without affix is already 45.4%. The rule of falling off -(I)p in the bases ending in a consonant has not yet been normalized and works sporadically; the spelling of the fused-separately multivariate form of the present tense on (–)char has not been normalized either.

Keywords: Turkic languages; young written Shor language; first books; parser; present tense.

For citation: Esipova A.V. Present tense of (–)char in the Young written Shor language. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 311-330 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-20

Acknowledgements. The work was supported by grants from the Government of the Russian Federation No. 14.Y26.31.0014 (project "Linguistic and ethno-cultural diversity of Southern Siberia in synchrony and diachrony: interaction of languages and cultures", collection and processing of material), and from Russian Science Foundation No. 18-18-00501 (project "Creation of an electronic dialectological atlas of the Turkic languages of Russia"). Author expresses her deep gratitude to these foundations for the opportunity to conduct these studies. Author also expresses her gratitude to A.V. Dybo and Yu.V. Normanskaya for their valuable comments that were taken into account during the creation of the automatic parser of the Shor Language Corpus and its description.

1. Введение

Шорский язык – язык миноритарного тюркского народа Южной Сибири, относительно близок хакасскому, чулымско-тюркскому и северным диалектам алтайского языка (кумандинскому, чалканскому, тубинскому). Это язык с неглубокой письменной традицией (почти 160 лет), переживший за свою историю несколько длительных периодов бесписьменности. В настоящее время численность шорцев составляет 13 000 человек. Согласно данным переписи 2010 г. шорцы входят в число двух народов РФ с отрицательным приростом населения [1], а шорский язык находится под угрозой исчезновения [2].

Одной из важнейших задач по сохранению исчезающих языков является их документация. Документация шорского языка и создание его электронного корпуса осуществляется разными группами ученых с большими перерывами с 1989 года[1]. В настоящее время на

[1] Проект «Создание электронного корпуса шорских текстов (литературных и образцов устной речи) с целью сохранения материалов исчезающего языка, и его лингвистическое описание», № 00-06-04005 ННИО а, поддержанный Немецким научным обществом (ННИО) и Российским фондом фундаментальных исследований (РФФИ) и другие более ранние и более поздние проекты

платформе «ЛингвоДок» был создан подкорпус первых православных книг на шорском наречии путем их оцифровки, сегментного анализа и грамматического описания (исполнитель д.ф.н. А.В. Есипова)[2]. Он включает две печатные книги на первом шорском алфавите, созданном Алтайскими миссионерами на кириллице на основе графики и орфографии русского языка до орфографической реформы 1917–1918 гг., когда еще сохранялись буквы ‹ѣ› (ять), ‹ѳ› (фита), ‹і› (‹и› десятеричное), а также некоторые другие, с добавлением графем ‹ӧ›, ‹ӱ›, ‹ҥ›, обозначающих шорские звуки, которых нет в русском языке (подробнее см.: [3]).

Наше исследование проводится на языковом материале, полученном путем сплошной выборки из 70 страниц текста первых православных книг «*Священная история на шорском наречии*» [4] и «*Указание пути в Царствие Небесное*» на шорском наречии [5]. Эти книги изданы Алтайскими миссионерами в последней трети XIX в. и представляют собой переводы православных текстов с алтайского на кондомское (шорское) наречие.

Переводы были сделаны священником Алтайской Духовной Миссии отцом Иоанном Матвеевичем Штыгашевым, носителем шорского языка, окончившим Казанскую учительскую семинарию. Они были подготовлены к изданию под руководством В. В. Вербицкого в сотрудничестве с Постоянной переводческой комиссией при Братстве святителя Гурия (г. Казань) на выработанных ею под руководством Н. И. Ильминского принципах. Одним из важнейших принципов была точная запись на слух живого разговорного языка, понятного простым людям, выполненная носителями этих языков.

Объектом нашего исследования является форма настоящего времени на *чар* в памятниках «шорского наречия» (1883-1884 гг.), которая в современном нормативном шорском языке отсутствует. Некоторые данные об этой форме в кондомском (шорском) наречии, а также в некоторых говорах современного шорского языка и его кондомском диалекте приведены авторами «*Грамматики алтайского языка*», Н. П. Дыренковой, Э. Ф. Чиспияковым и Ф. Г. Чиспияковой.

Первые сведения о шорской форме «именно теперь совершающегося действия» на *jap* обнаружены нами в «*Грамматике алтайского языка*», но представлены они не в системе, а распределены по разным разделам при описании грамматических форм алтайско-телеутского наречия [6: 16, 58, 60, 72, 98, 239]. Тем не менее, там содержатся важные сведения о данной форме. Авторы описали происхождение формы на *jap*, ее образование, спряжение компонента этой формы – вспомогательного глагола *jam-* в форме *jap*, предложили вариант ее орфографической записи через дефис. Однако исследование было проведено на базе всего девяти словоформ, отсутствует парадигма спряжения глаголов настоящего времени на *jap* в положительной и отрицательной форме, нет примеров-предложений.

В Грамматике шорского языка в примечаниях к форме настоящего времени на *-ча* впервые названы районы функционирования формы на *–чар*, указывается, что она является одной из форм настоящего времени в говорах шорцев рек Томи и Кондомы, а также верховьев реки Мрассу. В говоре шорцев реки Томи гласный аффикса *–чар* законам гармонии гласных не подчиняется. В говоре шорцев верховья реки Мрассу гласный аффикса гармонирует *–чар* / *–чер* (при передних гласных), и в аффиксе удерживается конечный *‹ы›*. В говоре шорцев реки Кондомы наряду с *–чар* встречаются варианты *–чер, –чыр, –чир*. Здесь приведена смешанная парадигма спряжения глаголов настоящего времени в говорах шорцев рек Томи и Мрассу, в которой форма на *–чар* отсутствует только во втором и третьем лице множественного числа [7: 195-196].

[2] Проект «Языковое и этнокультурное разнообразие Южной Сибири в синхронии и диахронии: взаимодействие языков и культур», грант Правительства РФ № 14. Y26.31.0014. Проект «Создание электронного диалектологического атласа тюркских языков России», РНФ № 18-18-00501.

В «*Учебном пособии по диалектологии шорского языка*» называются формы настоящего времени в кондомском диалекте современного нормированного шорского языка, в том числе и форма на *–чар*, а также приводится смешанная парадигма спряжения глаголов, в которой форма на *–чар* представлена не во всех лицах и числах [8: 23].

Форма настоящего времени в памятниках «шорского наречия», то есть в первых православных книгах на шорском языке, описывается впервые. Актуальность темы исследования определяется новизной проблемы и необходимостью в процессе создания подкорпуса первых православных книг на шорском наречии глоссировать тексты второй половины XIX в., грамматика которых отличается от грамматики нормированного шорского языка XX-XXI вв. и еще не описана.

Целью нашего исследования является совершенствование шорского парсера, расширение его возможностей в области автоматизированного анализа не только образцов современного шорского языка, но и более ранних изданий.

К задачам данного исследования относится изучение частотных особенностей, характерных для первого варианта письменного шорского языка, отличающих его от современного нормативного шорского языка, в частности формы настоящего времени на (–)*чар*, описание пути ее эволюции, введение в научный оборот материалов мало изученного младописьменного шорского языка последней трети XIX в., называемого кондомским или шорским наречием [6: I, 9: 1], а также пополнение шорского парсера отсутствующими в нем аффиксами глагольных форм.

Под *наречием* мы понимаем, вслед за авторами «*Грамматики алтайского языка*», язык одного из тюркских народов, населявших Алтай в XIX веке [6].

Кондомское (*шорское*) *наречие* – это язык шорцев, проживавших не только в бассейне реки Кондомы, но и в других регионах Шории. отдельные грамматические формы которого приведены в «*Грамматике алтайского языка*», и на котором написаны первые православные шорские книги. Во многом шорское наречие сохраняет особенности живой разговорной речи (с местными чертами) и произношения носителей шорского языка второй половины XIX в. Поднаречия в кондомском (шорском) наречии не выделялись, поэтому вопрос о диалектной принадлежности текстов является дискуссионным.

Кондомский диалект – это один из двух диалектов современного шорского языка. Понятия «кондомское наречие» и «кондомский диалект» не тождественны.

Младописьменный шорский язык – язык первых православных книг на шорском наречии. Он слабо нормирован и является, по сути, первым этапом письменности, возникшей путем создания Алтайскими миссионерами первого шорского алфавита. Во многом этот язык сохраняет особенности живой разговорной речи и произношения носителей шорского языка того времени. Это – письменный вариант кондомского (шорского) наречия.

Памятники «шорского наречия» – это первые печатные книги на шорском языке, а именно православные книги на шорском наречии: «*Священная история* …» [4] и «*Указание пути в Царствие Небесное* …» [5].

2. Первые сведения о шорской форме на jap (чар)

Первые сведения о шорской форме «именно теперь совершающегося действия» на *jap* содержатся в *Грамматике алтайского языка* [6: 16, 58, 60, 72, 98, 239]. Она употребляется в полной – *jадыр(ы)* или сокращенной – *jар(ы)* формах, «по кондомскому произношению начального ‹j› – *чадыры*, *чары*» [6: 58], образованных в процессе десемантизации самостоятельного глагола *jат-* ‘лежать’ и превращения его во вспомогательный глагол [6: 60]. Сравните, например: конд. *алып-jадыр*, *алып-jар* ‘он берет’ [6: 72].

Форма *jадыр(ы)* (*чадыр(ы)*) образуется от глагола *jат-* (*чат-*) путем присоединения аффикса настоящего времени *–ыр*, существующего в некоторых тюркских языках [6: 72]: конд. *jат-* + *-ыр* > *jадыр* > *jадыры.* Формы настоящего времени *jад(ы) и jар(ы)* являются образованными

от нее. Форма *јад(ы)* (*чад(ы)*) образовалась путем опущения слога ‹*ры*› и беглого ‹*ы*› в форме *јадыры*, а форма *јар* (*чар*) образовалась путем опущения в ней слога ‹*ды*› и беглого ‹*ы*› [6: 58]. Сравните: конд.

јадыры > *јады(ры)* > *јад(ы)* > *јат* (*чадыры* > *чады(ры)* > *чад(ы)* > *чат*) и

јадыры > *ја(ды)ры* > *јар(ы)* > *јар* (*чадыры* > *ча(ды)ры* > *чар(ы)* > *чар*).

Интересующая нас форма настоящего времени *јар* (*чар*) образована из аналитической конструкции *–(Ы)п јат-* (*чат-*), в которой трансформируется не только глагол *јат-* (*чат-*), но и аффикс соединительного деепричастия *-(Ы)п*. Сравните, например: конд. *сарна-п-јар* ‘он поет’, *ал-ып-јар* или *ал-јар* ‘он берет’, *таап-јар* ‘он находит’ (от *тап-* ‘находить’), *кел-јер* ‘он приходит’ [6: 16-17, 71-72].

Следует обратить внимание на то, что во всех содержащихся в *Грамматике алтайского языка* примерах показатель настоящего времени *јар* (*чар*) не пишется слитно с основой, а отделен от смыслового глагола дефисом, то есть *јар* является полноценным вспомогательным глаголом [6: 16-17, 71-72] и, значит, в аффикс, который не отделим от основы, еще не превратился. В. В. Радлов, поясняя появление дефиса между двумя ранее самостоятельными словами отмечает: «- (*дефис, прим. автора*) между двумя словами означает, что оба слова соединились почти в одно слово» [10: XVI]. Таким образом, с одной стороны, показатель *јар* (*чар*) является вспомогательным глаголом, а с другой стороны, он слился со смысловым глаголом почти в одно слово, то есть представляет собой продвинутую стадию превращения показателя настоящего времени – вспомогательного глагола *јат-* (*чат-*) в форме 3 лица единственного числа настоящего времени *јар* (*чар*) – в аффикс настоящего времени *-јар* (*-чар*).

Показатель *јар* употребляется нами без инициального (*-јар*) или финального (*јар-*) дефиса, так как он еще не аффикс, и поэтому ему не может предшествовать дефис – маркер аффиксов, но и не основа глагола, маркером которой является конечный дефис.

Более последовательно процесс превращения самостоятельного глагола *чат-* ‘быть, пребывать, существовать, жить’ [6: 238] в аффикс настоящего времени *-чар* отражен в орфографии первых православных книг на шорском наречии, то есть, по нашей терминологии, в младописьменном шорском языке, где форма на *-чар* является единственной формой, выражающей «именно теперь совершающееся действие» и отсутствующей в современном нормативном шорском языке. Следует отметить, что процесс превращения глагола *чат-* в полноценный аффикс еще не завершен ни в младописьменном (*чар*), ни в современном (*-ча*) шорском языке, так как гласный формантов *чар* / *-ча* не подчиняется рядному сингармонизму. Однако такие процессы были отмечены Н.П. Дыренковой в говорах шорского языка. Так, например, в говоре шорцев верховья реки Мрассу факультативно могут быть отмечены формы на *че* / *чер* / *чир* (*кел-че, кел-че-р, кел-чи-р* ‘приходит’), а в говоре шорцев верховья реки Томи – формы на *чар* / *чыр* (*сура-п-чар* / *сура-п-чыр* ‘спрашивает’, но *кел-чар* ‘приходит’) [подробнее см. 7: 195].

3. Настоящее время на (–)чар в младописьменном шорском языке

В памятниках «шорского наречия» [4-5], то есть в младописьменном шорском языке функционирует только одна, сокращенная, форма настоящего времени *(–)чар(ы)*, хотя, как отмечалось выше, в работе [6], написанной на базе других источников – записей шорского фольклора В.В. Радлова [10] и записей миссионеров, в том числе и бытовых часто употребляемых фраз, указывается, что эта форма употребляется как в полной – *јадыр(ы),* так и в сокращенной форме – *јад(ы), јар(ы)* [6: 58].

