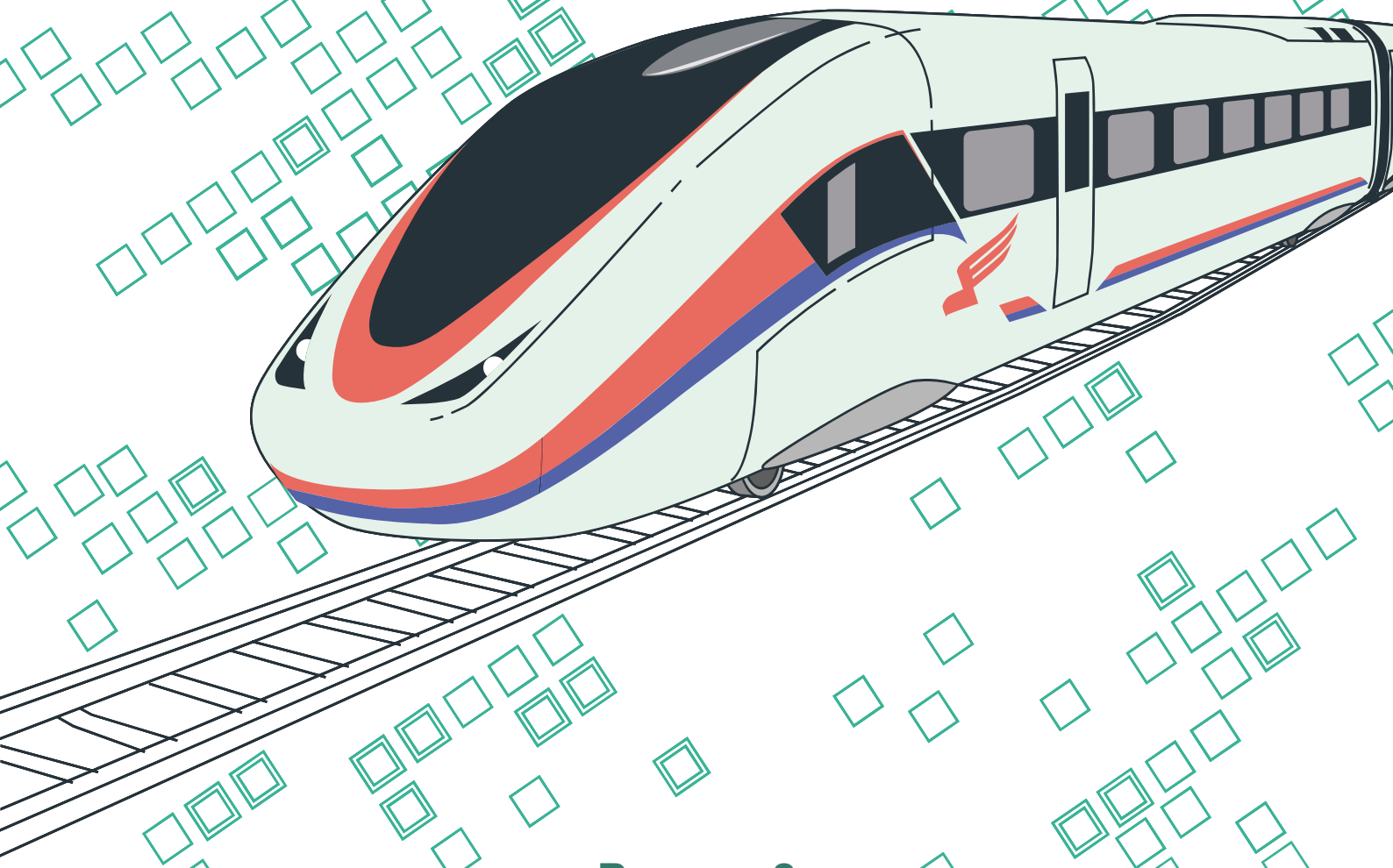
# Интеллектуальные технологии на транспорте

Intellectual Technologies on Transport



Выпуск 2
2024

**ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ НА ТРАНСПОРТЕ**
**(сетевой электронный научный журнал)**
**Выпуск 2 (38), 2024**

ISSN 2413-2527

Сетевой электронный научный журнал, свободно распространяемый через интернет. Публикуются статьи на русском и английском языках с результатами исследований и практических достижений в области интеллектуальных технологий и сопутствующих им научных исследований. Журнал основан в 2015 году.

**Учредитель**
Федеральное государственное бюджетное образовательное учреждение высшего образования «Петербургский государственный университет путей сообщения Императора Александра I» (ФГБОУ ВО «ПГУПС»)

**Издатель**
ООО «Медиа-Сервис» по договору № ЭА00271 от 19.12.2023

Периодичность выхода — 4 номера в год.

**Адрес редакции:**
190031, Санкт-Петербург, Московский пр., 9
e-mail: itt-pgups@yandex.ru
Телефон: +7 (812) 457-86-06

Сетевое издание «Интеллектуальные технологии на транспорте (сетевой электронный научный журнал), Intellectual Technologies on Transport» зарегистрировано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.
Свидетельство Эл № ФС77–61707 от 7 мая 2015 г.

Журнал зарегистрирован в Российском индексе научного цитирования (РИНЦ).

Информация предназначена для детей старше 12 лет.

Выпуски журнала доступны на сайте http://itt-pgups.ru

**INTELLECTUAL TECHNOLOGIES ON TRANSPORT**
**(Network electronic scientific journal)**
**Issue 2 (38), 2024**
ISSN 2413-2527

Network electronic scientific journal, open access.
It publishes articles in Russian and English with the results of research and practical achievements in the field of intelligent technologies and associated research.
Founded in 2015.

**Founder**
Federal State Budgetary Educational Institution of Higher Education "Emperor Alexander I St. Petersburg State Transport University"

**Publisher**
Media Service LLC No. ЭА00271, 19.12.2023

Frequency of release — 4 issues per year.

**Editorial adress:**
190031, St. Petersburg, Moskovsky ave., 9
e-mail: itt-pgups@yandex.ru
Phone: +7 812 457 86 06

The online journal "Intellectual Technologies on Transport" is registered by the Federal Service for Supervision of Communications, Information Technologies, and Mass Media.
El No. FS77-61707 Testimony from May 7, 2015.

The journal is registered in the Russian Science Citation Index (RSCI).

The content is for children over the age of 12.
Issues of the magazine are available at http://itt-pgups.ru

# Содержание

# Contents

УДК 317.4

# Автоматизация процессов в системах дистанционного обучения: современные решения и перспективы

**Баталов Дмитрий Иннокентьевич** — кандидат технических наук, доцент кафедры «Информационные и вычислительные системы». E-mail: d.i.batalov@yandex.ru

**Руссу Юлия Витальевна** — бакалавр 4-го курса направления 09.03.01 «Информатика и вычислительная техника». E-mail: mio.kiokoko@yandex.ru

Петербургский государственный университет путей сообщения Императора Александра I, Россия, Санкт-Петербург

**Для цитирования:** Баталов Д. И., Руссу Ю. В. Автоматизация процессов в системах дистанционного обучения: современные решения и перспективы // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 5–12. DOI: 10.20295/2413-2527-2024-238-5-12

***Аннотация.*** *Рассматривается роль автоматизации в системах дистанционного обучения (СДО) и ее влияние на образовательные процессы. Обсуждаются современные тенденции в области дистанционного образования и роль программных роботов в повышении эффективности учебного процесса. Приводится анализ популярных СДО, таких как Moodle и Google Classroom, и перспектив их развития с использованием инновационных технологий. В статье также рассматриваются проблемы и возможности автоматизации процессов в образовательных учреждениях.*

***Ключевые слова:*** *системы дистанционного обучения, роботизированная автоматизация процессов, RPA, автоматизация образования, Moodle, электронные образовательные ресурсы, учебные процессы, преподаватели, программные роботы.*

## Введение

С развитием информационных технологий образование все чаще переходит в онлайн-формат, что способствует широкому распространению систем дистанционного обучения (СДО). Такие системы позволяют студентам получать доступ к учебным материалам из любой точки мира в любое удобное время, что особенно важно для работающих студентов и тех, кто живет в удаленных районах. Важной задачей становится автоматизация процессов в СДО, направленная на повышение эффективности учебного процесса и снижение нагрузки на преподавателей [1].

Одной из ключевых технологий, применяемых для автоматизации рутинных и повторяющихся задач, является роботизированная автоматизация процессов (RPA). Программные роботы могут выполнять различные административные задачи, такие как обработка документов, регистрация студентов на курсы, отправка уведомлений и многое другое [2]. Внедрение таких технологий позволяет существенно сократить временные затраты на выполнение рутинных операций и улучшить качество образовательного процесса [3, 4].

В данной статье рассматриваются современные решения в области дистанционного обучения и автоматизации образовательных процессов. Приводится анализ функциональных возможностей СДО, а также примеры использования программных

роботов для оптимизации работы учебных заведений. Особое внимание уделяется таким популярным системам, как Moodle и Google Classroom, а также перспективам их дальнейшего развития и интеграции новых технологий.

**Обзор существующих решений**

В современных условиях образование все чаще становится доступным онлайн, что требует автоматизации процессов в системах дистанционного обучения (СДО). Большинство учебных заведений только начинает осваивать дистанционные технологии, что приводит к техническим проблемам и повышенной нагрузке на пользователей. Грамотная работа информационно-образовательной среды (ИОС) включает организацию и проведение онлайн-курсов, поддержку студентов и мониторинг их прогресса.

СДО предоставляют широкий спектр возможностей для эффективного обучения, включая доступ к учебным материалам в любое время и из любой точки мира. Функциональные возможности СДО включают инструменты для создания курсов, проведения тестов и оценки успеваемости студентов, что помогает персонализировать обучение [5].

Программные роботы, или роботизированная автоматизация процессов (RPA), представляют собой инновационные инструменты для автоматизации рутинных задач в образовании. Они помогают в автоматизации административных задач, сборе и анализе данных, а также поддержке студентов через чат-боты. Применение RPA в образовании способствует повышению эффективности и конкурентоспособности учебных заведений [6].

В транспортной отрасли RPA используется для повышения производительности и эффективности. Например, ОАО «РЖД» активно внедряет технологии RPA для оптимизации различных процессов [7]. Программные роботы могут автоматизировать множество задач, включая обработку данных и выполнение пользовательских действий, что значительно ускоряет работу и снижает вероятность ошибок [8].

Автоматизация в СДО играет ключевую роль в повышении качества обучения и снижении нагрузки на преподавателей. Инструменты автоматизации помогают в создании и управлении контентом, а также в автоматической оценке и предоставлении обратной связи студентам. Системы, оснащенные искусственным интеллектом, могут подстраиваться под индивидуальные потребности учащихся.

Примеры сервисов для автоматизации СДО включают Moodle, Google Classroom и Turnitin. Эти инструменты помогают преподавателям эффективно управлять учебным процессом, создавая задания, тесты и интерактивные уроки. Применение RPA в СДО позволяет автоматизировать рутинные задачи, что значительно упрощает работу преподавателей [6].

Автоматизация процессов в СДО предоставляет новые возможности для улучшения качества образования. Применение RPA позволяет сократить временные затраты на выполнение рутинных задач и повысить эффективность учебного процесса. Однако важно учитывать возможные проблемы, связанные с интеграцией новых технологий и адаптацией пользователей к новым условиям.

**Проектирование программного робота**

Процесс проектирования программного робота для формирования курсов дисциплин в системе дистанционного обучения СДО ПГУПС начинается с анализа преподавательской деятельности и выявления задач, которые занимают значительное время и могут быть автоматизированы. Основные задачи включают:

– структурирование контента курса;
– проверка и оценка работ студентов;
– общение с учащимися;
– организация и загрузка материалов.

Автоматизация этих процессов, особенно загрузки файлов и заполнения курса дисциплины, может значительно облегчить труд преподавателей.

Основные требования к роботу включают шесть пунктов (рис. 1):

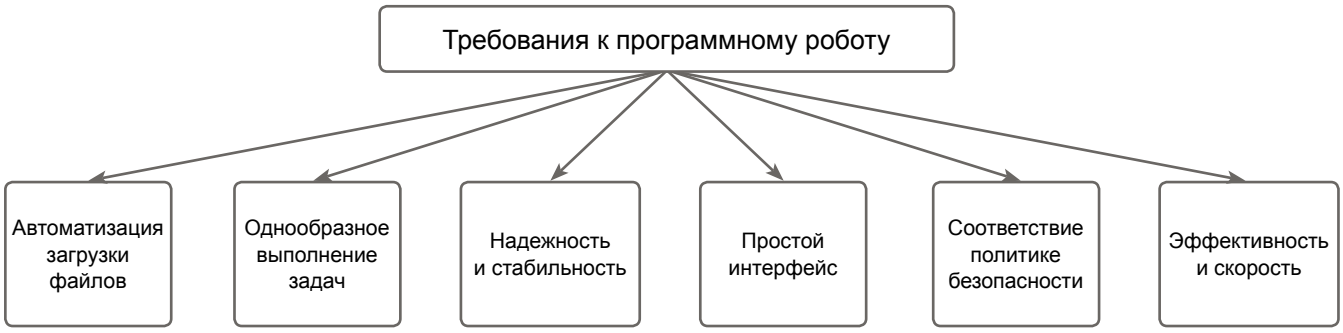



*Рис. 1.* Требования к программному роботу

Автоматизация загрузки файлов: система должна автоматически загружать файлы из подготовленного архива в систему дистанционного обучения.

Надежность и стабильность: приложение должно работать без сбоев, обладать механизмами для обработки ошибок.

Простота интерфейса: интерфейс должен быть интуитивно понятным, чтобы преподаватели могли легко управлять процессом.

Безопасность данных: приложение должно соблюдать политику безопасности университета и обеспечивать защиту данных студентов и учебных материалов.

Эффективность и скорость: робот должен выполнять задачи быстро, минимизируя время, затрачиваемое на процесс загрузки файлов.

Основные функциональные возможности робота включают шесть пунктов (рис. 2).

Выбор платформы и инструментов для разработки программного робота включает:

1. Совместимость с СДО: Гарантия гладкого взаимодействия робота с основной системой.

2. Гибкость и масштабируемость: Возможность адаптировать робота под различные курсы и дисциплины.

3. Надежность и безопасность: Гарантия стабильной работы и защиты данных.

Основные платформы RPA (Robotic Process Automation) для разработки [9]:

1. Lexema RPA: Платформа для создания программных роботов, выполняющих рутинные операции на компьютере.

2. Sherpa RPA: Российская платформа для интеллектуальной автоматизации бизнес-процессов с использованием ИИ.

3. Robin RPA: Российская платформа, предлагающая простую настройку в визуальном конструкторе.

В ходе исследования рынка платформ создания автоматизированных решений разработки программного робота для формирования курса дисциплины в СДО ПГУПС была выбрана платформа Sherpa RPA, в связи с ее доступностью, возможностями работать с веб-приложениями, с визуальными интерфейсами и высокой степенью контроля над роботом, что обеспечит успешное внедрение и эффективное функционирование робота в учебном процессе [10].

В процессе разработки были выделены следующие программные модули и компоненты программного робота (рис. 3).

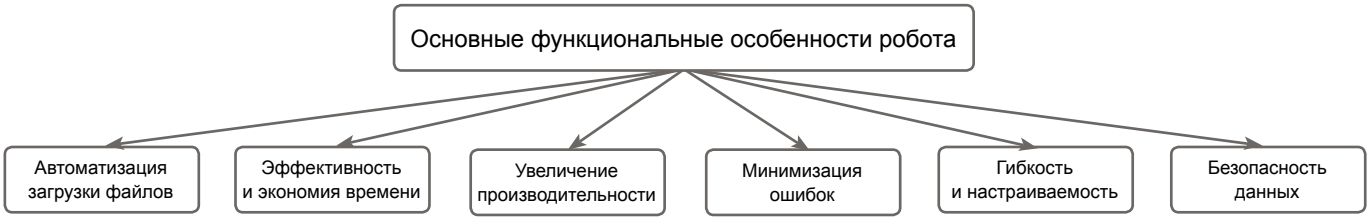



*Рис. 2.* Основные функциональные особенности робота

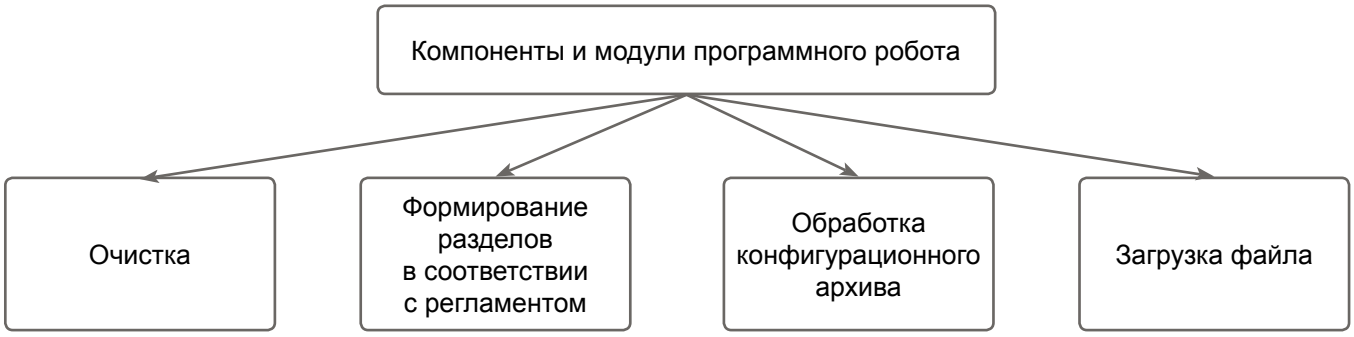



*Рис. 3.* Компоненты и модули программного робота

*Модуль очистки* предназначен для удаления всего содержимого существующего курса, чтобы подготовить платформу к загрузке обновленных материалов. За создание структуры курса на основе утвержденного регламента отвечает *модуль «Формирование разделов»*. *Модуль обработки конфигурационного архива* отвечает за распределение файлов по соответствующим разделам курса. *Модуль загрузки файлов* выполняет действия на сайте по загрузке учебных материалов.

Для наглядного представления структуры модулей, последовательности действий программного робота и упрощения процесса разработки были созданы UML-диаграммы (рис. 4–6).

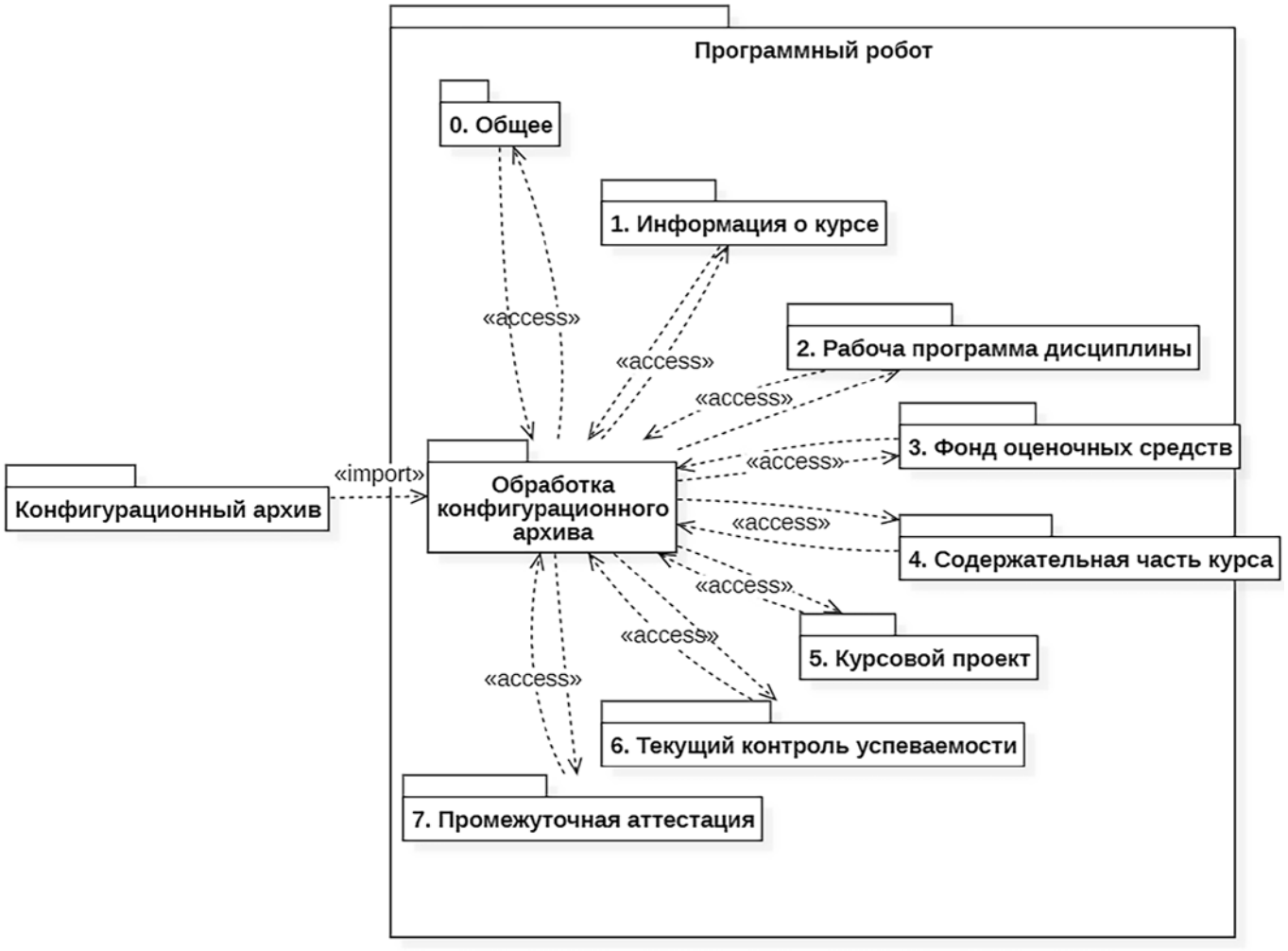



*Рис. 4.* Диаграмма пакетов модуля «Формирование разделов»

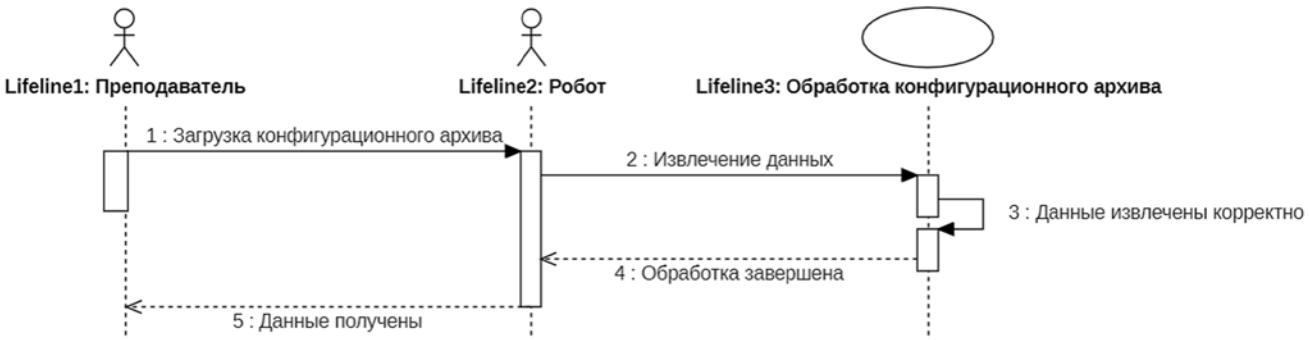



*Рис. 5.* Диаграмма последовательностей модуля «Обработка конфигурационного архива»

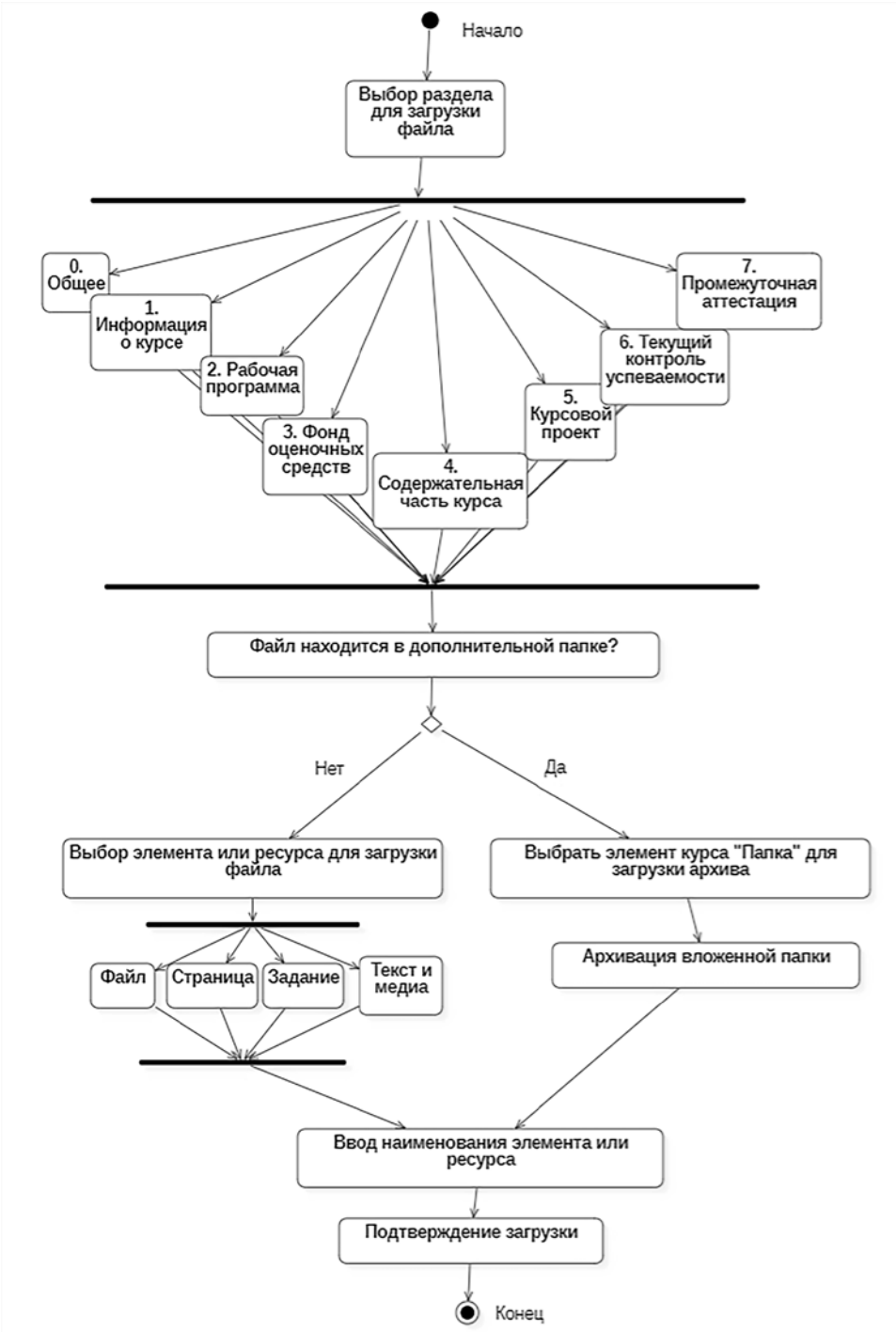



*Рис. 6.* Диаграмма активностей для модуля «Загрузка файла»

Взаимодействие робота с системой дистанционного обучения (СДО) ПГУПС представляет собой важную часть его функционала, так как это обеспечивает необходимую интеграцию и доступ к ресурсам для выполнения задач и использует поверхностное взаимодействие с системой, управляя внешними устройствами компьютера, как делает это обычный пользователь.

Эмуляция пользовательских действий — это способ, при котором программные роботы используют средства автоматизации для, например, кликов мыши, прокрутки страниц или ввода данных.

Основное преимущество такого подхода заключается в безопасности. Поскольку робот работает только на уровне интерфейса пользователя, он не вмешивается во внутренние процессы и данные СДО. Это исключает возможность несанкционированного доступа к конфиденциальной информации студентов и преподавателей.

Поскольку робот не имеет доступа к внутренним базам данных и только имитирует действия пользователя, вероятность ошибок или утечек данных значительно снижается. Все данные, используемые роботом, поступают из заранее подготовленного конфигурационного архива. Это обеспечивает контроль над содержимым и его соответствие требованиям безопасности и конфиденциальности.

Основные аспекты этого взаимодействия позволят выполнять открытие курса, удаление или добавление материалов, возможность редактирования информации, но также включают возможные проблемы, способы предотвращения которых необходимо рассмотреть.

При выполнении сценария программным роботом существует несколько ключевых этапов: инициализация задачи, навигация, взаимодействие с элементами интерфейса и проверка корректности выполнения задачи.

Преподаватель, инициируя процесс автоматизации, может столкнуться с неожиданными событиями, например, когда процесс запущен не вовремя.

Для решения проблемы с неожиданными событиями и некорректными запусками необходимо настроить четкие триггеры и условия для запуска задач. Также следует использовать логические операторы для контроля выполнения задач, чтобы минимизировать вероятность ошибок.

## Разработка программного робота

Созданный программный робот имитирует действия преподавателя при формировании курса в СДО, используя стандартизированные интерфейсы для взаимодействия с пользовательским интерфейсом. Набор действий, которые может выполнять робот, включает компоненты для обеспечения работоспособности, такие как взаимодействие с веб-интерфейсами и обработка исключений.

Программный робот прошел тестирование на различных этапах разработки для подтверждения его работоспособности и соответствия требованиям к действиям преподавателя в СДО ПГУПС. Для этого использованы модули робота, выполняющие вместо преподавателя такие действия, как очистка курса, формирование разделов, обработка архивов и загрузка файлов.

Разработка программного робота для автоматизации действий преподавателя в СДО ПГУПС потребовала комплексного подхода, включающего анализ требований, проектирование, тестирование и обеспечение совместимости. Созданный робот позволяет оптимизировать управление курсами и повысить эффективность образовательного процесса.

## Заключение

Автоматизация процессов в системах дистанционного обучения играет ключевую роль в современном образовании. Использование программных роботов позволяет значительно повысить эффективность учебного процесса, снизить нагрузку на преподавателей и улучшить качество обучения. Примеры таких систем демонстрируют, как автоматизация может помочь преподавателям и студентам достичь лучших результатов. Однако для успешной реализации проектов автоматизации необходимо учитывать индивидуальные особенности каждого учебного заведения и правильно адаптировать технологии под их нужды. В целом развитие автоматизации в образовании открывает новые возможности для улучшения доступности и качества образования.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Ванюхина Н. В., Абдуллин А. И., Габитов А. И. Проблема внедрения электронных образовательных ресурсов как инновационных средств обучения // Ученые записки Санкт-Петербургского государственного института психологии и социальной работы. 2016. Т. 25, № 1. С. 140–147. EDN RZFSXZ.

2. Osman C. C. Robotic Process Automation: Lessons Learned from Case Studies // Informatica Economica. 2019. Т. 23. № 4.

3. Гузий М. В., Андрющенко А. А. RPA — роботизация в области образования и управления кадрами // Междисциплинарные подходы в современной науке: вызовы, достижения и перспективы: сборник статей Международной научно-практической конференции, Воронеж, 25 февраля 2024 года. Уфа: общество с ограниченной ответственностью «Аэтерна», 2024. С. 28–30. EDN EHYIBA.