3.1. Образование настоящего времени на (–)чар в памятниках шорского наречия

В памятниках «шорского наречия» [4-5] утвердительная форма настоящего времени на *-чар* образуется из деепричастия на *-(Ы)п,* в полной (младп. шор. *сал-**ып чар*** ‘кладет’, *кыйна-**п-чар*** ‘мучает’) или усеченной (без аффикса деепричастия: младп. шор. *айт-**чар*** ‘говорит’) форме, и вспомогательного глагола *чат-* в форме *чар,* который часто пишется в первых православных книгах на шорском наречии слитно с деепричастием (младп. шор. *сура**пчар*** ‘просит’), но, нередко, и раздельно с ним (младп. шор. *сана**п чар*** ‘думает’), а иногда – через дефис (младп. шор. *кöдүр**ӱп-чар***. ‘поднимает’). Во избежание путаницы в употреблении маркеров аффикса и вспомогательного глагола в составе словоформы мы используем в первом случае короткий дефис, то есть дефис нормальной длины, а во втором – длинный дефис, обозначаемый через короткое тире. Сравните: при делении на морфемы написание *кöдүр-**ӱп-чар*** означало бы слитное написание ***чар*** и соединительного деепричастия: *кöдүр**ӱпчар***, а написание через длинный дефис – не слитное, но не раздельное, написание ***чар*** и соединительного деепричастия *-(Ы)П*: *кöдүр**ӱп–чар***, то есть через дефис. Поэтому в младописьменном шорском языке показатель *чар* целесообразно употреблять, на наш взгляд, с инициальным длинным дефисом в круглых скобках: *(–)чар,* как следствие неустоявшейся орфографии, отражающей нормы произношения носителей шорского языка того времени. При таком обозначении круглые скобки свидетельствуют о факультативности короткого дефиса, как маркера аффикса, а также о факультативности длинного дефиса, как маркера вспомогательного глагола на пути превращения в аффикс. Обозначение *(–)чар* объединяет три варианта написания показателя *чар* в одно: *чар*, *–чар*, *-чар = (–)чар*.

Полная форма деепричастия на *-(Ы)П* всегда присоединяется к основам на гласный, при слитном или же раздельном написании компонента *(–)чар*, например: млад. *сакта**пчар*** ‘ожидает’ и *сыкта**п чар*** ‘рыдает’ (все примеры см. в табл. 5)). Однако иногда полную форму имеют и глаголы с финальным согласным при раздельном написании основы и показателя *(–)чар*, например: младп. шор. *кал**ып чар*** ‘остается’. Нам встретился также один пример написания полной формы деепричастия на *-(Ы)П* и показателя *(–)чар* через дефис: младп. шор. *кöдүр-**ӱп–чар***. ‘поднимает’. В основном же, глаголы с согласным в ауслауте имеют усеченную форму деепричастия, как при слитном, так и при раздельном написании с показателем настоящего времени *(–)чар*: младп. шор. *кел**чар*** ‘приходит’, *курут **чар*** ‘сушит’. При раздельном написании глагольная основа часто содержит аффикс залогового типа: младп. шор. *арда-**т чар*** ‘портит’, *піл-**дір чар*** ‘означает’, *эді-**н чар*** ‘делает себе’, *арыгла-**л чар*** ‘очищается’.

Отрицательная форма настоящего времени на *(–)чар* образуется путем присоединения к основе смыслового глагола аффикса *–ПАн*, который пишется как слитно, так и раздельно или через дефис с показателем *(–)чар* (младп. шор. *піл**бенчар*** ‘не знает’, *четпол**бан чар*** ‘не может достичь’, *тутур**бан–чар*** ‘не держит’).

Полезно сравнить образование форм настоящего времени на *(–)чар*, описанных в *Грамматике алтайского языка* [6] и встречающихся в памятниках «шорского наречия» (см. табл. 1).

Примеры в табл. 1 расположены от наиболее полной формы деепричастия на *-(Ы)п* к сокращенной, а также от раздельного написания показателя *(–)чар* к слитному. Расположение форм по способу написания показателя настоящего времени отражает разные стадии процесса превращения его из вспомогательного глагола *чар-* в аффикс *-чар*.

У основ с финальным согласным ‹*п*› полная форма соединительного деепричастия вообще не встречается, так как буква ‹*п*› озвончается, ‹*б*› в интервокальном положении исчезает, а следующая за ним ‹*ы*› уподобляется предыдущей гласной: в первом примере – ‹*а*›, во втором – ‹*е*› [6: 17]. Например, конд. *тап* ‘найди’ > *тап-ып* > *таб-ып* > *таып* > *таап* ‘найдя’; *теп* ‘толкни’ > *теп-ін* > *теб-ін* > *тейн* > *теен* ‘толкнув’. Такие примеры в памятниках «шорского наречия» [4-5] нам пока не встретились.

Как показано в табл. 1, соединительное деепричастие во всех источниках образуется одинаково, однако в орфографии показателей *jap / чар* имеются отличия. В [6] гласный показателя *jap* подчиняется законам гармонии гласных (*jap / jep*): конд. *ал–**ja**p, кел–**je**p, албан–**ja**p, келбен–**je**p* [6: 58], что является признаком превращения его в аффикс, а показатель *(–)чар* нет. Кроме того, в [6] представлен только один способ написания показателя *jap* – через дефис, хотя в работе Радлова [10], которая послужила основным источником примеров для нее [6: VI], данный показатель пишется тремя способами, как и в памятниках «шорского наречия» [4-5], которые являются объектом нашего исследования. Данные факты свидетельствуют о том, что степень нормализации шорского наречия выше в [6].

Табл. 1. Образование настоящего времени на –jap / –чар в шорском наречии
Table 1. The formation of the present tense on –jar / –char in the Shor dialect

Утвердительная форма			
Грамматика алтайского языка, 1869 [6]		Священная история, 1883 [4]; Указание пути в Царствие небесное, 1884 [5]	
Форма	Перевод	Форма	Перевод
*кел**in–jep***	‘приходит’	*кал-**ып чар***	‘остается’
*сарна**n–jap***	‘поет’	*көдүр-**үп–чар***	‘поднимает’
*таа**n–jap***	‘находит’	*сыкта-**п чар***	‘рыдает’
*тее**n–jep***	‘толкает’	*сакта**пчар***	‘ожидает’
*ал–**jap***	‘берет’	*арыгла-л **чар***	‘очищается’
*кел–**jep***	‘приходит’	*арда-т **чар***	‘портит’
		*піл-дір **чар***	‘означает’
		*эді-н **чар***	‘делает себе’
		*кел**чар***	‘приходит’
Отрицательная форма			
*ал**бан–jap***		*четпол-**бан чар***	‘не может достичь’
*кел**бен–jep***		*тутур-**бан–чар***	‘не держит’
		*піл**бенчар***	‘не знает’

В приведенных ниже примерах 1-5 из памятников «шорского наречия» [4-5] показатели *чар*, а также Pres употребляются с длинным дефисом в скобках: *(–)чар* и (–)Pres, как маркером единства компонентов аналитической и синтетической конструкции.

(1) « <…> *ол поз-ы-нга <…> кыял эд-ін чар-ы*» [5: 17].
<…> ол поз-(з)Ы-н(К)А <…> кыял эд-(Ы)н(–)чар-ы.
<…> тот сам-3Pos-Dat <…> грех делать-Refl(–)Pres-Pred
‘<…> тот себе самому <…> грех делает’.

(2) «<…> *Кудай Поз-ы-ныҥ чакшы-зы-была көл-еп ча.р-ы*» [4: 17].
<…> Кудай Поз-(з)Ы-НЫҥ чакшы-(з)Ы-П(Ы)лА көле-(Ы)П(–)чар-ы
<…> Бог Свой-3Pos-Gen милость-3Pos-Instr одаривать-NF$_{II}$(–)Pres-Pred.
‘<…> Бог Своей милостью одаривает’.

(3) «<…> *чакшы эт-кен-ібіс Кудай-даҥ пол-чар*» [5: 37].
<…> чакшы эт-ГАн-(Ы)бЫС Кудай-ДАҥ пол(–)чар.
<…> добро делать-Past-1pos.pl Бог-Abl быть(–)Pres.
‘<…> добро, (которое) сделали-мы, от Бога бывает’.

(4) «<…> *Кудай чүрек-ты өтүре піл-чар-ы;* <…>» [5: 21].
<…> Кудай чүрек-НЫ өтүре піл(–)чар-ы; <…>.
<…> Господь сердце-Асс насквозь знать(–)Pres-Pred; <…>.
‘<…> Господь сердце насквозь знает (видит) <…>’,

(5) «<…> *сөөг-і-даа чаткан черді пірдаа кіжі пілбен чар*» [4: 53].
<…> сөөг-(з)Ы-ТА(А) чат-ГАн чер-НЫ пірдаа кіжі піл-П(А)Ан(–)чар.
<…> кость-3Pos-Add лежать-Past место-Асс никто знать-Conv.Neg(–)Pres.
‘<…> место, (где) кости его лежат, ни один человек (никто) не знает’.

3.2 Спряжение глаголов настоящего времени на (–)чар в шорском наречии

В шорском наречии в настоящем времени глаголы спрягаются путем присоединения к показателю *jap* / *(–)чар* аффиксов лица. В памятниках «шорского наречия» [4-5] глаголы настоящего времени встречаются в утвердительной и отрицательной форме в разных лицах и числах. На основе этих примеров можно составить парадигмы спряжения глаголов настоящего времени на *(–)чар* в младописьменном шорском языке в обеих формах.

3.2.1 Утвердительная форма спряжения глаголов настоящего времени на –jap / (–)чар

Полную парадигму спряжения глаголов настоящего времени в утвердительной форме только на основе примеров из памятников «шорского наречия» [4-5] составить невозможно из-за отсутствия в них глаголов второго лица единственного числа. Это обстоятельство обусловлено характером православных текстов, в которых ситуации, где может появиться данная форма, в исследуемых источниках нам не встретились. Для определения аффикса второго лица единственного числа целесообразно сравнить спряжение глаголов настоящего времени на *jap* / *(–)чар* в шорском наречии, представленное в [6] и составленное нами на базе примеров, извлеченных из памятников «шорского наречия» [4-5], так как они относятся к одному периоду времени. Правда, в [6] не описано спряжение глаголов настоящего времени, но спряжение формы на *jap*, как его составляющей, показано в таблице [6: 59].

Сравним спряжение глаголов настоящего времени на *–jap / (–)чар* в [6] и в памятниках «шорского наречия» [4-5].

Табл. 2. Спряжение глаголов настоящего времени на –jap / (–)чар в шорском наречии утвердительная форма)
Table 2. Conjugation of verbs of the present tense on –jar / (–)char in the Shor dialect (affirmative form)

Грамматика алтайского языка, 1869 [6]		Памятники «шорского наречия» 1883-1884 [4-5]			
Утвердительная форма					
		Квазианалитическая форма		Синтетическая форма	
Единственное число					
1 л.	*jap-ым*	*санап чар-ым*	‘думаю’	*айтчар-ым*	‘говорю’
2 л.	*jap-ың* *jap-зың*	-		–	
3 л.	*jap-ы,* *jap*	*ӱргет чар-ы* *салып чар*	‘учит’ ‘кладет’	*айтчар-ы* *айтчар*	‘говорит’
Множественное число					
1 л.	*jap-ык* *jap-ыбіс*	*теп чар-быс*	‘говорим’	*айтчар-быс*	‘говорим’
2 л.	*jap-ыңар* *jap-зылар*	*калып чар-зар*	‘остаетесь’	*кӧрчар-зар*	‘видите’
3 л.	*ja-лыр* (*je-лір*)	*чіскінчар-лар*	‘гнушаются’	*айтчар-лар*	‘говорят’

Примечание 1 к табл. 2. Мы сравниваем между собой только формы шорского наречия, а формы современного нормативного шорского языка, как формы более высокого уровня нормализации, к сравнению не привлекаем.

Примечание 2 к табл. 2. Под квазианалитическими конструкциями мы понимаем достаточно широко представленные в православных источниках на шорском наречии формы, которые еще не сформировались окончательно и находятся в процессе превращения аналитической глагольной формы в синтетическую.

Примечание 3 к табл. 2. В *Грамматике алтайского языка* форма на *jap* называется, собственно, кондомской формой [6: 59] настоящего времени, поэтому в дальнейшем мы будем употреблять термин «кондомское наречие», который используется в грамматике чаще, чем синонимичный ему термин «шорское наречие».

Парадигма спряжения глаголов настоящего времени на *(-)чар* в памятниках «шорского наречия» [4-5] нами составлена из разных глаголов, так как один и тот же глагол во всех лицах и числах настоящего времени в наших материалах отсутствует.

В табл. 2 в памятниках «шорского наречия» [4-5] не представлена дефисная форма показателя *(–)чар*, что обусловлено малочисленностью таких примеров в наших материалах (см. табл. 5). Однако как синтетические, так и квазианалитические формы имеют почти полную парадигму настоящего времени на *(–)чар*. Отсутствует только форма второго лица единственного числа. Аффиксом 2-го лица единственного числа будет, скорее всего, аффикс *–зың*, который имеется в парадигме спряжения компонента настоящего времени *jap* (*jap-зың*) в кондомском наречии. Сравните, также, отрицательные формы глаголов настоящего времени, отмеченные Н. П. Дыренковой в говорах шорского языка, к*ӧр-беен-чар-зынъ* ‘ты не видишь’ и *сура-баан-чар-зынъ* ‘ты не спрашиваешь’ [7: 196 §129, примечание].

В кондомском наречии впервые фиксируется сокращенная форма показателя *jap – ja / je,* которая в современном нормативном шорском языке является единственной формой настоящего описательного времени [7: 193], однако в отличие от кондомского (шорского) наречия гласный ‹*а*› аффикса законам гармонии гласных не подчиняется**.** В кондомском наречии (см. табл. 2) эта форма присутствует в 3-ем лице множественного числа: *ja-лыр* (*je-лір*). В памятниках же «шорского наречия» сокращенная форма показателя настоящего времени на *(–)чар* нам не встретилась. Причем в кондомском наречии гласный показателя *jap* законам гармонии гласных подчиняется, а в памятниках «шорского наречия» нет. Сравните формы настоящего времени глагола *кел-* из таблицы 1 (конд. *келіп–јер* и *кел–јер* ‘приходит’) и показатель 3-его лица множественного числа из таблицы 2 (*ja-лыр* (*je-лір*). Как видно, в кондомском наречии гармонирует не только гласный показателя *ja*, но и, закономерно, аффикс 3-его лица множественного числа. В то же время в сочетании с полной формой показателя настоящего времени *jap* переднерядные варианты аффиксов лица отсутствуют в таблице, хотя можно предположить, что они возможны, судя по формам 3-его лица единственного числа: конд *келіп–јер* и *кел–јер* ‘приходит’. Однако из-за малочисленности примеров говорить об этом с уверенностью невозможно.

Наблюдаются различия и в аффиксах первого и второго лица множественного числа. В памятниках «шорского наречия» [4-5] в отличие от [6] в первом лице множественного числа отсутствует аффикс *–ык* - «обыкновенная татарская и киргизская приставка этого лица» [6: 54, 58], а также аффикс *–ыбіс* употребляется без беглого ‹*ы*› - *–быс* (см. табл. 2). Во втором лице множественного числа используется только одна, стяженная, форма этого аффикса - *-зар* по сравнению с [6], где отмечены две полные формы данного аффикса *-ыңар* и *–зылар.*

Особое внимание следует обратить на факультативное употребление аффикса *–ы* в третьем лице единственного числа, как в кондомском наречии, так и в памятниках «шорского наречия». Как поясняют авторы работы [6], «Беглый звук ‹*ы*› мы поставили как личное окончание. Подобное ‹*ы*› находится в 3-ем лице настоящего времени изъявительного наклонения … Это ‹*ы*›, можно полагать, есть 3-е лице нашей второй группы личных приставок» [6: 93], которая «не переносит на себя ударение» [6: 91].

В памятниках «шорского наречия» количество примеров с аффиксом *–ы* в квазианалитических конструкциях с показателем *(–)чар(ы)* существенно превышает количество примеров без данного аффикса. А в синтетических конструкциях количество примеров с аффиксом 3 л. ед. ч. только на два больше, чем без него. Представим процентное соотношение рассматриваемых конструкций в табл. 3.

Табл. 3. Соотношение форм 3 лица единственного числа с –ы и без –ы в формах настоящего времени на (–)чар в памятниках «шорского наречия» [4-5]
Table 3. Correlation between the forms of the 3rd person singular with–s and without –s in the forms of the present tense on (–)char in the monuments of the "Shor dialect" [4-5]

Формы 3 л.ед.ч. на *(–)чар*			Квазианалитические *(-NF$_n$) чар*			Синтетические *(-NF$_n$)-чар*		
Всего	с *-ы*	без *-ы*	Всего	с *-ы*	без *-ы*	Всего	с *-ы*	без *-ы*
34	22	12	12	10	2	22	12	10
100 %	64,7 %	35,3 %	100 %	83,3 %	1,7 %	100 %	54,5 %	45,4 %

Примечание 1 к табл. 3. Дефисные формы настоящего времени с показателем *(–)чар(ы)* не включены в таблицу из-за их малочисленности.

Как показано в табл. 3, общее количество глаголов настоящего времени 3-его лица единственного числа составляет в наших материалах 34 словоформы, причем глаголы с аффиксом *–ы* составляют в них 64,7%, а с нулевым показателем почти вдвое меньше – 35,3%. В квазианалитических конструкциях картина другая: словоформы с нулевым показателем составляют всего 1,7%, а преобладают словоформы с аффиксом *–ы* (83,3%). В синтетических формах с аффиксом *-чар* количество словоформ с нулевым аффиксом резко возрастает и приближается к 50%, составляя 45,4%, в то время как глаголов с аффиксом 3-го лица *–ы* немногим больше 50% (54,5%). Полученные данные свидетельствуют о том, что в младописьменном шорском языке наблюдается тенденция к элиминации аффикса 3-го лица единственного числа *-ы* в синтетических формах глаголов настоящего времени на *(–)чар.*

Сравнение форм настоящего времени на *jap* и *(–)чар* в разных источниках шорского наречия выявило их общие и отличительные черты. В работе [6] содержатся первые сведения о грамматических формах настоящего времени кондомского (шорского) наречия, разбросанные по разным разделам [6] и подчеркивающие особенности аналогичных форм алтайско-телеутского наречия. Это первый опыт нормализации шорского наречия на базе разнообразных аутотентичных материалов, опубликованных В.В. Радловым [10], а также записанных В.И. Вербицким и другими сотрудниками Алтайской Духовной Миссии [6: 58, 7: 4, примеч. 3] от носителей шорского наречия на обширной территории, не только в бассейне реки Кондомы, но и в других районах проживания шорцев. Этими факторами обусловлено, на наш взгляд, наличие полных (*jap* / *jep*) и сокращенных (*ja* / *je*) форм настоящего времени и более широкого набора личных окончаний в кондомском наречии.

В памятниках же «шорского наречия» – православных текстах, переложенных отцом Иоанном Штыгашевым с алтайско-телеутского на кондомское наречие, представлено только одно родное поднаречие автора переводов и его особенности. Язык памятников является первым опытом нормализации шорского наречия на основе первого шорского алфавита, созданного Алтайскими миссионерами. Это – образец младописьменного шорского языка, первый этап шорской письменности. На материале памятников «шорского наречия» [4-5] целесообразно написать первую грамматику шорского языка, и описание формы настоящего времени на *(–)чар* – первый шаг к ее созданию.

В некоторых грамматических описаниях современного шорского языка содержатся и сведения о диалектной форме настоящего времени на *–чар(ы)* [7, 8]. В случае с материалами из памятников «шорского наречия» [4-5] сложно говорить о диалектах, так как в то время литературный, а точнее нормированный, шорский язык, по отношению к которому выделяются диалекты, еще не сформировался. Сравнение формы настоящего времени на *-чат* в шорском наречии (*(–)чар*) и в диалектах шорского языка (*-чар* / *-ча*) целесообразно провести в другой работе.

В источниках, где содержатся сведения о диалектной форме настоящего времени на *–чар*, в качестве примера приводится лишь одно предложение с этой формой:

(6) «*Иир-де одунъ ал-чар-ы, оданъ-а таш-чар-ы*» [8: 195].
Иир-ТА одуң ал-ча, оданъ-(К)А таш-ча.
Вечер-Loc дрова брать-Pres, балаган-Dat таскать-Pres.

‘Вечером берет дрова и таскает (их) к балагану’.

Приведем и другие примеры с формой настоящего времени на *(–)чар*, но уже из памятников «шорского наречия».

Квазианалитические формы:

(7) «<…> *де-п сана-п чар-ым, де-ен*» [4: 30].
<…> де-(Ы)П сана-(Ы)П(–)чар-ым, де-(Г)Ан.
<…> говорить-Conv$_{П}$ думать-NF$_{П}$(–)Pres-1.sg.br, говорить-Past.
‘<…> я думаю, что <…> сказал’.

(8) «<…> *чӱрег-і дезе Каан Кудай-даӈ ра-п чар-ы, де-ен*» [5: 6].
<…> чӱрег-(з)Ы дезе Каан Кудай-ДАӈ ра-(Ы)П(–)чар-ы, де-(Г)Ан.
<…> сердце-3Pos а Господь Бог-Abl удалять-NF$_{П}$(–)Pres-Pred, говорить-Past.
‘<…> а сердце-свое от Господа Бога удаляет, сказал»’.

(9) «<…> *алым-нар-ыбіс-ті ташта де-п чар-быс;* <…>» [5: 6].
<…> алым-ЛАр-(Ы)бЫС-НЫ ташта де-(Ы)П(–)чар-ПЫс <…>»;
<…> долг-Pl-1Pos.Pl-Acc прощать говорить-NF$_{П}$(–)Pres-1Pl <…>»;
‘<…> (как) мы свои долги (себе) прощаем, говорим; <…>’.

(10) «- *Слер бӱнь ногерек беде сагыш-ка кал-ып чар-зар де-ен*» [4: 27].
« - Слер бӱнь ногерек беде сагыш-КА кал-(Ы)П(–)чар-С(А)Ар де-(Г)Ан»
«- Вы сегодня отчего так дума-Dat остаться-NF$_{П}$(–)Pres-2.pl.mix говорить-Past».
‘- Вы сегодня отчего так беспокоитесь (заботитесь)? – сказал.’

(11) «<…> *Агарыг Тын-ныӈ чакшы-зы-нга арыгла-т чар-лар*», [5: 12].
<…> Агарыг Тын-НЫӈ чакшы-(з)Ы-(н)(К)А арыгла-(Ы)Т(–)чар-ЛАр,
<…> Святой Дух-Gen благодать-3Pos-Dat очищать-Caus(–)Pres-PredPl,
‘<…> Святого Духа благодатью они очищаются, <…>’.

Переходные формы:

(12) <…> «*четі чыл кӧп табак шыг-ар-ы-н білдір–чар-ы*; <…>» [4: 28]
<…> четі чыл кӧп табак шыг-Ар-(з)Ы-Н(Ы) білдір(–)чар-ы; <…>.
<…> семь год много плод родить-Fut-3Pos-Acc означать(–)Pres-Pred; <…>.
‘<…> семь лет великого изобилия означают; <…>’.

(13) «<…> *тегре-е Кудай-дыӈ Ангел-ар-ы шыгар–чар-лар*» [4: 23].
<…> тегре-(К)А Кудай-НЫӈ Ангел-(Л)Ар-(з)Ы шыгар(–)чар-ЛАр.
<…> небо-Dat Бог-Gen Ангел-Pl-3Pos поднимать(–)Pres-PredPl.
‘<…> на небо Божьи Ангелы поднимают’.

Синтетические формы:

(14) «<…> *Кудай-даӈ корук-чар-ым* <…>» [4: 30-31].
<…> Кудай-ДАӈ корук(–)чар-ым <…>
<…> Бог-Abl бояться(–)Pres-1.sg.br: <…>
‘<…> Я Бога боюсь: <…>’.

(15) «<…> *шошка* <…>, *пазок* <…> *малчак-а кір-чар*; <…>» [5: 23].
<…> шошка <…>, пазок <…> малчак-(К)А кир(–)чар; <…>.
<…> свинья <…>, снова <…> грязь-Dat входить(–)Pres; <…>.
‘<…> свинья <…>, опять в грязи пачкается ; <…>’.

(16) «<…> *корыг-ып кыял-ыбіс айдын-чар-быс*; <…> [5: 6].
<…> корыг-(Ы)П кыял-(Ы)бЫС айдын(–)чар-ПЫС; <…>.
<…> бояться-Conv$_{П}$ грех-1Pos.Pl признаваться(–)Pres-1.pl.mix ...
‘<…> со страхом в грехах наших признаемся: <…>’.

(17) «<…> - *Слер ногере те-ген чат-чар-зар*?» [4: 30].
<…> - Слер ногере те-ГАн чат(–)чар-С(А)Ар?
<…> - Вы почему говорить-Past лежать(–)Pres-2.pl.mix?
‘<…> - Вы почему, сказал, лежите?’

(18) «<…> *лар* <…> *карын-ар-ы-нга Кудай-даӈ артык чобал-чар-лар*» [5: 13].
<…> ылар <…> карын-(Л)Ар-(з)Ы-(н)(К)А Кудай-ДАӈ артык чобал(–)чар-ЛАр.
<…> они <…> утроба-Pl-3Pos-Dat Бог-Abl лучше служить(–)Pres- PredPl.
‘<…> они своим утробам лучше, чем Богу, служат (утруждаются)’.

3.2.2 Отрицательная форма спряжения глаголов настоящего времени на (–)jap / (–)чар

Глаголов настоящего времени на *(–)чар* в отрицательной форме в наших материалах всего 4, причем они имеют разную орфографию, являясь синтетическими (2), дефисными (1) или квазианалитическими (1) глагольными конструкциями. Однако, если объединить в одну парадигму все орфографические варианты разных глаголов, то можно составить почти полную парадигму спряжения глаголов настоящего времена на *(–)чар* в отрицательной форме (см. табл. 4). В ней будут отсутствовать только формы второго лица единственного и множественного числа. Это связано с тем, что в памятниках «шорского наречия» ситуации, где могут появиться такие формы глагола, нам не встретились. Если же включить в табл. 4 примеры из [6], то можно установить некоторые особенности спряжения таких глаголов в памятниках «шорского наречия», а поместив в таблицу выявленный на примерах вариант аффикса отрицательного деепричастия на -(Ы)П с личными окончаниями 2 лица единственного и множественного числа утвердительной формы, получаем полную парадигму спряжения глаголов настоящего времени на *(–)чар* в отрицательной форме.

Табл. 4. Спряжение глаголов настоящего времени на –jap / (–)чар в [6] и в памятниках «шорского наречия» [4-5] (отрицательная форма)
Table 4. Conjugation of verbs of the present tense on –jar / (–)char in [6] and in the monuments of the "Shor dialect" [4-5] (negative form)

Грамматика алтайского языка, 1869 [6]				Памятники «шорского наречия» 1883-1884 [4-5]	
Отрицательная форма					
Единственное число					
Лицо	Аффиксы лица	Форма	Перевод	Форма	Перевод
1 л.	*jap-ым*			*четполбан чар-ым*	‘не смогу достичь’
2 л.	*jap-ың* *jap-зың*			– *(-ПАн(–)чар-зың)*	
3 л.	*jap-ы,* *jap*	*албан–jap* *келбен–jep*	‘не берет’ ‘не приходит’	*пілбенчар*	‘не знает’
Множественное числ					
1 л.	*jap-ык* *jap-ыбіс*			*пілбенчар-быс*	‘не знаем’
2 л	*jap-ыңар* *jap-зылар*			– *(-ПАн(–)чар-зар)*	
3 л.	*ja-лыр (je-ліp)*			*тутурбан–чар-лар*	‘не держат’

Примечание 1 к табл. 4. Минус (–) в табл. 4 означает отсутствие в наших материалах из памятников «шорского наречия» примеров на отрицательную форму глаголов настоящего времени второго лица единственного и множественного числа.

Примечание 2 к табл. 4. Под минусами в скобках представлены аффиксы второго лица единственного и множественного числа, которые будут присоединяться к любой орфографической форме любой глагольной лексемы, о чем свидетельствует длинный дефис в скобках перед показателем *(–)чар.*

Как показано в табл. 4, спряжение отрицательной формы глаголов настоящего времени в *Грамматике алтайского языка* не представлено, но приведены два глагола в отрицательной форме третьего лица единственного числа. Они свидетельствуют о том, что и аффикс *–бАн*, и показатель *–jap* имеют как заднерядные (*-бан–jap*), так и переднерядные (*-бен–jep*) варианты.