4. Кутуков Н. Ю. Место технологии RPA в современной системе образования и возможности ее применения // Развитие современной науки и технологий в условиях трансформационных процессов: сборник материалов Международной научно-практической конференции, Москва, 15 апреля 2022 года. М.: ИП Овчинников Михаил Артурович (типография «Алеф»), 2022. С. 33–35. DOI: 10.34755/IROK.2022.10.38.023. EDN NEQJSP.

5. Оболенский Д. М., Шевченко В. И. Концептуальная модель интеллектуальной образовательной экосистемы // Экономика. Информатика. 2020. Т. 47, № 2. С. 390–401. DOI: 10.18413/2687–0932-2020-47-2-390-401. EDN SNSYNO.

6. Новицкий А. А. Современные тенденции в инвестиционной политике российских публичных компаний // Трансформация экономики и финансового сектора России: вызовы и тренды: сборник материалов Всероссийской научно-практической конференции, Москва, 30 марта 2023 года. М.: Московский финансово-промышленный университет «Синергия», 2023. С. 280–281. EDN PZGCHU.

7. Каргина Л. А., Ионова Т. В., Лебедева С. Л. Роль технологий RPA в цифровой трансформации ОАО «РЖД» // Экономика железных дорог. 2022. № 8. С. 62–69. EDN RMQKFH.

8. Ермаков С. Г., Баталов Д. И., Мельников И. С. Использование платформы Robin RPA в процессе цифровой трансформации транспортных компаний // Интеллектуальные технологии на транспорте. 2023. № 1 (33). С. 5–14. DOI: 10.24412/2413–2527-2023-133-5-14. EDN YVWRCE.

9. Невес А., Араухо В. Умная автоматизация в интересах кибербезопасности // Форсайт, 2023. Т. 17, № 1. С. 89–97. DOI: 10.17323/2500–2597.2023.1.89.97. EDN CJCHLS.

10. Захаров Н. А. Отечественные платформы для роботизации бизнес-процессов // Автоматизация в промышленности. 2022. № 5. С. 59–61. DOI: 10.25728/avtprom.2022.05.15. EDN TJVXRI.

Дата поступления: 11.06.2024
Решение о публикации: 14.06.2024

# Automation of Processes in Distance Learning Systems: Modern Solutions and Prospects

**Dmitry I. Batalov** — Candidate of Technical Sciences, Associate Professor of the Department of Information and Computing Systems. E-mail: d.i.batalov@yandex.ru

**Yulia V. Russu** — Bachelor of the 4th year of the 09.03.01 direction “Computer Science and Computer Engineering”. E-mail: mio.kiokoko@yandex.ru

Emperor Alexander I Petersburg State Transport University, Saint Petersburg, Russia

**For citation:** Batalov D. I., Russu Yu. V. Automation of processes in distance learning systems: modern solutions and prospects // Intelligent technologies on transport. 2024. No. 2 (38). P. 5–12. (In Russian). DOI: 10.20295/2413-2527-2024-238-5-12

***Abstract.*** *The role of automation in distance learning systems and its impact on educational processes is considered. Current trends in the field of distance education and the role of software robots in improving the effectiveness of the educational process are discussed. The analysis of popular SDS, such as Moodle and Google Classroom, and the prospects for their development using innovative technologies are presented. The article also discusses the problems and possibilities of automating processes in educational institutions.*

***Keywords:*** *distance learning systems, robotic automation of processes, RPA, automation of education, Moodle, electronic educational resources, educational processes, teachers, software robots.*

**REFERENCES**

1. Vanyuhina N. V., Abdullin A. I., Gabitov A. I. Problema vnedreniya elektronnyh obrazovatel'nyh resursov kak innovacionnyh sredstv obucheniya // Uchenye zapiski Sankt-Peterburgskogo gosudarstvennogo instituta psihologii i social'noj raboty. 2016. T. 25, № 1. S. 140–147. EDN RZFSXZ. (In Russian)

2. Osman C. C. Robotic Process Automation: Lessons Learned from Case Studies // Informatica Economica. 2019. T. 23, № 4.

3. Guzij M. V., Andryushchenko A. A. RPA — robotizaciya v oblasti obrazovaniya i upravleniya kadrami // Mezhdisciplinarnye podhody v sovremennoj nauke: vyzovy, dostizheniya i perspektivy: sbornik statej Mezhdunarodnoj nauchno-prakticheskoj konferencii, Voronezh, 25 fevralya 2024 goda. Ufa: obshchestvo s ogranichennoj otvetstvennost'yu “Aeterna”, 2024. S. 28–30. EDN EHYIBA. (In Russian)

4. Kutukov N. Yu. Mesto tekhnologii RPA v sovremennoj sisteme obrazovaniya i vozmozhnosti eyo primeneniya // Razvitie sovremennoj nauki i tekhnologij v usloviyah transformacionnyh processov: sbornik materialov Mezhdunarodnoj nauchno-prakticheskoj konferencii, Moskva, 15 aprelya 2022 goda. M.: IP Ovchinnikov Mihail Arturovich (tipografiya “Alef”), 2022. S. 33–35. DOI: 10.34755/IROK.2022.10.38.023. EDN NEQJSP. (In Russian)

5. Obolenskij D. M., Shevchenko V. I. Konceptual'naya model' intellektual'noj obrazovatel'noj ekosistemy // Ekonomika. Informatika. 2020. T. 47, № 2. S. 390–401. DOI: 10.18413/2687–0932-2020-47-2-390-401. EDN SNSYNO. (In Russian)

6. Novickij A. A. Sovremennye tendencii v investicionnoj politike rossijskih publichnyh kompanij // Transformaciya ekonomiki i finansovogo sektora Rossii: vyzovy i trendy: sbornik materialov Vserossijskoj nauchno-prakticheskoj konferencii, Moskva, 30 marta 2023 goda. M.: Moskovskij finansovo-promyshlennyj universitet “Sinergiya”, 2023. S. 280–281. EDN PZGCHU. (In Russian)

7. Kargina L. A., Ionova T. V., Lebedeva S. L. Rol' tekhnologij RPA v cifrovoj transformacii OAO “RZHD” // Ekonomika zheleznyh dorog. 2022. № 8. S. 62–69. EDN RMQKFH. (In Russian)

8. Ermakov S. G., Batalov D. I., Mel'nikov I. S. Ispol'zovanie platformy Robin RPA v processe cifrovoj transformacii transportnyh kompanij // Intellektual'nye tekhnologii na transporte. 2023. № 1 (33). S. 5–14. DOI: 10.24412/2413–2527-2023-133-5-14. EDN YVWRCE. (In Russian)

9. Neves A., Arauho V. Umnaya avtomatizaciya v interesah kiberbezopasnosti // Forsajt, 2023. T. 17, № 1. S. 89–97. DOI: 10.17323/2500–2597.2023.1.89.97. EDN CJCHLS. (In Russian)

10. Zaharov N. A. Otechestvennye platformy dlya robotizacii biznes-processov // Avtomatizaciya v promyshlennosti. 2022. № 5. S. 59–61. DOI: 10.25728/avtprom.2022.05.15. EDN TJVXRI. (In Russian)

Received: 11.06.2024
Accepted: 14.06.2024

УДК 004.75

# Algorithms for Assessing Quality Indicators Functioning of a Distributed System Storing Confidential Data

**Alexander G. Basyrov**[1] — Doctor of Technical Sciences, Professor; professor of the Department of the A. F. Mozhaysky Military Space Academy. Research interests: parallel computing, information systems. E-mail: alexandrerbas@mail.ru

**Igor N. Koshel**[2] — Candidat of Technical Sciences. Head of the Department of the A. F. Mozhaysky Military Space Academy. Research interests: parallel information processing planning, information systems. E-mail: kin1470@mail.ru

**Valery V. Abramenkov**[1] — Head of FSAU "Central Department of Housing and Social Infrastructure (Complex)". Research interests: information systems, data storage systems. E-mail: info@fgaul.ru

[1] A. F. Mozhaysky Military Space Academy, Saint Petersburg, Russia

[2] FSAU "Central Department of Housing and Social Infrastructure (Complex)", Moscow, Russia

**For citation:** Basyrov A. G., Koshel I. N., Abramenkov V. V. Algorithms for Assessing Quality Indicators Functioning of a Distributed System Storing Confidential Data // Intelligent technologies on transport. 2024. No. 2 (38). P. 13–19. (In Russian). DOI: 10.20295/2413-2527-2024-238-13-19

***Abstract.*** *The article presents algorithms for assessing the quality indicators of distributed data storage systems used to accumulate and process data from automated information systems for various purposes, serving a large number of geographically distributed clients. Mathematical expressions for indicators of confidentiality, availability and cost of data storage are given. The proposed algorithms make it possible to analyze the distribution of information resources among the elements of the data storage system in order to select rational solutions for organizing the storage of confidential information.*

***Keywords:*** *distributed data storage system, data confidentiality.*

## Introduction

Modern automated information systems that serve a large number of users contain data storage systems (DSS), the quality of which is subject to high requirements [1–3].

In order to analyze the values of quality indicators of the functioning of storage systems, a methodological apparatus is needed for assessing the following indicators of storing data located in storage systems: confidentiality, availability and cost.

A feature of the proposed approach is the use of quantitative estimates of the indicators under consideration.

The article discusses mathematical expressions that make it possible to quantify the values of the listed indicators and proposes algorithms for their calculation.

## Quality indicators of a distributed data storage system

A distributed information system (RIS), including client automated workstations (AWS) and data processing centers (DPCs), united by a global computer network (fig. 1), allows solving applied problems based on the collection, storage, processing and transmission of target information [4–7].

We will assume that the operating technology of such a RIS involves collecting information from client workstations and processing it in one or more data centers. In this case, information is accumulated in a database (DB) and stored in a data center.

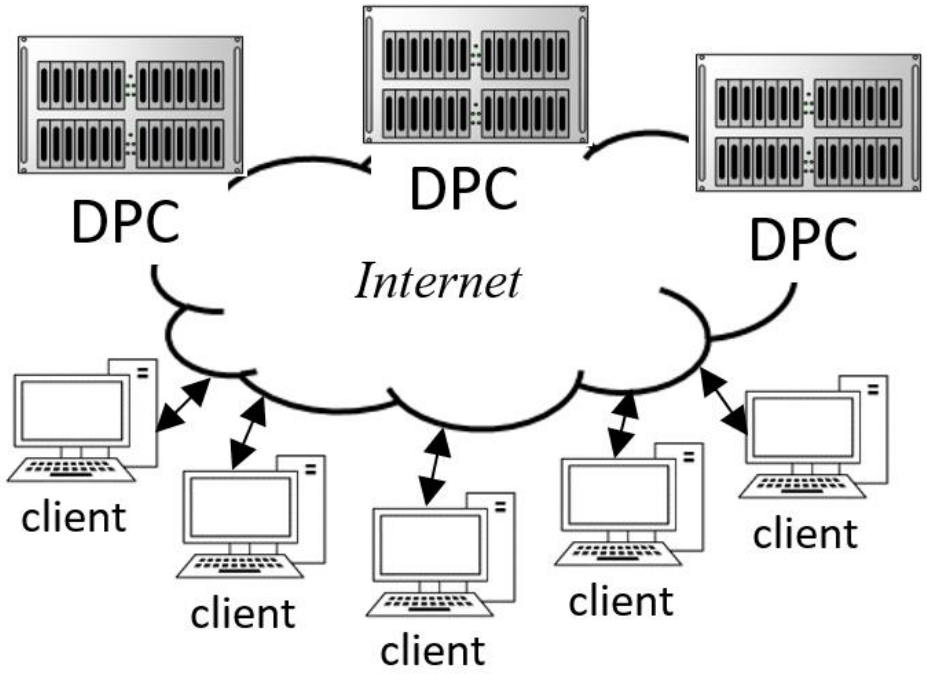



*Fig. 1.* Distributed information system

Information received from clients is stored in relational databases with a table structure (fig. 2). The database consists of n records or their blocks, each of which contains service information from one RIS client.

| Records | Recording storage location | | | |
|---|---|---|---|---|
| | $\alpha_1$ | $\alpha_2$ | ... | $\alpha_m$ |
| $r_1$ | 0 | 1 | | 0 |
| $r_2$ | 1 | 0 | | 0 |
| ... | | | | |
| $r_n$ | 1 | 0 | | 1 |

*Fig. 2.* Structure of stored information in the database

In order to increase reliability, database records can be stored on several elements of a data storage system, which can have a different architecture and represent a data storage of various capacities, accessed by users via the global Internet. At the same time, storing data on each storage system element has a certain cost, depending on the volume of stored information and the quality of the services provided for its storage (reliability, confidentiality, communication channel capacity, etc.).

In normal mode, RIS uses a database located in the data center, and the remaining data centers are used for backup information storage.

Each data center can be subject to destructive effects, both external (cyber-attack, terrorist attack, natural disaster, etc.) and internal (breakdown, failure, unauthorized access to data), as a result of which the integrity, availability and confidentiality of stored information can be violated (or parts thereof), which leads to denial of service to the RIS.

In order to meet the requirements for the quality of information storage, database records can be distributed among storage elements (data center and workstation) in such a way as to ensure restoration of access to data in the event of a data center service failure within a given time, subject to restrictions on the cost and confidentiality of their storage.

## Data storage confidentiality indicator

Let a rectangular matrix be given $H_{[n,m]}$, each element $h_{ij}$ of which is equal to 1 if the $i$-th record, $1 \le i \le n$, is placed on the $j$-th element of the storage system, $1 \le j \le m$ and is equal to 0 otherwise.

Let also be given a vector $C_{\langle m \rangle} = \langle c_1, c_2, ..., c_m \rangle$, each element $c_j$ of which is the probability (*guarantee level*)

of ensuring confidentiality of data storage on the *j*-th element of the storage system.

Let's consider two options for ensuring the confidentiality of data storage, each of which is advisable to use in appropriate conditions [8–12].

*Option 1*. The confidentiality requirement applies to each record separately.

Data confidentiality indicator — the minimum level of confidentiality *L* for all database records will be

$$L = \min_i \left( \prod_{j=1}^{m} \beta\left(h_{ij}, c_j\right) \right), i \in [1, n],$$

where $\beta(x,y) = \begin{cases} 1 \text{ at } x = 0; \\ 1 \text{ at } x = 1 \end{cases}$

*Option 2*. The confidentiality requirement applies to the entirety of the records.

Data confidentiality indicator — average value of information storage confidentiality:

$$L = \frac{1}{2^m} \sum_{k=1}^{2^m} \sum_{i=1}^{n} \prod_{j=1}^{m} \beta\left(h_{ij} \cdot comb_{jk}, c_j\right),$$

where $comb_{jk}$ is the value of the *j*-t*h* digit (0 or 1) in the binary representation (combination) of the number *k*.

**Data storage cost indicator**

The cost of data storage is determined by the total cost of storing it on all storage elements. If the cost of storage is calculated through the price $Z_{\langle m \rangle} = \langle z_1, z_2, ..., z_m \rangle$ for storing a unit of data volume on the *j*-th storage element, then the total cost of storing all records on all storage systems will be:

$$S = \sum_{j=1}^{m} \left( z_j \sum_{i=1}^{n} h_{ij} \cdot v_i \right),$$

where $z_j$ is the cost of storing a unit of volume (GB, TB) of data on the *j*-th storage system;

$v_j$ — vector component;

$V_n = v_1, ..., v_n$ — volume of the *i*-th record.

**Data availability indicator**

The availability of data depends on the readiness of the storage system elements to provide the databases stored in it. This indicator can be estimated based on the availability factors *g* of the corresponding storage elements [13–15]:

$$g = \frac{t_p}{t_p + t_в},$$

$t_p$ — the time of regular operation of the RIS, during which the requested database records must be provided with the established efficiency;

$t_в$ — time during which the requested database records are unavailable.

If the availability coefficient of the *j*-th storage element is $g_j$, then the minimum level of availability *G* for all data records will be

$$G = \min_i \left( 1 - \prod_{j=1}^{m} \beta\left(h_{ij}, 1 - g_j\right) \right), i \in [1, n].$$

Obviously, this indicator depends on the placement of database records, the readiness and state of storage elements (serviceable/faulty).

To increase data availability, storage elements are combined into clusters, duplicating information on each element of such a cluster. However, duplication of information increases the risk of unauthorized access to it, which leads to a decrease in the confidentiality of data storage.

Thus, the optimization problem arises of placing information on storage elements in order to ensure its maximum confidentiality under restrictions on the availability and cost of data storage.

To ensure the adequacy of the solution to this problem, algorithms are proposed for assessing the confidentiality, availability and cost of data storage, the use of which will make it possible to select a rational configuration for placing information on storage elements.

The following describes the step-by-step operation of algorithms for assessing quality indicators of storage systems, common to which are the following notations: matrix $H_{[n,m]}$ — matrix for placing records on storage systems; *n* — number of database records (blocks); *m* — number of storage elements.

**Algorithm for assessing information availability in a distributed data storage system**

In the presented algorithm $g_i$, the availability coefficient of the $i$-th storage element.

**Algorithm 1.**

Step 1. Start.

Step 2. $i$ :=1 , $G$ :=1.

Step 3. $j$ :=1, $d$ :=1.

Step 4. If $h_{ij} = 1$, then $d := d \cdot (1 - g_i)$.

Step 5. $j := j + 1$.

Step 6. If $j > N$, then go to step 7, otherwise — to step 4.

Step 7. $G := min(G,1 - d)$.

Step 8. $i := i + 1$.

Step 9. If $i > m$, then go to step 10, otherwise — to step 3.

Step 10. End.

As a result of the algorithm's operation, the variable $G$ will have the value of the minimum level of data recording availability. By comparing this value with the required one, it is possible to assess the suitability of data distribution among storage elements from the point of view of information availability requirements. The computational complexity of the algorithm is $O(mn)$.

**Algorithm for estimating the cost of information storage in a distributed data storage system**

In the presented algorithm $z_j$, is the price of storing a unit of data volume on the $j$-th storage element, $v_i$ and is the volume of the $i$-th data block.

**Algorithm 2.**

Step 1. Start.

Step 2. $i := 0$, $S := 0$.

Step 3. $j := 1$.

Step 4. If $h_{ij} = 1$, then $S := S + z_j \cdot v_i$.

Step 5. $j := j + 1$.

Step 6. If $j > N$, then go to step 7, otherwise — to step 4.

Step 7. $i := i + 1$.

Step 8. If $i > m$, then go to step 10, otherwise — to step 3.

Step 9. Finish.

At the end of the algorithm, the variable will have the value of the cost of data storage. By comparing this value with the required one, it is possible to assess the suitability of data distribution among storage elements from the point of view of requirements for the cost of information storage. The computational complexity of the algorithm is $O(mn)$.

**Algorithms for assessing information confidentiality in a distributed data storage system**

In the algorithms under consideration $c_i$, the level of confidentiality of information storage on the $i$-th storage element.

As a result of the operation of the algorithms, the variable $C$ will have a value of a certain level of data storage confidentiality. Comparing this value with the required one, it is possible to evaluate the value of the confidentiality of data distributed among storage elements in terms of information confidentiality requirements.

**Algorithm 3.** Estimation of the minimum level of data confidentiality.

Step 1. Start.

Step 2. $i := 1$, $C := 1$.

Step 3. $j := 1$, $d := 1$.

Step 4. If $h_{ij} = 1$, then $d := d \cdot c_i$.

Step 5. $j := j + 1$.

Step 6. If $j > N$, then go to step 7, otherwise — to step 4.

Step 7. $C := min(C,d)$.

Step 8. $i := i + 1$.

Step 9. If $i > m$, then go to step 10, otherwise — to step 3.

Step 10. Finish.

Thus, the variable $C$ will have the value *of the minimum* level of confidentiality of data storage. The computational complexity of the algorithm is $O(m^2)$.

**Algorithm 4.** Estimating the average level of information confidentiality (brute force).

Step 1. Start.

Step 2. $i := 1$, $C := 0$.

Step 3. $j := 1$, $u := 0$.

Step 4. $k := 1$, $d := 1$.

Step 5. If $h_{ik} = 1 \cap comb(i,k - 1) = 1$, then $d := d \cdot c_i$.

Step 6. $k := k + 1$.

Step 7. If $k > m$, then go to step 8, otherwise — to step 5.

Step 8. $u := u + d$.

Step 9. $j := j + 1$.

Step 10. If $j > N$, then go to step 11, otherwise — to step 4.

Step 11. $C := C + u$.

Step 12. $i := i + 1$.

Step 13. If $i > 2^m - 1$, then go to step 14, otherwise — to step 3.

Step 14. If C := $C / \left(n\left(2^m - 1\right)\right)$.

Step 15. End.

In the presented algorithm, the value of the function β is equal to the value β of the digit $\beta \in \{0,1\}$, in the binary code of the integer α, $0 \le \alpha \le 2^m - 1$.

As a result of the algorithm's operation, the variable *C* will have the value *of the average* level of confidentiality of data storage.

It should be noted that the algorithm has high computational complexity $O(2^m)$ and is applicable for relatively small values of the number of storage elements.

Since in practice there is a need to assess the confidentiality of data storage on a significant number of storage elements, it is advisable to use an approximate approach to assessment based on "greedy algorithms". The algorithm proposed below, which belongs to this type, implements a strategy of "greedy" data collection on storage elements with the goal of unauthorized collection of the maximum amount of information at minimal cost.

**Algorithm 5.** Evaluating the confidentiality of data storage (greedy algorithm).

Step 1. Start.

Step 2. Define the vector $F = f_1, f_2, ..., f_m, f_i \in \{0,1\}$.

Step 3. $F := 0$, $C := 0$, $k := 1$.

Step 4. $i := 1$, $d := 0$.

Step 5. If $f_i = 1$, then go to step 13, otherwise — to step 6.

Step 6. $j := 1$, $u := 0$.

Step 7. $u := u + H(j,i)$.

Step 8. $j := j + 1$.

Step 9. If $j > N$ , then go to step 10, otherwise — to step 7.

Step 10. If $u \cdot (1 - c_i) \ge d$, then go to step 11, otherwise — to step 13.

Step 11. $d := u \cdot (1 - c_i)$, $l := i$.

Step 12. $f_l := 1$, $c := c + d$.

Step 13. $i := i + 1$.

Step 14. If i > $m$, then go to step 15, otherwise — to step 5.

Step 15. If $h_{jl} = 1$, then $h_{jr} = 0$ $\forall r = 1,...,m$.

Step 16. $k := k + 1$.

Step 17. *If* $k > m$, then go to step 18, otherwise — to step 4.

Step 18. End.

As a result of the algorithm's operation, the variable *C* will have the value of *the pessimistic (minimum)* level of data storage confidentiality. The computational complexity of the algorithm is $O(m^2)$.

Thus, the presented algorithms provide estimates of data storage quality indicators.

## Conclusion

The given algorithms have a practical focus on studying the effectiveness of using storage systems for collecting, storing and processing confidential information.

The main problem of using algorithms is obtaining initial data, namely, estimates of the values of indicators of the availability of storage elements and the level of their provision of confidentiality of stored information. Availability estimates can be obtained based on software agents that operate on each storage element and keep track of the active location of the corresponding storage element on the network. Obtaining confidentiality estimates can be found using fuzzy data analysis models [16], taking into account many parameters of their storage on storage elements.

**REFERENCES**

1. Suhoroslov O. V. Novye tekhnologii raspredelennogo hraneniya i obrabotki bol'shih massivov dannyh. M.: Institut sistemnogo analiza RAN, 2021. 40 s. (In Russian)

2. Radchenko G. I. Raspredelennye vychislitel'nye sistemy. Chelyabinsk: Fotohudozhnik, 2012. 184 s. (In Russian)

3. Dokuchaev V. A., Kal'fa A. A., Maklachkova V. V. Arhitektura centrov obrabotki dannyh. M.: Goryachaya liniya — Telekom, 2023. 240 s. (In Russian)

4. Cil'ker B. Ya., Orlov S. A. Organizaciya EVM i sistem: uchebnik dlya vuzov. 2-e izd. SPb.: Piter, 2011. 668 s. (In Russian)

5. Melekhin V. F., Pavlovskij E. G. Vychislitel'nye mashiny, sistemy i seti. M.: Akademiya, 2007. 560 s. (In Russian)

6. Horoshevskij V. G. Arhitektura vychislitel'nyh sistem: ucheb. posobie. 2-e izd. M.: MGTU im. N. E. Baumana, 2008. 520 s. (In Russian)

7. Kosyakov M. S. Vvedenie v raspredelennye vychisleniya. SPb.: NIU ITMO, 2014. 115 s. (In Russian)

8. Amelin R. V. Informacionnaya bezopasnost'. M.: Yurist", 2010. S. 121. (In Russian)

9. Danilov A. D. Cennost' informacii. Tekhnologii informacionnogo obshchestva. 2011. № 10. S. 137–140. (In Russian)

10. Stepanov E. A., Korneev I. K. Informacionnaya bezopasnost' i zashchita informacii: uchebnoe posobie. M.: Infra-M. 2010. 336 s. (In Russian)

11. Fat'yanov A. A. Problemy zashchity konfidencial'noj informacii, ne sostavlyayushchej gosudarstvennuyu tajnu. Informacionnoe obshchestvo. 2010. № 5. S. 49–56. (In Russian)

12. Informacionnaya bezopasnost'. M.: Oruzhie i tekhnologii, 2011. S. 35–44. (In Russian)

13. Rahman P. A. Koefficient gotovnosti sistemy obrabotki dannyh s osnovnym i rezervnym uzlami. Mezhdunarodnyj zhurnal prikladnyh i fundamental'nyh issledovanij. 2015. № 9. S. 608–611. (In Russian)

14. Polovko A. M., Gurov S. V. Osnovy teorii nadezhnosti. 2-e izd., pererab. i dop. SPb.: BHV-Peterburg, 2006. 704 s. (In Russian)

15. Shishmarev V. Yu. Nadezhnost' tekhnicheskih sistem: uchebnik dlya stud. vyssh. ucheb. zavedenij. M.: Izdatel'skij centr “Akademiya”, 2010. 304 s. (In Russian)

16. Bronevich A. G., Lepskij A. E. Nechetkie modeli analiza dannyh i prinyatiya reshenij: uchebnoe posobie. Nac. issled. un-t “Vysshaya shkola ekonomiki”. M.: Izd. dom Vysshej shkoly ekonomiki, 2022. 264 s. (In Russian)

Received: 23.04.2024
Accepted: 20.05.2024

# Алгоритмы оценивания показателей качества функционирования распределенной системы хранения конфиденциальных данных

**Басыров Александр Геннадьевич**[1] — доктор технических наук, профессор. Профессор кафедры Военно-космической академии имени А. Ф. Можайского. E-mail: Vka_24kaf@mil.ru

**Кошель Игорь Николаевич**[2] — канд. техн. наук. Начальник факультета Военно-космической академии имени А. Ф. Можайского. E-mail: kin1470@mail.ru

**Абраменков Валерий Валерьевич**[1] — начальник ФГАУ «Центральное управление жилищно-социальной инфраструктуры (комплекса)». E-mail: info@fgaul.ru

[1] Военно-космическая академия имени А. Ф. Можайского, Россия, Санкт-Петербург

[2] ФГАУ «Центральное управление жилищно-социальной инфраструктуры», Россия, Москва

**Для цитирования:** Басыров А. Г., Кошель И. Н., Абраменков В. В. Алгоритмы оценивания показателей качества функционирования распределенной системы хранения конфиденциальных данных // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 13–19. DOI: 10.20295/2413-2527-2024-238-13-19

***Аннотация.*** *В статье представлены алгоритмы оценивания показателей качества распределенных систем хранения данных, применяемых для накопления и обработки данных автоматизированных информационных систем различного назначения, обслуживающих большое количество территориально распределенных клиентов. Приведены математические выражения показателей конфиденциальности, доступности и стоимости хранения данных. Предложенные алгоритмы позволяют проанализировать распределение информационных ресурсов по элементам системы хранения данных для выбора рациональных решений по организации хранения конфиденциальной информации.*

***Ключевые слова:*** *распределенная система хранения данных, конфиденциальность данных.*

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Сухорослов О. В. Новые технологии распределенного хранения и обработки больших массивов данных. М.: Институт системного анализа РАН, 2021. 40 с.

2. Радченко Г. И. Распределенные вычислительные системы. Челябинск: Фотохудожник, 2012. 184 с.

3. Докучаев В. А., Кальфа А. А., Маклачкова В. В. Архитектура центров обработки данных. М.: Горячая линия — Телеком, 2023. 240 с.

4. Цилькер Б. Я., Орлов С. А. Организация ЭВМ и систем: учебник для вузов. 2-е изд. СПб.: Питер, 2011. 668 с.

5. Мелехин В. Ф., Павловский Е. Г. Вычислительные машины, системы и сети. М.: Академия, 2007. 560 с.

6. Хорошевский В. Г. Архитектура вычислительных систем: учеб. пособие. 2-е изд. М.: МГТУ им. Н. Э. Баумана, 2008. 520 с.

7. Косяков М. С. Введение в распределенные вычисления. СПб.: НИУ ИТМО, 2014. 115 с.

8. Амелин Р. В. Информационная безопасность. М.: Юристъ, 2010. С. 121.

9. Данилов А. Д. Ценность информации. Технологии информационного общества. 2011. № 10. С. 137–140.

10. Степанов Е. А., Корнеев И. К. Информационная безопасность и защита информации: учебное пособие. М.: Инфра-М. 2010. 336 с.

11. Фатьянов А. А. Проблемы защиты конфиденциальной информации, не составляющей государственную тайну. Информационное общество. 2010. № 5. С. 49–56.

12. Информационная безопасность. М.: Оружие и технологии, 2011. С. 35–44.

13. Рахман П. А. Коэффициент готовности системы обработки данных с основным и резервным узлами. Международный журнал прикладных и фундаментальных исследований. 2015. № 9. С. 608–611.

14. Половко А. М., Гуров С. В. Основы теории надежности. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2006. 704 с.

15. Шишмарев В. Ю. Надежность технических систем: учебник для студ. высш. учеб. заведений. М.: Издательский центр «Академия», 2010. 304 с.

16. Броневич А. Г., Лепский А. Е. Нечеткие модели анализа данных и принятия решений: учебное пособие. Нац. исслед. ун-т «Высшая школа экономики». М.: Изд. дом Высшей школы экономики, 2022. 264 с.

Дата поступления: 23.04.2024
Решение о публикации: 20.05.2024

УДК 656.073

# Математическая модель выкладки товаров на полочном пространстве торговой сети с наличием расширяемых и сужаемых сегментов для определенных видов товаров

**Черняховская Екатерина Сергеевна** — PhD, преподаватель Вроцлавского университета экономики и бизнеса.
E-mail: kateryna.czerniachowska@ue.wroc.pl

Вроцлавский университет экономики и бизнеса, Вроцлав, Республика Польша

**Для цитирования:** Черняховская Е. С. Математическая модель выкладки товаров на полочном пространстве торговой сети с наличием расширяемых и сужаемых сегментов для определенных видов товаров // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 20–29. DOI: 10.20295/2413-2527-2024-238-20-29

***Аннотация.*** *Целью исследования является разработка математической модели выкладки товара на доступном полочном пространстве магазина. Рассмотрены категории товаров (местные, повседневного спроса), которые должны располагаться в определенных сегментах полок. Характерной особенностью модели является возможность расширения и сужения данных сегментов в зависимости от количества ассортимента или сезонных изменений спроса на товар. Цель ретейлера — максимизация прибыли со сбыта товаров при выполнении ограничений на размещение товаров на полках. В обсуждении даются достоинства и недостатки приведенной модели выкладки товара. Исследование имеет важное значение для розничной торговой сети.*

***Ключевые слова:*** *математическое моделирование, оптимизация, распределение места на полках.*

## Введение

Полочное пространство — это дефицитный ресурс, которым ретейлер должен управлять в розничной торговой сети. Поэтому практичные и рациональные мерчандайзинговые решения, а также эффективное управление ограниченным полочным пространством имеют ключевое значение [1]. Оптимизация полочного пространства в магазине позволяет увеличить продажи за счет более эффективного размещения популярных и высокомаржинальных товаров. Тщательно продуманная выкладка товаров не только привлекает внимание покупателей, но и улучшает их опыт покупок, делая поиск нужных товаров проще и быстрее. Это, в свою очередь, способствует увеличению среднего чека и повышению общей лояльности клиентов к магазину.