В памятниках «шорского наречия» аффикс *–ПАн* тоже подчиняется законам гармонии гласных, а вот показатель *(–)чар* в отличие от форм, содержащихся в *Грамматике …,* - нет. Как и в положительной форме, он имеет как слитное, так и раздельное написание, а также пишется через дефис с деепричастием на *-ПАн*.

Приведем примеры употребления отрицательной формы настоящего времени на *(–)чар* в памятниках «шорского наречия»:

(19) «*аба-лар-ым-ныӊ чаш-тар-ы-нга чет-пол-бан чар-ым,*» [4: 39]
аба-ЛАр-(Ы)м-НЫӊ чаш-ЛАр-(з)Ы-(н)(К)А чет-пол-П(А)Ан(–)чар-ым,
отец-Pl-1Pos.sg-Gen год-Pl-3Pos-Dat достигать мочь-Conv.Neg(–)Pres-1.sg.br,
‘отцов-моих возраста не могу достичь-я,’

(20) «*<…> сööг-і-даа чаткан черді пірдаа кіжі піл-бен чар*» [4: 53]
<…> сööг-(з)Ы-ТА(А) чат-ГАн чер-НЫ пірдаа кіжі *піл-П(А)Ан(–)чар.*
<…> кость-3Pos-Add лежать-Past место-Acc никто знать-Conv.Neg(–)Pres.
‘<…> место, (где) кости его лежат, ни один человек (никто) не знает’.

(21) «*кайдып кап-тар-ыбіс-ка кір-ген-і-н піл-бен-чар-быс,*» [4: 34]
кайдып кап-ЛАр-(Ы)бЫС-КА кір-ГАн-(з)Ы-Н(Ы) піл-П(А)Ан(–)чар-ПЫс,
как мешок-Pl-1pos.pl-Dat входить-Past-3Pos-Acc знать-Conv.Neg(–)Pres-1.pl.br,
‘как в мешки-наши попало не знаем-мы, <…>’.

4. Процессы синтезации мультивариативной формы настоящего времени на (–)чар

Процессы синтезации формы настоящего времени на *(–)чар* происходят как посредством редукции аффикса деепричастия на *–(Ы)п*, так и путем сокращения вариативности орфографии данной формы. Названные процессы целесообразно рассмотреть на базе всех имеющихся у нас материалов, представив их в отдельной таблице 5 (в скобках указано количество повторяющихся форм).

Табл. 5. Сводная таблица глаголов настоящего времени на (–)чар из памятников «шорского наречия»
Table 5. Summary table of verbs of the present tense on (–)char from the monuments of the "Shor dialect"

Утвердительная форма			
Квазианалитическая		Синтетическая	
Форма	Перевод	Форма	Перевод
ардат чар-ы	‘портит’	*алгыш четірчар-быс*	‘благодарим’
арыглал чар-ы	‘очищается’	*айдынчар-быс*	‘каемся’
арыглат чар-лар	‘очищаются’	*айдып салчар-ы*	‘говорит’
көлеп чар-ы	‘одаривает’	*айтчар-ым*	‘говорю’
курут чар-ы	‘сушит’	*айтчар*	‘толкует’
öктендир чар-ы	‘подражает’	*айтчар-ы*	‘говорит’
пілдір чар-ы	‘означает’	*айтчар-быс*	‘говорим’
рап чар-ы	‘удаляется’	*айтчар-лар*	‘говорят’
рап чар-ы	‘удаляется’	*езен чӧрчар*	‘здравствует’
сагышка калып чар-зар	‘задумываетесь’	*келчар*	‘приходит’
салып чар	‘кладет’	(*малчака) кірчар*	‘пачкается (в грязь входит)’
санап чар-ым	‘думаю’	*корукчар-ым*	‘боюсь’
сыктап чар	‘рыдает’	*көöп салчар-лар*	‘закапывают’
теп чар-быс	‘говорим’	*көрчар-зар*	‘видите’
шыныктап чар-лар	‘действительно появляются’	*көрӱнчар-ы*	‘кажется’
ӱргет чар-ы	‘учит’	*кыйнапчар*	‘мучает’
эдін чар-ы	‘делает себе’	*нііктетчар-ы*	‘облегчает’
Переходная форма (через дефис)		*өрӱн чӧрчар-ы*	‘радуется’
көдӱрӱп–чар-ым	‘поднимаю’	*парчарым (2)*	‘иду’
пілдір–чар-ы	‘означает’	*парчар-ы (2)*	‘отходит’
		перчарым	‘даю’

		пілчар-ы	‘знает’
		піріктірчар	‘соединяет’
		пожытчар-быс	‘отпускаем’
		полчар	‘бывает’
		полчар-ы (2)	‘бывает; имеется’
		полчар-лар (3)	‘имеются’
		чӧрчар-ым	‘живу’
		сактапчар	‘ожидает’
		сурапчар-быс	‘просим’
		тӱжӱрчар (3)	‘ввергает; погружает’
		ӱргетчар	‘учит’
		чатчар-ы	‘лежит’ ‘лежат’
		чатчар-зар	‘лежите’
		чіскінчар-лар	‘гнушаются’
		чобалчар-лар	‘утруждаются’
		шыкчар	‘выходит’
		ылгап чӧрчар-ы	‘плачет’
		ылгап шыкчар-ы	‘сильно плачет’
		этчар-ы	‘делает’
Отрицательная форма			
четполбан чар-ым	‘не смогу достичь’	*пілбенчар*	‘не знает’
Переходная форма (через дефис)		*пілбенчар-быс*	‘не знаем’
тутурбан–чар-лар	‘не держат’		
Всего	21	49	
Итого	70		

Формы глаголов, представленных в табл. 5, свидетельствует о том, что в памятниках «шорского наречия» в качестве показателя настоящего времени используется редуцированная форма показателя *чадыр(ы) – чар(ы).*

Согласно табл. 5, показатель *чар(ы)* употребляется только в полной форме и не подчиняется законам рядного сингармонизма. Он пишется раздельно (младп. шор. *салып чар* ‘кладет’, *пілдір чар-ы* ‘означает’, *эдін чар-ы* ‘делает себе’), через дефис (младп. шор. *кӧдӱрӱп–чар-ым* ‘поднимаю’, *пілдір–чар-ы* ‘означает’) или слитно (младп. шор. *шыкчар* ‘выходит’, *этчар-ы* ‘делает’, *пілдірчар-ы* ‘означает’, *айдынчар-быс* ‘каемся’) с полной или сокращенной формой соединительного деепричастия (*-(Ы)п*). Данные факты свидетельствуют о том, что в памятниках шорского наречия орфография слитно-раздельно еще не была нормирована.

Очевидно, что в памятниках «шорского наречия» настоящее время на *(-)чар(ы)* является мультивариативной формой, включающей аналитические, переходные в синтетические, и синтетические формы. Аналитические формы, содержат вспомогательный глагол *чат-* в форме настоящего времени на *–ыр – чар(ы).* Переходные формы, в которых показатель *чар(ы)* объединен с основой посредством дефиса, обозначенного коротким тире (–), является формой вспомогательного глагола *чат-*, почти слившегося с основой и ставшего «почти аффиксом» - *–чар(ы)*. Количество переходных форм в наших материалах незначительно. В синтетических формах показатель *чар(ы)* пишется слитно с основой и является аффиксом – *-чар(ы)*.

В табл. 6 представлено соотношение разных типов конструкций с показателем настоящего времени *(–)чар*. Как показывают данные, собранные в этой таблице, количество квазианалитических форм настоящего времени на *(–)чар* в памятниках «шорского наречия» значительно, но существенно меньше, чем синтетических. Этот факт, а также наличие

небольшого количества форм, с объединяющим дефисом между деепричастием на *–(Ы)п* и показателем *чар*, в сочетании с десемантизацией смыслового глагола *јат- / чат-* «лежать» и редукции вспомогательного глагола (из конд. *јадыры > ја(ды)ры > јАр(ы) > јАр > јА* и младп. шор. *чадыры > ча(ды)ры > чар(ы) > чар*) в формах презенса при сохранении формы на *јадыр / чадыр* (с определенным модальным оттенком) может служить, на наш взгляд, подтверждением гипотезы авторов *Грамматики алтайского языка* [6] об идущих в кондомском (шорском) наречии второй половины XIX в. процессах превращения аналитической глагольной формы на *чар* в синтетическую.

Табл. 6. Соотношение квазианалитических, переходных и синтетических форм настоящего времени на (–)чар в памятниках «шорского наречия [4-5]

Table 6. The ratio of quasianalytical, transitional and synthetic forms of the present tense on (-)char in the monuments of the "Shor dialect [4-5]

Всего	Квазианалитические *(NF_n) чар*	Переходные (дефисные) *(NF_n)–чар*	Синтетические *(NF_n)-чар*
70	18	3	49
100 %	25,7 %	4,3 %	70 %

Примечание 1 к табл. 6. *NF_n* – not finished | non finitum – это грамматикализованный показатель, исторически восходящий к соединительному деепричастию *((-Ы)п)*, который используется для образования ряда синтетических и аналитических словоформ и указывает на незаконченность словоформы.

Конечно, надежным подтверждением этой гипотезы являлись бы данные экспериментальной фонетики, доказывающие наличие / отсутствие паузы между основой и показателем *чар* и ее длину, а также регулярность ее наличия. Однако по причине отсутствия таких данных мы можем полагаться в данный момент только на восприятие устной речи шорцев тюркологами (В.В. Радлов) миссионерами (В.И. Вербицкий и другие), записывавшими речь носителей шорского языка, или на интуицию самих шорцев, переводивших христианские книги с алтайского на шорское наречие и подготовивших их к изданию.

Методами экспериментальной фонетики были изучены фонетические трансформации глагольных аналитических конструкций (ГАК) в современных южносибирских тюркских языках в сопоставлении с языком фольклора. Новосибирскими учеными было установлено, что позиционно обусловленные фонетические изменения звуковых оболочек ГАК, являются своеобразными маркерами степени их синтезации [11: 124]. Было выявлено также, что «в основе инновационных модификаций звукового облика ГАК лежат процессы десемантизации входящих в их состав глагольных форм, ослабления их статуса, устранения межсловных пауз с последующим слиянием компонентов ГАК в единое фонетическое слово, преобразования внешнего сандхи во внутреннее, реализации ассимилятивных закономерностей, порождаемых сформировавшимся в результате стяжения фонетическим контекстом» [11: 124-125]. Исследователями были зафиксированы «различные стадии процесса синтезации ГАК – от полной утраты прозрачности структуры словосочетания в результате стяжения <…> до более ранних этапов их упрощения по пути потери лексическими компонентами самостоятельности <…> через промежуточную стадию в языке шорцев» [11: 124-125].

Результаты исследований в области экспериментальной фонетики были получены на материале других ГАК шорского языка, форма на *(–)чар* не входит в их число. Однако полученные данные схожи с результатами наших исследований по выявлению преобразований формы на *(–)чар* в процессе синтезации. В.В. Радловым и авторами *Грамматики алтайского языка* были отмечены процессы десемантизации самостоятельного глагола *чат-* в составе дефисных глагольных форм настоящего времени на *јат* и *јар* и ослабления его статуса. А нами было установлено, что в младописьменном шорском языке форма на *(–)чар* является мультивариативной, включающей конструкции разной степени синтезации: аналитическую, дефисную (переходную) и синтетическую. Процесс перехода

аналитической глагольной конструкции с показателем *чар* в синтетическую отражается в орфографии памятников «шорского наречия» и обусловлен сохранением, ослаблением или утратой самостоятельности, составляющих ее лексических компонентов. Пробел между деепричастием *–(Ы)п* и вспомогательным глаголом *чат-* в форме настоящего времени на *чар* свидетельствует о сохранении лексическими компонентами ГАК своей самостоятельности. Отсутствие пробела и появление на его месте дефиса в переходных формах обозначает ослабление самостоятельности лексических компонентов. Отсутствие пробела и дефиса между *–(ы)п* и *чар* > *–(ы)пчар*, сопровождающееся слиянием компонентов рассматриваемой ГАК в единое слово, – маркер утраты лексическими компонентами самостоятельности.

Методами экспериментальной фонетики процессы синтезации мультивариативной формы настоящего времени на *(–)чар* подтвердить нельзя ввиду отсутствия звуковых записей памятников «шоского наречия», а также других источников того времени, в которых содержатся данные формы. Являются ли описанные процессы отражением разной степени синтезации мультивариативной формы на *(–)чар* или результатом неустоявшейся орфографии – вопрос открытый и требует дополнительных исследований.

Процессы синтезации формы настоящего времени на *(–)чар* сопровождаются редукцией аффикса деепричастия *-(Ы)п.* Табл. 5 наглядно показывает, что аффикс *-(Ы)п* употребляется после всех основ на гласный и незначительного количества основ на согласный. Он отсутствует у большинства основ на согласный, в том числе у всех основ с аффиксами залогового типа в ауслауте, а также у вспомогательных глаголов в аналитических глагольных конструкциях, при сохранении *-(Ы)п* у основного глагола. Например: младп. шор. *көле-п чар-ы* ‘одаривает’, *ра-п чар-ы* ‘удаляется’; *сал-ып чар* ‘кладет’, *көдӱр-ӱп-чар-ым* ‘поднимаю’, но *айт-чар* ‘толкует’, *корук-чар-ым* ‘боюсь’, *чат-чар-ы* ‘лежит’; *тӱж-ӱр-чар* ‘ввергает; погружает’ (от *тӱш-* ‘спускаться’), *пожы-т-чар-быс* ‘отпускаем’ (от *пожа-* к. ‘освободиться’); *айдып* ***сал****-чар-ы* ‘говорит’, *көөп* ***сал****-чар-лар* закапывают’, но *сагышка кал-****ып*** *чар-зар* ‘задумываетесь’ (от сложного глагола *сагышка кал-* ‘заботиться’).

В табл. 7 представлены результаты процесса элиминации аффикса деепричастия *–(Ы)п* в памятниках «шорского наречия».

Табл. 7. Элиминация аффикса -(Ы)п после основ на согласный в формах настоящего времени на (–)чар в памятниках «шорского наречия»

Table. 7. Elimination of the affix -(I)p after the bases on the consonant in the forms of the present tense on (–)char in the monuments of the "Shor dialect"

Основы с согласным в ауслауте			Квазианалитические *(-NF$_п$) чар*			Синтетические *(-NF$_п$)-чар*			
Всего	(Voice) NF$_п$	Voice / (Voice) (NF$_п$)	Всего	(Voice) NF$_п$	Voice (NF$_п$)	Всего	NF$_п$	Voice (NF$_п$)	(Voice) (NF$_п$)
49	3	46	12	3	9	37	-	10	27
100 %	6,1 %	93,9 %	100 %	25 %	75 %	100 %	0 %	27 %	73 %

Примечание 1 к табл. 7. Глосса Voice означает залог. В наших материалах встречаются основы с аффиксами большинства залогов: пассивного, возвратного и побудительного. Основы с аффиксом взаимного залога нам пока не встретились.

Как показано в табл. 7, полная форма деепричастия на *-ып* от основ с согласным в ауслауте встречается в наших материалах всего в трех случаях, что составляет 6,1 % от основ данного типа. Причем она возможна только в квазианалитических конструкциях без залогового аффикса, где она составляет 25 %. В синтетических формах настоящего времени она отсутствует как в основах с залоговыми показателями, так и в основах без них. Следовательно, тенденция к элиминации *–ып* в основах, кончающихся на согласную, в младописьменном шорском языке уже существовала, и была обусловлена формой глагола настоящего времени. В квазианалитических формах она присутствует у всех глаголов без залогового показателя (25 %) и отсутствует у всех основ, содержащих таковые (75 %), в

синтетических формах она не встречается вообще. Следовательно, в памятниках «шорского наречия» тенденция к элиминации *–ып* в основах, кончающихся на согласную, уже существовала, но правило о падении *-ып* работало спорадически.

5. Заключение

Проведенное исследование позволяет нам сделать ряд выводов.

Полученные результаты вносят заметный вклад в документацию языка малочисленного тюркского народа Южной Сибири – шорского, имеющего неглубокую письменную традицию и находящегося под угрозой исчезновения. Введение в научный оборот нового обширного эмпирического материала по кондомскому (шорскому) наречию – первому варианту письменного шорского языка, а также грамматическое описание созданного на платформе «ЛингвоДок» подкорпуса первых православных книг на шорском наречии способствует решению одной из важнейших задач лингвистики – сохранению исчезающих языков.

Первые письменные памятники шорского языка *Священная история на шорском наречии* [4] и *Указание пути в Царствие Небесное на шорском наречии* [5] – переводы православных текстов с алтайского на кондомское (шорское) наречие, сделанные носителем шорского языка о. Иоанном Матвеевичем Штыгашевым и изданные Алтайскими миссионерами в последней трети XIX в., являются образцами младописьменного шорского языка.

Младописьменный шорский язык – это слабо нормированный шорский язык, функционировавший с 1883 г. и до появления первых советских учебников шорского языка в 20-х годах XX в., формировавшийся путем создания миссионерами Алтайской Духовной Миссии первого шорского алфавита на кириллице на основе графики и орфографии русского языка до орфографической реформы 1917–1918 гг. с добавлением графем ‹ӧ›, ‹ӱ›, ‹ҥ› для шорских звуков в тот период, когда грамматика шорского языка еще не была написана, и поэтому точно фиксировавший особенности произношения носителей шорского языка. Это – письменный вариант кондомского (шорского) наречия, который практически не изучен.

Первый вариант письменного шорского языка имеет ряд существенных отличий от современного нормативного шорского языка, в том числе и в области временных глагольных форм. Так, из трех форм настоящего времени, отмеченных в кондомском (шорском) наречии авторами *Грамматики алтайского языка*, *јадыр(ы), јад(ы) и јар(ы)* (произносятся: *чадыры*, *чады* и *чары*), в младописьменном шорском языке имеется только одна форма настоящего времени с показателем *(–)чар*, которая в современном нормативном шорском языке отсутствует, но является одной из форм настоящего времени в кондомском диалекте современного шорского языка. В случае с материалами из памятников «шорского наречия» сложно говорить о диалектах, так как в то время литературный, а точнее нормированный, шорский язык, по отношению к которому выделяются диалекты, еще не сформировался.

Форма на *(–)чар(ы)* является мультивариативной, объединяющей квазианалитические (младп. шор. *кӧлеп чар-ы* ‘одаривает’), дефисные, переходные к синтетическим (младп. шор. *пілдір–чар-ы* ‘означает’), и синтетические (младп. шор. *айтчар* или *айтчар-ы* ‘говорит’) глагольные конструкции настоящего времени с одним и тем же показателем.

В памятниках «шорского наречия» квазианалитические и синтетические формы настоящего времени на *(–)чар* имеют почти полную парадигму спряжения. Отсутствуют только формы второго лица единственного числа в утвердительной форме, а также формы второго лица единственного и множественного числа в отрицательной форме. В исследуемых источниках нам не встретились ситуации, где бы они могли появиться. Наблюдается тенденция к элиминации аффикса третьего лица единственного числа *-ы* в синтетических формах настоящего времени на *(–)чар*.

Для младописьменного шорского языка характерны процессы синтезации формы настоящего времени на *(–)чар(ы)*, происходящие: 1) как за счет редукции аффикса деепричастия на *–(Ы)п* после основ на согласный, 2) так и путем сокращения вариативности орфографии данной формы вследствие утраты пробела и дефиса между деепричастием на *-(Ы)п* и показателем *(–*

)чар(ы), сопровождающееся слиянием компонентов рассматриваемой ГАК в единое слово и обусловленной сохранением, ослаблением или утратой самостоятельности, составляющих ее лексических компонентов – основного и вспомогательного глаголов. Правило отпадения *-(Ы)п* в основах, оканчивающихся на согласную, в младописьменном шорском языке еще не нормировано и работает спорадически; не нормирована и орфография слитно-раздельно мультивариативной формы настоящего времени на *(–)чар*.

В статье предложена форма представления показателя настоящего времени *(–)чар* и формы его обозначения в квазианалитических, дефисных и синтетических конструкциях настоящего времени в младописьменном шорском языке, что расширяет возможности автоматизированного Парсера шорского языка в анализе не только современных, но и ранних шорских текстов.

Список сокращений

Глоссы

Abl – аблатив (исходный)
Acc – аккузатив (винительный)
Add – аддитиная частица (и, ни, же, ведь, и т. д.)
Caus – каузатив, побудительный залог
Conv – деепричастие
$Conv_п$ – деепричастие на *–(Ы)П* (последовательное)
Conv.Neg – отрицательная форма деепричастия
Dat – датив (дательный)
Fut – будущее время
Gen – генитив (родительный)
Instr – инструментальный (творительный)
Loc – локатив (местный)
NF – неконечная форма (not finished | not finitum), грамматикализованный показатель
$NF_п$ – (not finished | not finitum), грамматикализованный показатель, исторически восходящий к $Conv_п$, используемый для образования ряда синтетических и аналитических словоформ, указывает на незаконченность словоформы
Past – прошедшее время
Pl – множественное число
Pos. – принадлежность
Pred – предикак
PredPl – 3 лицо множественное число предиката
Pres – настоящее время
Refl – рефлексив, возвратный залог
Sg – единственное число
3.Sg.Pred – 3 лицо единственное число предиката
br, – краткие личные окончания
mix – смешанные личные окончания

Общие сокращения

1, 2, 3 – лица
ГАК – глагольная аналитическая конструкция
ед. ч. – единственное число
л. – лицо

Языки и диалекты

к. – кондомский диалект шорского языка

конд. – кондомское, то есть шорское наречие
младп. шор. – младописьменный шорский язык
шор. – шорский язык

Список литературы / References

[1]. Перепись 2010 – Всероссийская перепись населения 2002 г. Официальный сайт – https://www.gks.ru/free_doc/new_site/perepis2010/perepis_itogi1612.htm / All-Russian Population Census 2002 Official website – https://www.gks.ru/free_doc/new_site/perepis2010/perepis_itogi1612.htm /

[2]. Есипова 2016 – Есипова А.В. Шорский язык / «Язык и общество. Энциклопедия». Отв. ред. В.Ю.Михальченко. М.: «Азбуковник», 2016. – 872 с. [РИНЦ] – Стр. 571-576. / A.V. Esipova, "Language and society. Encyclopedia". Ed. by V.Y.Mikhalchenko. M.: "Azbukovnik", 2016. – 872 p. [RSCI] – PP. 571-576.

[3]. Есипова, Норманская 2022 – Есипова А.В., Норманская Ю.В. Архаические графико-фонетические особенности в первых православных памятниках на шорском наречии (на материале книги «Указание пути в Царствие Небесное...») // Кириллические памятники на уральских и алтайских языках. - I том. - Графико-фонетические особенности книг XIX в. — Москва: Издательская группа «Альма Матер», 2022. — С. 493-515. / A.V. Esipova, Yu.V. Normanskaya. Archaic graphic and phonetic features in the first Orthodox monuments in the Shor dialect (based on the book “Indicating the Way to the Kingdom of Heaven...”) // Cyrillic monuments in the Ural and Altai languages. - Volume I. - Graphic and phonetic features of the books of the XIX century. - Moscow: Alma Mater Publishing Group, 2022. - PP. 493-515.

[4]. История 1883 – Священная исторія на шорскомъ нарѣчіи для инородцевъ восточной половины кузнецкаго округа. Изданіе Православнаго Миссионерскаго Общества. – Казань: Типография В.М. Ключникова, 1883. – 207 с. / The Sacred history in the Shor dialect for allogeneous people of the eastern part of the Kuznetsk district. Publication of the Orthodox Missionary Society. – Kazan: V.M. Klyuchnikov Printing House, 1883. – 207 p.

[5]. Указание 1884 – Указание пути в Царствие Небесное на шорском наречии. Тегрідіҥ чарыгынга кірчеҥ чолды кödӱсче. Издание Православного Миссионерского Общества. – Казань: Типография В.М. Ключникова. – 1884. / Indication of the Way into the Kingdom of Heaven in the Shor dialect. Tegridiŋ čarygynga kirčeŋ čoldy ködüsče. Publication of the Orthodox Missionary Society. – Kazan: V.M. Klyuchnikov Printing House, 1884.

[6]. Грамматика 1869 – Грамматика алтайского языка / Составлена членами Алтайской миссии. – Казань: Унив. тип., 1869. – 299 с. / Grammar of the Altai language / Comp. by members of the Altai mission. – Kazan: Univ. typography, 1869. – 299 p.

[7]. Вербицкий 1884 – Вербицкий В.И. Словарь алтайского и аладагского наречий тюркского языка. – Казань: Православ. миссионер. о-во, 1884. – 494 с. / V.I. Verbitskiy. Dictionary of the Altai and Aladag dialects of the Turkic language. – Kazan: Orthodox missionary community, 1884. – 494 p.

[8]. Дыренкова 1941 – Дыренкова Н.П. Грамматика шорского языка. – М.–Л.: Изд-во Акад. наук СССР, 1941. – 306 с. / N.P. Dyrenkova. Grammar of the Shor language. – Moscow–Leningrad: USSR Academy of Sciences Publishing House, 1941. – 306 p.

[9]. Чиспиякова 1991 – Чиспиякова Ф.Г. Учебное пособие по диалектологии шорского языка. – Новокузнецк: [б. и.], 1991. – 47 с. / F.G. Chispiyakova. Textbook on dialectology of the Shor language. – Novokuznetsk: [no publisher], 1991. – 47 p.

[10]. Радлов 1866 – Радлов В. В. Образцы народной литературы тюркских племен, живущих в Южной Сибири и Дзунгарской степи. Ч. 1. Поднаречия Алтая: алтайцев, телеутов, черневых и лебединских татар, шорцев и саянцев. Proben der Volksliteratur der türkischen Stänne Süd-Sibiriens, gesammelt und übersetzt von D. W. Radloff. I. Teil: Die Dialecte des eigentlichen Altai: der Altaier, Teleuten, Lebed-Tataren, Schoren und Sojonen. – Санктпетербург, 1866. – 420 с. / V. Radloff. Samples of folk literature of Turkic tribes living in Southern Siberia and the Dzungarian steppe. Part 1. Sub-dialects of Altai: Altaians, Teleutes, Chernev and Lebedinsky Tatars, Shors and Sayans. Proben der Volksliteratur der türkischen Stänne Süd-Sibiriens, gesammelt und übersetzt von D. W. Radloff. I. Teil: Die Dialecte des eigentlichen Altai: der Altaier, Teleuten, Lebed-Tataren, Schoren und Sojonen. – St. Petersburg, 1866. – 420 p.

[11]. Уртегешев, Селютина 2023 – Уртегешев Н. С., Селютина И. Я., Добринина А. А., Рыжикова Т. Р. Фонетические трансформации в аналитических формах тюркского глагола / Отв. ред. д-р филол. наук И. В. Шенцова. Ин-т филологии Сиб. отд-ния Рос. акад. наук. – Новосибирск: Академиздат,

2021. – 256 с. / N. S. Urtegeshev, I. Ya. Selyutina, A. A. Dobrinina, T. R. Ryzhikova Phonetic transformations in analytical forms of the Turkic verb / Ed. Doctor of Philology I. V. Shentsova. Institute of Philology of the Siberian Branch of the Russian Academy of Sciences. Novosibirsk: Akademizdat, 2021. 256 p.

Информация об авторах / Information about authors

Алиса Васильевна ЕСИПОВА – доктор филологических наук, доцент, ведущий научный сотрудник Отдела урало-алтайских языков Института языкознания РАН. Сфера научных интересов: общее языкознание, типология, тюркские языки, шорский язык, словообразование, морфология, синтаксис, документация исчезающих языков, электронные корпуса тюркских языков.

Alisa Vasil'yevna ESIPOVA – PhD in Philology, Associate Professor, Leading Research Associate of Department of Ural-Altaic languages of Institute of Linguistics, Russian Academy of Sciences. Research interests: general linguistics, typology, Turkic languages, Shor language, word formation, morphology, syntax, documentation of endangered languages, electronic corpus of Turkic languages.

DOI: 10.15514/ISPRAS-2023-35(6)-21

Рефлексы прамонгольских гласных в южно-монгольских языках

З.И. Чушкаева, ORCID: 0000-0003-0061-5278 <*ms.zayanaa@mail.ru*>

Калмыцкий научный центр РАН,
Россия, 358000, Элиста, ул. И. К. Илишкина, д. 8.