На предприятиях розничной торговли решения о распределении полочных площадей принимают на двух уровнях:

1. Определение количества полочного пространства для категории товаров.

2. Определение количества полочного пространства для отдельно взятого товара в пределах каждой товарной категории [2].

Классическим инструментом планирования полочного пространства является планограмма. Она представляет собой иллюстрацию полок, показывающую, где на полке будет физически выставлен товар и какое количество этого товара будет на полке. При создании планограммы ретейлеру необходимо тщательно спланировать расположение товаров, количество видимых покупателям

товаров (фейсингов), количество товаров, уложенных позади и над каждым рядом фейсингов, стиль упаковки, а также возможную ориентацию (лицом, боком, верхом, низом) и т. д. [3–4].

Использование данных о покупательском поведении помогает определить наиболее перспективные места для выкладки товаров, привлекая внимание клиентов и стимулируя импульсные покупки. Анализируя маршруты передвижения покупателей и точки их внимания, ретейлеры могут размещать товары, обеспечивая их максимальную заметность. Это не только увеличивает вероятность спонтанных покупок, но и позволяет оптимально использовать полочное пространство, способствуя росту продаж и улучшению общей эффективности магазина. Технологии автоматизации и алгоритмы машинного обучения становятся неотъемлемой частью управления полочным пространством, повышая его эффективность и снижая затраты на ручное регулирование.

**Постановка задачи**

Описываемая задача ранее была опубликована в [5–7]. Пусть будет: $S$ — количество полок; $P$ — количество товаров; $i$ — индекс полки; $i = 1,...,S$; $j$ — индекс товара; $j = 1,...,P$; $g$ — индекс сегмента полки, $g = 1,...,V_i$, $m$ — индекс сегмента и индекс типа товара, $m = 1,...,5$ ($m = 1$ — местный сегмент/товар; $m = 2$ — сегмент/товар повседневного спроса; $m = 3$ — сегмент/товар в центральной части полки; $m = 4$ — сегмент/товар в начале аллейки; $m = 5$ — сегмент/товар в конце аллейки); $n$ — индекс размера сегмента полки, $n = \{1,2\}$; левая ($n = 1$) и правая ($n = 2$) граница; $r$ — индекс подмножества; $r = 1,...,3$ ($r = 1$ — подмножество $A$, подмножество перед специальным сегментом на полке, $r = 2$ — подмножество $C$, подмножество внутри специального сегмента на полке, $r = 3$ — подмножество $B$, подмножество после специального сегмента на полке).

Параметры полки $i$, используемые в модели: $s_i^l$ — длина; $s_i^h$ — высота; $s_i^b$ — грузоподъемность полки; $s_i^{bh}$ — полка для крупногабаритных товаров; $s_i^{fm}$ — полка на нижнем уровне (быстро сбываемые товары); $s_i^{be}$ — полка на уровне глаз (брендовые товары).

Параметры сегмента полки $i$: $V_i$ — количество сегментов на полке; $v_i^w = s_i^l : V_i$ — ширина сегмента; $s^{lc}$ — индекс полки, где выделен местный сегмент, $s^{lc} = \{1,...,S\}$; $v^{lc}$ — индекс местного сегмента на полке $i = s^{lc}$, $v^{lc} = \{1,...,V_i\}$ ; $s^v$ — индекс полки, где находится сегмент для товаров повседневного спроса, $s^v = \{1,...,S\}$; $v^v$ — индекс сегмента товаров повседневного спроса на полке $i = s^v$, $v^v = \{1,...,V_i\}$; $z_{mn}$ — левая ($n = 1$) и правая ($n = 2$) горизонтальные координаты границы сегмента типа $m$.

Левая и правая координаты отрезков вычисляются следующим образом:

$z_{11} = (v^{lc} - 1) \cdot v_i^w$, $z_{12} = v^{lc} \cdot v_i^w$ — для местного сегмента;

$z_{21} = (v^v - 1) \cdot v_i^w$, $z_{22} = v^v \cdot v_i^w$ — для сегмента товаров повседневного спроса;

$z_{31} = v_i^w$, $z_{32} = s_i^l - v_i^w$ — для сегментов в средней части полок;

$z_{41} = 0$, $z_{42} = v_i^w$ — для сегмента возле начала аллейки;

$z_{51} = s_i^l - v_i^w$, $z_{52} = s_i^l$ — для сегмента в конце аллейки.

Параметры товара, используемые в модели: $p_j^w$ — ширина; $p_j^h$ — высота; $p_j^b$ — масса; $p_j^s$ — лимит поставки; $p_j^u$ — прибыль; $p_j^n$ — коэффициент вложенности, $p_j^n < 1$ или $p_j^n = 0$, если товар не может быть вложен; $f_j^{min}$ — минимальное количество фейсингов; $f_j^{max}$ — максимальное количество фейсингов; $c_j^{min}$ — минимальное количество каппингов на одну группу фейсингов; $c_j^{max}$ — максимальное количество каппингов на одну группу фейсингов; $n_j^{min}$ — минимальное количество нестингов в одном фейсинге; $n_j^{max}$ — максимальное количество нестингов в одном фейсинге; $s_j^{min}$ — минимальное количество полок для размещения товара; $s_j^{max}$ — максимальное количество полок для размещения товара; $\lceil p_j^h / p_j^w \rceil$ — необходимое количество фейсингов для поддержки каппингов, положенных над ними; $p_j^{bh}$ — товар должен быть размещен на полке для крупногабаритных товаров; $p_j^{fm}$ — товар должен быть размещен на более низком уровне из-за частоты его покупки, цены или функций; $p_j^{be}$ — товар должен быть размещен на уровне глаз из-за его бренда или цены; $p_j^{af}$, $p_j^{al}$ — товар должен быть размещен рядом с началом/концом аллейки в виду направления движения покупателей,

цены, сезонных характеристик или акций; $p_j^c$ — товар должен располагаться в средней части полки (не возле прохода); $p_j^{lc}$ — местный товар, должен быть размещен в сегменте для местных товаров; $p_j^v$ — товар повседневного спроса, должен быть размещен в сегменте для товаров повседневного спроса.

Переменные решения:

$$x_{ij} = \begin{cases} 1, \text{ если товар } j \text{ положен на полку } i \\ 0, \text{ в противном случае} \end{cases}.$$

$f_{ij}$ — количество фейсингов товара; $c_{ij}$ — количество каппингов товара; $n_{ij}$ — количество нестингов (вложений) товара.

$$y_{mijr}^{ABC} = \begin{cases} 1, \text{ если товар } j \text{ принадлежит множеству } r \\ \quad \text{типа } m \text{ на полке } i \\ 0, \text{ в противном случае} \end{cases}.$$

Существует набор товаров $P$, которые должны быть размещены на $S$ полках на планограмме, на которой каждая полка разделена на $V_i$ сегментов без фиксированной границы между ними. Это позволяет увеличивать или уменьшать сегмент. Цель ретейлера — максимизировать общую прибыль после размещения товаров на планограмме. На планограмме сегмент полки может быть выделен рядом с началом или концом аллейки или в средней части. На планограмме также имеется два сегмента специального назначения, которые можно выделить в любой части полки — местные товары и товары повседневного спроса.

Фейсинг — основная видимая единица товара. Одни товары могут быть размещены поверх фейсинга в боковом положении — это каппинг (рис. 1, *а*). Другие товары могут быть размещены внутри фейсинга — это нестинг (рис. 1, *б*). Коэффициент вложенности для товаров равен $p_j^n < 1$, в противном случае $p_j^n = 0$. Выражение $\lceil p_j^h / p_j^w \rceil$ указывает необходимое количество фейсингов, чтоб можно было положить на них каппинги. Минимальное $c_j^{min}$ и максимальное $c_j^{max}$ количество каппингов на фейсинг, а также минимальное $n_j^{min}$ и максимальное $n_j^{max}$ количество нестингов на фейсинг означает, может ли товар быть вложен один в другой. Величина $c_j^{max}$ ограничивает количество каппингов, которые можно разместить сверху, чтобы фейсинги снизу не деформировались, а каппинги сверху не падали с полки. Величина $n_j^{max}$ ограничивает количество нестингов, которые можно поместить внутрь самого нижнего фейсинга на полке, чтобы нижний товар не деформировался. Общее количество фейсингов изделия является суммой единиц фейсингов $f_{ij}$, каппингов $c_{ij}$ и нестингов $n_{ij}$ товара.

На рис. 2, *а* показано распределение сегментов полок на планограмме. На рис. 2, *б* показаны возможности расширения и сужения сегментов. Если товар должен быть помещен в сегмент, его центр должен находиться внутри сегмента, ограниченного значениями $[z_{m1}; z_{m1}]$. На рис. 2, *б* показаны возможности расширения сегмента в начале аллейки и сужения сегмента в конце аллейки. Сегмент для повседневных товаров расширен. Кроме того, на рис. 2, *б* показан сегмент полки без товаров, расположенных в конце аллейки. Такие случаи могут возникать и с другими типами товаров, такими как местные товары или товары повседневного спроса, в начале аллейки, если товаров не хватает или они размещены на других полках.

Для размещения товаров в сегментах необходимо разделить их на 3 подмножества. $A$ — подмножество товаров, размещенных перед определенным сегментом на полке; $B$ — подмножество товаров, размещенных после определенного сегмента на полке; $C$ — подмножество товаров, размещенных внутри определенного сегмента и помеченных как специфические (местные, товары повседневного спроса, центральная часть полки, первый и последний сегменты). Для решения этой задачи ретейлеру необходимо найти количество фейсингов $f_{ij}$, каппингов $c_{ij}$ и нестингов $n_{ij}$ товара $j$, размещенного на полке $i$, с учетом следующих категорий ограничений: ограничения на полки, ограничения на тип полок, ограничения на товар и ограничения на сегменты полок. Затем товар $j$ должен быть отнесен к определенному типу подмножества $m$ (местный, повседневного спроса, центральная часть полки, в начале или в конце аллейки) подмножества $r$ (то есть до ($A$), после ($B$) или внутри ($C$) конкретного подмножества) на полке $i$ $y_{mijr}^{ABC}$.

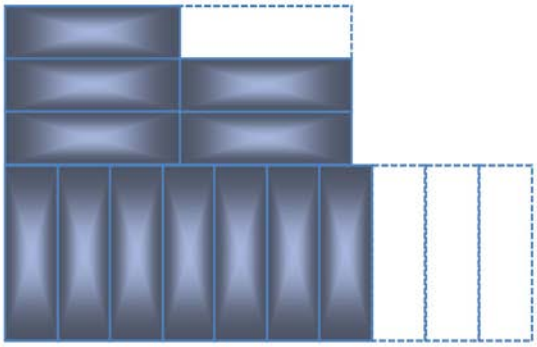


*Рис. 1, а*. Каппинг

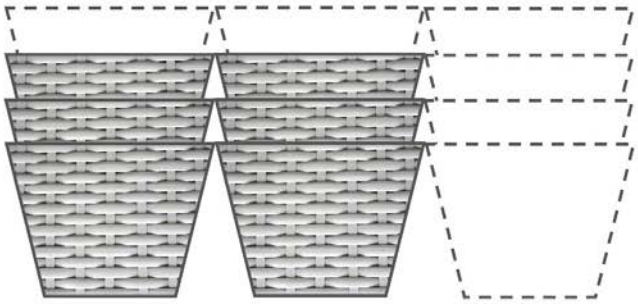


*Рис. 1, б*. Нестинг

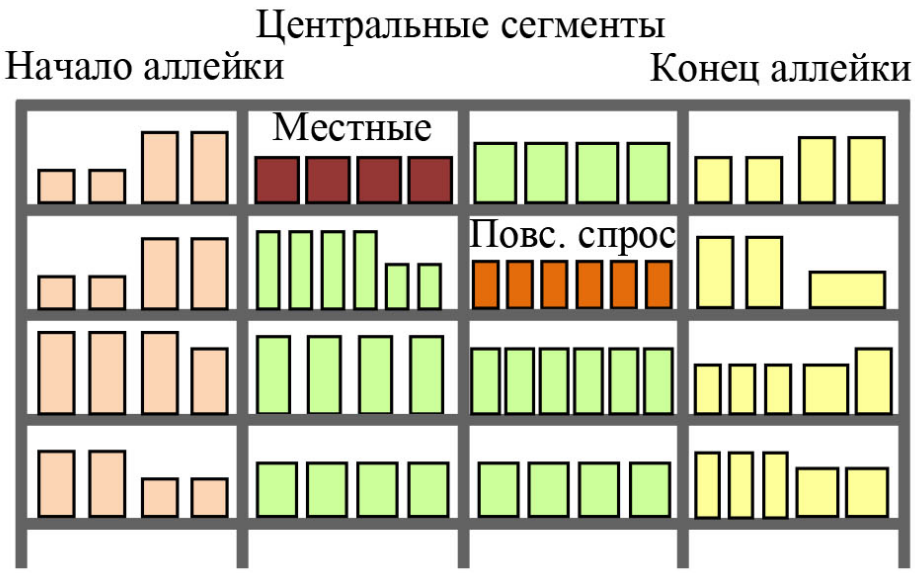



*Рис 2, а*. Сегменты полок на планограмме

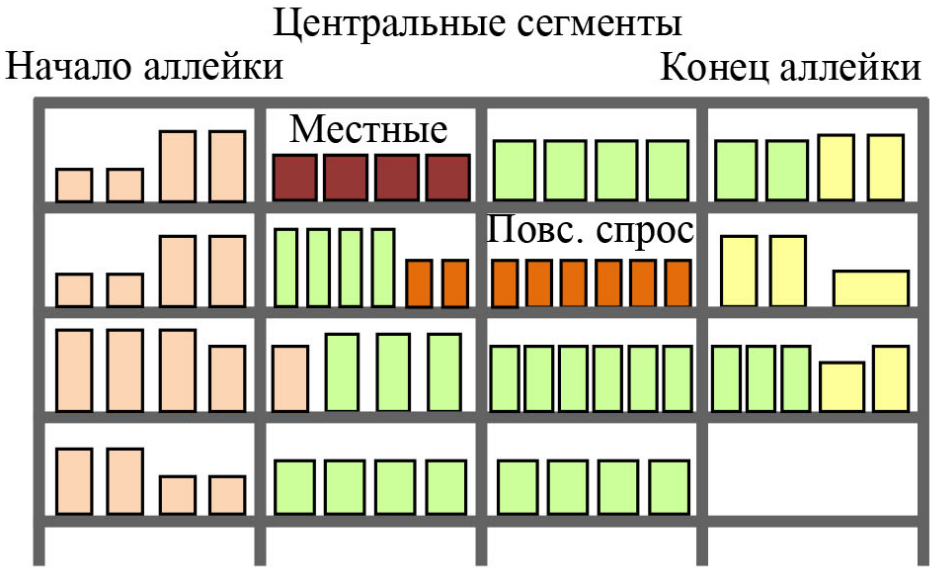



*Рис. 2, б*. Расширение и сужение сегментов полок на планограмме

Критериальная функция — максимизация прибыли:

$$\max\sum_{j=1}^{P}\sum_{i=1}^{S} x_{ij} p_j^{u}(f_{ij} + c_{ij} + n_{ij}). \quad (1)$$

1. Ограничения по полке.

Общая высота товара, каппинга (рис. 1, *а*) и нестинга (рис. 1, *б*) не выше высоты полки:

$$\forall(i,j)[x_{ij}\,(p_j^{h} + \left\lceil \frac{c_{ij}x_{ij}}{\max(\left\lceil \frac{f_{ij}p_j^{w}}{p_j^{h}} \right\rceil,1)} \right\rceil \cdot p_j^{w} + \left\lceil \frac{n_{ij}x_{ij}}{\max(f_{ij},1)} \right\rceil \cdot p_j^{h} p_j^{n}) \le s_i^{h}]. \quad (2)$$

Выражение $\frac{x_{ij}}{\max(f_{ij},1)}$ означает обход деления 0, если товар не поставлен на полку.

Масса товаров на полке:

$$\forall(i)[\sum_{j=1}^{P}(f_{ij} + c_{ij} + n_{ij})p_j^{b} \le s_j^{b}]. \quad (3)$$

Длина полки:

$$\forall(i)[\sum_{j=1}^{P} f_{ij} p_j^{w} \le s_j^{l}]. \quad (4)$$

2. Ограничения по типу полки.

Малогабаритный товар не может быть размещен на полке для крупногабаритных товаров:

$$\forall(j : p_j^{bh} = 0)\forall(i : s_j^{bh} = 1)[x_{ij} = 0]. \quad (5)$$

Крупногабаритный товар не может быть размещен на полке для малогабаритных товаров:

$$\forall(j : p_j^{bh} = 1)\forall(i : s_j^{bh} = 0)[x_{ij} = 0]. \quad (6)$$

Определенный товар должен быть размещен на уровне глаз, другие товары также могут быть размещены на уровне глаз:

$$\forall(j : p_j^{be} = 1)\forall(i : s_j^{be} = 0)[x_{ij} = 0]. \quad (7)$$

Товар должен быть размещен на нижней полке для быстро сбываемых товаров, другие товары также могут быть размещены на этом уровне при наличии места на полке:

$$\forall(j : p_j^{fm} = 1)\forall(i : s_i^{fm} = 0)[x_{ij} = 0]. \quad (8)$$

Местные товары не могут быть размещены в других сегментах полок, кроме как в соответствующем сегменте:

$$\forall(j : p_j^{lc} = 1)\forall(i : i \ne s^{lc})[x_{ij} = 0]. \quad (9)$$

Товары повседневного спроса не могут быть размещены в других сегментах полок, кроме как в соответствующем сегменте:

$$\forall(j : p_j^{v} = 1)\forall(i : i \ne s^{v})[x_{ij} = 0]. \quad (10)$$

3. Ограничения на товар.

Минимальное и максимальное количество полок:

$$\forall(j)[s_j^{min} \le \sum_{i=1}^{S} x_{ij} \le s_j^{max}]. \quad (11)$$

4. Лимит поставки.

$$\forall(j)[\sum_{i=1}^{S}(f_{ij} + c_{ij} + n_{ij}) \le p_j^{s}]. \quad (12)$$

Минимальное и максимальное количество фейсингов:

$$\forall(j)[f_j^{min} \le \sum_{i=1}^{S} f_{ij} \le f_j^{max}]. \quad (13)$$

Минимальное и максимальное количество каппингов:

$$\forall(i,j)[c_j^{min} \le c_{ij} \le c_j^{max} \cdot \left\lfloor \frac{f_{ij}p_j^{w}}{p_j^{h}} \right\rfloor]. \quad (14)$$

Минимальное и максимальное количество нестингов:

$$\forall(i,j)[n_j^{min} \le n_{ij} \le n_j^{max} f_{ij}]. \quad (15)$$

5. Ограничения на сегменты полки:

Обычный товар относится к подмножеству *A* или *B*, если он размещен на полке:

$$\forall(m,i)\forall(j : p_j^{lc} = 0, p_j^{v} = 0, p_j^{c} = 0, p_j^{af} = 0, p_j^{al} = 0)[(y_{mij1}^{ABC} + y_{mij3}^{ABC} = x_{ij}) \wedge (y_{mij2}^{ABC} = 0)]. \quad (16)$$

Специальный товар относится к подмножеству *C*, если он размещен на полке:

$$\forall(m,i)\forall(j : p_j^{lc} = 1, p_j^{v} = 1, p_j^{c} = 1, p_j^{af} = 1, p_j^{al} = 1)[(y_{mij1}^{ABC} = 0) \wedge (y_{mij2}^{ABC} = x_{ij}) \wedge (y_{mij3}^{ABC} = 0)]. \quad (17)$$

Обычный товар на полке для местных товаров относится к подмножеству *A* или *B*, если он размещен на полке:

$$\forall(m)\forall(i : i = s^{lc})\forall(j : p_j^{lc} = 0)[(y_{mij1}^{ABC} + y_{mij3}^{ABC} = x_{ij}) \wedge (y_{mij2}^{ABC} = 0)]. \quad (18)$$

Местный товар на полке для местных товаров относится к подмножеству *C*, если он размещен на полке:

$$\forall(m)\forall(i : i = s^{lc})\forall(j : p_j^{lc} = 1)[(y_{mij1}^{ABC} = 0) \wedge (y_{mij2}^{ABC} = 1) \wedge (y_{mij3}^{ABC} = 0)]. \quad (19)$$

Обычный товар на полке товаров повседневного спроса относится к подмножеству *A* или *B*, если он размещен на полке:

$$\forall(m)\forall(i : i = s^{v})\forall(j : p_j^{v} = 0)[(y_{mij1}^{ABC} + y_{mij3}^{ABC} = x_{ij}) \wedge (y_{mij2}^{ABC} = 0)]. \quad (20)$$

Товары повседневного спроса на соответствующей полке относятся к подмножеству *C*, если они размещены на полке:

$$\forall(m)\forall(i : i = s^{v})\forall(j : p_j^{v} = 1)[(y_{mij1}^{ABC} = 0) \wedge (y_{mij2}^{ABC} = 1) \wedge (y_{mij3}^{ABC} = 0)]. \quad (21)$$

Размер сегментов центральной части полки:

$$\begin{aligned}
&\forall(m : z_{m1} > 0 \wedge z_{m2} < s_i^l)\forall(i) \\
&[\ (\ (\sum_{j=1}^{P} y_{mij1}^{ABC} f_{ij} p_j^w + \frac{\sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w}{2} \le z_{m1}) \wedge (\sum_{j=1}^{P} y_{mij3}^{ABC} f_{ij} p_j^w \le s_i^l - (z_{m1} + \frac{\sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w}{2}))\\
&\ )\ \vee \\
&\quad \vee\ (\ (\sum_{j=1}^{P} y_{mij1}^{ABC} f_{ij} p_j^w + \frac{\sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w}{2} \ge z_{m1}) \wedge (\sum_{j=1}^{P} y_{mij1}^{ABC} f_{ij} p_j^w + \frac{\sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w}{2} \le z_{m2}) \wedge \\
&\qquad \wedge (\sum_{j=1}^{P} y_{mij3}^{ABC} f_{ij} p_j^w \le s_i^l - \max(\sum_{j=1}^{P} y_{mij1}^{ABC} f_{ij} p_j^w + \sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w, z_{m1} + \frac{\sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w}{2}))\\
&\qquad )].
\end{aligned} \quad (22)$$

Первая часть ограничения представляет случай, когда на полке слишком много свободного места, вторая часть представляет случай частого расположения товаров, когда полка почти полностью заполнена. Определенные товары (подмножество *C*) размещаются внутри определенного сегмента полки таким образом, чтобы центр каждого товара находился в границах конкретного сегмента. Таким образом, товары из этого подмножества могут выходить за границы сегмента не более чем на $\sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w/2$. Следовательно, максимальная ширина специального сегмента имеет следующие координаты: $[z_{m1} - \sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w/2;\ z_{m2} + \sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w/2]$. Остальные товары распределяются по подмножествам *A* и *B*. Правила сужения и расширения этих сегментов аналогичны случаям с подмножеством *C*.

Размер сегментов в начале аллейки:

$$\forall(m : z_{m1} = 0)\forall(i)[(\sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w \le \frac{3}{2} \cdot z_{m2}) \wedge (\sum_{r=1}^{3}\sum_{j=1}^{P} y_{mijr}^{ABC} f_{ij} p_j^w \le s_i^l)]. \quad (23)$$

Размер сегментов в конце аллейки:

$$\forall(m : z_{m2} = s_i^l)\forall(i)[(\sum_{j=1}^{P} y_{mij2}^{ABC} f_{ij} p_j^w \le \frac{3}{2} \cdot (s_i^l - z_{m1})) \wedge (\sum_{r=1}^{3}\sum_{j=1}^{P} y_{mijr}^{ABC} f_{ij} p_j^w \le s_i^l)]. \quad (24)$$

Ограничение и написаны для обозначения начала и конца аллейки (первый и последний сегменты). Общая ширина товаров, отнесенных к 3 подмножествам, не должна превышать длину полки. Расширение первого сегмента допустимо до $\frac{3}{2} \cdot z_{m2}$, его максимальная ширина равна максимальной ширине последнего сегмента $\frac{3}{2} \cdot [s_i^l - z_{m1}]$. Это гарантирует, что центр товара будет находиться внутри крайних сегментов.

6. Ограничения отношений:

$$\forall(i,j)[x_{ij} \cdot \frac{s_i^l}{p_j^w} \ge f_{ij}], \quad (25)$$

$$\forall(i,j)[x_{ij} \le f_{ij}], \quad (26)$$

$$\forall(i,j)[c_{ij} \le x_{ij} \cdot c_j^{max} \cdot \left\lfloor f_{ij} \cdot \frac{p_j^w}{p_j^h} \right\rfloor], \quad (27)$$

$$\forall(i,j)[n_{ij} \le x_{ij} \cdot n_j^{max} \cdot f_{ij}], \quad (28)$$

$$\forall(m,i,j)[\sum_{r=1}^{3} y_{mijr}^{ABC} = x_{ij}]. \quad (29)$$

7. Переменные решения.

Товар размещен на полке:

$$x_{ij} \in \{0,1\}\forall(i,j). \quad (30)$$

Количество фейсингов товара на полке:

$$f_{ij} = \{f_j^{min} \ldots f_j^{max}\}\forall(i,j). \quad (31)$$

Количество каппингов товара на полке:

$$c_{ij} = \{c_j^{min} \ldots c_j^{max} \cdot \lfloor f_j^{max} \cdot p_j^w / p_j^h \rfloor\}\forall(i,j). \quad (32)$$

Количество нестингов товара на полке:

$$n_{ij} = \{n_j^{min} \ldots n_j^{max}\}\forall(i,j). \quad (33)$$

Подмножество товара на полке:

$$y_{mijr}^{ABC} \in \{0,1\}\forall(m,i,j,r). \quad (34)$$

**Обсуждение**

Организация выкладки товаров на полочном пространстве в магазинах с выделением отдельных категорий товаров, которые должны быть размещены в определенных сегментах полок, приводит к лучшей видимости данных товаров, а следовательно, большей вероятности их покупки. Расширяемые и сужаемые сегменты полок дают возможность гибко адаптировать выкладку

в зависимости от сезонных изменений спроса и маркетинговых акций.

Приведенный способ выкладки товаров на полки с использованием расширяемых и сужаемых полочных сегментов имеет следующие достоинства:

1. Повышение продаж специальных товаров. Математическая модель позволяет выделить специальные товары и разместить их в отдельных сегментах, которые можно расширить или сузить, но месторасположение сегмента остается неизменным. Это привлекает внимание покупателей и способствует увеличению продаж в том числе уникальных товаров (местных, повседневного спроса), которые доступны только в данном магазине.

2. Оптимизация использования полочного пространства. Модель учитывает возможность сужения или расширения сегментов в зависимости от спроса на товары, что позволяет более эффективно использовать полочное пространство. Это особенно полезно в условиях ограниченного пространства в магазине.

3. Улучшение клиентского опыта. Расположение часто покупаемых товаров повседневного спроса в удобных для доступа местах способствует улучшению клиентского опыта, так как покупатели могут быстро найти необходимые товары.

4. Гибкость и адаптивность. Модель позволяет легко адаптировать выкладку товаров в зависимости от изменения спроса, сезонности или маркетинговых акций, что обеспечивает гибкость управления ассортиментом.

5. Стимулирование импульсных покупок. Стратегическое размещение специальных товаров в конкретных сегментах полки может стимулировать импульсные покупки, увеличивая общий объем продаж.

Несмотря на все достоинства, приведенный способ выкладки товаров может иметь и недостатки, среди которых можно выделить следующие:

1. Определенная сложность в реализации и управлении. Реализация такой модели требует дополнительных ресурсов для анализа спроса, постоянного мониторинга продаж и корректировки выкладки, что может быть трудоемким и затратным процессом.

2. Сложности с обучением персонала. Персонал магазина может потребовать дополнительного обучения для правильного управления и адаптации полочных сегментов в соответствии с моделью, что требует времени и ресурсов.

Математическая модель выкладки товаров с использованием расширяемых и сужаемых полочных сегментов представляет собой инновационный подход к управлению ассортиментом, который имеет потенциал для значительного увеличения продаж и улучшения клиентского опыта. Однако для успешной реализации данной модели необходимо учитывать ее сложность, необходимость точного прогнозирования спроса и возможные риски, связанные с доступностью товаров и перегрузкой специальных сегментов.

**Заключение**

Грамотное использование полочного пространства включает в себя анализ покупательских маршрутов и предпочтений для максимизации доступности ключевых товаров. В данной работе приводится математическая модель организации выкладки товаров на полки в магазине. Характерной особенностью такой модели является наличие полочных сегментов без фиксированного размера, то есть сегменты на полках можно расширять и сужать в зависимости от наличия товарных категорий. Следующей особенностью модели является расстановка единиц товара на полке как фейсинг, нестинг и каппинг. Это позволяет сэкономить место при размещении большего количество товара на одной и той же торговой поверхности.

Дальнейшие исследования будут направлены на создание алгоритмов прогноза спроса на различные категории товаров, что необходимо для обеспечения работоспособности приведенной модели.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК:**

1. Zhang W., Rajaram K. Managing limited retail space for basic products: space sharing vs. space dedication // European Journal of Operational Research. 2017. Vol. 263. P. 768–781.

2. Bianchi-Aguiar T., Silva E., Guimarães L., et al. Using analytics to enhance a food retailer's shelf-space management // Interfaces. 2016. Vol. 46, no. 5. P. 424–444.

3. Bianchi-Aguiar T., Silva E., Guimarães L., et al. Alocating products on shelves under merchandising rules: Multi-level product families with display directions // Omega. 2018. Vol. 76. P. 47–62.

4. Reyes P. M., Frazier G. V. Goal programming model for grocery shelf space allocation // European Journal of Operational Research. 2007. Vol. 181, no. 2. P. 634–644.

5. Czerniachowska K. A. Genetic algorithm for the retail shelf space allocation problem with virtual segments // OPSEARCH. 2022. Vol. 59, no. 1. P. 364–412.

6. Czerniachowska K., Lutosławski K., Kozina A., et al. Shelf space allocation problem with horizontal shelf division // Procedia Computer Science. 2021. Vol. 192. P. 1550–1559.

7. Czerniachowska K., Wichniarek R. Shelf space allocation models in retail enterprise // Zarządzanie Przedsiębiorstwem. Enterprise Management. 2020. Vol. 23, no. 2. P. 11–15.