Abstract. Монгольские языки – языковая семья, включающая в себя несколько близкородственных языков Монголии, Китая, России и Афганистана. Согласно данным лексикостатистики, они распались около V в. н. э. К южно-монгольским языкам принято относить дагурский, шира-югурский, дунсянский, баоаньский, язык ту (монгорский). Южно-монгольские языки, по сравнению с другими группами монгольских языков, подверглись влиянию китайских и тюркских языков, различные исторические события оказали несомненное влияние на них.

Ключевые слова: дунсянский язык; шира-югурский язык; баоаньский язык; монгольский язык; словарь; рефлексы.

Для цитирования: Чушкаева З. И. Рефлексы прамонгольских гласных в южно-монгольских языках. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 331–336. DOI: 10.15514/ISPRAS–2023–35(6)–21.

Благодарности: Исследование проведено при финансовой поддержке РНФ в рамках научного проекта № 22-78-10152 «Корпус старокалмыцких текстов на „ясном письме“ на платформе Lingvodoc: новые подходы к цифровизации письменного наследия».

Reflexes of proto-Mongolian vowels in South Mongolian languages

Z.I. Chushkaeva, ORCID: 0000-0003-0061-5278 <*ms.zayanaa@mail.ru*>

Kalmyk Scientific Centre of Russian Academy of Sciences,
8, Ilishkin st., 358000, Elista, Russia.

Abstract. The Mongolian languages are a language family that includes several closely related languages of Mongolia, China, Russia and Afghanistan. According to lexicostatistics, they broke up around the 5th century AD. Dagur, Shira-Yugur, Dongxiang, Bao'an, Tu (Monguor) are commonly referred to as South Mongolian languages. The South Mongolian languages, in comparison with other groups of Mongolian languages, were influenced by Chinese and Turkic languages, various historical events had an undoubted influence on them.

Keywords: Dongxiang language, Shira-Yugur language, Baoan language, Mongolian language, dictionary, reflexes.

For citation: Chushkaeva Z. I. Reflexes of proto-Mongolian vowels in South Mongolian languages. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 6, 2023. pp. 331-336 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-21.

Acknowledgements: The reported study was funded by Russian Science Foundation, project no. 22-78-10152 ‘Corpus of Old Kalmyk Clear Script Texts on the Lingvodoc Linguistic Platform: New Approaches to the Digitization of Written Heritage’.

1. Введение

Языки южно-монгольской группы (дунсянский, шира-югурский, баоаньский) имеют большое значение в реконструкции протомонгольского языка, этимологии лексем монгольских языков. Материалы по указанным языкам до сих пор не введены в научный оборот, и, следовательно, сами языки недостаточно исследованы: в настоящее время имеется небольшое количество работ в основном по дунсянскому и шира-югурскому языкам по сравнению с относительной изученностью других монгольских языков. В отечественной литературе наиболее известны труды Б. Х. Тодаевой. Благодаря ее исследованиям в монголоведной науке появились новые данные о районах расселения монгольских народностей на территории КНР, первые системные описания их фонетики и грамматики. Именно из ее работ, основанных на лингвистическом материале, собранном ею лично в ходе экспедиций 1954–1957-х гг. на северо-западе Китая, начиная с Маньчжурии и до Синьцзян-Уйгурского автономного района, мировая общественность узнала об особенностях дагурского, дунсянского, баоаньского, монгорского языков и других языков и наречий всех монголоязычных народностей, представленных на территории Китая. Ее труды внесли значительный вклад в отечественную и мировую монголоведную науку [1–5].

Несмотря на то, что монгольские языки исследуются достаточно длительное время, мнения ученых относительно статуса некоторых монгольских языков и диалектов до сих пор разнятся. Это касается и языков южно-монгольской группы.

Обозначенная проблема требует особого внимания и дальнейших исследований монгольского языка и его диалектов на материале описываемых языков с привлечением компьютерных программ платформы Lingvodoc. На материале южно-монгольских языков были выявлены некоторые соответствия на основе установленных связей между словами языков южно-монгольской группы.

Автором статьи были переведены на русский язык с китайского языка три словаря (Дунсянско-китайский словарь, Шира-югурский словарь, Баоаньский словарь), затем они были загружены на платоформу Lingvodoc, с помощью функционала платформы проведены сравнения дунсянского, шира-югурского и баоаньского языков. С помощью анализа когнатов была выявлена значительная разница в транскрипции. Тем не менее, мы смогли выявлять ряды соответствий, которые имеются в языке, а также достаточно быстро анализировать похожие звуки в рядах соответствий, выстраивая таблицы соответствий.

На первых этапах работы были переведены с китайского языка на русский два издания «Дунсянско-китайского словаря», под редакцией Ма Го Чжуна и Чен Юань Луна [6].В первом издании словаря описывается более 10800 слов, а во втором издании более 11733 слов (включая производные слова, составные слова, названия местностей, небольшое количество идиом и различные формы произношения некоторых слов). Наличие языковых контактов играют важную роль в развитии языка, поэтому в речи дунсян немалое количество заимствований. Дунсяне также являются мусульманами, поэтому их лексика содержит множество терминов, связанных с исламом, что затрудняло процесс перевода.

Следующая работа, загруженная на платформу — «Шира-югурский письменно-монгольский словарь», содержащий 3616 слов. Работа над сбором лексики проводилась аспирантами из Университета Внутренней Монголии в июне 1980 г [7].

«Баоаньский словарь» — результат работы группы ученых по исследованию баоаньского языка из Университета Внутренней Монголии. Словарь содержит 5078 слов, который также был переведен с китайского языка на русский и загружен на платформу Лингводок [8].

Для изучения транскрипций, сделанных китайскими учеными, была просмотрена работа «Introduction, Grammar, and Sample Sentences for Baoan» (У Чаолу) [9], которая была написана с помощью МФА. Для более точного понимания была проверена точность записей (У Чаолу)

[9] по сравнению с (Бао Чао Лу) [7] и (Жалсаном) [10], где были выявлены соответствия и различия, которые указаны ниже в таблице.

В табл. 1 приведены некоторые слова из трех словарей шира-югурского словаря. Интересно то, что у (У Чаолу) [9] множество сходств с обозначением транскрипций китайского ученого (Бао Чао Лу) [7] , однако некоторые слова отсутствуют в другом китайском источнике у (Жалсана) [10].

В результате сравнения были обнаружены интересные различия, которые не встречаются в других источниках. Полученные результаты позволили составить таблицу, в которой показаны соответствия гласных в монгольских языках из EDAL (Алтайского этимологического словаря) [11], праформы из работы Х. Нугтерена «Mongolic Phonology and the Qinghai-Gansu Languages LOT dissertations» [12] и южно-монгольских языков (см. табл. 2).

Из табл. 2 видно, что соответствия праформ гласных значительно отличаются от транскрипций в китайских источниках. Если же в первых примерах такие соответствия, как *u-, *e, *a, *ö, *o, имеются во всех языках, то соответствия *eü/*öe, *üü и *idu у Х. Нугтерена вызывает некоторые вопросы, которые требуют дальнейшего исследования.

Табл. 1. Соответствия и различия в транскрипциях в словарях шира-югурского языка
Table 1. Results for different strategies

	Introduction, Grammar, and Sample Sentences for Jegtin Yogu [9]	Шира-югурский словарь [7]	Шира-югурский словарь [10]		Introduction, Grammar, and Sample Sentences for Jegtin Yogu [9]	Шира-югурский словарь [7]	Шира-югурский словарь [10]
[a]	xan- ‘satisfy’	xan- ‘удовлетворять’	–	[a:]	xa:n ‘king’	xa:n ‘хан, император’	–
[ə]	ʤəgə ‘call goat’’	–	–	[ə:]	ʤəgə: 'call baby goat’	hʤedʒe: ‘звать овец’	–
[e]	sergə ‘wake’	sergə-‘возрождаться’	ser- ‘просыпаться’	[e:]	se:rgə ‘close'	se:rgə ‘недавно’	–
[i]	ʃike ‘big’	ʃike ‘большой’	ʃge- ‘большой’	[i:]	ʃi:kə- ‘urinate’	ʃi:- ‘мочиться’	ʃii- ‘ходить по малой нужде’
[ɔ]	ɔrɔ 'enter'	ɔrɔ- 'вступать, входить'	oro- 'входить'	[ɔ:]	ɔrɔ 'untrained'	ɔrɔ: 'непослушный’	–
[ø]	tølø - 'pay for'	tølø- 'возмещать'	-	[ø:]	tølø: 'in order to'	tølø: 'для, ради'	-
[u]	ula 'sole'	ula 'подошва'	ula 'стопа'	[u:]	u:la 'mountain'	u:lo 'гора'	uula 'гора'
[ʉ]	ʤʉ:g 'stable'	ʤʉ:g 'только'	-	[ʉ:]	dsʉger	ʤʉge:r 'спокойный'	-
[y]	dyge 'period'	dyge 'во время'	-	[y:]	dy:ge 'younger brother'	dy:ge 'младший брат'	dyy 'младший'

Табл. 2. Соответствия гласных в монгольских языках
Table 2. Matching vowels in Mongolian languages

Пра-монгольский [12]	Пра-монгольский [11]	Шира-югурский словарь [10]	Шира-югурский словарь [7]	Баоаньский словарь [1]	Баоаньский словарь [8]	Дунсянский словарь [2]	Дунсянский словарь [6]
*u	*u	[u]	[u]	[u]	[u]	[u]	[u]
*e	*e в анлауте	[e]	[e]	[e]	[e]	[e]	[e]
*e	*e в инлауте	[e]	[e]	[a]	[e]	[ie]	[ie]
*a	*a	[a]	[a]	[a]	[a]	[a]	[a]
*ö	*ö	[ø]	[ø]	[o]	[ø]	[o]	[o]
*o	*o	[o]	[ɔ]	[o]	[ø]	[o]	[o]
*o	*aɣu, *ow	[uu]	[uː]	[au]	[o]	[o]	[u]
*i	*i	[ə]	[ə]	[i]	[i]	[ɨ]	[i]
*eü/*öe	*üɣü/*eɣü	[yy]	[y]	?	?	[u]	[u]
*üü	*eɣü/*öɣü	[yy]	[yː]	?	?	[u]	?
*idu	*iCü	[u]	[ʉ]	[e]	[u]	[u]	[u]

2. Заключение

Загруженные китайские словари южно-монгольских языков имеют высокий потенциал в сравнительно-исторических исследованиях, кроме того, они имеют большой объем, и, следовательно, такой большой массив данных впервые может стать материалом для комплексного исследования фонетики южно-монгольских языков.

Список литературы / References

[1]. Тодаева Б. Х. Баоаньский язык. М.: Наука, 1964. 158 с.
[2]. Тодаева Б. Х. Дунсянский язык. М.: Восточная литература, 1961. 137 с.
[3]. Тодаева Б.Х. Монгорский язык. М.: Изд-во «Наука». Главная редакция восточной лит-ры, 1973, 391 с.
[4]. Тодаева Б. Х. Язык монголов Внутренней Монголии. Очерк диалектов. М.: Наука, 1985, 134 с.
[5]. Тодаева Б.Х. Дагурский язык. М.: Наука. Гл. ред. вост. лит., 1986, 189 с.
[6]. Ma Guo Zhong, Chen Yuan Long. 东乡语汉语词典 (Dongxiang-Chinese dictionary). Lanzhou: Gansu minzuchubanshe 2011, 500 p.
[7]. Bao Chaolu. 东部裕固语词汇（Eastern Yogur Vocabulary）. Hohhot: Nei menggu renmin chubanshe. 1985, 157 p.
[8]. Chen Naixiong. 保安语词汇(Bao'an Dictionary). Hohhot: Nei Meng Gu Ren Min Chubanshe, 1986. 239 p.
[9]. Chaolu Wu. Introduction, Grammar, & Sample Sentences for Dongxiang. In Victor H. Mair. (ed.) Sino-Platonic Papers. 1994 (55), 1–34. University of Pennsylvania.
[10]. Zhaonasitu. 东部裕固语简志 (Introduction to the Eastern Yugur language). Beijing: Minzu Chubanshe. 1981, 122 p.
[11]. S.A. Starostin, A. V. Dybo, O. A. Mudrak. An etymological dictionary of Altaic languages. Vol. I. Leiden, Boston, 2003.
[12]. Nugteren H. Mongolic phonology and the Qinghai-Gansu languages. Doctoral dissertation, Leiden University, 2011. 563 p.

Информация об авторах / Information about authors

Заяна Игоревна ЧУШКАЕВА – младший научный сотрудник отдела языкознания Калмыцкого научного центра РАН. Сфера научных интересов: лингвистика, изучение языков, их структуры, происхождения, истории, функционирования и законов развития.

Zayana Igorevna CHUSHKAEVA – Junior researcher at the Department of Linguistics of the Kalmyk Scientific Center of the Russian Academy of Sciences. Research interests: linguistics, study of languages, their structure, origin, history, functioning and laws of development.

DOI: 10.15514/ISPRAS-2023-35(6)-22

Решение проблемы многоязычия в международном научно-техническом информационном пространстве

[1] *К.К. Колин*, ORCID: 0000-0002-8187-9314 <kolinkk@mail.ru>
[1,2] *А.А. Хорошилов*, ORCID: 0000-0003-4885-3232 <khoroshilov@mail.ru>
[2,3] *А.В. Кан,* ORCID: 0000–0001–9410–406X <kanav@nrczh.ru>
[1] *Ю.В. Никитин,* ORCID: 0000-0002-7641-0247 <yuri.v.nikitin@gmail.com>

[1] *Федеральный исследовательский центр "Информатика и управление" РАН, 119333, Россия, г. Москва, Вавилова, д.44, кор.2.*
[2] *Национальный исследовательский центр «Институт имени Н.Е. Жуковского», 125319, Россия, г. Москва, ул. Викторенко, д.7.*
[3] *Московский авиационный институт (национальный исследовательский университет), 125993, Россия, г. Москва, Волоколамское шоссе, д. 4.*

Аннотация. Решение проблемы многоязычия в международном научно-техническом информационном пространстве связано с технологиями машинного перевода (МП). Процесс перевода в рамках концепции фразеологического концептуального перевода текстов можно представить, как процесс передачи смыслового содержания исходного текста, средствами выходного языка. В рамках этой концепции перевод текстов обеспечивается понятийным анализом исходного текста и трансформацией его смыслового содержания на целевой язык. Этот подход базируется на понимании закономерностей функционирования естественных языков и теоретических преставлений о смысловой структуре текстов. Основой подхода являются технологии автоматизированного формирования тематических словарных баз, адекватно отражающих понятийный состав различных тематических областей.

Ключевые слова: проблемы многоязычия в международном научно-техническом информационном пространстве; машинный перевод; естественный язык; машинные грамматики; семантико-синтаксический и концептуальный анализ текстов; фразеологический концептуальный перевод текстов.

Для цитирования: Колин К.К., Хорошилов А.А., Кан А.В., Никитин Ю.В. Решение проблемы многоязычия в международном научно-техническом информационном пространстве. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 337–346. DOI: 10.15514/ISPRAS–2023–35(6)–22.

Solving the Multilingualism Problem in the International Scientific and Technical Information Space

[1] *K.K. Kolin*, ORCID: : 0000-0002-8187-9314 <kolinkk@mail.ru>
[1,2] *A.A. Khoroshilov*, ORCID: 0000-0003-4885-3232 <khoroshilov@mail.ru>
[2,3] *A.V. Kan*, ORCID: 0000–0001–9410–406X <kanav@nrczh.ru>
[1] *Yu.V. Nikitin*, ORCID: 0000-0002-7641-0247 <yuri.v.nikitin@gmail.com>

[1] *Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44 building 2, Vavilova st., Moscow, 119333, Russia.*
[2] *Analytical Department of the National Research Center "Zhukovsky Institute", 7, Viktorenko st., Moscow, 125319, Russia.*
[3] *Moscow Aviation Institute (National Research University), 4, Volokolamskoye Shosse, Moscow, 125993, Russia.*

Abstract. The solution to the problem of multilingualism in the international scientific and technical information space is connected with machine translation (MP) technologies. The translation process within the framework of the concept of phraseological conceptual translation of texts can be represented as the process of transmitting the semantic content of the source text by means of the output language. Within the framework of this concept, the translation of texts is provided by the conceptual analysis of the source text and the transformation of its semantic content into the target language. This approach is based on understanding the laws of the functioning of natural languages and theoretical ideas about the semantic structure of texts. The basis of the approach is the technology of automated formation of thematic dictionary bases that adequately reflect the conceptual composition of various thematic areas.

Keywords: problems of multilingualism in the international scientific and technical information space, machine translation, natural language, machine grammars, semantic-syntactic and conceptual analysis of texts, phraseological conceptual translation of texts.

For citation: Kolin K.K., Khoroshilov A.A., Kan A.V., Nikitin Yu.V. Solving the multilingualism problem in the international scientific and technical information space. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 6, 2023. pp. 337-246 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-22.

1. Введение

Современное международное информационное пространство быстро развивается в результате все более широкого распространения компьютерных телекоммуникаций, которые сегодня охватывают все страны мира и становятся неотъемлемой частью их культуры, научно-технологической и социально-экономической деятельности [1]. При этом особую важность приобретает научно-техническая информация, которая содержит сведения о новых достижениях в области науки и технологий, здравоохранения, организации общественного производства, а также о методах противодействия новым вызовам и угрозам XXI века.

Серьезная лингвистическая проблема использования такой информации специалистами различных стран состоит в том, что она, как правило, содержит большое количество специальных терминов, требующих адекватного перевода. А этого современные средства перевода текстов в необходимой степени еще не обеспечивают. Поэтому проблема повышения качества перевода текстов научно-технической информации и является той актуальной и стратегически важной проблемой, без решения которой эффективное использование передовых достижений научно-технического прогресса и международное научно-техническое сотрудничество практически невозможно.

Необходимо отметить, что попытки решения этой проблемы предпринимались неоднократно, начиная с середины минувшего века, когда появились средства вычислительной техники, и продолжаются до сих пор. Однако полученные в них результаты еще нельзя признать удовлетворительными. Наглядным примером здесь может служить современное состояние этой проблемы в странах Европейского экономического союза, для

которых систему высококачественного автоматизированного перевода текстов создать пока еще не удалось [2].

Аналогичная проблема существует и в странах Евразийского экономического союза, а также в странах, которые являются членами БРИКС, ШОС и СНГ. Причем, здесь она осложняется еще и существенным различием алфавитов, на которых представлена текстовая информация. Так, например, в Китае используются иероглифы, в Индии – слоговое письмо, а в других странах этих новых объединений государств – латиница и кириллица. Все это привело к созданию так называемого языкового барьера при доступе к разноязычной информации. В настоящее время преодоление этого барьера осуществляется с помощью систем машинного перевода.

В настоящей работе показано состояние этой проблемы в России и возможности использования уже полученных результатов в интересах развития многоязычного международного пространства научно-технической информации.

2. Современное состояние проблемы машинного перевода текстов

Анализ подходов к решению проблемы машинного перевода, приведенный в работе [3] показал, что в настоящее все большую популярность завоевывают системы машинного перевода, базирующиеся на нейросетевых подходах (Neural Machine Translation, NMT) [4-6]. Так, не выдержав конкуренции с современными системами NMT, с IT-рынка ушли традиционные системы на основе правил (Rule-based Machine Translation, RBMT) [7,8,9], а также системы статистического перевода (Statistical based machine translation, SMT)[10]. Действительно, NMT системы, примером которых может служить портальный сервис перевода Google Translate, показывают удовлетворительные результаты при переводе текстов по некоторым тематическим областям в ряде направлений перевода. В основу этих систем положена модель глубокого обучения, представляющая собой нейросеть, состоящую из ряда слоев, на каждом из которых производится извлечение большого числа признаков из текстовых данных, необходимых для реализации процесса перевода аналогичных текстовых ситуаций.

Широкому применению этих технологий способствовало наличие большого числа свободно распространяемых программных средств, обеспечивающих обучения моделей перевода, а также возможности их реализации и встраивания в любые технологические процессы. Для обучения модели необходимы только исходные данные, представляющие собой тексты, фрагментированные на предложения на исходном языке, и переводы этих предложений на целевом языке.

Обычно для создания NMT-переводчика достаточно только «дообучить» одну из доступных предобученных моделей и использовать программные средства, обеспечивающие реализацию процесса перевода. При этом полностью исключаются трудоемкие работы по создания языковых моделей и технологий трансформации текстов с одного ЕЯ на другой. Но при этом всегда нужно было принимать во внимание, что обученная модель реализует только *те возможности, которые были заложены в нее в процессе обучения*.

Казалось бы, замечательная технология – без значительных затрат решает все проблемы перевода с любых языков по любой тематике. Но, как оказалось при ближайшем рассмотрении, эти технологии, изначально при их разработке, потребовали огромные финансовые и интеллектуальные ресурсы, посильные лишь таким крупным транснациональным компаниям, как Google, IBM и др. Поэтому необходимо понимать, что, если возникнет необходимость создать «с нуля» новое направление перевода по новой тематической области, то, даже при наличии соответствующего программного обеспечения, потребуется специализированное дорогостоящее оборудование и огромные массивы размеченных двуязычных параллельных текстов. В случае, если не окажется в наличии одного из этих компонентов, то этой технологией невозможно будет воспользоваться.

Другими словами, технологии NMT базируются на принципе *перевода последовательности слов на основе аналогии с ранее выполненными переводами*, которые в явном виде не предполагает реализацию основополагающего принципа профессионального перевода: *передачу смыслового содержания исходного текста средствами другого языка*.

А как быть с учетом ряда таких сложных явлений ЕЯ, как вариативность представления смысла в текстах, явления синонимии, гипонимии в ЕЯ? Необходим также учет явления пресуппозиции и ряда других аномальных ситуаций в языке и речи, которые нужно учитывать при переводе текстов с одного ЕЯ на другой.

Таким образом, как не хотелось бы сэкономить на качественном переводе и обойтись только дешевыми технологиями NMT, без трудозатратных работ по созданию полноценных языковых моделей и адекватных двуязычных фразеологических и терминологических словарей, качественного терминологически обоснованного перевода научно-технических текстов получить невозможно.

Однако, многих из выше перечисленных проблем удалось избежать в технологии *фразеологического машинного перевода*, которая, по сути, является дальнейшим развитием традиционного лингвистического перевода, основанного на правилах, но полностью переосмыслена и дополнена рядом гибких языковых моделей, которые базируются на принципе лингвистической аналогии с возможностью значительного расширения их признакового пространства, необходимого для решения задач смыслового анализа текстов.

3. Концепция фразеологического машинного перевода

Впервые возможность получения высококачественного перевода научно-технических текстов предложил и обосновал военный ученый профессор Г.Г. Белоногов[1] в 1975 году в рамках ***концепции фразеологического машинного перевода*** (Phraseological Machine Translation, PMT) [11-12]. В основу этой концепции было положено понимание того факта, что в процессе перевода необходимо выявить смысловое содержание исходного текста и передать его фразеологическими конструкциями того языка, на который переводится текст.

В рамках этой концепции, реализовывалась возможность моделирования деятельности человека-переводчика при переводе текстов. Она включает процедуры восприятия смыслового содержание переводимого текста (анализ смыслового содержания текста) и «пересказывания» содержания этого текста с помощью той понятийной терминологической системы, к тематике которой относился переводимый текст. При этом осуществляется преобразование понятийной структуры исходного текста в понятийную структуру на целевом языке и выполняется порождение осмысленного терминологически и грамматически связанного текста.

В концепции PMT в качестве основных единиц смысла используются *фразеологические словосочетания, выражающие понятия*. Ведь именно понятия являются теми элементарными мыслительными образами, используя которые можно строить более сложные мыслительные образы, соответствующие переводимому тексту. Поэтому перспективные системы МП должны переводить *не слова и их последовательности, а мыслительные образы, выраженные словами и словосочетаниями* [11-12].

Проф. Г.Г. Белоногов в концепции PMT связал воедино иерархию смысловых единиц текста в формализованную многослойную двуязычную модель текста. По его мнению, иерархия понятийной системы текста составляет основу сложного мыслительного образа всего текста. А фразеологические понятия являются теми базовыми «строительными блоками», на основе

[1] Проф. Белоногов Г.Г. – известный советский и российский ученый в области информатики, компьютерной лингвистики и машинного перевода, доктор технических наук, профессор, академик Международной академии информационных процессов и технологий, один из основоположников отечественной информатики, признанный как у нас в стране, так и за рубежом.

которых формируются смысловые единицы более высоких уровней – предложения, сверхфразовые единства, входящие с состав текста [11-12].

При этом формальным инвариантом смысловой структуры предложения является его предикатно-актантная структура (ПАС). Ее компонентами служат понятия-предикаты (признаки и отношения) и понятия-актанты, выступающие в роли описываемых объектов. Использование в процессе машинного перевода моделей ПАС предложений в качестве смысловых единиц обеспечивает возможность адекватной передачи смыслового содержания исходного текста терминологическими и фразеологическими конструкциями на целевом языке.

В работах [11-12] показано, что, согласно этой концепции, «*...система РМТ должна включать в свой состав многоязычную понятийную базу, содержащую переводные эквиваленты часто встречающихся терминологических словосочетаний, фрагментов фраз, служебных конструкций и отдельных слов, средства анализа смысловой структуры исходного текста, средства формирования понятийной структуры на целевом (языке перевода) и средства порождения (генерации) текста на целевом языке*».

В процессе перевода текстов система использует хранящиеся в этой базе переводные эквиваленты в следующем порядке: сначала для очередного предложения исходного текста делается попытка перевести его как целостную фразеологическую единицу. Далее, в случае неудачи – входящие в его состав наиболее длинные синтаксические конструкция, при их отсутствии – более короткими словосочетаниями, и, наконец, осуществляется пословный перевод тех фрагментов предложения, которые не удалось перевести первыми тремя способами. Фрагменты выходного текста, полученные всеми рассмотренными способами, должны грамматически согласовываться друг с другом (с помощью процедур морфологического и синтаксического синтеза) [11-12].

Нужно также отметить, что проф. Белоногов еще рубеже 50-60 гг. прошлого века разработал уникальную машинную грамматику русского языка, ориентированную на широкое применение принципов лингвистической аналогии при реализации , жесткое соответствие между формой представления слов и их грамматической информацией позволило создать на этой основе новые классы – «*…классы слов, имеющие одинаковые наборы грамматических признаков, соответствующие их формам представления в сходных контекстных окружениях...*»[13].

Идея создания новых классов слов, ориентированных на схожесть грамматических признаков слов и схожесть их синтаксических функций в предложении, была впервые предложена в работе [14] для разрешения грамматической омонимии английских слов. Для решения этой задачи и ряда аналогичных задач, была разработана универсальная центроидно-контекстная языковая модель (ЦКЯМ), основная идея которой заключается *в возможности однозначного выявления свойств модели по ее контекстному окружению*.

Каждая языковая модель использует определенное признаковое пространство, в пределах которого и разрешается конкретная языковая ситуация. В частности, для разрешения грамматической омонимии английских слов использовался набор грамматических характеристик английских слов в конкретных текстовых ситуациях. Решением каждой ситуации было однозначное определение грамматических характеристик слов в конкретном контексте, а также и в значительном числе аналогичных контекстов.

В синтаксической модели текстов на основе использования механизма обобщенных синтагм была разработана иерархия синтаксических конструкций предложений, обеспечивающая при синтаксическом анализе возможность адекватного построения синтаксических структур предложений любой сложности. Эта модель решает задачу построения синтаксической структуры предложения в рамках следующего утверждения: «*...представление синтаксической структуры текстов в виде последовательности контактно расположенных двухбайтовых индексов обобщенных синтагм, обладающих*

грамматическими свойствами конкретных слов-эталонов, позволяет фиксировать грамматические и синтаксические свойства различных отрезков реальных текстов, а также дает возможность в ряде задач распознавать аналогичные по заданным свойствам отрезки текстов...» [15].