Дата поступления: 05.06.2024
Решение о публикации: 13.06.2024

# Mathematical Model of Product Display on the Shelf Space of a Retail Chain with the Presence of Expandable and Contractible Segments for Definite Types of Products

**Ekaterina S. Chernyakhovskaya** — PhD in Management and Quality, Lecturer at the Wroclaw University of Economics and Business. E-mail: kateryna.czerniachowska@ue.wroc.pl

Wroclaw University of Economics and Business

**For citation:** Chernyakhovskaya E. K. Mathematical model of product display on the shelf space of a retail chain with the presence of expandable and contractible segments for definite types of products // Intelligent technologies on transport. 2024. No. 2 (38). P. 20–29. (In Russian). DOI: 10.20295/2413-2527-2024-238-20-29

***Abstract.*** *The goal of the study is to develop a mathematical model for displaying products on the existing shelf space of a store. Categories of products (local, everyday demand products) that should be located in certain segments of shelves are considered. A characteristic feature of the model is the possibility of expanding and narrowing these segments depending on the number of assortments or seasonal changes in demand for products. The goal of the retailer is to maximize profits from the sale of products while complying with restrictions on the placement of products on shelves. The discussion gives the advantages and disadvantages of the given product display model. The study is important for retail chains.*

***Keywords:*** *mathematical modelling, optimization, shelf space allocation.*

**REFERENCES**

1. Zhang W., Rajaram K. Managing limited retail space for basic products: space sharing vs. space dedication // European Journal of Operational Research. 2017. Vol. 263. P. 768–781.

2. Bianchi-Aguiar T., Silva E., Guimarães L., et al. Using analytics to enhance a food retailer's shelf-space management // Interfaces. 2016. Vol. 46, no. 5. P. 424–444.

3. Bianchi-Aguiar T., Silva E., Guimarães L., et al. Alocating products on shelves under merchandising rules: Multi-level product families with display directions // Omega. 2018. Vol. 76. P. 47–62.

4. Reyes P. M., Frazier G. V. Goal programming model for grocery shelf space allocation // European Journal of Operational Research. 2007. Vol. 181, no. 2. P. 634–644.

5. Czerniachowska K. A. Genetic algorithm for the retail shelf space allocation problem with virtual segments // OPSEARCH. 2022. Vol. 59, no. 1. P. 364–412.

6. Czerniachowska K., Lutosławski K., Kozina A., et al. Shelf space allocation problem with horizontal shelf division // Procedia Computer Science. 2021. Vol. 192. P. 1550–1559.

7. Czerniachowska K., Wichniarek R. Shelf space allocation models in retail enterprise // Zarządzanie Przedsiębiorstwem. Enterprise Management. 2020. Vol. 23, no. 2. P. 11–15.

Received: 05.06.2024
Accepted: 13.06.2024

УДК 004.052

# Оптимизация параллельной обработки информации в отказоустойчивой вычислительной системе мобильного объекта с временной избыточностью вычислительного процесса

**Захаров Иван Вячеславович** — докт. техн. наук, доцент, профессор кафедры математического и программного обеспечения. E-mail: x.vano-z80@yandex.ru

**Мусаллам Али** — адъюнкт кафедры математического и программного обеспечения. E-mail: alloushi1987@gmail.com

Военно-космическая академия имени А. Ф. Можайского, Россия, Санкт-Петербург

**Для цитирования:** Захаров И. В., Мусаллам А. Оптимизация параллельной обработки информации в отказоустойчивой вычислительной системе мобильного объекта с временной избыточностью вычислительного процесса // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 30–36. DOI: 10.20295/2413-2527-2024-238-30-36

***Аннотация.*** *Задачи эффективного применения многопроцессорных вычислительных систем (ВС) мобильных объектов требуют рациональной организации вычислительных процессов на борту. В статье рассмотрен параллельный вычислительный процесс с временной избыточностью, предусматривающий периодическое сохранение текущего состояния вычислений с возможностью их восстановления в целях обеспечения отказоустойчивости. Кратко проанализировано ранее известное решение частной оптимизационной задачи выбора периодичности сохранения текущего состояния вычислительного процесса в условиях возможных отказов. Предложен новый усовершенствованный способ решения указанной задачи. Представлены сравнительные результаты ее решения различными способами и сделаны выводы о целесообразности использования предложенных подходов.*

***Ключевые слова:*** *многопроцессорная вычислительная система, параллельные вычисления, контрольные точки, отказоустойчивость вычислений.*

## Введение

Актуальным направлением современных технологий является создание мобильных объектов с высокой степенью автономности, управляемых интеллектуальными комплексами управления на базе бортовых вычислительных систем (ВС), функционирующих в режимах реального времени. К таким ВС предъявляются жесткие требования по надежности и производительности, что весьма тесно связано с проблемами обеспечения их отказоустойчивости. Важным аспектом реализации данного направления является диспетчеризация вычислительного процесса на основе учета условий функционирования.

Очевидно, что невозможно создать абсолютно надежную аппаратуру, в особенности сложное оборудование, поэтому важна разработка программных средств обеспечения отказоустойчивости. С этой точки зрения существенное значение имеют методы

восстановления, основанные на так называемых контрольных точках (КТ) [1]. Данный подход предполагает следующее. В составе ВС имеется запоминающее устройство, к которому имеют доступ все ВМ системы. Оно предназначается для сохранения промежуточных результатов вычислений и других необходимых для управления вычислительным процессом параметров. К нему возможны обращения со стороны ВМ, которые могут быть назначены для продолжения решения прерванной задачи. Таким образом, механизмы управления ходом вычислений в системе обеспечивают периодическую запись состояния вычислительного процесса в централизованное запоминающее устройство [1, 2].

В то время как традиционные подходы предполагают решение прерванной отказом ВМ задачи на исправном с ее начала, вызывая тем самым существенные временные потери и соответствующее снижение производительности, в данном случае восстановление хода решения будет производиться с крайнего сохранения, то есть последней КТ. При этом потери времени, то есть время вычислений, обесцененное отказами ВМ, значительно снижается. В наихудшем случае оно будет соответствовать интервалу между соседними сохранениями — интервалу между КТ. Таким образом, в условиях возможных отказов того или иного количества вычислительных модулей (ВМ) системы возможно обеспечить параллельную обработку информации на основе указанного подхода.

Следует, однако, отметить, что сложность организации вычислительного процесса при этом неизбежно возрастает. Ведущая роль здесь отводится программному обеспечению ВС, которое осуществляет формирование КТ, совокупность процедур автоматического восстановления вычислений, а также в значительной степени реализует протоколы межмодульного взаимодействия [1].

Нужно иметь в виду, что увеличение количества КТ позволяет затрачивать меньше времени на завершение вычислений прерванной отказом ВМ задачи, но, с другой стороны, создание каждой КТ вносит временную избыточность в вычислительный процесс. Таким образом, возникает задача оптимизации количества КТ. Очевидно, что решение этой задачи зависит от оценки опасности отказов ВМ, а также временных затрат на сохранение и восстановление вычислений, а также других параметров ВС и вычислительных процессов. В [3, 4] предложен подход к решению указанной задачи и оценен ожидаемый эффект. Он, по нашему мнению, безусловно, интересен и полезен, однако не лишен некоторых недостатков, имея потенциал к совершенствованию.

## Подход к оптимизации количества контрольных точек при отказоустойчивой параллельной обработке информации

Будем считать для определенности, что ВС, которая должна выполнить определенный набор вычислительных задач, однородна и состоит из некоторого числа ВМ. Предположим, что вычислительные задачи взаимонезависимы и имеют одинаковую вычислительную трудоемкость (объем вычислений). В соответствии с установленными правилами диспетчеризации вычислительного процесса каждый ВМ решает в любой момент времени только одну задачу (или вовсе простаивает, будучи исправным, или же находится в состоянии отказа).

В рамках реализации рассматриваемого подхода каждый ВМ записывает на общее запоминающее устройство состояние выполняемой им программы вычислительной задачи (например, промежуточные результаты вычислений, дампы своей памяти, своих регистров и т. п.), то есть создает ее КТ. Периодичность этого действия определена и задается соответствующим алгоритмом.

Выход из строя ВМ вызывает прерывание задачи, решаемой на нем. Те задачи, которые были спланированы на отказавшие ВМ, выполняются на работоспособных ВМ с момента крайней КТ прерванной задачи. В силу высказанного выше допущения об однородности ВС все ВМ могут выполнить любую задачу. Таким образом, за счет наличия временной избыточности возможно обеспечить выполнение набора задач при отказах ВМ.

Общее время выполнения всех задач будет зависеть не только от времени отказов ВМ, но и от количества создаваемых при решении каждой задачи КТ и временных затрат на эти действия. Поскольку увеличение количества КТ уменьшает,

с одной стороны, остаточный объем вычислений при отказах, с другой — само по себе требует временных затрат, встает задача поиска оптимального количества КТ для заданных параметров ВС, вычислительных задач и вероятностно-временных характеристик отказов ВМ.

Итак, авторы [3, 4], в общем, справедливо полагают, что период решения задачи состоит из $k$ интервалов, каждый из которых, кроме последнего, включает интервал $\tau$ выполнения задачи и интервал $\Delta$ создания КТ. Для определенности считается, что отказы ВМ происходят в один и тот же, но заранее не известный случайный момент времени. Тогда, если отказ одного ВМ происходит в момент времени $\xi$, общее время решения задачи составит

$$\theta = \xi + T - (i-1)\tau = \xi + T\left(1 - \frac{1}{k}\left\lfloor \frac{k\,\xi}{T + k\Delta} \right\rfloor\right), \quad (1)$$

где $T = k\,\tau$ — время, необходимое для решения задачи без учета отказов;

$i = \lfloor \xi/(\tau + \Delta) \rfloor + 1$ — номер интервала, на котором произошел отказ ВМ;

$\lfloor \square \rfloor$ — ближайшее целое, меньшее или равное.

Если известна плотность распределения $f(\xi)$ момента времени отказа ВМ, то математическое ожидание $\overline{\theta}$ времени выполнения задачи составит $\overline{\theta} = \int_0^T f(\xi)\theta(k,\xi)\partial\xi$.

Для оценки $\overline{\theta}$ и решения оптимизационной задачи $\overline{\theta}(k) \to min$ авторы [3, 4] поступают следующим образом. Среднее значение общего времени $\overline{\theta}$, затрачиваемого на решение задачи, определяется как

$$\overline{\theta} = (1 - q^2)\overline{\theta}_0 + 2q(1-q)\overline{\theta}_1, \quad (2)$$

где $\overline{\theta}_1$ — среднее время выполнения задачи при отказе одного из двух ВМ;

$\overline{\theta}_0$ — время выполнения задачи при безотказной работе ВС;

$q$ — вероятность отказа ВМ.

Надо полагать, что здесь авторы рассмотрели два ВМ из состава ВС, не учитывая в оценке $\overline{\theta}$ случай отказа обоих ВМ в предположении, что общее число ВМ достаточно велико и задача будет в любом случае переназначена на исправный модуль. Заметим, что, на наш взгляд, полученная таким путем оценка $\overline{\theta}$ может оказываться излишне оптимистичной, так как не учитывает полную группу событий. Далее оценка $\overline{\theta}_0$ рассчитывается довольно тривиально, а $\overline{\theta}_1$ авторы рассчитали для равномерного закона распределения момента отказа как [3, 4]

$$\overline{\theta}_1 = \frac{1}{2}\left(3T + (k-1)\Delta - \frac{T(T + k\Delta)(k-1)}{k(T + (k-1)\Delta)}\right). \quad (3)$$

Используя (2), (3) и дифференцируя полученную сумму по $k$, оптимальное количество КТ возможно найти путем численного решения относительно $k$ уравнения четвертой степени $\frac{\partial\overline{\theta}}{\partial k} = 0$:

$$\Delta k^2(k^2\Delta^2 + 2k\Delta T - 2k\Delta^2 - 2\Delta T + \Delta^2 + T^2) -$$

$$- qT^2(2k\Delta - \Delta + T) = 0. \quad (4)$$

При этом значение искомого корня уравнения округляется до натурального числа с учетом того, что количество КТ на одну меньше количества интервалов $k$ [4].

### Уточнение оценки средних потерь времени вычислительного процесса при отказах

На наш взгляд, вместо соотношения (2) и вытекающих из него результатов более корректно рассуждать следующим образом. Оценим среднее время вычислений, потерянное («обесцененное») в результате отказа ВМ, величиной $\tau/2$, а его ожидаемая величина с учетом риска отказа — $q\tau/2$. Время, затраченное на создание КТ, составит $(k-1)\Delta$. Таким образом, оценкой математического ожидания избыточного времени решения задачи будет выступать величина

$$\overline{\beta} = \frac{q\tau}{2} + (k-1)\Delta = \frac{q\mathrm{T}}{2k} + (k-1)\Delta. \quad (5)$$

Приравнивая к нулю производную по $k$ выражения (5), получим

$$-\frac{q\mathrm{T}}{2k^2} + \Delta = 0,\ k = \sqrt{\frac{qT}{2\Delta}} = \sqrt{\frac{qT}{2\delta}}, \quad (6)$$

где $\delta = \Delta/T$ — доля временных затрат на создание одной КТ относительно времени «чистого» решения задачи.

Поскольку $k > 0$, округление будем производить в бóльшую сторону.

По нашему мнению, предложенный способ получает преимущество за счет более корректной оценки средних потерь времени вычислительного процесса при отказах. Кроме того, значительно упрощаются анализ и прикладные расчеты. Укажем, что взгляд на соотношение (6) демонстрирует соответствие ожидаемым теоретическим представлениям и здравому смыслу. Так, количество КТ растет с увеличением опасности отказов и уменьшением временных затрат на их создание. Создание КТ не имеет смысла при нулевой вероятности отказа и при времени создания одной КТ более половины времени непосредственного решения задачи. Ожидаемо размерность (масштаб) единиц времени не играет роли.

Проведем сравнительный анализ исходного и предлагаемого способов нахождения оптимального числа КТ при помощи имитационного моделирования.

Необходимо отметить следующее. Как в случае использования (4), так и (6), $k$ не зависит от количества ВМ и количества задач, что следует из принятых допущений. Однако достигаемый эффект будет различным и при получении его оценок указанные величины требуются для расчетов на имитационной модели. Для определенности целесообразно принимать количество ВМ M равным числу задач N, а затем исследовать случай, при котором M > N (случай M < N не соответствует поставленной задаче, так как требует увеличения рассматриваемого интервала T).

Суть имитационного моделирования состоит в следующем. Разыгрывается случайное время $\xi \in [0;T]$ возможных с вероятностью $q$ отказов ВМ. Задачи с отказавших ВМ переназначаются на исправные по порядку очередности в силу допущения об их однородности. Достигаемый эффект будем оценивать как выигрыш $\sigma$ по средней производительности на интервале решения набора задач. (Оценка эффекта по среднему времени вычислительного процесса, как исследовали авторы [3, 4], представляется не вполне корректной, так как в случаях отказов всех ВМ время решения стремится в бесконечность.)

Графики с примером зависимости ожидаемого относительного выигрыша в производительности

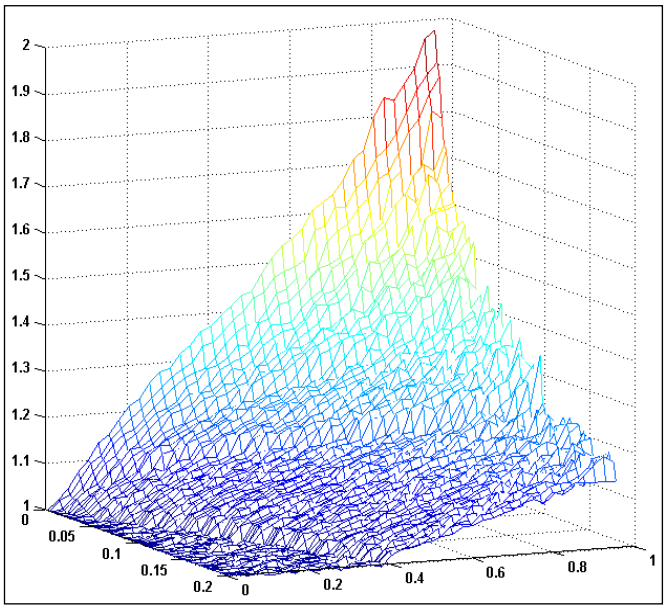



*Рис. 1.* Зависимость относительного выигрыша в производительности от реализации отказоустойчивого вычислительного процесса с временной избыточностью от вероятности отказа ВМ при M = N = 8

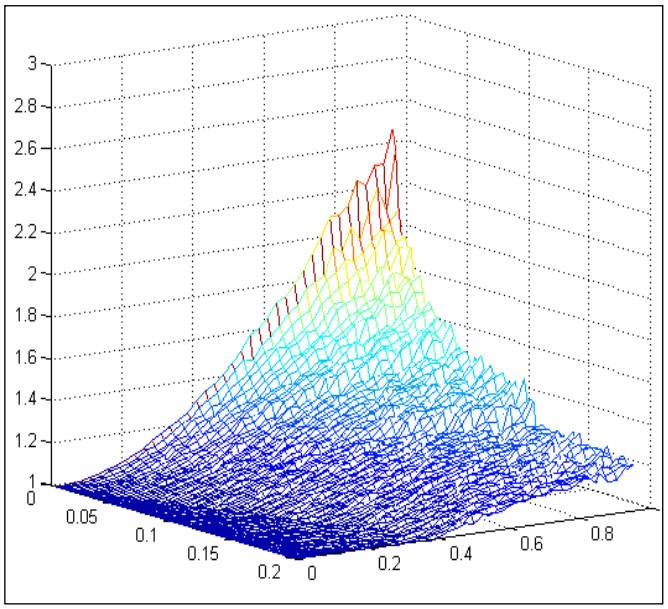



*Рис. 2.* Зависимость относительного выигрыша в производительности от реализации отказоустойчивого вычислительного процесса с временной избыточностью от вероятности отказа ВМ при M = 6; N = 3

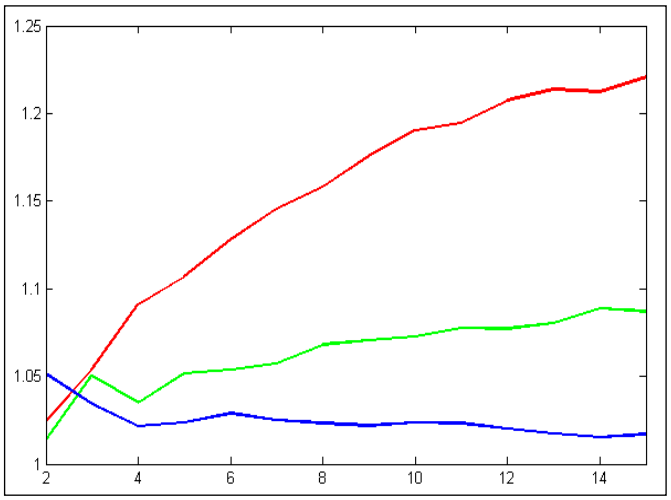



*Рис. 3.* Зависимость относительного выигрыша в производительности от реализации отказоустойчивого вычислительного процесса с временной избыточностью от количества ВМ в ВС при $q = 0{,}2$; $\delta = 0{,}01$

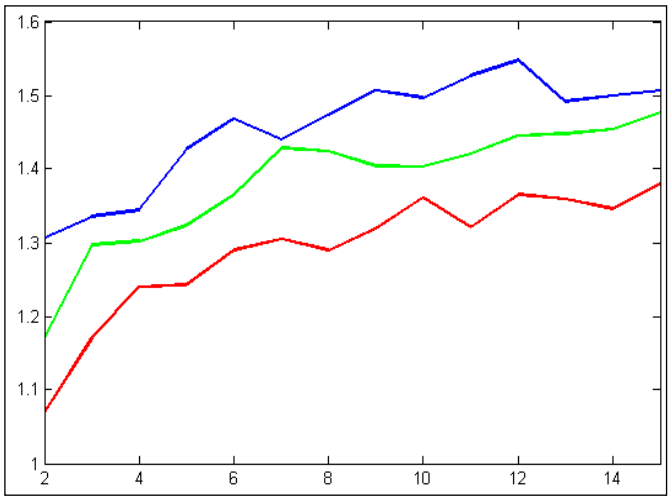



*Рис. 4.* Зависимость относительного выигрыша в производительности от реализации отказоустойчивого вычислительного процесса с временной избыточностью от количества ВМ в ВС при $q = 0{,}5$; $\delta = 0{,}01$

$\sigma(\delta,q)$ при некоторых различных наборах количества ВМ и количества задач для предлагаемого способа оптимизации количества КТ представлены на рис. 1 и 2.

На рис. 3 и 4 представлены графики зависимости $\sigma$ от количества ВМ M при M = N, а также при случае наличия ВМ в холодном резерве (для определенности двукратного M = 2N и трехкратного M = 3N) при различных вероятностях отказа $q$. На рис. 3 и 4 красная линия соответствует случаю M = N, зеленая — M = 2N, синяя — M = 3N.

Данные примеры показывают возможность существенного снижения потерь производительности для указанных условий и допущений. Отличия в характере поведения графиков для рассмотренных случаев на рис. 3 и 4 можно объяснить целесообразностью увеличения резерва ВМ с увеличением опасности отказов, в то время как наличие значительного резерва при низком риске отказов снижает выигрыш от применения механизма сохранения и восстановления при помощи КТ. Отметим явную тенденцию выхода графиков на «насыщение» с ростом количества ВМ в случаях определенного соотношения параметров (например, как видно из рис. 2, вариант параметров $q = 0{,}2$; $\delta = 0{,}01$ не является существенно выигрышным для M = 3N — трехкратный аппаратный резерв значительно парирует отказы и снижает актуальность реализации механизма КТ). Наблюдаемые немонотонные выбросы значений $\sigma$ можно объяснить дискретным изменением числа $k$ интервалов и не в полной мере учтенными, по-видимому, особенностями его округления до целого в соотношении (6). Таким образом, целесообразно в дальнейшем рассмотреть вопросы эффективности технологии сохранения и восстановления вычислительных процессов на основе КТ во взаимосвязи с параметрами структур ВС.

## Заключение

Анализ предложенного способа оптимизации количества КТ показал целесообразность его использования при разработке алгоритмов управления вычислительным процессом ВС мобильных объектов, функционирующих в условиях опасного деструктивного влияния неблагоприятных факторов различного происхождения.

Результаты моделирования показывают возможность существенного снижения потерь производительности ВС при отказах ее компонентов в сравнении с организацией вычислений без сохранения состояния вычислительного процесса,

а также в сравнении с ранее известным способом расчета оптимальной периодичности создания КТ. Таким образом, следовательно, обеспечивается соответствующее снижение потерь от срывов выполнения вычислительных задач и повышение коэффициента оперативной готовности ВС. Тем не менее обсуждаемые вопросы требуют дальнейшего исследования в рамках развития теоретических подходов и прикладных методик организации отказоустойчивых вычислительных систем [5, 6]. Так, например, целесообразно рассмотреть неоднородные ВМ и неоднородные вычислительные задачи, учесть директивные сроки выполнения задач, учесть применение различных дисциплин обслуживания, промоделировать отказы с разнообразными законами распределения и стохастическим потоки задач [7] и т. д. По-видимому, в этих целях будет необходимо использовать более сложные модели ВС и вычислительных процессов. Но решение указанных вопросов создаст предпосылки к совершенствованию алгоритмов оперативного управления ВС, повышая устойчивость вычислительных процессов на борту мобильных объектов.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Поляков А. Ю., Данекина А. А. Оптимизация времени создания и объема контрольных точек восстановления параллельных программ. Вестник СибГУТИ. 2010. № 2. С. 87–100.

2. Бондаренко А. А., Якобовский М. В. Обеспечение отказоустойчивости высокопроизводительных вычислений с помощью локальных контрольных точек. Вестник ЮУрГУ. Серия «Вычислительная математика и информатика». 2014. Т. 3. № 3. С. 20–36.

3. Басыров А. Г., Зыкова С. С., Кошель И. Н. и др. Метод отказоустойчивой параллельной обработки информации в бортовых вычислительных системах летательных аппаратов на основе временной избыточности вычислительного процесса. Авиакосмическое приборостроение. 2023. № 6. С. 33–39.

4. Зыкова С. С. Модель и алгоритм планирования параллельной обработки информации в отказоустойчивой бортовой вычислительной системе на основе временной избыточности вычислительного процесса. Интеллектуальные технологии на транспорте. 2023. № 4. С. 28–34.

5. Koren I., Mani Krishna C. Fault-Tolerant Systems. USA, Morgan Kaufmann Publishers, 2020. 416 p.

6. Rathore N. Checkpointing: Fault Tolerance Mechanism. i-manager's Journal on Cloud Computing. January-June, 2017. Vol. 4. No. 1. P. 28–35. [Электронный ресурс]. URL: https://www.researchgate.net/publication/322760706 (дата обращения: 15.05.2024).

7. Захаров И. В., Терехов В. Г., Соколовский А. Н. и др. Реконфигурация бортового комплекса подвижного объекта на основе моделирования вариантов его структурной деградации. Вестник Российского нового университета. Серия «Сложные системы: модели, анализ и управление». 2022. № 4. С. 39–46.

Дата поступления: 20.05.2024
Решение о публикации: 17.06.2024

# Optimization of Parallel Information Processing in a Fault-tolerant Computing System of a Mobile Object with Temporal Redundancy of the Computing Process

**Ivan V. Zakharov** — Doctor of Technical Sciences, Associate Professor, Professor of the Department of Mathematical and Software Engineering. PIN code: 7246-7546. E-mail: x.vano-z80@yandex.ru

**A. Mousallam** — Postgraduate Student at the Department of Mathematics and Software Engineering. E-mail: alloushi1987@gmail.com

A. F. Mozhaisky Military Space Academy, Saint Petersburg, Russia

**For citation:** Zakharov I. V., Mousallam A. Optimization of parallel information processing in a fault-tolerant computing system of a mobile object with temporal redundancy of the computing process // Intelligent technologies on transport. 2024. No. 2 (38). P. 30–36. (In Russian). DOI: 10.20295/2413-2527-2024-238-30-36

**Abstract.** *The tasks of effective use of multiprocessor computing systems of mobile objects require rational organization of computing processes on board. The article discusses a parallel computing process with temporal redundancy, which provides for periodic saving of the current state of calculations with the possibility of their restoration in order to ensure fault tolerance. A previously known solution to a specific optimization problem of choosing the frequency of saving the current state of a computing process under conditions of possible failures is briefly analyzed. A new improved method for solving this problem is proposed. Comparative results of its solution in various ways are presented and conclusions are drawn about the advisability of using the proposed approaches.*

**Keywords:** *multiprocessor computing system, parallel computing, checkpoints, computational fault tolerance.*

**REFERENCES**

1. Polyakov A. Yu., Danekina A. A. Optimizaciya vremeni sozdaniya i ob"ema kontrol'nyh tochek vosstanovleniya parallel'nyh programm. Vestnik SibGUTI. 2010. № 2. S. 87–100. (In Russian)

2. Bondarenko A. A., Yakobovskij M. V. Obespechenie otkazoustojchivosti vysokoproizvoditel'nyh vychislenij s pomoshch'yu lokal'nyh kontrol'nyh tochek. Vestnik YUUrGU. Seriya “Vychislitel'naya matematika i informatika”. 2014. T. 3. № 3. S. 20–36. (In Russian)

3. Basyrov A. G., Zykova S. S., Koshel' I. N. i dr. Metod otkazoustojchivoj parallel'noj obrabotki informacii v bortovyh vychislitel'nyh sistemah letatel'nyh apparatov na osnove vremennoj izbytochnosti vychislitel'nogo processa. Aviakosmicheskoe priborostroenie. 2023. № 6. S. 33–39. (In Russian)

4. Zykova S. S. Model' i algoritm planirovaniya parallel'noj obrabotki informacii v otkazoustojchivoj bortovoj vychislitel'noj sisteme na osnove vremennoj izbytochnosti vychislitel'nogo processa. Intellektual'nye tekhnologii na transporte. 2023. № 4. S. 28–34. (In Russian)

5. Koren I., Mani Krishna C. Fault-Tolerant Systems. USA, Morgan Kaufmann Publishers, 2020. 416 p.

6. Rathore N. Checkpointing: Fault Tolerance Mechanism. i-manager’s Journal on Cloud Computing. January-June, 2017. Vol. 4. No. 1. P. 28–35. [Электронный ресурс]. URL: https://www.researchgate.net/publication/322760706 (дата обращения: 15.05.2024).

7. Zaharov I. V., Terekhov V. G., Sokolovskij A. N. i dr. Rekonfiguraciya bortovogo kompleksa podvizhnogo ob"ekta na osnove modelirovaniya variantov ego strukturnoj degradacii. Vestnik Rossijskogo novogo universiteta. Seriya “Slozhnye sistemy: modeli, analiz i upravlenie”. 2022. № 4. S. 39–46. (In Russian)

Received: 28.05.2024
Accepted: 17.06.2024

УДК 004.41

# Использование продвинутых функций Git при разработке программного обеспечения

**Хомоненко Анатолий Дмитриевич**[1,2] — доктор технических наук, профессор, профессор кафедр «Информационные и вычислительные системы» и «Математическое и программное обеспечение». E-mail: khomon@mail.ru

**Каратаев Евгений Николаевич**[1] — магистрант 2-го курса направления 09.04.02 «Информационные системы и технологии». E-mail: zhenya-karat@yandex.ru

[1] Петербургский государственный университет путей сообщения Императора Александра I, Россия, Санкт-Петербург

[2] Военно-космическая академия имени А. Ф. Можайского, Россия, Санкт-Петербург

**Для цитирования:** Хомоненко А. Д., Каратаев Е. Н. Использование продвинутых функций Git при разработке программного обеспечения // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 37–48. DOI: 10.20295/2413-2527-2024-238-37-48

***Аннотация.*** *Рассматриваются ключевые аспекты использования продвинутых функций системы управления версиями Git при разработке программного обеспечения: модели ветвления, настройка git-конфигурации и методы исправления ошибок, полезные для разработчиков всех уровней.* ***Цель исследования:*** *демонстрация возможностей Git для повышения эффективности и гибкости разработки программного обеспечения. Рассмотрены вопросы интеграции продвинутых функций Git в процесс разработки программного обеспечения, представляющие интерес для профессионалов в области IT.* ***Практическая значимость:*** *обусловлена тем, что рассмотрены полезные параметры конфигурации, способствующие максимизации производительности и удобства использования Git. Приведены примеры команд и настроек системы Git, повышающие доступность и эффективность ее применения на практике.* ***Обсуждение:*** *через анализ технических и практических сторон показаны преимущества продвинутых функций Git и их вклад в улучшение процессов разработки. Показано, что использование продвинутых функций Git позволяет оптимизировать процессы разработки, повысить эффективность командных взаимодействий и обеспечить высокое качество кода.*

**Ключевые слова:** *Git, разработка программного обеспечения, управление версиями, Git-конфигурация, Git Flow, Trunk-Based Development.*

## Введение

В современной разработке программного обеспечения системы управления версиями играют критическую роль, обеспечивая координацию работы команды, сохранность кода и возможность эффективного возвращения к предыдущим версиям продукта. Среди таких систем Git зарекомендовал себя как один из самых мощных и гибких инструментов в индустрии.

Цель настоящей статьи — исследовать и продемонстрировать, как в современном процессе разработки программного обеспечения применяются продвинутые возможности Git, которые могут быть использованы для повышения эффективности и гибкости в процессах разработки программного обеспечения.

Особое внимание уделяется различным моделям ветвления, в частности, Git Flow и Trunk-Based Development, которые предлагают структурированные подходы к организации работы с кодом,

способствуя повышению продуктивности и упрощению сотрудничества в командах разработчиков.

В дополнение в статье освещена значимость настройки Git-конфигурации и методологии, применяемые в этом процессе. Акцент сделан на том, как адекватно настроенные области видимости и конфигурационные файлы могут способствовать повышению эффективности использования системы. В заключительной части статьи рассмотрены полезные параметры конфигурации, способствующие максимизации производительности и удобства использования Git. Статья нацелена на разработчиков, стремящихся углубить свои знания и умения в работе с Git, раскрывая его потенциал в полной мере.

В статье показывается, как продвинутые функции Git интегрируется в современную разработку программного обеспечения, подчеркивая их роль в оптимизации процессов и повышении эффективности командных взаимодействий.

## Обзор основных концепций Git

Git, разработанный Линусом Торвальдсом, является одной из наиболее широко используемых **систем контроля версий** в мире программирования. Эта система позволяет разработчикам отслеживать и управлять изменениями в коде, обеспечивая эффективную координацию работы в командах и сохранение истории проекта. В этом разделе подробно рассмотрены ключевые концепции Git, понимание которых необходимо для освоения продвинутых функций, описанных в последующих разделах [2].

*Репозиторий* — это основной элемент системы Git, представляющий собой центральное хранилище кода и его истории. Репозиторий содержит все версии файлов и информацию о их изменениях. В Git существуют локальные и удаленные репозитории, что позволяет разработчикам работать с кодом независимо, а затем синхронизировать изменения.

*Коммиты* — это «снимки» состояния репозитория в определенный момент времени. Каждый коммит содержит информацию об изменениях, авторе, дате и имеет уникальный идентификатор (hash). Через коммиты можно отслеживать историю изменений, возвращаться к предыдущим версиям и восстанавливать утерянные данные.

*Ветвление и слияние*. Ветвление позволяет разработчикам работать над разными задачами параллельно, не влияя на основной код проекта (master branch). Каждая ветка представляет собой отдельную линию разработки. Слияние (*merge*) — это процесс интеграции изменений из одной ветки в другую, позволяющий объединить разработки из разных веток.

*Удаленные репозитории* используются для обмена данными между участниками команды. Они служат централизованным хранилищем кода, к которому могут обращаться разработчики. Наиболее распространенным примером удаленного репозитория является GitHub. Команды `git push` и `git pull` позволяют синхронизировать локальные изменения с удаленным репозиторием.

*Индексация и состояние файлов*. Git отслеживает состояние файлов в репозитории, которые могут быть в трех состояниях: зафиксированные (*committed*), измененные (*modified*) и подготовленные к коммиту (*staged*). Индексация (*staging*) — это процесс добавления измененных файлов в промежуточное состояние перед фиксацией изменений коммитом.

*Конфликты слияния* возникают, когда изменения в одной ветке противоречат изменениям в другой. Git не может автоматически решить, какие изменения следует сохранить, поэтому требуется вмешательство разработчика для их разрешения [1–2].

## Модели ветвления в Git

Модели ветвления в Git представляют собой структурированные подходы к управлению ветками, которые помогают координировать работу разработчиков, упорядочивать процесс интеграции изменений и поддерживать высокий уровень качества продукта. Без четко определенной модели ветвления команды разработчиков могут столкнуться с проблемами в управлении зависимостями, разрешении конфликтов и отслеживании изменений, что в конечном итоге может привести к снижению продуктивности и увеличению вероятности ошибок.

Существует несколько моделей ветвления, каждая из которых предлагает различные стратегии для управления изменениями и интеграции кода. Выбор конкретной модели зависит от множества факторов, включая размер команды, сложность проекта, частоту релизов и предпочтения команды. Примеры наиболее популярных моделей ветвления:

1. Git Flow: популярная модель, которая вводит строгую структуру ветвления, включая отдельные ветки для разработки, релизов, хотфиксов и функциональностей.

2. Trunk-Based Development: модель, акцентирующая внимание на короткоживущих ветках и частом слиянии с главной веткой (trunk), что способствует быстрой интеграции изменений и минимизации конфликтов.

3. Feature Branch Workflow: подход, при котором для каждой новой функции создается отдельная ветка, что позволяет изолированно разрабатывать и тестировать нововведения, прежде чем интегрировать их в основной код [4].

В статье рассматриваются две основные модели ветвления: Git Flow и Trunk-Based Development, так как Feature Branch Workflow схожа с моделью Trunk-Based Development.

- **Git Flow**

Git Flow выделяется как одна из наиболее структурированных и популярных моделей ветвления, разработанная и популяризированная Винсентом Дриссеном в 2010 году. Эта модель представляет собой точно определенный подход к ветвлению и слиянию, который способствует организации рабочего процесса разработки, особенно в командах среднего и большого размера.

***Описание модели Git Flow***

Основой Git Flow является деление рабочего процесса на несколько ключевых веток, каждая из которых служит определенной цели и имеет строгое назначение (рис. 1):

1. Основная ветка (master). В этой ветке содержится стабильная версия кода, готовая к выпуску. Изменения в эту ветку попадают только из ветки release или hotfix.

2. Ветка разработки (develop). Основная ветка для разработки, в которой происходит слияний изменений из различных веток функциональностей. После достижения определенного этапа развития проекта изменения из ветки develop могут быть перенесены в ветку release для последующего выпуска версии.

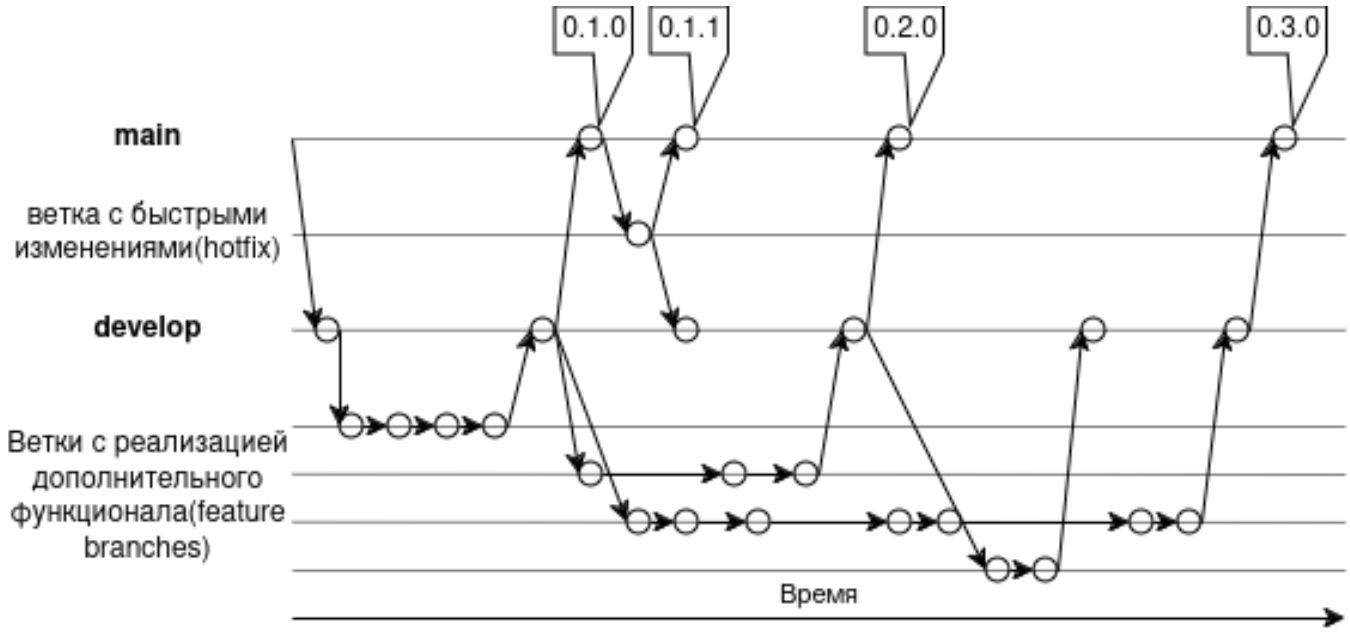



*Рис. 1.* Модель ветвления Git Flow

3. Ветки функциональностей (feature branches): Отдельные ветки, создаваемые для разработки новых функций или компонентов. После завершения работы функциональность интегрируется обратно в ветку develop.

4. Ветки выпуска (release branches). Эти ветки используются для подготовки новых версий продукта к выпуску. В них могут производиться последние исправления, документация и другие задачи, связанные с выпуском. После завершения изменения мерджатся как в master, так и в develop.

5. Ветки исправлений (hotfix branches). В случае обнаружения срочных ошибок в стабильной версии продукта создаются ветки hotfix, в которых происходит работа над исправлениями. После завершения исправлений, изменения интегрируются в master и develop, обеспечивая актуализацию кодовой базы [6].

***Достоинства модели Git Flow***

1. Параллельная разработка: возможность одновременной работы над различными функциями в отдельных ветках без влияния на основной код. Это ускоряет процесс разработки и позволяет разработчикам работать независимо.

2. Улучшенное тестирование: отделение новых разработок в отдельные ветки позволяет проводить тестирование изолированно от основного кода, что повышает качество тестирования и уменьшает вероятность ошибок в рабочей среде (production environment).

***Недостатки модели Git Flow***

1. Сложность.

Git Flow имеет относительно сложную структуру с множеством типов веток (feature, release, develop, hotfix, master), что может вызывать путаницу, особенно у новичков или в маленьких проектах, где такая структура может быть излишней.

2. Замедление разработки.

Постоянное переключение между ветками и необходимость соблюдения определенных правил могут замедлить процесс разработки, особенно в быстро меняющихся или маленьких проектах.

3. Избыточность.

Для маленьких или менее сложных проектов структура Git Flow может быть излишне сложной и тяжеловесной, что приводит к неоправданному увеличению работы с проектом [7].

***Характеристика работы с моделью***

*Основные ветки*. ***Master*** является основной веткой, в которой содержится стабильная версия кода, готовая к выпуску. Эта ветка служит конечной точкой для всех изменений, проходящих через процесс разработки и тестирования. ***Develop***, в свою очередь, представляет собой ветку для разработки, где собираются и интегрируются все изменения из вспомогательных веток, предварительно подготавливая их к переносу в основную ветку master.

*Вспомогательные ветки* (***feature branches***) создаются для разработки новых функциональностей. Они порождаются от ветки develop и после завершения работы над новой функцией сливаются обратно в нее. ***Release branches*** отвечают за подготовку новой версии продукта к выпуску. Они начинаются от ветки develop и позволяют вносить последние изменения перед слиянием в master и develop. ***Hotfix branches*** предназначены для оперативного устранения неполадок в уже выпущенных версиях. Создаются непосредственно от master и после внесения необходимых исправлений сливаются обратно в master и develop.

Процесс работы с Git Flow начинается с инициализации репозитория, что задает структуру веток. Далее в ходе разработки создаются ветки для новых функций (***feature***) от ***develop***, в которых и ведется вся работа. При подготовке к выпуску готовые и протестированные функции сливаются в ветку release, где происходит их окончательная доработка. После успешного тестирования и утверждения изменений ветка release интегрируется в master и помечается тегом с номером версии. В случае обнаружения срочных ошибок в произведенном релизе, применяются горячие исправления (***hotfix***), которые после устранения сливаются в master и develop, обеспечивая актуальность и стабильность кода [7].

- **Trunk-Based Development**

Trunk-Based Development (TBD) представляет собой подход к разработке программного обеспечения, который подчеркивает важность коротких циклов ветвления и слияния. Он ориентирован на упрощение процесса разработки путем минимизации продолжительности и сложности ветвлений, тем самым ускоряя слияние изменений и улучшая совместимость кода.

***Описание TBD***

Trunk-Based Development является методологией разработки программного обеспечения, при которой разработчики интегрируют свои изменения в одну центральную ветку в системе контроля версий, известную как trunk или main (рис. 2).

Основные принципы TBD включают следующие аспекты:

1. Централизация в одной ветке.

Все изменения напрямую интегрируются в главную ветку. Это снижает сложность управления множеством параллельных веток и уменьшает риск конфликтов при слиянии.

2. Частые коммиты.

Разработчики совершают коммиты в главную ветку как можно чаще, предпочтительно несколько раз в день. Это обеспечивает актуализацию кода у всех участников проекта и позволяет быстро выявлять и исправлять ошибки.

3. Минимальное время жизни веток.

Если создаются дополнительные ветки для специфических задач, их существование должно быть максимально коротким — от нескольких часов до пары дней. Эти ветки быстро интегрируются обратно в главную ветку, что поддерживает непрерывность и скорость разработки.

4. Контроль качества.

Несмотря на частоту коммитов, контроль качества не уступает по строгости. Автоматизированные тесты и проверки кода на предмет соответствия стандартам проводятся непрерывно, что гарантирует высокий уровень качества выпускаемого продукта.

5. Feature flags.

Для управления функционалом, который еще не готов к полноценному запуску, используются feature flags (флаги поведения или флаги функций). Это позволяет интегрировать код новых функций непосредственно в главную ветку, не ожидая их полного завершения, и активировать их в нужное время [8].

***Достоинства и недостатки модели TBD***

Ключевым достоинством модели Trunk-Based Development является повышение скорости внедрения новых возможностей и исправлений. Отказ от сложной ветвистой структуры позволяет команде быстрее реагировать на изменения требований, ускорять процесс непрерывной интеграции и развертывания.

К *достоинствам* этой модели ветвления можно отнести:

1. Ускоренный процесс разработки.

TBD позволяет разработчикам интегрировать изменения непосредственно в главную ветку, что минимизирует необходимость управления множеством параллельных веток. Это сокращает время для реализации нового функционала и уменьшает время выхода его в рабочую среду (production environment). Также уменьшение количества активных веток упрощает управление проектом.

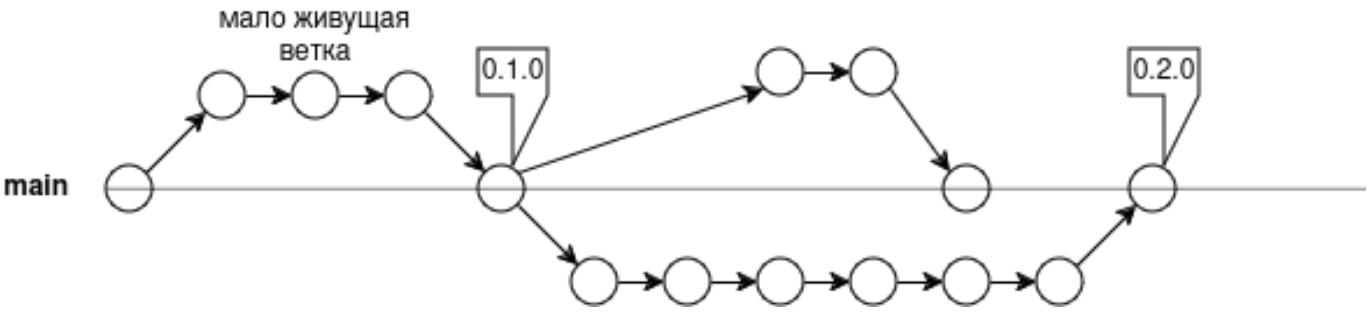



***Рис. 2.*** Модель ветвления Trunk-Based Development

С малым количеством веток навигация и ориентирование в проекте становятся более интуитивными, особенно для новых участников команды.

2. Меньше конфликтов слияния.

Поскольку изменения вносятся часто и в малых объемах, вероятность возникновения сложных конфликтов слияния существенно уменьшается. Это способствует более простому процессу разработки и сокращает время, потраченное на разрешение конфликтов [9–11].

В то же время модель Trunk-Based Development имеет и некоторые *недостатки:*

1. Требования к дисциплине.

TBD требует от разработчиков высокой степени ответственности за вносимые изменения, поскольку любые ошибки сразу влияют на главную ветку. Это может увеличить количество проблем в рабочей ветке. Для эффективной работы с TBD необходима строгая система автоматизированного тестирования. Без надлежащего тестового покрытия, частое слияние изменений может привести к внедрению ошибок и нестабильности основного продукта.

2. Возможные проблемы с масштабированием.

В больших командах, где множество разработчиков одновременно работают над различными аспектами проекта, TBD может вызвать управленческие и технические трудности.

***Процесс работы с моделью можно описать следующим образом:***

1. Разработчики создают локальные ветки для реализации новых функций или исправления ошибок.

2. После завершения работы над изменениями, они производят слияние (***merge***) своей ветки в ***master***.

3. Автоматизированные системы сборки и тестирования проверяют интегрированные изменения на соответствие требованиям качества.

4. При успешном прохождении всех проверок, изменения считаются готовыми к релизу.

5. Новая версия продукта создается из среза ***master*** ветки [8].

**Эффективность моделей ветвления**

Внедрение моделей ветвления, таких как Git Flow и Trunk-Based Development, может повысить эффективность процессов разработки программного обеспечения. Структурированный подход к управлению ветками улучшают качества кода, ускоряет разработку новых версий продукта и упрощает взаимодействие в команде разработчиков.

Чтобы продемонстрировать преимущества использования моделей ветвления, проведен эксперимент в среднем проекте с командой из 10 разработчиков. В течение 6 месяцев команда не использовала модель ветвления. В следующие 6 месяцев, была внедрена модель Trunk-Based Development с коротким циклом ветвления.

В табл. 1 демонстрируются изменения в производительности и качестве работы команды до и после внедрения передовых практик ветвления.

Результаты эксперимента демонстрируют, что внедрение современных моделей ветвления, таких как Trunk-Based Development, повышает эффективность процессов разработки программного обеспечения. Структурированный подход к управлению ветками способствует ускорению выпуска новых версий продукта, повышению качества кода и улучшению сотрудничества в командах разработчиков.

*Таблица 1*

**Результаты эксперимента**

| Параметр оценки | До введения | После введения | Изменение (%) |
|---|---|---|---|
| Время на разработку (дни) | 45 | 38 | -15.6% |
| Обнаруженные ошибки (шт.) | 65 | 42 | -35.4% |
| Время на исправление ошибок (часы) | 120 | 84 | -30.0% |
| Уровень сотрудничества (по шкале от 1 до 10) | 5 | 8 | +60.0% |

**Исправление действий в Git**

В процессе разработки программного обеспечения с использованием распределенной системы управления версиями Git неизбежно возникают ситуации, когда разработчики совершают ошибочные действия. Эти ошибки могут негативно повлиять на процесс разработки и привести к нежелательным последствиям. Поэтому в статье рассмотрены наиболее распространенные ошибки, возникающих в процессе использования Git, и методов их решения.

**Исправление ошибок в коммите**

Еще одной из наиболее распространенных проблем, с которой сталкиваются пользователи Git, являются неправильные коммиты. Для решения данной задачи Git предлагает использовать команду `git commit --amend`, позволяющую вносить изменения в последний коммит без модификации его уникального хеша. Данная функциональность обеспечивает возможность быстрого исправления опечаток, неточностей в сообщении коммита, а также добавления упущенных файлов. При выполнении команды `git commit --amendd` автоматически открывается текстовый редактор, в котором разработчик может отредактировать содержимое последнего коммита, сохраняя при этом линейность истории репозитория [2, 12, 13].

Кроме того, в Git существует команда `git reset HEAD~1 --soft`, которая также может быть полезна при исправлении ошибок другого типа. Данная команда позволяет вернуть изменения в рабочую область, не затрагивая при этом историю коммитов. Это дает возможность разработчикам удалить нежелательные файлы или изменения, которые случайно попали в коммит, а затем создать новый, исправленный коммит [2, 12].

**Исправление ошибок в ветке**

Одним из распространенных сценариев является случайное внесение изменений в ветку master, вместо создания новой ветки для разработки функционала. В подобных ситуациях можно воспользоваться командой `git reset`, позволяющей откатить последние коммиты. Для решение данной проблемы нужно выполнить последовательно данные команды:

1. `Git branch future-brunch`. Создает новую ветку с названием future-brunch на основе текущей активной ветки, но без переключения на нее.

2. `Git reset HEAD~ --hard`. Сбрасывает состояние рабочего каталога и зону отслеживаемых изменений (staging area) к состоянию предпоследнего коммита в активной ветке.

3. `Git checkout future-brunch` переключает активную ветку на future-brunch [2, 12, 14].

Соответственно, все изменения перешли в ветку `future-brunch`, а в ветке `master` этих изменений нет. Также, если нужно откатить состояние на определенное количество коммитов назад, то перед `HEAD~` поставить число, указывающее, на сколько коммитов нужно вернуться. Например, для отката на три коммита назад используйте команду:

4. `Git reset HEAD~3 --hard`.

Другой распространенной ошибкой является некорректное название создаваемой ветки. В таком случае можно воспользоваться командой `git branch -m`, позволяющей переименовать существующую ветку. Например, команда `git branch -m future-brunch feature-branch` переименует ветку `future-brunch` в `feature-branch`. Если ветка уже была отправлена в удаленный репозиторий, потребуется также удалить старое название ветки с помощью `git push origin --delete future-brunch` и затем отправить переименованную ветку `git push origin feature-branch` [2].

**Отмена действий в Git-репозитории**

Когда после действий в Git-репозитории состояние стало еще хуже, чем было в начале. Для восстановления состояния используются такие инструменты, такие как `git reflog` и `git reset`.

`Git reflog` позволяет просмотреть журнал всех выполненных команд Git, включая перемещения между ветками, изменения указателя HEAD и другие операции. Используя эту информацию, разработчик может вернуться к любому предыдущему состоянию репозитория [2].

Используя эту информацию, разработчик может вернуться к любому предыдущему состоянию репозитория, воспользовавшись командой `git reset HEAD@{index}`, где `index` — это соответствующий индекс из вывода `git reflog` [2, 15].

Например, если в результате неверных действий в репозитории возникла нежелательная ситуация, разработчик может выполнить `git reflog`, чтобы увидеть историю изменений, и затем восстановить предыдущее, корректное состояние с помощью `git reset HEAD@{index}`. Данный подход позволяет эффективно «путешествовать во времени» и возвращаться к любым ранее существовавшим версиям проекта.

Следует отметить, что манипуляции с историей коммитов, такие как git reset, следует производить с особой осторожностью, поскольку они могут привести к потере данных, особенно если изменения уже были опубликованы в удаленном репозитории. В таких случаях рекомендуется использовать более безопасные методы, например, git revert, которые создают новый коммит, отменяющий предыдущие изменения [2].

**Конфигурация Git**

Несмотря на то что при первом знакомстве с системой контроля версий Git большинство пользователей ограничиваются только минимальной настройкой, необходимой для начала работы, углубленное понимание процесса конфигурации Git позволяет эффективно использовать данный инструмент в современной разработке.

Конфигурация Git представляет собой многогранный процесс, затрагивающий различные аспекты использования данной системы. От определения идентификационных данных пользователя до настройки стилей коммитов и интеграции со сторонними инструментами каждый из этих элементов вносит вклад в повышение удобства и эффективности работы разработчика.

Более того, возможность сохранения конфигурации Git в виде файлов способствует повышению переносимости среды разработки. Разработчики могут быстро восстановить свое рабочее окружение на новом компьютере или сервере, минимизируя затраты времени на повторную настройку.

**Уровни конфигурации**

Конфигурации Git могут быть заданы на трех уровнях: системном (system), глобальном (global) и локальном (local). Каждый уровень конфигурации имеет свою область действия и приоритет [1].

Системный уровень конфигурации затрагивает всех пользователей и все репозитории на данной машине. Настройки этого уровня применяются в случае, если не заданы конфигурации на глобальном или локальном уровнях. Для создания системной конфигурации используется команда `git config --system`. Файл системной конфигурации обычно находится по пути /etc/gitconfig.

Глобальный уровень конфигурации относится к конкретному пользователю и применяется ко всем репозиториям этого пользователя, если не переопределен на локальном уровне. Глобальные настройки удобны для задания персональных предпочтений, таких как имя пользователя, email и редактор по умолчанию. Для установки глобальной конфигурации используется команда `git config --global`. Файл глобальной конфигурации располагается в домашней директории пользователя: ~/.gitconfig или ~/.config/git/config.

Локальный уровень конфигурации имеет наивысший приоритет и применяется только к конкретному репозиторию. Эти настройки хранятся в файле .git/config внутри репозитория. Локальная конфигурация позволяет задать специфичные для репозитория параметры, такие как удаленные репозитории (remotes) и настройки ребейза. Для создания локальной конфигурации используется команда `git config --local` или просто `git config` без флагов, находясь внутри репозитория [16].

Приоритет уровней конфигурации следующий: локальный > глобальный > системный. То есть, если определенный параметр задан на нескольких уровнях, будет использовано значение с наивысшим приоритетом.

**Файл конфигурации**

Файл конфигурации представляет собой основное хранилище параметров, определяющих поведение Git во время выполнения различных операций. Формат конфигурационного файла основан на синтаксисе INI, состоящем из секций, обозначенных заголовками в квадратных скобках, и ключ-значных пар внутри этих секций. Пример:

```
[user]
    name = John Doe
    email = johndoe@example.com
[core]
    editor = vim
```

**Типовые параметры конфигурации**

Некоторые типовые параметры, которые можно установить в файле конфигурации Git [5]:

1. User.name и user.email

Эти параметры задают имя и email пользователя, которые будут использоваться при создании коммитов. Пример:

```
[user]
    name = John Doe
    email = johndoe@example.com
```

2. Core.Editor

Определяет текстовый редактор, который будет использоваться для редактирования сообщений коммитов и других текстовых данных. Пример:

```
[core]
    editor = vim
```

3. Credential.helper

Указывает способ хранения учетных данных для аутентификации при работе с удаленными репозиториями. `cache` указывает, что учетные данные сохраняются в памяти на определенный период времени (по умолчанию 15 минут). `store` указывает, что учетные данные сохраняются в файле на диске в открытом виде. `manager` указывает, что используется встроенный в операционную систему менеджер учетных данных. Также можно указать путь к внешней программе или скрипту, который будет отвечать за хранение и предоставление учетных данных. Пример:

```
[credential]
    helper = cache --timeout =3600
```

4. Core.autocrlf

Управляет автоматическим преобразованием символов окончания строк при чтении и записи файлов. При параметре `true` Git будет автоматически преобразовывать окончания строк из формата LF (Linux, macOS) в формат CRLF (Windows) при чтении файлов из репозитория и обратно при записи файлов в репозиторий. Это удобно для разработчиков, работающих в Windows. При параметре `false`: Git не будет выполнять никаких преобразований окончаний строк. Разработчики должны самостоятельно следить за согласованностью окончаний строк в файлах. При параметре `input`: Git будет преобразовывать окончания строк из формата CRLF в формат LF при записи файлов в репозиторий, но не будет выполнять преобразования при чтении файлов. Это удобно для разработчиков, работающих в Linux или macOS и сотрудничающих с разработчиками, использующими Windows. Пример:

```
[core]
    autocrlf = true
```

5. Core.whitespace

Настраивает обработку пробельных символов в файлах. Может включать опции для предупреждения или автоматического исправления нежелательных пробельных символов. Пример:

```
[core]
    whitespace = fix,-indent-with-
non-tab,trailing-space,cr-at-eol
```

Отметим, что локальные параметры переопределяют глобальные для конкретного репозитория. Многие GUI-клиенты Git также предоставляют интерфейсы для настройки этих параметров.

**Заключение**

Использование таких возможностей, как модели ветвления, исправление ошибок и настройка конфигурации, может значительно повысить эффективность и гибкость процессов разработки. Анализ моделей ветвления, таких как Git Flow и Trunk-Based Development, продемонстрировал их роль в структурировании работы с кодом и улучшении командного взаимодействия. Применение этих подходов позволяет оптимизировать процессы разработки и упростить сотрудничество между членами команды.

Правильное использование методов исправления ошибок в коммитах, ветках и отмены действий в git-репозитории помогает поддерживать целостность кода и обеспечивает возможность эффективного возврата к предыдущим версиям продукта. Адекватно настроенные области видимости и конфигурационные файлы способствуют повышению эффективности использования системы. Выделены полезные параметры конфигурации, которые могут значительно улучшить производительность и удобство работы с Git. Показано, что использование продвинутых функций Git играет важную роль в современной разработке программного обеспечения. Применение этих возможностей позволяет оптимизировать процессы разработки, повысить эффективность командных взаимодействий и обеспечить высокое качество кода.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Фишерман Л. В. Git. Практическое руководство. Управление и контроль версий в разработке программного обеспечения // Наука и техника, 2022. 304 с.

2. Chacon S., Straub B. Pro Git. Apress, 2014. Версия 2.1.104-2-g74b0d66, 06.09.2022. 538 с.

3. Shakikhanli U., Bilicki V. Optimizing branching strategies in mono and multi-repository environments: a comprehensive analysis // Computer Assisted Methods in Engineering and Science. 2024.

4. Гаспарян А. В., Тимошина Н. В. Совместная разработка по с использованием Git // ИТ-портал. 2017. № 1 (13) [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/sovmestnaya-razrabotka-po-s-ispolzovaniem-git (дата обращения: 26.04.2024).

5. Вьюшкова М. В., Чернова С. В. Принцип работы системы контроля версий Git // Теория и практика современной науки. 2019. № 10 (52) [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/printsip-raboty-sistemy-kontrolya-versiy-git (дата обращения: 26.04.2024).

6. Kummer D. Шпаргалка по git-flow. 2016 [Электронный ресурс]. URL: http://danielkummer.github.io/git-flow-cheatsheet/index.ru_RU.html (дата обращения: 26.04.2024).

7. Driessen V. A Successful git branching model. 2010 [Электронный ресурс]. URL: https://nvie.com/posts/a-successful-git-branching-model/ (дата обращения: 26.04.2024).

8. Trunk Based Development [Электронный ресурс]. URL: https://trunkbaseddevelopment.com/ (дата обращения: 26.04.2024).

9. Климентьев А. Пожалуйста, перестаньте рекомендовать Git Flow. 2020 [Электронный ресурс]. URL: https://habr.com/ru/companies/flant/articles/491320/ (дата обращения: 26.04.2024).

10. Александров А. Почему Trunk Based Development — лучшая модель ветвления. 2020 [Электронный ресурс]. URL: https://habr.com/ru/articles/519314/ (дата обращения: 26.04.2024).

11. Решетников С. Мой опыт перевода команд разработки на trunk-based development, 2024 [Электронный ресурс]. URL: https://habr.com/ru/articles/794246/ (дата обращения: 26.04.2024).

12. Beckham S. Git happens! 6 типичных ошибок Git и как их исправить. 2018 [Электронный ресурс]. URL: https://habr.com/ru/companies/flant/articles/419733/ (дата обращения: 26.04.2024).

13. Hexlet. Изменение последнего коммита [Электронный ресурс]. URL: https://ru.hexlet.io/courses/intro_to_git/lessons/git-commit-amend/theory_unit (дата обращения: 26.04.2024).

14. git scm. Git-git-branch Documentation. 2024 [Электронный ресурс]. URL: https://git-scm.com/docs/git-branch (дата обращения: 26.04.2024).

15. git scm. Git-git-reset Documentation. 2024 [Электронный ресурс]. URL: https://git-scm.com/docs/git-reset (дата обращения: 26.04.2024).

16. git scm. Git-git-config Documentation. 2024 [Электронный ресурс]. URL: https://git-scm.com/docs/git-config (дата обращения: 26.04.2024).