Широкое применение принципа лингвистической аналогии и использование динамических последовательно контекстных и центроидно-контекстных языковых моделей (ЦКЯМ), базирующихся на значительном признаковом пространстве, обеспечило возможность однозначного разрешения сложных текстовых ситуаций на различных этапах анализа текста.

4. Основные модели системы РМТ

Функциональные *модули перевода* в процессе многоступенчатой обработки текстов исходного текстов формируют ряд моделей исходного текста – модели его трансформации и модели порожденного текста на целевом языке.

Морфологическая модель слов исходного текста представляет собой (так же, как и в модели NTM) вектор характеристик слов фиксированной длины. Состав этих характеристик имеет различную природу и четкое распределение между грамматическими и семантическими признаками слов. В модели также обозначены их словообразующие и формообразующие характеристики, а также определен набор формальных характеристик, отображающих смысл слов и характеристики их конкретных форм [14].

Семантико-синтаксическая модель предложений исходного текста представляет собой двухмерную матрицу, в которой позициям токенов предложения поставлены в соответствие расширенные векторы их характеристик. На основе этой матрицы реализован ряд синтаксических моделей, а также осуществляется с помощью центроидно-контекстных моделей извлечение и классификация синтаксических конструкций предложения, определяется их роль в предложении, выявляются именные и глагольные словосочетания и устанавливается их структура.

На завершающем этапе производится выявление смыслового каркаса предложения и установление связей между его элементами. Результатом обработки предложений является формирование нескольких логически связанных формальных представлений: в виде бинарных отношений токенов предложения (дерева зависимостей), в виде смыслового каркаса («скелета») предложения, в виде предикатно-актантной структуры. [15].

Модель концептуального анализа исходного текста в качестве основной задачи ставит выявление текстовой системы понятий и преобразование ее в унифицированное формализованное представление. Также в рамках этой модели устанавливаются парадигматические и синтагматические связи между понятиями. На конечном этапе анализа цифровые характеристики всей иерархии моделей исходного текста преобразуются в многослойную цифровую модель метаданных [12].

Сформированная *модель метаданных исходного текста* обеспечивает возможность передачи информации об анализируемом исходном тексте в трансформационную модель. В *трансформационной модели* основным процессом является механизм вероятностного соотнесения понятийной модели исходного текста с понятийной моделью целевого языка. В процессе этого соотнесения учитываются принадлежность понятий к типу двуязычного словаря и степень покрытия синтаксической структуры предложений исходного текста. При этом предполагается, что большая длина текстовых представлений понятий обеспечивает более точный перевод. Обязательным условие смысловой связанности переведенного текста является включение переводных соответствий в единое понятийное пространство. Использование многофакторной вероятностной модели в процессе соотнесения понятий исходного языка с понятиями целевого языка обеспечивает более точный терминологический перевод текста [12].

Основой этой модели является многомиллионный комплекс двуязычных словарей наименований понятий, охватывающий широкий спектр тематических областей. Этот комплекс словарей включает четыре уровня словарей: политематический, тематический пользовательский и словарь ТМ (Translation Memory – память переводчика). Перевод осуществляется в соответствии с приоритетами словарей в следующем порядке: словарь ТМ, пользовательский словарь, тематический словарь и, наконец, политематический словарь.

Словарь ТМ подключается по желанию пользователя. Приоритет словарей обеспечивает возможность пользователю влиять на процесс перевода, путем оперативного формирования словарных статей или установления приоритетов переводных соответствий, имеющихся в словарной базе.

Завершает процесс перевода *модель порождения текста* на целевом языке. В этой модели производятся необходимые локальные перестановки слов внутри словосочетаний и глобальные перестановки словосочетаний в пределах предложения, а также осуществляется грамматическая трансформация форм слов предложения в соответствие с грамматическим строем целевого языка. Этот комплекс трансформаций текстового представления позволяет сформировать терминологически и грамматически связанный переведенный на целевой язык исходный текст.

5. Основные модели системы РМТ

Основой любой системы перевода является его словарная база, каком бы виде она не была бы представлена. Качество перевода обеспечивается наличием в ней наиболее часто встречающихся представлений текстовых ситуаций и возможностью оперативного пополнения отсутствующих в словаре необходимых представлений ситуаций. Текстовые представления ситуаций в словарных базах представлены различными текстовыми конструкция: отдельными словами, именными и глагольными словосочетаниями, словосочетаниями в контекстных окружениях и целыми предложениями. Система в процессе перевода должна извлекать из словарной базы наиболее адекватные словарные конструкции и располагать их в той последовательности, которая будет соответствовать наиболее точной передачи смыслового содержания текста.

Процесс извлечения наиболее адекватных словарных конструкций производится в модели трансфера – в соответствии с принципами, заложенными в трансформационной модели. Но для полноценного функционирования этого модуля должна быть подготовлена соответствующая двуязычная словарная база.

В концепции ФМТ большое внимание уделено разработке *автоматизированных технологий создания двуязычных словарей*. Основным требованием к этим технологиям является возможность адекватного отображения понятийного состава предметной области в словарной базе системы ФМТ. Другими словами, в тематической словарной базе должен содержаться основной понятийный и терминологический состав отрасли, контекстная система отношений между наименованиями понятий, а также необходимый набор фразеологических конструкций, обеспечивающий связывание переводных соответствий в грамматически согласованный переведенный текст.

В настоящее время авторами разработаны основные технологии создания политематической, тематической и пользовательской словарных баз. Наиболее важной из этих технологий является *технология создания тематической словарной базы*. Исходными данными для ее создания является репрезентативный корпус текстов на языке оригинала по заданной тематике. В монографии [12] детально описан технологический процесс создания такой словарной базы.

Необходимо отметить, что процесс извлечения фразеологического и терминологического понятийного состава тематики производится полностью автоматически точными и предиктивными методами концептуального анализа текстов. В рамках этого процесса

предусмотрены механизмы оценки необходимых трудозатрат для получения такой словарной базы, которая обеспечит требуемое качество перевода отраслевых документов.

Политематическая словарная база формируется путем слияния всех имеющихся тематических словарных баз по строго установленной технологии, учитывающий приоритеты смежных тематик по отношению к заданной тематике. Эта технология позволяет обеспечить формирование единой словарной базы, в которой исключены все дублирующие переводные эквиваленты, а оставшиеся эквиваленты выстроены в соответствии с их приоритетами. В каждой сессии перевода выполняется автоматическое формирование конфигурации виртуальной словарной базы, позволяющей произвести однократный поиск переводных соответствий одновременно как в тематической, так и политематической словарных базах.

Пользовательская словарная база обеспечивает возможность получения высококачественного перевода в режиме диалогового общения пользователя с системой РМТ. Этот режим позволяет пользователю в процессе перевода формировать свой высокоприоритетный пользовательский словарь, в который он может включать любые текстовые конструкции, практически любой длины.

По усмотрению пользователя, часто встречающиеся предложения с их переводами могут им помещаться в *словарь ТМ*. Пользователь также имеет уникальную возможность оперативной коррекции политематических и тематических словарей. И, наконец, пользователю предоставлена возможность оперативной коррекции языковых моделей исходных и целевых ЕЯ.

Необходимо отметить, что отличительной особенностью словарей РМТ является простая структура словарных статей, входами в которые могут быть любые текстовые фрагменты, а в качестве их переводных соответствий - эквивалентные по смыслу переводные соответствия, специфичные для данной предметной области. Причем, в словарях РМТ полностью отсутствует какая-либо сопутствующая грамматическая или семантическая информация, а наименования понятий и их переводные эквиваленты на целевом языке могут быть представлены в любой грамматической форме.

Все эти особенности технологий и структур словарей позволяют в короткие сроки создавать словарные базы больших объемов. Так, например, к настоящему времени создано более 150 таких тематических словарей по различным направлениям развития научно-технического прогресса: авиация, космонавтика, вычислительная техника, ядерная энергетика и т.п. Их общий объем составляет более 7.5 млн. словарных статей.

6. Заключение

В современную эпоху создания многополярного мироустройства особое значение приобретает международное научно-технологическое сотрудничество с дружественными странами - членами БРИКС, ШОС и СНГ. Возникающий при этом языковый барьер можно преодолеть путем разработки современных гибридных технологий машинного перевода, базирующихся как традиционных технологиях машинного перевода, так и на технологиях NMT. Такие гибридные технологии мы применили при разработке программного комплекса «Умный текстовый процессор» в АО «НПК «ВТ и СС», который был удостоен Национальной премии в области информационных технологий «Приоритет: Цифра-2023» в номинации «Искусственный интеллект».

Список литературы / References)

[1]. Колин К. К., Урсул А. Д. Информация и культура. Введение в информационную культурологию. М.: Изд-во Стратегические приоритеты, 2015. – 300 с.

[2]. Колин К. К., Хорошилов А. А. Проблема многоязычия в информационном общества и интеллектуальные переводческие технологии // Информационное общества, 2012, № 1. С. 56-61.

[3]. Искусственный интеллект в технологиях машинного перевода / Колин К.К., Хорошилов Ал-др. А., Никитин Ю.В., Пшеничный С.И., Хорошилов Алексей А. // Социальные новации и социальные науки. – Москва: ИНИОН РАН, 2021. – № 2.
[4]. Johnson M., Schuster M., Le Q. V., et al. Google's multilingual neural machine translation system: Enabling zeroshot translation // T. Assoc. Computational Linguistics, 2017. Vol. 5.
[5]. Хобсон Л., Ханнес Х., Ховард К. Обработка естекственного языка в действие. Изд. Питер, 2020.
[6]. Ганегедара Т. Обработка естественного языка с TensorFlow. М. ДМК Пресс, 2020.
[7]. Мельчук И. А. Опыт теории лингвистических моделей «Смысл <=> текст». «Наука», Москва, 1974 г.
[8]. Кулагина О. С. Исследования по машинному переводу. «Наука», Москва, 1979.
[9]. Пиотровский Р. Г. Новые горизонты машинного перевода. Сб. «Научно-техническая информация», сер. 2, № 1, ВИНИТИ, 2002.
[10]. Белоногов Г. Г., Хорошилов Ал-др А., Хорошилов Ал-сей А., Козачук М. В., Рыжова Е. Ю., Гуськова Л. Ю. Каким быть машинному переводу в XXI веке // Перевод: традиции и современные технологии. – М.: ВЦП, 2002.
[11]. Белоногов Г. Г., Калинин Ю. П., Хорошилов Ал-др А., Хорошилов Ал-ей А. Системы фразеологического машинного перевода. Изд. Русский мир, Москва, 2007.
[12]. Хорошилов Ал-др А., Кан А.В., Хорошилов А.А. Фразеологический машинный перевод. – М.: Изд-во «Директ-Медиа», 2019.
[13]. Аблов И.В., Козичев В.Н., Ширманов А.В., Хорошилов А.А., Хорошилов А.А. Средства машинной грамматики русского языка (по Г.Г. Белоногову) // Сб. "Научно-техническая информация", Серия 2, № 6, ВИНИТИ, 2018.
[14]. Белоногов Г. Г., Калинин Ю. П., Хорошилов А. А. Компьютерная лингвистика и перспективные информационные технологии. Теория и практика построения систем автоматической обработки текстовой информации. Изд. Русский мир, Москва, 2004.
[15]. Кан А. В., Ревина В. Д., Руснак В. И., Хорошилов Александр А., Хорошилов Алексей А. Автоматическое формирование синтаксической модели языка для задач машинного перевода и информационного поиска. Сб. «Научно-техническая информация», Сер. 2, № 12, ВИНИТИ, 2018.
[16]. Захаров В. Н., Никитин Ю. В., Хорошилов Александр А., Хорошилов Алексей А. Технологии создания новых направлений перевода для системы МетаФраз (на примере казахско-русского перевода). Сб. «Научно-техническая информация», Сер. 2, № 9, ВИНИТИ, 2017.
[17]. Хорошилов А.А., Никитин Ю.В., Лазарев А.С. и др. Программный комплекс "Умный текстовый процессор" (ПК УТП). Свидетельство о регистрации программы для ЭВМ RU 2023611992, 26.01.2023. Заявка № 2022683941 от 06.12.2022.

Информация об авторах / Information about authors

Константин Константинович КОЛИН – Доктор технических наук, профессор, главный научный сотрудник Федерального исследовательского центра "Информатика и управление" Российской академии наук. Сфера научных интересов: философия информации, философские и социальные проблемы информатики.

Konstantin Konstantinovich KOLIN – Dr. Sci. (Tech.), Professor, Chief Researcher of the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences. Research interests: philosophy of information, philosophical and social problems of computer science.

Александр Алексеевич ХОРОШИЛОВ – Доктор технических наук, профессор Московского авиационного института, ведущий научный сотрудник Федерального исследовательского центра "Информатика и управление" Российской академии наук, старший научный сотрудник 27-го Центрального научно-исследовательского института Министерства обороны России. Сфера научных интересов: системный анализ, машинный перевод и искусственный интеллект.

Alexanser Alexeevich KHOROSHILOV – Dr. Sci. (Tech.), Professor of the Moscow Aviation Institute (National Research University), Lead Researcher of the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Senior Researcher of the 27

Central Research Institute of the Ministry of Defense of the Russian Federation. Research interests: system analysis, machine translation and artificial intelligence.

Анна Владимировна КАН – кандидат технических наук, доцент Московского авиационного института, начальник аналитического отдела Научно-исследовательского центра “Институт имени Н. Е. Жуковского”. Сфера научных интересов: системный анализ, имитационное моделирование и искусственный интеллект.

Anna Vladimirovna KAN – Cand. Sci. (Tech.), Associate Professor of the Moscow Aviation Institute, Head of the Analytical Department of the National Research Center “Zhukovsky Institute”. Research interests: system analysis, simulation and artificial intelligence.

Юрий Викторович НИКИТИН – научный сотрудник Федерального исследовательского центра “Информатика и управление” Российской академии наук, руководитель группы разработчиков Научно-Промышленной компании “Высокие технологии и стратегические системы”. Сфера научных интересов: компьютерная лингвистика, технологии автоматической обработки и семантического анализа текстов.

Yuri Viktorovich NIKITIN – Researcher of the Federal Research Center “Computer Science and Control” of the Russian Academy of Sciences, Development Team Leader of the Scientific and Industrial Company “High Technologies and Strategic Systems”. Research interests: computational linguistics, technologies of automatic processing and semantic analysis of texts.