Дата поступления: 13.05.2024
Решение о публикации: 27.05.2024

# Using Advanced Git Features in Software Development

**Anatoly D. Khomonenko**[1,2] — Doctor of Technical Sciences, Professor, Professor of the departments "Information and Computing Systems" and "Mathematical and Software". E-mail: khomon@mail.ru

**Evgenij N. Karataev**[1] — 2nd year undergraduate student of the direction 09.04.02 "Information systems and technologies". E-mail: zhenya-karat@yandex.ru

[1] Emperor Alexander I Petersburg State Transport University, Saint Petersburg, Russia
[2] A. F. Mozhaisky Military Space Academy, Saint Petersburg, Russia

**For citation:** Khomonenko A. D., Karataev E. N. Using advanced git features in software development // Intelligent technologies on transport. 2024. No. 2 (38). P. 37–48. (In Russian). DOI: 10.20295/2413-2527-2024-238-37-48

***Abstract.*** *The key aspects of using the advanced features of the Git version control system in software development are considered: branching models, Git configuration settings and error correction methods useful for developers at all levels. The purpose of the study is to demonstrate the capabilities of Git to increase the efficiency and flexibility of software development. The issues of integrating advanced Git functions into the software development process, which are of interest to IT professionals, are considered. Practical significance: due to the fact that useful configuration parameters are considered that help maximize the performance and usability of Git. Examples of commands and settings of the Git system are given, which increase the accessibility and effectiveness of its application in practice. Discussion: through the analysis of technical and practical aspects, the advantages of advanced Git functions and their contribution to improving development processes are shown. It is shown that the use of advanced Git functions makes it possible to optimize development processes, increase the efficiency of team interactions and ensure high code quality.*

***Keywords:*** *Git, software development, version control, Git configuration, Git Flow, Trunk-Based Development.*

**REFERENCES**

1. Fisherman L. V. Git. Prakticheskoe rukovodstvo. Upravlenie i kontrol' versij v razrabotke programmnogo obespechenija // Nauka i tehnika, 2022. 304 s. (In Russian)

2. Chacon S., Straub B. Pro Git. Apress, 2014. Versija 2.1.104-2-g74b0d66, 06.09.2022. 538 s.

3. Shakikhanli U., Bilicki V. Optimizing branching strategies in mono and multi-repository environments: a comprehensive analysis // Computer Assisted Methods in Engineering and Science. 2024.

4. Gasparjan A. V., Timoshina N. V. Sovmestnaja razrabotka po s ispol'zovaniem Git // IT-portal. 2017. № 1 (13) [Jelektronnyj resurs]. URL: https://cyberleninka.ru/article/n/sovmestnaya-razrabotka-po-s-ispolzovaniem-git (data obrashhenija: 26.04.2024). (In Russian)

5. V'jushkova M. V., Chernova S. V. Princip raboty sistemy kontrolja versij Git // Teorija i praktika sovremennoj nauki. 2019. № 10 (52) [Jelektronnyj resurs]. URL: https://cyberleninka.ru/article/n/printsip-raboty-sistemy-kontrolya-versiy-git (data obrashhenija: 26.04.2024). (In Russian)

6. Kummer D. Shpargalka po git-flow. 2016 [Jelektronnyj resurs]. URL: http://danielkummer.github.io/git-flow-cheatsheet/index.ru_RU.html (data obrashhenija: 26.04.2024). (In Russian)

7. Driessen V. A Successful git branching model. 2010 [Jelektronnyj resurs]. URL: https://nvie.com/posts/a-successful-git-branching-model/ (data obrashhenija: 26.04.2024).

8. Trunk Based Development [Jelektronnyj resurs]. URL: https://trunkbaseddevelopment.com/ (data obrashhenija:26.04.2024).

9. Kliment'ev A. Pozhalujsta, perestan'te rekomendovat' Git Flow. 2020 [Jelektronnyj resurs]. URL: https://habr.com/ru/companies/flant/articles/491320/ (data obrashhenija: 26.04.2024). (In Russian)

10. Aleksandrov A. Pochemu Trunk Based Development — luchshaja model' vetvlenija. 2020 [Jelektronnyj resurs]. URL: https://habr.com/ru/articles/519314/ (data obrashhenija: 26.04.2024). (In Russian)

11. Reshetnikov S. Moj opyt perevoda komand razrabotki na trunk-based development, 2024 [Jelektronnyj resurs]. URL: https://habr.com/ru/articles/794246/ (data obrashhenija: 26.04.2024). (In Russian)

12. Beckham S. Git happens! 6 tipichnyh oshibok Git i kak ih ispravit'. 2018 [Jelektronnyj resurs]. URL: https://habr.com/ru/companies/flant/articles/419733/ (data obrashhenija: 26.04.2024). (In Russian)

13. Hexlet. Izmenenie poslednego kommita [Jelektronnyj resurs]. URL: https://ru.hexlet.io/courses/intro_to_git/lessons/git-commit-amend/theory_unit (data obrashhenija: 26.04.2024). (In Russian)

14. git scm. Git-git-branch Documentation. 2024 [Jelektronnyj resurs]. URL: https://git-scm.com/docs/git-branch (data obrashhenija: 26.04.2024).

15. git scm. Git-git-reset Documentation. 2024 [Jelektronnyj resurs]. URL: https://git-scm.com/docs/git-reset (data obrashhenija: 26.04.2024).

16. git scm. Git-git-config Documentation. 2024 [Jelektronnyj resurs]. URL: https://git-scm.com/docs/git-config (data obrashhenija: 26.04.2024).

Received: 13.05.2024
Accepted: 27.05.2024

УДК 681.327.1/13

# Архитектурные подходы и технологические решения в создании инновационных веб-приложений для голосовых и текстовых чатов. Анализ, перспективы и реализация

**Васильков Сергей Константинович** — студент, бакалавр, кафедра «Информационные и вычислительные системы». E-mail: z2013zz@bk.ru

**Забродин Андрей Владимирович** — кандидат исторических наук, доцент, кафедра «Информационные и вычислительные системы». E-mail: zabrodin@pgups.ru

Петербургский государственный университет путей сообщения Императора Александра I, Россия, Санкт-Петербург

**Для цитирования:** Васильков С. К., Забродин А. В. Архитектурные подходы и технологические решения в создании инновационных веб-приложений для голосовых и текстовых чатов. Анализ, перспективы и реализация // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 49–56. DOI: 10.20295/2413-2527-2024-238-49-56

***Аннотация.*** *В статье на примере реализованного решения подробно рассматривается технологии, используемые в инновационном сервисе, реализующим голосовые и текстовые чаты в виде веб-приложения. При этом основное внимание уделяется анализу архитектурных и технологических подходов, используемых при разработке веб-приложений для голосовых и текстовых чатов. Достаточно подробно анализируются функциональные возможности и масштабируемость таких систем. Объектом для анализа служит сервис, с рабочим названием «Сервис с голосовым и текстовым чатами в контексте импортозамещения», имеющий рабочее название Ruscord, который несмотря на схожесть по функциональности и пользовательскому опыту с ведущими платформами общения, такими как Discord, Telegram, «ВКонтакте», реализован с учетом специфических требований и новейших технологий в области веб-разработки.*

*В статье описываются ключевые компоненты системы, включая серверную логику, клиентские приложения и инфраструктуру данных. Особое внимание уделено вопросам интеграции современных технологий в реальные продукты, включая использование облачных сервисов, микросервисной архитектуры и современных фреймвоков и протоколов.*

*В рамках данного исследования анализируются не только технические аспекты, но и вопросы удобства использования, доступности и включенности, что позволяет создавать более эффективные и удовлетворяющие потребности пользователей сервисы.*

**Ключевые слова:** *архитектурные подходы, технологические решения, веб-приложения, голосовые чаты, текстовые чаты, WebRTC, Socket, React, инновационные технологии, платформы общения.*

## Введение

В условиях бурного развития цифровых технологий и возрастающих требований к коммуникационным инструментам, разработка веб-приложений для голосовых и текстовых чатов становится предметом особого внимания. Эти сервисы, представляющие собой сложные многоуровневые системы. Они включают в себя множество компонентов, каждый из которых требует тщательного анализа и интеграции современных технологий [1].

Статья структурирована следующим образом: в начале рассматривается общая модель системы, описывающая и перечисляющая ее основные компоненты. Далее каждый компонент будет рассмотрен в отдельности с точки зрения его функциональности, технических требований и взаимодействия с другими элементами системы. Отдельная часть посвящена механизмам соединения этих компонентов для обеспечения надежной и эффективной работы всего приложения. Особое внимание уделено обсуждению пользовательского интерфейса как ключевого аспекта, определяющего удобство и функциональность конечного продукта.

Таким образом, статья предоставит читателю полное представление о том, как строится современное веб-приложение для голосовых и текстовых чатов, акцентируя внимание на наиболее значимых аспектах и решениях, которые могут быть использованы для создания эффективных и безопасных коммуникационных решений.

## Архитектура приложения

Архитектура компонентов реализованной системы представляет собой комплексный набор организационных принципов и структурных решений, ориентированных на обеспечение эффективного функционирования приложения. Эта архитектура включает в себя две ключевые составляющие: архитектуру компонентов сервера (рис. 1) и архитектуру компонентов клиента (рис. 2).

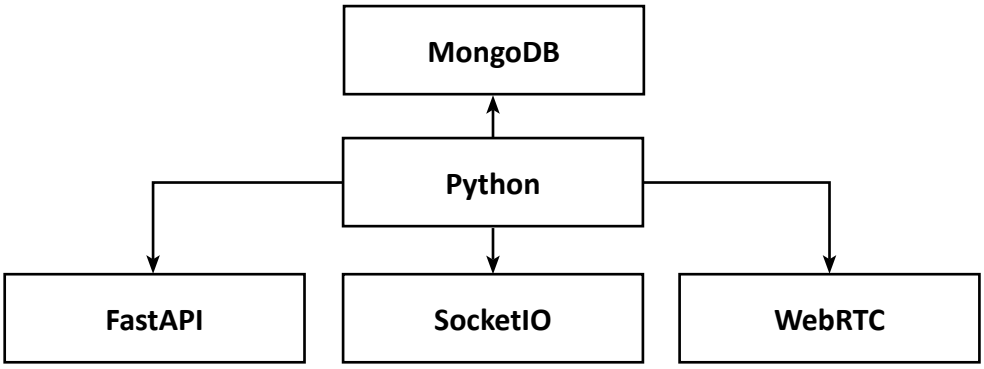



*Рис. 1.* Архитектура компонентов сервера

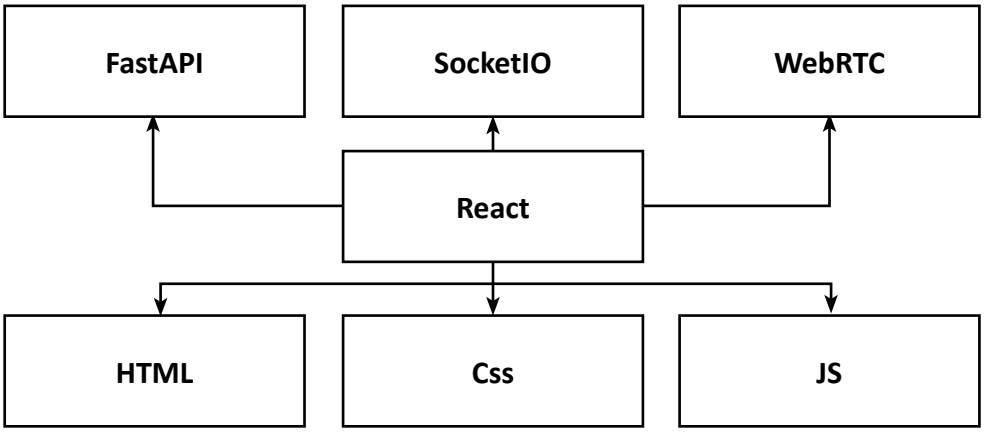



*Рис. 2.* Архитектура компонентов клиента

В представленной на рис. 1 и 2 архитектурах каждый из компонентов представляет собой определенную абстракцию, обеспечивающую конкретную функциональность и взаимодействие в рамках системы.

**Python.** Язык программирования, который служит основой для всего серверного взаимодействия и логики приложения, представляет собой ключевой элемент, определяющий функциональность и возможности системы в целом [2].

**FastAPI.** Современный, высокопроизводительный фреймворк для создания API с Python 3.7+, основанный на стандартных типах Python для валидации данных и аннотаций типов. FastAPI упрощает разработку и предоставляет автоматическую документацию в формате Swagger — инструмента, который позволяет автоматически создавать интерактивную документацию для вашего API.

**SocketIO.** Программная библиотека, позволяющая реализовывать веб-сокеты, то есть двусторонние и непрерывные каналы связи между клиентом и сервером. Этот механизм обеспечивает возможность серверу и клиентам обмениваться сообщениями в режиме реального времени.

**WebRTC.** Технология, позволяющая организовывать передачу потоковых данных (аудио-, видеофайлов) между браузерами без необходимости использования посредников [3]. В контексте реализованного сервиса WebRTC может быть использовано для реализации функционала видео- и аудиосвязи в приложении.

**React.** JavaScript-библиотека для разработки пользовательских интерфейсов. Она позволяет строить веб-интерфейсы, которые могут эффективно обновляться и управлять своим состоянием при взаимодействии с пользователем и сервером [4].

**HTML, CSS, JS**. Технологии, используемые для создания и стилизации веб-страниц на клиентской стороне. HTML определяет структуру содержимого страницы, CSS отвечает за ее визуальное оформление, а JavaScript управляет поведением страницы и обеспечивает асинхронное взаимодействие с сервером.

**MongoDB.** Современная, документоориентированная база данных, которая использует формат JSON-подобных документов со схемами. Она позволяет разработчикам строить масштабируемые приложения с гибкими запросами и индексацией. MongoDB предлагает мощные возможности агрегации и транзакций, поддерживая разнообразные операции данных. Это делает ее идеальной для приложений, требующих высокой производительности и легкой интеграции с большим количеством данных. MongoDB также поддерживает репликацию и шардинг, обеспечивая надежность и доступность данных в распределенных системах.

**Алгоритм взаимодействия компонентов**

В этой части работы рассматривается детальное взаимодействие между ключевыми компонентами системы, основываясь на диаграммах последовательности, описывается последовательность операций и обмен данных между серверными и клиентскими частями приложения. Это позволит лучше понять, как компоненты взаимодействуют друг с другом для предоставления необходимой функциональности пользователю. Рис. 3 наглядно иллюстрирует взаимодействие между ключевыми компонентами системы.

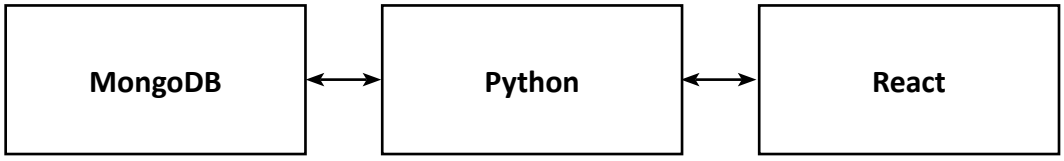



*Рис. 3.* Взаимодействие основных компонентов системы

Клиент и сервер непрерывно обмениваются данными через сокеты, позволяя приложению работать в режиме реального времени, что идеально подходит, например, для чатов, игр и совместной работы в интернете.

Алгоритм взаимодействия:

1. Python использует MongoDB для хранения и управления данными через библиотеку pymongo, которая представляет собой драйвер для работы с MongoDB. Это позволяет Python-приложениям легко взаимодействовать с базой данных, выполнять запросы, обновления, удаление и агрегацию данных [5]. MongoDB поддерживает асинхронное взаимодействие с базой данных, что совместимо с асинхронной природой FastAPI, улучшая производительность приложений, работающих в реальном времени, и позволяет связывать одну базу с несколькими серверами.

2. Сервер на Python использует FastAPI для обработки HTTP-запросов и может отправлять или получать данные в реальном времени через SocketIO [6].

3. WebRTC используется для прямой передачи потоковых данных между пользователями (например, для видеозвонков) [7].

4. React на клиентской стороне общается с сервером через HTTP и веб-сокеты для получения данных и управления состоянием интерфейса.

5. HTML, CSS и JavaScript используются для построения и оформления пользовательского интерфейса, который отображается в веб-браузере.

Клиент и сервер непрерывно обмениваются данными через сокеты, позволяя приложению работать в режиме реального времени, что идеально подходит, например, для чатов, игр и совместной работы в интернете.

Рассмотрим примеры реакций сервера на отдельные действия пользователя при взаимодействии с приложением.

На диаграмме взаимодействия (рис. 4) показан процесс отправки сообщений.

Алгоритм обмена сообщениями в реальном времени через веб-приложение включает следующие шаги:

1. Пользователь входит на сайт, авторизуется и открывается Socket-соединение.

2. Пользователь пишет сообщение и нажимает кнопку «отправить».

3. На сервер FastAPI приходит POST-запрос с сообщением в теле запроса [8].

4. Сервер проверяет состояние авторизации и записывает сообщение в БД.

5. Сервер отправляет сообщение в сокет клиента, находя его Socket ID по User ID.

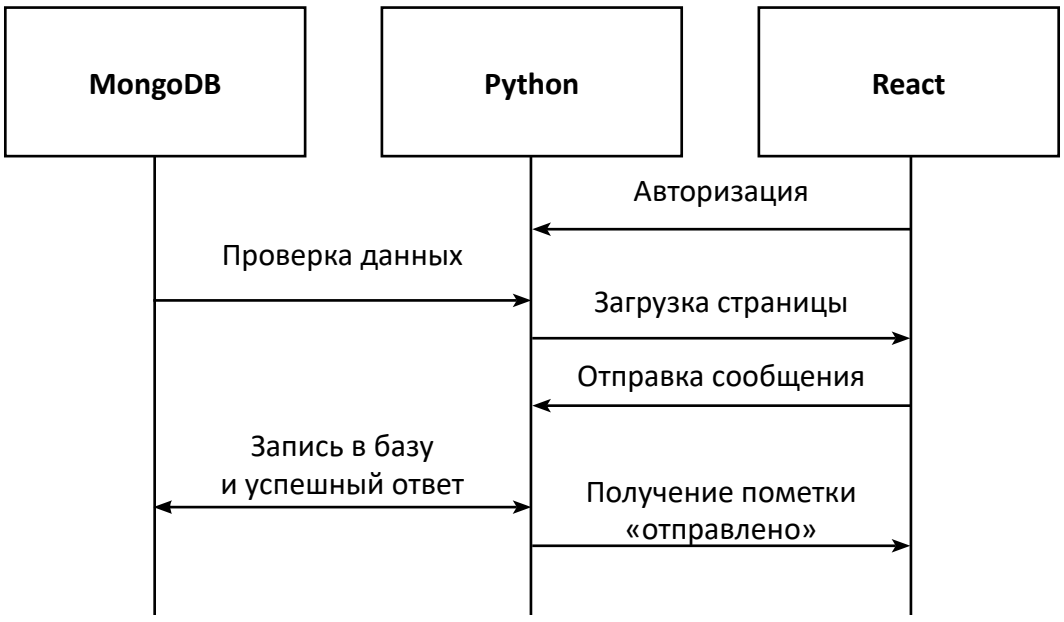



*Рис. 4.* Диаграмма взаимодействий при отправке сообщения

На диаграмме взаимодействия (рис. 5) продемонстрирован процесс добавления в друзья между двумя клиентами.

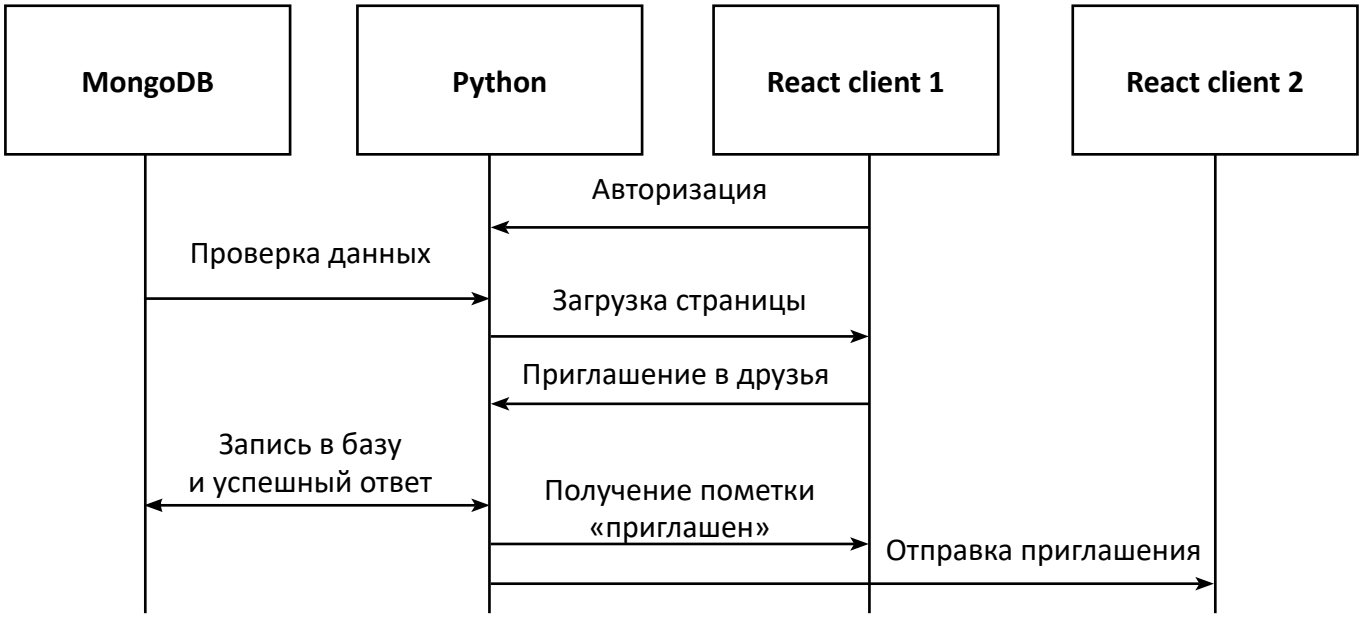



*Рис. 5.* Диаграмма взаимодействия при приглашении в друзья

Алгоритм включает следующие шаги:

1. Пользователь входит на сайт, авторизуется и открывается Socket-соединение.

2. Пользователь открывает список друзей и заходит во вкладку приглашения друзей.

3. Сервер присылает список пользователей в соответствии с введенным запросом пользователя.

4. Пользователь приглашает другого пользователя, нажимая на специальную кнопку в веб-интерфейсе.

5. На сервер FastAPI приходит POST-запрос с ID приглашенного пользователя в теле запроса.

6. Сервер проверяет состояние авторизации и записывает приглашение в друзья в БД [9].

7. Сервер отправляет сообщение в сокет клиента, находя его Socket ID по User ID, чтобы сменить кнопку приглашения на галочку. Так человек понимает, что он успешно пригласил другого пользователя.

8. Приглашение доходит до приглашенного человека через Socket ID, если он в сети.

Одним из ключевых преимуществ реализованной программы является ее способность функционировать в мультисерверной среде. Это означает, что система поддерживает одновременную работу нескольких бэкэнд-серверов, которые совместно используют одну базу данных MongoDB. Такая архитектура обеспечивает высокую масштабируемость и отказоустойчивость приложения (рис. 6).

Благодаря встроенным в код модулям синхронизации взаимодействие backend- и frontend-серверов происходит без сбоев и задержек. Эти модули отвечают за согласованность данных и предотвращение конфликтов, что особенно важно в многосерверной среде. Использование механизмов распределенного кэширования позволяет минимизировать задержки при доступе к данным. Таким образом, независимо от количества подключенных backend-серверов, данные в базе данных остаются согласованными, а пользователи получают стабильный и бесперебойный доступ к функционалу приложения. Следует отметить также, что данное решение не только подчеркивает гибкость и адаптивность системы, но и демонстрирует ее способность к оперативной реакции на изменяющиеся условия.

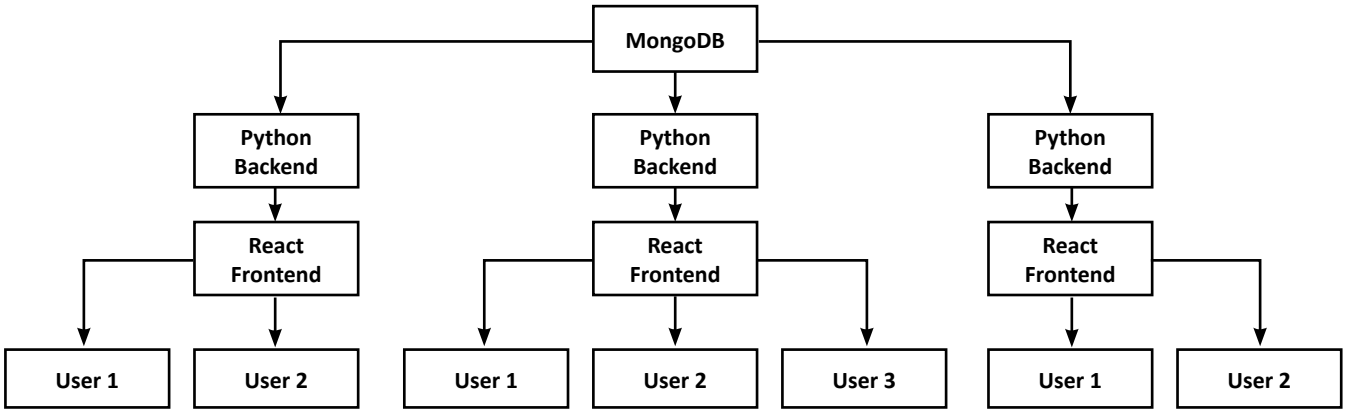



*Рис. 6.* Диаграмма мультисерверной реализации

В результате программа способна поддерживать высокие скорости обработки данных и стабильную работу даже при значительном увеличении количества пользователей и объема обрабатываемой информации.

Подводя итог вышесказанному: главная ценность проекта состоит в реализованном алгоритме взаимодействия между различными программными компонентами системы для обеспечения функциональности приложения. Именно правильная реализация и оптимизация взаимодействия обеспечивают высокую производительность, стабильность и отзывчивость системы, что является ключевым фактором в удовлетворении потребностей пользователей и достижении поставленных целей проекта.

## Заключение

Следует отметить, что разработка инновационного веб-сервиса для голосовых и текстовых чатов требует комплексного подхода к выбору технологий и архитектурных решений. Проанализированный в статье проект демонстрирует, как современные технологические решения, такие как Python, FastAPI, SocketIO, WebRTC, React, а также MongoDB, могут быть эффективно интегрированы для создания масштабируемой, удобной и функционально удобной платформы. Это позволяет не только достичь высокого уровня пользовательского опыта, аналогичного ведущим мировым платформам, но и учитывать специфические требования и контексты использования, включая аспекты импортозамещения.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Автоматизация, безопасность, онлайн-знакомства: для чего еще используют голосовые технологии в 2021 году. URL: https://habr.com/ru/articles/558774/ (дата обращения: 10.04.2024).

2. Федоров Д. Ю. Программирование на языке высокого уровня Python: учебное пособие для прикладного бакалавриата / 2-е изд., перераб. и доп. М.: Юрайт, 2019. 161 с. URL: https://urait.ru/bcode/437489 (дата обращения: 26.04.2024).

3. Streaming protocols and ultra-low latency including #webrtc. URL: https://webrtcbydralex.com/index.php/2018/05/15/streaming-protocols-and-ultra-low-latency-including-webrtc/ (дата обращения: 20.04.2024).

4. Top-10 Best Voice Chat APIs for Mobile and Web Apps. URL: https://habr.com/ru/articles/538150/ (дата обращения: 10.04.2024).

5. Рамальо Л. Совершенный Python. Пер. с англ. / СПб.: Питер, 2017. 800 с.

6. Как создать многопользовательский чат с помощью WebSocket. URL: https://dzen.ru/a/Y-AXCcMKtSLcYSJx (дата обращения: 10.04.2024).

7. Просто о WebRTC. URL: https://forasoft.github.io/webrtc-in-plain-russian/ (дата обращения: 10.04.2024).

8. Лутц М. Программирование на Python. Пер. с англ. / СПб.: Символ-Плюс, 2011. Том I, 4-е издание. 992 с.

9. Лутц М. Программирование на Python. Пер. с англ. / СПб.: Символ-Плюс, 2011. Том II, 4-е издание. 992 с.

Дата поступления: 28.05.2024
Решение о публикации: 17.06.2024

# Architectural Approaches and Technological Solutions in Creating Innovative Web Applications for Voice and Text Chats. Analysis, Prospects, and Implementation

**Sergey K. Vasilkov** — Student, Bachelor, Department of Information and Computing System.
E-mail: z2013zz@bk.ru

**Andrey V. Zabrodin** — Candidate of Historical Sciences, Associate Professor.
E-mail: zabrodin@pgups.ru

Emperor Alexander I Petersburg State Transport University, Saint Petersburg, Russia

**For citation:** Vasilkov S. K., Zabrodin A. V. Architectural Approaches and Technological Solutions in Creating Innovative Web Applications for Voice and Text Chats. Analysis, Prospects, and Implementation // Intelligent technologies on transport. 2024. No. 2 (38). P. 49–56. (In Russian). DOI: 10.20295/2413-2527-2024-238-49-56

***Abstract.*** *The article provides a detailed examination of the technologies used in an innovative service that implements voice and text chats as a web application, using an already implemented solution as an example. The main focus is on analyzing the architectural and technological solutions employed in the development of innovative web applications for voice and text chats. The article extensively discusses the functional capabilities, scalability of such systems. The service, provisionally named "Voice and Text Chat Service in the Context of Import Substitution", serves as the object of analysis. Despite its functional similarities and user experience with leading communication platforms such as Discord, Telegram, and VKontakte, it is developed with specific requirements and the latest technologies in web development in mind.*

*The article meticulously reviews the key components of the system, including server logic, client applications, and data infrastructure. Special attention is given to the integration of modern technologies into real products, including the use of cloud services, microservices architecture, and contemporary frameworks and protocols.*

*The analysis covers not only technical aspects but also issues of usability, accessibility, and inclusivity, enabling the creation of more effective services that meet user needs. The article also highlights potential directions for further development of such platforms and presents examples of best practices in this field.*

***Keywords:*** *architectural approaches, technological solutions, web applications, voice chats, text chats, WebRTC, Socket, React, innovative technologies, communication platforms.*

**REFERENCES**

1. Avtomatizaciya, bezopasnost', onlajn-znakomstva: dlya chego eshche ispol'zuyut golosovye tekhnologii v 2021 godu. URL: https://habr.com/ru/articles/558774/ (data obrashcheniya: 10.04.2024). (In Russian)

2. Fedorov D. YU. Programmirovanie na yazyke vysokogo urovnya Python: uchebnoe posobie dlya prikladnogo bakalavriata / 2-e izd., pererab. i dop. M.: YUrajt, 2019. 161 s. URL: https://urait.ru/bcode/437489 (data obrashcheniya: 26.04.2024). (In Russian)

3. Streaming protocols and ultra-low latency including #webrtc. URL: https://webrtcbydralex.com/index.php/2018/05/15/streaming-protocols-and-ultra-low-latency-including-webrtc/ (дата обращения: 20.04.2024).

4. Top-10 Best Voice Chat APIs for Mobile and Web Apps. URL: https://habr.com/ru/articles/538150/ (data obrashcheniya: 10.04.2024).

5. Ramal'o L. Sovershennyj Python. Per. s angl. / SPb.: Piter, 2017. 800 s. (In Russian)

6. Kak sozdat' mnogopol'zovatel'skij chat s pomoshch'yu WebSocket. URL: https://dzen.ru/a/Y-AXCcMKtSLcYSJx (data obrashcheniya: 10.04.2024). (In Russian)

7. Prosto o WebRTC. URL: https://forasoft.github.io/webrtc-in-plain-russian/ (data obrashcheniya: 10.04.2024). (In Russian)

8. Lutc M. Programmirovanie na Python. Per. s angl. / SPb.: Simvol-Plyus, 2011. Tom I, 4-e izdanie. 992 s. (In Russian)

9. Lutc M. Programmirovanie na Python. Per. s angl. / SPb.: Simvol-Plyus, 2011. Tom II, 4-e izdanie. 992 s. (In Russian)

Received: 28.05.2024
Accepted: 17.06.2024

УДК 004.415.53

# Обновление стека инструментов для тестирования: причины и шаги перехода с Selenium на Selenide

**Маркевич Даниил Владимирович**[1] — магистрант кафедры «Информационные и вычислительные системы». E-mail: dmarkevich811@mail.ru

**Хомоненко Анатолий Дмитриевич**[1, 2] — доктор технических наук, профессор, профессор кафедры «Информационные и вычислительные системы» Петербургского государственного университета путей сообщения Императора Александра I, профессор кафедры «Математическое и программное обеспечение» Военно-космической академии имени А. Ф. Можайского. E-mail: khomonenko@pgups.ru

[1] Петербургский государственный университет путей сообщения Императора Александра I, Россия, Санкт-Петербург

[2] Военно-космическая академия имени А. Ф. Можайского, Россия, Санкт-Петербург

**Для цитирования:** Маркевич Д. В., Хомоненко А. Д. Обновление стека инструментов для тестирования: причины и шаги перехода с Selenium на Selenide // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 57–68. DOI: 10.20295/2413-2527-2024-238-57-68

***Аннотация.*** *Автоматизированное тестирование с использованием Selenium было стандартом в разработке ПО, но с ростом сложности приложений возникли потребности в более продвинутых инструментах, таких как Selenide. Рассматривается переход к Selenide, подчеркиваются расширенные возможности и удобство использования Selenide для автоматизированного тестирования.* ***Цель исследования:*** *демонстрация преимуществ перехода на Selenide для автоматизированного тестирования браузера, повышение стабильности тестирования и предоставление руководства для миграции.* ***Методы и средства:*** *включают настройку среды тестирования, перенос тестовых сценариев, оптимизацию и рефакторинг тестов. Используются такие функции Selenide, как автоматическое ожидание, сжатый синтаксис и улучшенная обработка ошибок. Приведены примеры и конфигурации ряда инструментов: Maven, Gradle и Allure. Исследование отражает процесс перехода на Selenide, демонстрируя улучшения в стабильности и удобочитаемости тестов. Приведены примеры тестовых сценариев, оптимизированных для повышения производительности и удобства обслуживания.* ***Практическая значимость:*** *заключается в повышении эффективности тестов. Рассмотрен комплексный процесс миграции, описаны этапы настройки, миграции сценариев и оптимизации, а также проблемы во время перехода и решения. Дальнейшие исследования целесообразно направить на оптимизацию производительности Selenide в крупномасштабных приложениях и изучение дополнительных функций.*

***Ключевые слова:*** *автоматизированное тестирование, Selenium, Selenide, оптимизация тестов, Allure-отчеты.*

## Введение

В отрасли разработки программного обеспечения всегда необходимо стремиться к развитию в соответствии с современными тенденциями. Развивающиеся и реализованные проекты становятся все сложнее, что означает необходимость разрабатывать более надежные и эффективные тесты. Selenium — одна из самых популярных платформ для тестирования веб-приложений,

уже несколько лет является стандартом в отрасли. С появлением новых технологий возникли новые инструменты, которые обеспечивают расширенную функциональность и удобный пользовательский интерфейс [1].

В этой статье рассмотрен один из них — Selenide, являющийся надстройкой над Selenium и представляющий собой надежную платформу для упрощенного написания автоматизированных тестов. Легко сопровождающиеся тесты, интуитивно понятный API, встроенный механизм ожидания и поддержка динамически изменяющихся элементов позволяют Selenide стать достойной альтернативой традиционному тестированию на Selenium [2].

В последующих разделах рассмотрено, какие преимущества существуют при переходе с Selenium на Selenide. Также представлено подробное сравнение двух инструментов и пошаговое руководство, которое помогает существенно упростить процесс проверки на создание более стабильных и поддерживаемых тестов.

## Причины перехода на Selenide в автоматизации тестирования

Selenide разработан в 2011 году для обеспечения более простого и понятного синтаксиса по сравнению с Selenium. Недостатки Selenium, такие как написание большого количества шаблонного кода, ручное управление ожиданиями и взаимодействием с браузером, могли быть трудоемкими для тестировщиков, именно поэтому, для того чтобы избавить пользователя от этих проблем, создавался Selenide, который позволяет упрощать процесс написания и поддержки автоматизированных тестов [3].

Для полного осознания преимуществ Selenide необходимо иметь понимание фундаментальных отличий между Selenium и Selenide. Несмотря на то что обе платформы имеют одну и ту же главную цель (автоматизированное вебтестирование), предложенные ими подходы и функциональные возможности значительно отличаются [4].

Первым отличием Selenide является синтаксис и дизайн API. Хотя API Selenium эффективен, он иногда может быть громоздким и повторяющимся. Для выполнения стандартного теста в Selenium необходимо написать ряд строк кода, которые определяют местоположение элементов на странице, выполняют действия с ними и проверяют условия. В отличие от предшественника Selenide обладает более кратким и адаптивным интерфейсом программирования, что позволяет уменьшить объем кода, необходимого для решения аналогичных задач. Примеры с поиском элемента и выполнением действия с помощью Selenium и Selenide представлены на рис. 1 и 2.

```
WebDriver driver = new ChromeDriver();
driver.get("http://example.com");
WebElement element = driver.findElement(By.id("someId"));
element.click();
```

*Рис. 1.* Реализация поиска элемента с последующим щелчком через Selenium

```
open( relativeOrAbsoluteUrl: "http://example.com");
$(id("someId")).click();
```

*Рис. 2.* Реализация поиска элемента с последующим щелчком через Selenide

Следующий отличительный аспект — встроенные механизмы ожидания. При проведении тестирования веб-сайтов часто возникают сложности с обработкой динамического контента и асинхронных событий. Если настройка явных или неявных ожиданий в Selenium будет выполнена неправильно, это может привести к сбоям в тестировании. Selenide же имеет встроенные механизмы ожидания, которые автоматически справляются со временем загрузки элементов и предотвращают возникновение ошибок при взаимодействии с ними [5].

На рис. 3 и 4 приведены примеры использования Selenium и Selenide, где происходит ожидание отображения элемента с последующим щелчком.

Далее стоит рассмотреть удобство чтения и сопровождения тестов, так как благодаря гибкому API Selenide тестовые сценарии становятся более понятными и выразительными. Это особенно важно при работе с большим количеством тестов по мере развития проекта. В связи с этим тестировщики могут легко понимать и изменять тесты, без необходимости разбираться в сложном коде [6].

Пример с подтверждением наличия ожидаемого текста с помощью Selenium и Selenide представлен на рис. 5 и 6.

Также важным отличительным фактором является настройка веб-драйвера и браузера. Время на установку и обслуживание может быть выше из-за необходимости тестировщиков Selenium контролировать настройки WebDriver и браузера. Selenide решает эту проблему, так как предоставляет умные настройки по умолчанию и автоматическое управление конфигурацией браузера [7].

Рис. 7 и 8 демонстрируют примеры настройки WebDriver с использованием библиотек Selenium и Selenide. После анализа основных различий между Selenium и Selenide отметим конкретные преимущества, благодаря которым использование Selenide становится значительно более предпочтительным вариантом для автоматизированного тестирования [8].

Основное преимущество Selenide заключается в его упрощенном синтаксисе, который заметно сокращает количество обычного кода для написания тестов.

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement element = wait.until(ExpectedConditions
        .visibilityOfElementLocated(By.id("someId")));
element.click();
```

*Рис. 3.* Реализация ожидания отображения элемента с щелчком через Selenium

```
open( relativeOrAbsoluteUrl: "http://example.com");
$(id("someId")).click();
```

*Рис. 4.* Реализация ожидания отображения элемента с щелчком через Selenide

```
WebElement element = driver.findElement(By.id("someId"));
String text = element.getText();
assertEquals( expected: "Expected Text", text);
```

*Рис. 5.* Реализация проверки наличия ожидаемого текста через Selenium

```
$(id("someId")).shouldHave(text("Expected Text"));
```

*Рис. 6.* Реализация проверки наличия ожидаемого текста через Selenide

```
ChromeOptions options = new ChromeOptions();
options.addArguments("--headless=new");
WebDriver driver = new ChromeDriver(options);
driver.get("http://example.com");
```

*Рис. 7.* Реализация настройки WebDriver через Selenium

```
Configuration.headless = true;
open( relativeOrAbsoluteUrl: "http://example.com");
```

*Рис. 8.* Реализация настройки WebDriver через Selenide

Для долгосрочного успеха любого проекта при автоматизации тестирования является важным иметь код, который возможно прочитать и поддерживать. Использование Selenide повышает понятность тестовых сценариев, делая их более интуитивными и простыми для чтения. На рис. 9 показано, как осуществляется отправка формы через Selenide.

Далее рассмотрим такое преимущество, как автоматическая обработка ожиданий и синхронизация. В Selenium частой проблемой является обработка динамического контента и синхронизация. Для обработки этих сценариев Selenium требует применения явного или неявного ожидания, что может вызывать ошибки и некорректное выполнение теста. Selenide легко решает данную проблему благодаря своим встроенным механизмам ожидания. Selenide автоматически ожидает каждого взаимодействия, чтобы элемент достиг требуемого состояния, прежде чем продолжить. Это обеспечивает стабильность и надежность тестов.

Также Selenide обеспечивает повышенную стабильность во время тестирования благодаря использованию встроенного режима ожидания и улучшенной обработке асинхронных событий. Большое значение имеет эта стабильность для крупных проектов, которые содержат сложные веб-приложения. В таких случаях проблемы с синхронизацией и динамическим контентом могут часто нарушить результаты тестов. При уменьшении вероятности ложных результатов тестов Selenide повышает надежность результатов выполнения всего набора тестирования [9].

Кроме того, Selenide обеспечивает дополнительные возможности для отчетов об ошибках и отладки, которые включают подробную информацию о сбоях тестирования. Если тест Selenide обнаруживает сбой, то он автоматически создает скриншот экрана и регистрирует состояние браузера на момент сбоя. Это может быть весьма полезным для определения причины возникновения. Отчеты об ошибках улучшают эффективность процесса тестирования путем быстрого выявления и устранения проблем со стороны тестировщиков [10]. Для настройки генерации скриншотов при сбое можно использовать всего одну команду, которая представлена на рис. 10.

```
$(id("text field element")).setValue("Example Text");
$(id("submit button element")).click();
```

*Рис. 9.* Реализация отправки формы через Selenide

```
Configuration.reportsFolder = "target/reports";
```

*Рис. 10.* Команда для генерации скриншотов в указанную директорию при сбое

Отметим, что Selenide с легкостью интегрируется с JUnit, TestNG и CI/CD-конвейерами для разработки и тестирования. С помощью этой интеграции команды могут легко внедрить Selenide в свои существующие рабочие процессы без проблем. Более того, благодаря совместимости Selenide со множеством популярных платформ разработки он становится универсальным решением для широкого спектра проектов [11].

Тем не менее у некоторых тестировщиков могут возникнуть опасения относительно перехода с Selenium, несмотря на преимущества Selenide.

1. Процесс обучения: Selenide использует нетрудный синтаксис и подход, так что тестировщики, знакомые с Selenium, легко адаптируются к ним. API Selenide является интуитивно понятным и легким для изучения благодаря своей конструкции, и многие специалисты считают, что изучение Selenide быстро окупится благодаря увеличению производительности и сокращению расходов на обслуживание [12].

2. Совместимость с имеющимися тестами Selenium: тестировщикам стоит обратить внимание на совместимость уже имеющихся тестов на Selenium с Selenide. Selenide позволяет постепенно преобразовывать существующий код Selenium без необходимости полного переписывания для Selenide, так как он создан поверх Selenium. Этот пошаговый подход позволяет каждой команде выполнять миграцию в собственном темпе, минимизируя возможные сбои при проведении тестирования. Перевод на Selenide не представляет сложности, и пример его реализации в одном методе показан на рис. 11 и 12.

3. Производительность: некоторые специалисты напрасно сомневаются в том, что функции Selenide (ожидания и автоматическое управление браузером) оказывают влияние на производительность. На практике использование Selenide для оптимизации часто приводит к более быстрому выполнению тестов, поскольку требуется меньше ручной обработки ожидания и сокращаются проблемы с синхронизацией. Также использование поддержки параллельного тестирования в Selenide может значительно увеличить производительность и делать его особенно подходящим для обработки больших наборов тестов [13].

4. Гибкость и индивидуальная настройка: Selenium обладает гибкостью, которая позволяет тестировщикам в значительной степени настраивать свои сценарии проверки. Selenide поддерживает эту гибкость, предлагая разумные настройки по умолчанию и встроенные функции, которые покрывают наиболее распространенные возможности использования. Если это необходимо, тестировщики все еще могут использовать базовое Selenium API для работы с уникальными или сложными сценариями тестирования, так как Selenide не имеет ограничений в этом отношении [14].

```
@Step("Выбор раздела 'Научные конференции'")
public MainPage scientificConferencesClick() {
    webDriverWait.until(ExpectedConditions.elementToBeClickable(scienceModule)).click();
    webDriverWait.until(ExpectedConditions.elementToBeClickable(scientificConferences)).click();
    return this;
}
```

*Рис. 11.* Существующий код Selenium для открытия раздела «Научные конференции»

```
@Step("Выбор раздела 'Научные конференции'")
public MainPage scientificConferencesClick() {
    $x(scienceModule).click();
    $x(scientificConferences).click();
    return this;
}
```

*Рис. 12.* Открытие раздела «Научные конференции» в результате перехода на Selenide

Наконец, Selenide предлагает привлекательную альтернативу Selenium, которая решает множество проблем и ограничений традиционных платформ веб-тестирования. Его упрощенный синтаксис, автоматическая обработка ожидания и повышенная стабильность делают его привлекательным выбором для команд, которые хотят улучшить свои возможности в автоматизированном тестировании. Selenide помогает тестировщикам сосредоточиться на гарантировании качества и надежности своих приложений, сокращая сложность написания и поддержки тестов [15].

**Шаги перехода от Selenium к Selenide**

Чтобы гарантировать бесперебойность и эффективность процесса, при переходе с Selenium на Selenide требуется провести несколько конкретных этапов. Главная цель этих этапов состоит в том, чтобы использовать расширенные функции Selenide с сохранением надежности и стабильности тестирования. В этом разделе будет приведен подробный анализ каждого шага с обозначением конкретных примеров и рекомендаций, которые помогут облегчить переход [16].

Для использования Selenide следует настроить рабочую среду, чтобы она была способна поддерживать новую платформу. Далее необходимо произвести обновление зависимостей проекта, настроить интегрированную среду разработки (IDE) и убедиться в полной совместимости уже используемых инструментов и инфраструктуры [17].

Прежде чем использовать Selenide, необходимо внести зависимости в проект. Если разработка осуществляется на Maven, необходимо прописать следующую зависимость в файле pom.xml (рис. 13).

В случае использования Gradle необходимо добавить зависимость в файл build, приведенную на рис. 14.

Также следует убедиться, что настройки в среде разработки IDE позволяют правильно распознавать библиотеку Selenide. При обновлении конфигурации проекта множество современных интегрированных сред разработки (IDE), например, IntelliJ IDEA и Eclipse, автоматически загружают и настраивают все необходимые зависимости [18].

Кроме того, необходимо проверить интеграцию Selenide с уже существующими инструментами и инфраструктурой. Это включает CI/CD-конвейеры, инструменты для создания отчетов о тестировании и все пользовательские утилиты, которые могли быть уже внедрены в проект.

После настройки среды следующим шагом является начало миграции уже имеющихся тестовых сценариев Selenium в Selenide. Для достижения правильной и эффективной работы требуется редактирование тестовых сценариев при использовании API-функций и Selenide [19].

Для знакомства с функциями и синтаксисом Selenide рекомендуется начать переносить несложные тестовые сценарии. В стандартных простых тестах обычно осуществляются проверки базовых действий, включая переход на страницу, нажатие кнопок и сравнение текста. Выполнение переноса простого теста иллюстрируется на рис. 15 и 16.

```
<dependency>
    <groupId>com.codeborne</groupId>
    <artifactId>selenide</artifactId>
    <version>7.3.1</version>
    <scope>test</scope>
</dependency>
```

*Рис. 13.* Зависимость Selenide для проекта на Maven

```
dependencies {
  testImplementation 'com.codeborne:selenide:7.3.1'
}
```

*Рис. 14.* Зависимости Selenide для проекта на Gradle

```java
@BeforeEach
public void getDriver() {
    driver = MANAGER.getDriver();
}

@DisplayName("Скачивание документа с планом научных событий")
@ParameterizedTest
@MethodSource("selenium.junit.PgupsTestData#test7TestData")
public void downloadingDocumentWithPlanOfScientificEvents(String expectedUrl) {
    WebDriver driver = MANAGER.getDriver();
    driver.manage().window().maximize();
    driver.get("https://www.pgups.ru/");

    WebDriverWait webDriverWait = new WebDriverWait(MANAGER.getDriver(), Duration.ofSeconds(10));

    webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(centralBanner));
    webDriverWait.until(ExpectedConditions.elementToBeClickable(scienceModule)).click();
    webDriverWait.until(ExpectedConditions.elementToBeClickable(scientificConferences)).click();

    webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(pageTitle));
    WebElement report = webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(planOfScientificEvents));
    MANAGER.getDriver().navigate().to(report.getAttribute( name: "href"));
    assertEquals(MANAGER.getDriver().getCurrentUrl(), expectedUrl);
}

@AfterEach
public void quitTest() {
    driver.quit();
}
```

*Рис. 15.* Тест на скачивание документа с планом научных событий через Selenium

```java
@DisplayName("Скачивание документа с планом научных событий")
@ParameterizedTest
@MethodSource("selenide.junit.PgupsTestData#test7TestData")
public void downloadingDocumentWithPlanOfScientificEvents(String fileName) {
    open( relativeOrAbsoluteUrl: "https://www.pgups.ru/");

    $x(centralBanner).should(visible, Duration.ofSeconds(20));
    $x(scienceModule).click();
    $x(scientificConferences).click();

    $x(pageTitle).should(visible, Duration.ofSeconds(20));
    File report = $x(planOfScientificEvents).download();
    checkIfFileExist(fileName);
}
```

*Рис. 16.* Тест на скачивание документа с планом научных событий на Selenide

После миграции тестовых примеров необходимо приступить к оптимизации и рефакторингу, чтобы максимально эффективно использовать функциональные возможности Selenide. В рамках этого шага улучшается читаемость тестов, снижается количество дублирования кода и обеспечивается стабильность во время тестирования, так как Selenide API предоставляет возможность разрабатывать собственные команды и утилиты, которые помогут упростить повседневные операции, и для повышения надежности тестирования начать пользоваться функциями ожидания и обработки ошибок из библиотеки [20].

Затем все тесты должны быть тщательно проверены на надежность и способность обрабатывать динамический контент, задержку в сети и другие распространенные проблемы, которые могут повлиять на стабильность теста за счет интеграции тестов Selenide в конвейер CI/CD, где происходит автоматическое выполнение тестов при каждой сборке. Это обеспечит постоянную связь и раннее обнаружение любых потенциальных проблем. Также стоит использовать инструменты Selenide для создания отчетов о тестировании и анализа результатов. Allure или TestNG предлагает подробные отчеты и информацию о выполнении тестов, которые помогают обнаруживать и устранять проблемы. Для создания отчетов Allure следует обратить внимание на зависимость, которая показана на рис. 17.

Для обеспечения эффективности формирования тестовой системы рекомендуется систематически выполнять тесты с применением Selenide и сразу же реагировать на какие-либо сбои. Проверку результатов тестов и мониторинга можно осуществить различными способами, например через использование интерфейса отчетов allure. Пример такого отчета изображен на рис. 18.

```
<dependency>
    <groupId>io.qameta.allure</groupId>
    <artifactId>allure-junit5</artifactId>
    <version>2.27.0</version>
</dependency>
```

*Рис. 17.* Зависимость для генерации allure-отчетов

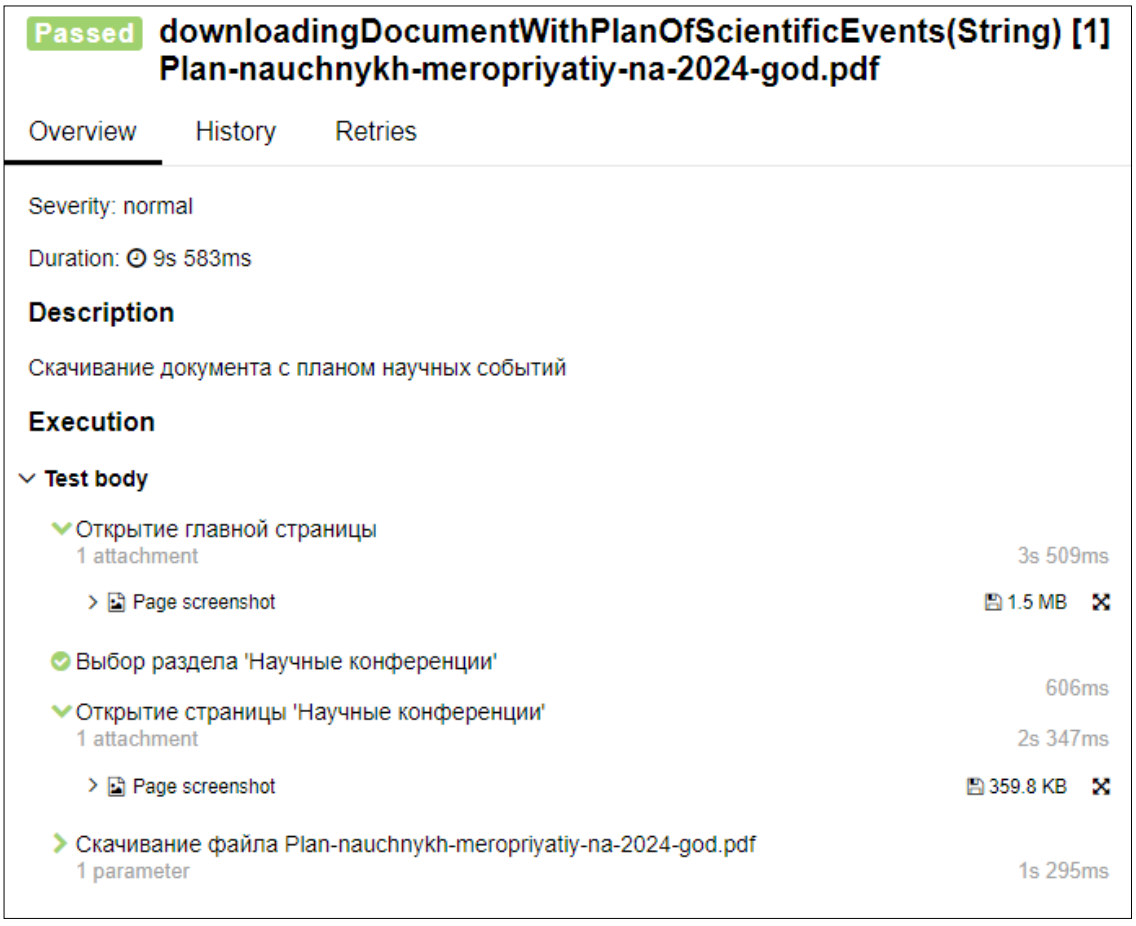



*Рис. 18.* Allure-отчет с результатами пройденного теста

Таким образом, переход к Selenide включает ряд хорошо определенных шагов, которые обеспечивают плавный и эффективный процесс миграции. Путем настройки среды, миграции тестовых скриптов, оптимизации и рефакторинга тестов и интеграции с конвейерами CI/CD можно использовать улучшенные возможности Selenide для улучшения стабильности, читабельности и поддерживаемости тестовой системы. С тщательным планированием и непрерывным развитием переход к Selenide позволит обеспечивать разработку программного обеспечения высокого качества с большей эффективностью [21].

**Заключение**

Переход от Selenium к Selenide является стратегическим шагом, который может значительно усилить качество автоматизированного тестирования. Перейдя на Selenide, можно получить доступ к более пользовательскому API, встроенным механизмам ожидания и улучшенной обработке ошибок, что поспособствует созданию более стабильных и поддерживаемых тестовых скриптов. В ходе проделанной работы очерчены критические шаги, необходимые для совершения этого перехода, — от настройки среды и миграции тестовых скриптов до оптимизации и интеграции с конвейерами CI/CD [22].

Тщательно планируя и выполняя каждый шаг, можно обеспечить плавный и эффективный переход к Selenide. Преимущества этого перехода станут очевидны по мере того, как тестовая система станет более прочной и легкой в обслуживании, что итоге приведет к более высокому качеству программного обеспечения и более быстрым циклам разработки.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Косов Е. С., Попов А. М. Разработка системы автоматизированного функционального тестирования интернет-магазина Брэндмэйкер // Пищевые инновации и биотехнологии. 2022. С. 30–31.

2. Букреева И. Р., Муртазина А. Р. Создание автоматизированных сценариев тестирования web-приложения на примере сайта «Магазин для творчества» // Инновационное развитие техники и технологий в промышленности (ИНТЕКС-2021). С. 61–63.

3. Петкун В. О. Применение типовых элементов при автоматизированном тестировании. БНТУ. 2022. С. 161–164.

4. Петрова А. И. Исследование методов и средств автоматизированного тестирования web-приложений // Новые информационные технологии в научных исследованиях (НИТ-2021). 2021. С. 119–121.

5. Архипов И. С. Внедрение автоматизированного тестирования в agile-разработке // Universum: технические науки. 2023. С. 25–30.

6. Gojare S., Joshi R., Gaigaware D. Analysis and design of selenium webdriver automation testing framework // Procedia Computer Science. 2015. Vol. 50. P. 341–346.

7. Яницкая Т. С., Моренов И. Р. Обзор и анализ существующих паттернов проектирования автоматизированных фреймворков тестирования API. МЦНП «Новая наука». 2023. С. 45–53.

8. Ramya P., Sindhura V., Sagar P. V. Testing using Selenium web driver // 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE, 2017. P. 1–7.

9. Кириллов С. С. Внедрение системы управления тестовыми данными в проект по автоматизации тестирования // Международный журнал гуманитарных и естественных наук. 2022. С. 150–152.

10. Маркевич Д. В., Хомоненко А. Д., Ермаков С. Г. От Foxpro к PostgreSQL: оптимизация, эффективное управление данными и генерация отчетов // Наукоемкие технологии в космических исследованиях Земли. 2024. № 1. С. 21–30.

11. Сударчиков Г. Е. Анализ инструментов для проведения автоматизированного функционального тестирования программного обеспечения // Проблемы развития современного общества. 2024. С. 129–132.

12. Васильев В. А. Автоматизация процесса тестирования информационных систем за счет разработки специализированных программных инструментов // Студенческая молодежь XXI века: наука, творчество, карьера, цифровизация. 2022. С. 63–72.

13. Галаган Т. А., Греков П. А. Проектирование системы автоматизированного тестирования задач по олимпиадному программированию // Вестник Амурского государственного университета, серия «Естественные и экономические науки». 2021. С. 42–45.

14. Биджиев М. Х., Ковалева К. А. Автоматизация тестирования при разработке ПО: инструменты и подходы // Актуальные проблемы научных исследований: теоретические и практические аспекты. 2023. С. 73–79.

15. Буравов А. А., Дузбаев Н. Т. Использование контейнеризации для автоматизированного тестирования программного обеспечения в онлайн-образовании // Universum: технические науки. 2022. С. 56–60.

16. Глухов К. А., Зарубин И. Б. Особенности формирования модульных и интеграционных тестов при разработке современных информационных систем. КОГРАФ-2022. 2022. С. 7–8.

17. Альтшулер И. О. Selenium WebDriver как инструмент функционального тестирования веб-приложений. Витебск: ВГУ имени П. М. Машерова. 2021. С. 286–288.

18. Смольский С. С. Роль, назначение и проблемы автоматизированного тестирования. БГУИР. 2022. С. 121–123.

19. Петренко С. А., Петренко А. А. Цифровая платформа тестирования и верификации программного кода на основе автомата динамического контроля // Дистанционные образовательные технологии. 2020. С. 400–405.

20. Кириллов С. С. Внедрение автоматизированных тестов в систему непрерывной интеграции // Научные вести. 2022. С. 37–44.

21. Бугаенко Р. С. Автоматизированное тестирование при разработке программного обеспечения // Международная научно-техническая конференция молодых ученых. 2020. С. 3835–3840.

22. Маркевич Д. В., Харланова В. В., Хомоненко А. Д. Интеграция систем бизнес-аналитики с системами управления базами данных на транспорте // Наукоемкие технологии в космических исследованиях Земли. 2023. Т. 15, № 2. С. 41–48.

Дата поступления: 03.06. 2024
Решение о публикации: 03.06.2024

# Updating the Stack of Testing Tools: Reasons and Steps for Switching from Selenium to Selenide

**Daniil V. Markevich**[1] — Master's student at the Department of Information and Computing systems of Emperor Alexander I St. Petersburg state transport university. E-mail: dmarkevich811@mail.ru

**Anatoly D. Khomonenko**[1, 2] — Doctor of Technical Sciences, Full Professor, Professor of the Department of Information and Computing systems of Emperor Alexander I St. Petersburg state transport university, Professor of the Department "Mathematics and Software" in VKA named after A. F. Mozhaisky. E-mail: khomonenko@pgups.ru

[1] Emperor Alexander I Petersburg State Transport University, Saint Petersburg, Russia
[2] A. F. Mozhaysky Military Space Academy, Saint Petersburg, Russia

**For citation:** Markevich D. V., Khomonenko A.D. Updating the stack of testing tools: reasons and steps for switching from Selenium to Selenide // Intelligent technologies on transport. 2024. No. 2 (38). P. 57–68. (In Russian). DOI:10.20295/2413-2527-2024-238-57-68

***Abstract.*** *Automated testing using Selenium was the standard in software development, but with the increasing complexity of applications, there was a need for more advanced tools such as Selenide. The transition to Selenide is considered, the advanced features and convenience of using Selenide for automated testing are emphasized. The purpose of the study is to demonstrate the benefits of switching to Selenide for automated browser testing, increase the stability of testing and provide guidance for migration.* ***Methods and tools.*** *These include setting up the testing environment, porting test scenarios, optimizing and refactoring tests. Selenide features such as automatic waiting, compressed syntax, and improved error handling are used. Examples and configurations of a number of tools are given: Maven, Gradle and Allure. The study reflects the process of switching to Selenide, demonstrating improvements in the stability and readability of texts. Examples of test scenarios optimized to increase productivity and ease of maintenance are given.* ***Practical significance.*** *It is to increase the effectiveness of tests. The complex migration process is considered, the stages of setup, scenario migration and optimization, as well as problems during the transition, and solutions are described. Further research should be directed to optimizing Selenide performance in large-scale applications and exploring additional features.*

***Keywords:*** *automated testing, Selenium, Selenide, optimization of test, Allure reports.*

**REFERENCES**

1. Kosov E. S., Popov A. M. Razrabotka sistemy avtomatizirovannogo funkcional'nogo testirovaniya internet-magazina Brendmejker // Pishchevye innovacii i biotekhnologii. 2022. S. 30–31. (In Russian)

2. Bukreeva I. R., Murtazina A. R. Sozdanie avtomatizirovannyh scenariev testirovaniya web-prilozheniya na primere sajta “Magazin dlya tvorchestva” // Innovacionnoe razvitie tekhniki i tekhnologij v promyshlennosti (INTEKS-2021). S. 61–63. (In Russian)

3. Petkun V. O. Primenenie tipovyh elementov pri avtomatizirovannom testirovanii. BNTU. 2022. S. 161–164. (In Russian)

4. Petrova A. I. Issledovanie metodov i sredstv avtomatizirovannogo testirovaniya web-prilozhenij // Novye informacionnye tekhnologii v nauchnyh issledovaniyah (NIT-2021). 2021. S. 119–121. (In Russian)

5. Arhipov I. S. Vnedrenie avtomatizirovannogo testirovaniya v agile-razrabotke // Universum: tekhnicheskie nauki. 2023. S. 25–30. (In Russian)

6. Gojare S., Joshi R., Gaigaware D. Analysis and design of selenium webdriver automation testing framework // Procedia Computer Science. 2015. T. 50. S. 341–346.

7. Yanickaya T. S., Morenov I. R. Obzor i analiz sushchestvuyushchih patternov proektirovaniya avtomatizirovannyh frejmvorkov testirovaniya API. MCNP “Novaya nauka”. 2023. S. 45–53. (In Russian)

8. Ramya P., Sindhura V., Sagar P. V. Testing using Selenium web driver // 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE, 2017. P. 1–7.

9. Kirillov S. S. Vnedrenie sistemy upravleniya testovymi dannymi v proekt po avtomatizacii testirovaniya // Mezhdunarodnyj zhurnal gumanitarnyh i estestvennyh nauk. 2022. S. 150–152. (In Russian)

10. Markevich D. V., Homonenko A. D., Ermakov S. G. Ot Foxpro k PostgreSQL: optimizaciya, effektivnoe upravlenie dannymi i generaciya otchetov // Naukoemkie tekhnologii v kosmicheskih issledovaniyah Zemli. 2024. № 1. S. 21–30. (In Russian)

11. Sudarchikov G. E. Analiz instrumentov dlya provedeniya avtomatizirovannogo funkcional'nogo testirovaniya programmnogo obespecheniya // Problemy razvitiya sovremennogo obshchestva. 2024. S. 129–132. (In Russian)

12. Vasil'ev V. A. Avtomatizaciya processa testirovaniya informacionnyh sistem za schet razrabotki specializirovannyh programmnyh instrumentov // Studencheskaya molodezh' XXI veka: nauka, tvorchestvo, kar'era, cifrovizaciya. 2022. S. 63–72. (In Russian)

13. Galagan T. A., Grekov P. A. Proektirovanie sistemy avtomatizirovannogo testirovaniya zadach po olimpiadnomu programmirovaniyu // Vestnik Amurskogo gosudarstvennogo universiteta, seriya “Estestvennye i ekonomicheskie nauki”. 2021. S. 42–45. (In Russian)

14. Bidzhiev M. H., Kovaleva K. A. Avtomatizaciya testirovaniya pri razrabotke PO: instrumenty i podhody // Aktual'nye problemy nauchnyh issledovanij: teoreticheskie i prakticheskie aspekty. 2023. S. 73–79. (In Russian)

15. Buravov A. A., Duzbaev N. T. Ispol'zovanie kontejnerizacii dlya avtomatizirovannogo testirovaniya programmnogo obespecheniya v onlajn-obrazovanii // Universum: tekhnicheskie nauki. 2022. S. 56–60. (In Russian)

16. Gluhov K. A., Zarubin I. B. Osobennosti formirovaniya modul'nyh i integracionnyh testov pri razrabotke sovremennyh informacionnyh sistem. KOGRAF-2022. 2022. S. 7–8. (In Russian)

17. Al'tshuler I. O. Selenium WebDriver kak instrument funkcional'nogo testirovaniya veb-prilozhenij. Vitebsk: VGU imeni P. M. Masherova. 2021. S. 286–288. (In Russian)

18. Smol'skij S. S. Rol', naznachenie i problemy avtomatizirovannogo testirovaniya. BGUIR. 2022. S. 121–123. (In Russian)

19. Petrenko S. A., Petrenko A. A. Cifrovaya platforma testirovaniya i verifikacii programmnogo koda na osnove avtomata dinamicheskogo kontrolya // Distancionnye obrazovatel'nye tekhnologii. 2020. S. 400–405. (In Russian)

20. Kirillov S. S. Vnedrenie avtomatizirovannyh testov v sistemu nepreryvnoj integracii // Nauchnye vesti. 2022. S. 37–44. (In Russian)

21. Bugaenko R. S. Avtomatizirovannoe testirovanie pri razrabotke programmnogo obespecheniya // Mezhdunarodnaya nauchno-tekhnicheskaya konferenciya molodyh uchenyh. 2020. S. 3835–3840. (In Russian)

22. Markevich D. V., Harlanova V. V., Homonenko A. D. Integraciya sistem biznes-analitiki s sistemami upravleniya bazami dannyh na transporte // Naukoemkie tekhnologii v kosmicheskih issledovaniyah Zemli. 2023. T. 15. № 2. S. 41–48. (In Russian)

Received: 03.06.2024
Accepted: 03.06.2024

УДК 004.056

# Подход к оцениванию функциональности доверенных программно-аппаратных комплексов

**Глухов Александр Петрович**[1] — докт. техн. наук, директор Центра критической инфраструктуры Передовой инженерной школы СВЧ-электроники.
E-mail: apg606@yandex.ru

**Белова Елена Ивановна**[2] — аспирант кафедры «Информатика и информационная безопасность».
E-mail: elenabelovavm@yandex.ru

**Глухов Александр Александрович**[3] — директор программ по информационно-телекоммуникационным системам.
E-mail: alexander.glukh0v@yandex.ru

[1] Российский технологический университет МИРЭА, Россия, Москва
[2] Петербургский государственный университет путей сообщения Императора Александра I, Россия, Санкт-Петербург
[3] АО «Научно-производственное объединение «Критические информационные системы», Россия, Москва

**Для цитирования:** Глухов А. П., Белова Е. И., Глухов А. А. Подход к оцениванию функциональности доверенных программно-аппаратных комплексов // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 69–76. DOI: 10.20295/2413-2527-2024-238-69-76

***Аннотация.*** *В статье представлены основные положения методического подхода к решению задачи оценивания функциональности доверенных программно-аппаратных комплексов по показателям, характеризующим выполнение ими функциональных задач. Основной целью исследования является создание системы мониторинга показателей функциональности объектов критического применения. Предлагаемый подход может применяться для оценивания состояния функциональности в условиях как точно заданных, так и нечетко определенных параметров информационной безопасности, работоспособности и надежности доверенных программно-аппаратных комплексов.*

***Ключевые слова:*** *доверенные программно-аппаратные комплексы, критическая информационная инфраструктура, функциональные задачи, показатели качества функционирования.*

## Введение

С 1 сентября 2024 года субъекты критической информационной инфраструктуры (КИИ) РФ могут использовать только доверенные программно-аппаратные комплексы (ДПАК), что обусловлено необходимостью решения проблем обеспечения безопасности КИИ [1].

Исходя из ГОСТ ПНСТ 905-2023 «Критическая информационная инфраструктура. Доверенные программно-аппаратные комплексы. Термины и определения» (срок действия — с 01.04.2024 по 01.04.2027) к ДПАК относятся:

— вычислительная техника;
— телекоммуникационное оборудование;
— автоматизированные системы управления;
— программное обеспечение;
— электронная компонентная база;
— аппаратно-программные платформы.

Основным признаком ДПАК (ГОСТ ПНСТ 905-2023) является решение ими заявленных функций (функциональных задач — ФЗ), что определяет необходимость перехода при оценивании соответствия ДПАК требованиям функциональности, надежности и защищенности, от оценки технических характеристик к оценке качественных и/или количественных показателей выполнения (невыполнения) ДПАК своих ФЗ.

Решение задачи оценивания функционально-технических характеристик ДПАК на техническом уровне требует разработки стандартов требований к функциональности различных типов ДПАК, состава оцениваемых параметров, моделей и методик оценивания, критериев оценки и других технических и организационных вопросов, в том числе проведения оценки функциональности в рамках процедур сертификации на соответствие требованиям доверия и мониторинга состояния ДПАК на различных этапах их жизненного цикла.

**Этапы оценивания функциональности ДПК**

Возможный подход оценивания функциональности по показателям, характеризующим выполнение ДПАК своей (своих) ФЗ может включать следующие основные этапы [2–4]:

*1-й этап*. Сбор и обработка данных об инцидентах, связанных с деструктивными воздействиями на ДПАК, от систем мониторинга IT-инфраструктуры и информационной безопасности.

*2-й этап*. Выбор интегральных показателей функциональности — показателей качества функционирования ДПАК (ПКФ ДПАК).

С учетом сложности и комплексного характера проблемы анализа функциональности ДПАК-уровня управления бизнес-процессов и решаемых ДПАК функциональных задач, произвести оценивание функциональности только на основе одного какого-либо показателя не всегда возможно. Целесообразно рассматривать совокупность показателей (в том числе и показатели надежности и защищенности, непосредственно влияющие на интегральные показатели функциональности), включающую как количественные, так и качественные характеристики [5].

Основными показателями работоспособности могут являться интегральные показатели производительности и надежности ДПАК (далее показатели качества функционирования ДПАК — ПКФ ДПАК):

— фактическая производительность ДПАК;

— вероятность безотказной работы;

— коэффициент готовности ДПАК;

— коэффициент технической готовности ДПАК [5].

Также такие показатели, в том числе качественные, как своевременность, полнота, достоверность, актуальность и др. данных формирования планов, проведения расчетов и т. п., определенные для решения ФЗ ДПАК.

*3-й этап*. Построение моделей оценивания влияния инцидентов на ПКФ ДПАК [6].

Необходимо отметить: практическое применение количественных методов оценивания часто осложнено неточностью, недостаточностью и неопределенностью исходной информации, отсутствием необходимой статистической информации по инцидентам.

В связи с этим для оценивания влияния инцидентов на показатели качества решения ФЗ предлагается следующая модель, характеризующая плавное изменение функции принадлежности (ФП) состояния выполнения ДПАК функциональной задачи от критического состояния к безопасному [7, 8] в зависимости от состояния ПКФ ДПАК (рис. 1).

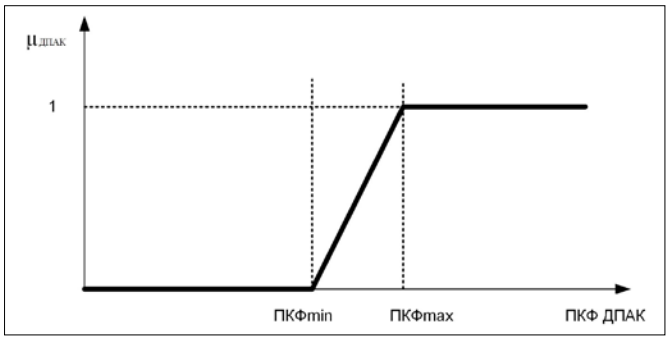



*Рис. 1.* Функция принадлежности состояния выполнения ДПАК функциональной задачи в зависимости от состояния ПКФ ДПАК

*4-й этап*. Определение эталонных и текущего состояний ПКФ ДПАК.

На уровне руководства организаций (предприятий) зачастую целесообразно давать оценку выполнения функциональных задач на основе количественных и качественных показателей, используя лингвистический подход с терминами «критическое (К)» состояние, «допустимое (Д)» и «безопасное (Б)» (при трехуровневом нечетком классификаторе) или с терминами «критическое (К)», «близкое к критическому (БК)», «допустимое (Д)», «близкое к безопасному (ББ)», «безопасное (Б)» (при пятиуровневом нечетком классификаторе).

На рис. 2 в качестве иллюстрации представлены функции принадлежности ПКФ ДПАК при трехуровневом нечетком классификаторе (ФП — треугольная функция, Тс — текущее состояние ПКФ ДПАК).

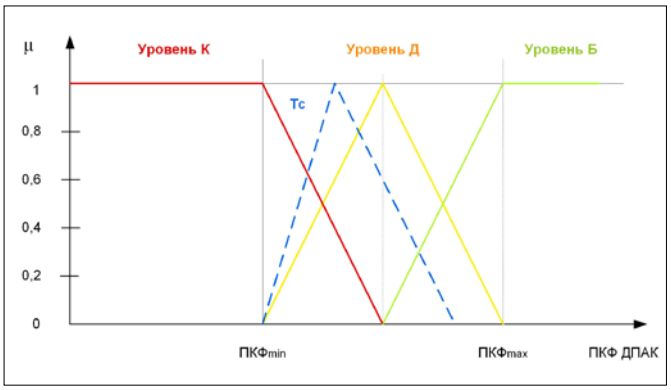



*Рис. 2.* Функции принадлежности для лингвистической переменной ПКФ ДПАК при трехуровневом нечетком классификаторе

*5-й этап*. Оценивание текущего состояния выполнения ДПАК функциональной задачи.

Текущее состояние $\widetilde{s_0}$ нечетко включается в состояние $\widetilde{s_1}$ при условии, что степень включения $\widetilde{s_0}$ и $\widetilde{s_1}$ не меньше некоторого порога $\rho$: $0{,}6 \le \rho \le$ .

Степень включения состояния $\widetilde{s_0}$ в состояние $\widetilde{s_l}$ определяется выражением:

$$v(\widetilde{s_0},\widetilde{s_l}) = \underset{y=Y}{\&} v(\mu_{s_0}(y), \mu_{s_i}),$$

где $v(\mu_{s_0}(y), \mu_{s_i})$ определяются следующим образом:

$$v(\mu_{s_0}(y), \mu_{s_i}) = \underset{l=L}{\&} v\left(\mu_{\mu s_0}(T_0^l) \to \mu_{\mu s_i}(T_i^l)\right).$$

Таким образом, состояние $\widetilde{s_0}$ нечетко включается в состояние $\widetilde{s_1}$, если степень включения $\widetilde{s_0}$ в $\widetilde{s_1}$ не меньше порогового $\rho_{пор} \in [0{,}6;\ 1]$, т. е. $v(\widetilde{s_0}, \widetilde{s_1}) \ge \rho_{пор}$.

Зачастую является актуальным и представляющим интерес для стратегического управления бизнесом (особенно для ДПАК-АСУ) оценивание функциональности ДПАК, в том числе в условиях деструктивных воздействий, на уровне бизнес-процессов и целей деятельности предприятия. Иерархическая модель активов для проведения таких оценок представлена на рис. 3. Данная модель является основой для построения иерархии показателей качества на уровнях цели, бизнес-процессов, ФЗ, ПКФ и функционально-технических характеристик ДПАК.

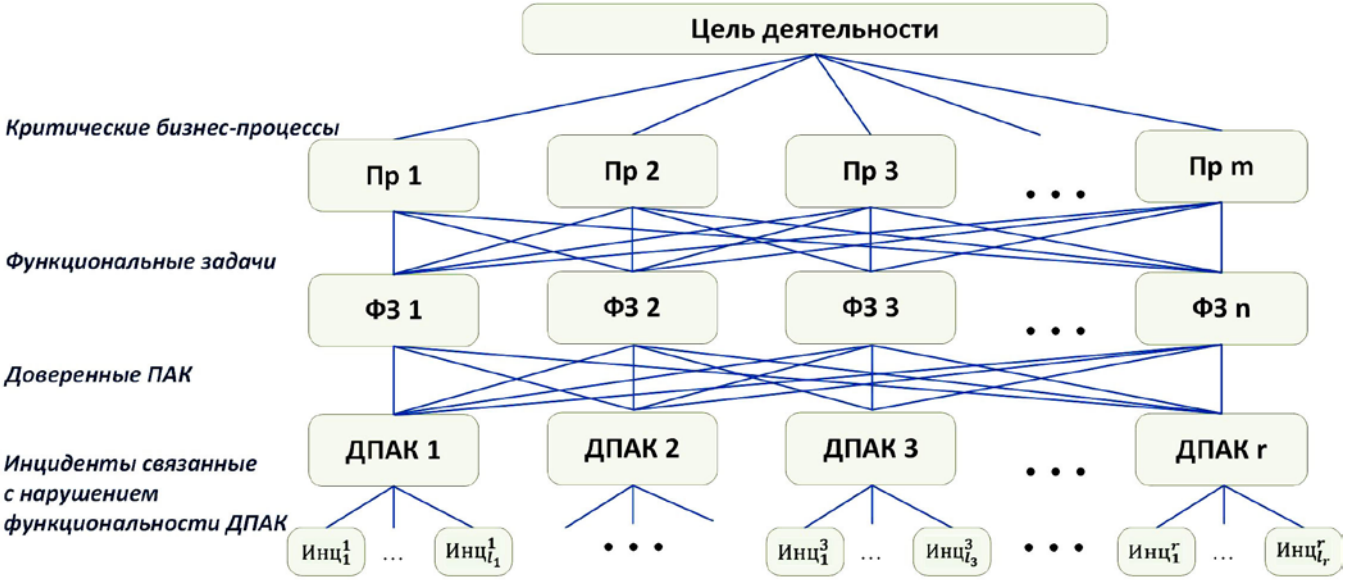



*Рис. 3.* Иерархическая модель активов для проведения оценивания функциональности ДПАК на уровне бизнес-процессов

В качестве основы модели определения уровней ИБ значимых активов пассажирских перевозок предлагается использовать продукционные правила и алгоритмы логического вывода [9].

В модель входят:

— иерархическая модель нечетко-продукционных правил;

— модель нечетко-продукционного правила;

— алгоритм логического вывода на нечетких правилах.

Особенностями предлагаемой иерархической модели продукционных правил являются:

1) модель представляет собой нечеткую иерархическую систему, которая включает в себя следующие уровни правил:

— для цели ПП;

— для бизнес-процессов ПП;

— для функциональных задач;

— для ДПАК;

— для деструктивных воздействий на ДПАК;

2) каждый выход одного правила (кроме самого верхнего) является входом для другого правила следующего уровня иерархии;

3) в иерархии нет независимых правил, у которых отсутствуют связи с другими правилами.

Формирование выходного множества показателей на разных уровнях иерархии можно производить:

— с использованием алгоритмов нечеткого вывода, например представленного выше;

— на основе статистических данных с применением методов машинного обучения;

— на основе экспертных оценок.

Основные этапы оценивания функциональности ДПАК по состоянию бизнес-процессов и целей деятельности представлены на рис. 4.

## Пример оценивания ПКФ ДПАК

Рассмотрим в качестве примера оценивание одного из ПКФ ДПАК с использованием аппарата нечетких категорий.

Пусть уровням безопасности (уровни решения ДПАК ФЗ) ПКФ ДПАК соответствуют нечеткие понятия: «безопасный», «допустимый» и «критический».

Совокупность этих состояний опишем векторной лингвистической переменной, $Z = < NZ =$ «уровень ИБ», $TZ, OZ, CZ >$, где $Tz = \{y_1, y_2, y_3, y_4\}$.


*Рис. 4.* Основные этапы оценивания функциональности ДПАК по состоянию бизнес-процессов

Пусть, например:

$y_1 = < N^{y_1} =$ «критическое», $O^{y_1} = [0,55]$, $П^{y_1} = \{< 1|0 >, < 0,9|15 >; < 0,65|25 >; < 0,4|35 >; < 0,2|45 >; < 0,1|50 >\}>$;

$y_2 = < N^{y_2} =$ «допустимое», $O^{y_2} = [35,75]$, $П^{y_2} = \{< 0,1|40 >, < 0,2|50>; < 0,45|55 >; <0,65|60>; < 0,8|65 >; < 0,9|70 >\}>$;

$y_3 = < N^{y_3} =$ «безопасное», $O^{y_3} = [45,100]$, $П^{y_3} = \{< 0,1|50 >, < 0,2|55 >; < 0,45|65 >; < 0,6|70 >; < 0,8|80 >; < 0,9|85 >; >; < 1|100 >\}>$;

$y_4 = < N^{y_4} =$ «текущее», $O^{y_4} = [45,75]$, $П^{y_4} = \{< 0,1|50 >, < 0,2|55 >; < 0,4|60 >; < 0,6|65 >; < 0,9|70 >\}>$

Функции принадлежности, характеризующие степень включения текущей ситуации состоянию «критическое», «допустимое», «безопасное», представлены в табл. 1.

*Таблица 1*

**Степень включения текущей ситуации**

| ФП | Степень включения состояния |
|---|---|
| $\mu^{41}(y_4, y_1)$ | 0,1 |
| $\mu^{41}(y_4, y_2)$ | 0,65 |
| $\mu^{41}(y_4, y_3)$ | 0,45 |

Проверяем соответствие функции принадлежности двух ситуаций порогу включения ситуаций. Максимальное значение степени включения состояния: $\mu(y_1, y_2, y_3) = max\ (0,1; 0,65; 0,45) = 0,65$.

Определяем принадлежность текущей ситуации ($y_4$) состоянию «допустимое» ($y_2$), при этом нечеткость с описания текущей ситуации снимается.

На рис. 5 представлены значения степеней включения текущего состояния и уровни функций принадлежности нечеткого ПКФ ДПАК. Проведенный расчет показал, что при сравнении экспертной оценки текущего состояния безопасности с допустимым состоянием было выявлено соответствие значению порога включения ситуаций, что свидетельствует о допустимом текущем состоянии ПКФ ДПАК.

## Заключение

Таким образом, для решения задач оценивания функциональности ДПАК с учетом взаимосвязи показателей функциональности, надежности и защищенности представляется целесообразным проведение работ по созданию интеллектуальной автоматизированной системы мониторинга и управления функциональностью доверенных программно-аппаратных комплексов (АСУ Ф ДПАК)

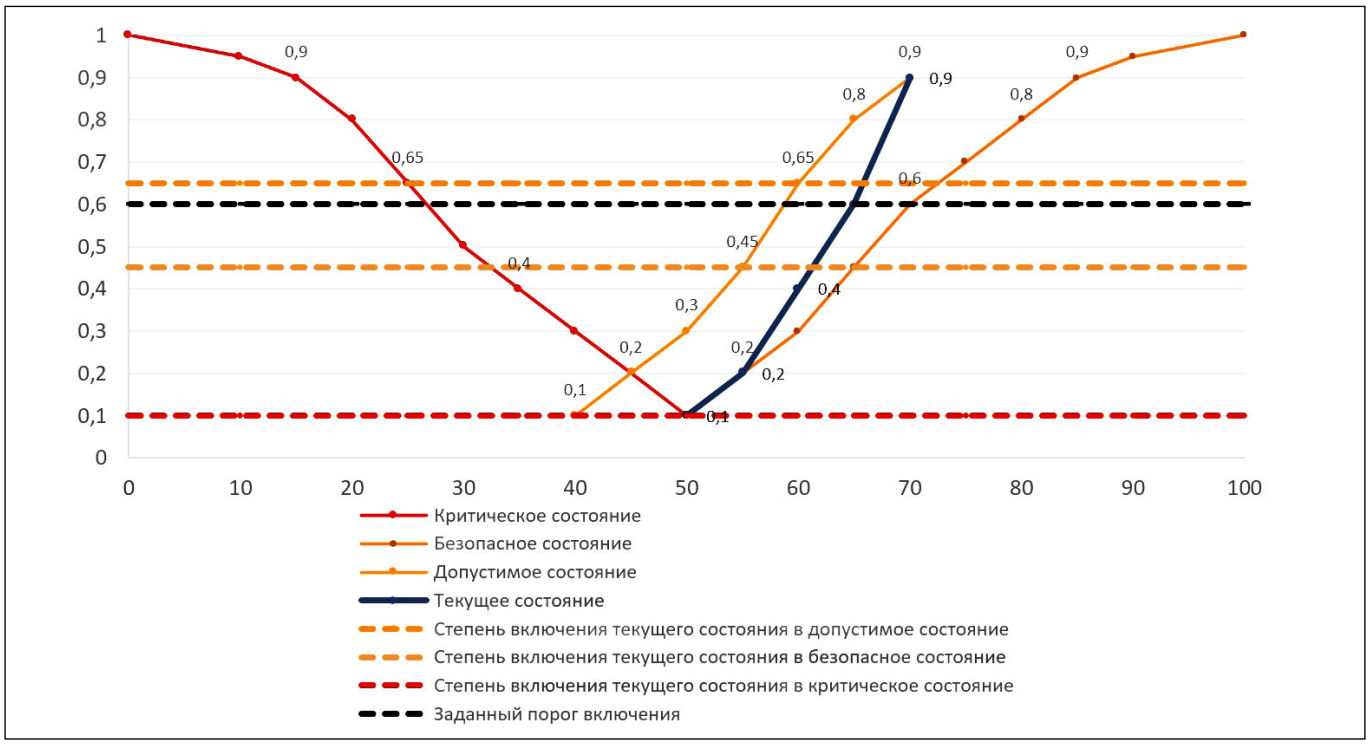



*Рис. 5.* Значения степеней включения текущего состояния и уровни функций принадлежности ПКФ ДПАК

и интеграции ее с системами управления и мониторинга ИТ-инфраструктуры и информационной безопасности. Применение предлагаемого подхода возможно в рамках разработки методологии построения риск-моделей функциональности, надежности и защищенности ДПАК, моделей и методик оценивания ценности и состояния бизнес-процессов, функциональных задач и ДПАК (элементов ДПАК), моделей и методик управления рисками и ресурсами.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. О порядке перехода субъектов критической информационной инфраструктуры РФ на преимущественное применение доверенных программно-аппаратных комплексов (ПАК) на принадлежащих им значимых объектах критической информационной инфраструктуры РФ: Постановление Правительства РФ от 14.11.2023 № 1912. Собр. законодательства Рос. Федерации. 2023. № 47, ст. 8423.

2. Глухов А. П., Корниенко А. А., Ададуров С. Е. и др. Оценивание информационной безопасности бизнес-процессов. Автоматика, связь, информатика. 2023. № 7. С. 17–20.

3. Глухов А. П., Сидак А. А., Василенко В. В. и др. Подходы к обеспечению безопасности критической информационной инфраструктуры железнодорожного транспорта. Двойные технологии. 2020. № 4. С. 69–74.

4. Белова Е. И. Модели и алгоритмы оценивания информационной безопасности автоматизированной системы управления пассажирскими перевозками железнодорожного транспорта. Двойные технологии. 2023. № 2. С. 48–54.

5. Белова Е. И., Глухов А. П., Корниенко А. А. Подход к оцениванию информационной безопасности автоматизированных систем управления пассажирским перевозками железнодорожного транспорта. Двойные технологии. 2023. № 1 (102). С. 71–77.

6. Зиновьев П. А., Мейко А. В., Моисеев В. С. Инженерные методы расчета функциональной надежности и живучести корпоративных информационных систем. Монография. Казань: Отечество. 2009. 256 с.

7. Борисов А. Н., Крумберг О. А., Федоров И. П. Принятие решений на основе нечетких моделей. Примеры использования. Рига: Зинантне, 1990. 184 с.

8. Долженко А. И. Оценка нефункциональных характеристик качества информационной системы на основе теории нечетких чисел. Известия вузов. Северо-Кавказский регион. Естественные науки. Приложение. 2006. № 8. С. 3–9.

9. Катасев А. С., Емалетдинова Л. Ю. Нечетко-продукционная каскадная модель диагностики состояния сложного объекта. Программные системы и вычислительные методы. 2013. № 1 (2). С. 69–81.

Дата поступления: 28.05.2024.
Решение о публикации: 17.06.2024.

# An Approach to Evaluating the Functionality of Trusted Software and Hardware Systems

**Aleksandr P. Glukhov**[1] — Doctor of Technical Sciences, Russian University of Technology MIREA.
E-mail: apg606@yandex.ru

**Elena I. Belova**[2] — Postgraduate student of the Department of Informatics and Information Security.
E-mail: elenabelovavm@yandex.ru

**Aleksandr A. Glukhov**[3] — Director of Information and Telecommunication Systems Programs.
E-mail: alexander.glukh0v@yandex.ru

[1] MIREA Russian Technological University, Moscow, Russia
[2] Emperor Alexander I Petersburg State Transport University, Saint Petersburg, Russia
[3] JSC Scientific and Production Association "Critical Information Systems", Moscow, Russia

**For citation:** Glukhov A. P., Belova E. I., Glukhov A. A. An approach to evaluating the functionality of trusted software and hardware complexes. Intelligent technologies on transport. 2024. No. 2 (38). P. 69–76. (In Russian). DOI: 10.20295/2413-2527-2024-238-69-76

***Abstract.*** *The main provisions of a methodological approach for the functionality evaluating of trusted software and hardware systems according to its functional tasks performance indicators are presented in the article.*
*The main purpose of the study is to create a critical objects functionality indicators monitoring system. The approach proposed is to assess the functionality status in conditions both precisely and vaguely defined parameters of information security, operability and reliability of trusted software and hardware systems.*

***Keywords:*** *trusted software and hardware systems, critical information infrastructure, functional tasks, performance indicators.*

**REFERENCES**

1. O poryadke perekhoda sub"ektov kriticheskoj informacionnoj infrastruktury RF na preimushchestvennoe primenenie doverennyh programmno-apparatnyh kompleksov (PAK) na prinadlezhashchih im znachimyh ob"ektah kriticheskoj informacionnoj infrastruktury RF: Postanovlenie Pravitel'stva RF ot 14.11.2023 № 1912. Sobr. zakonodatel'stva Ros. Federacii. 2023. № 47, st. 8423. (In Russian)

2. Gluhov A. P., Kornienko A. A., Adadurov S. E. i dr. Ocenivanie informacionnoj bezopasnosti biznes-processov. Avtomatika, svyaz', informatika. 2023. № 7. S. 17–20. (In Russian)

3. Gluhov A. P., Sidak A. A., Vasilenko V. V. i dr. Podhody k obespecheniyu bezopasnosti kriticheskoj informacionnoj infrastruktury zheleznodorozhnogo transporta. Dvojnye tekhnologii. 2020. № 4. S. 69–74. (In Russian)

4. Belova E. I. Modeli i algoritmy ocenivaniya informacionnoj bezopasnosti avtomatizirovannoj sistemy upravleniya passazhirskimi perevozkami zheleznodorozhnogo transporta. Dvojnye tekhnologii. 2023. № 2. S. 48–54. (In Russian)

5. Belova E. I., Gluhov A. P., Kornienko A. A. Podhod k ocenivaniyu informacionnoj bezopasnosti avtomatizirovannyh sistem upravleniya passazhirskim perevozkami zheleznodorozhnogo transporta. Dvojnye tekhnologii. 2023. № 1 (102). S. 71–77. (In Russian)

6. Zinov'ev P. A., Mejko A. V., Moiseev V. S. Inzhenernye metody rascheta funkcional'noj nadezhnosti i zhivuchesti korporativnyh informacionnyh sistem. Monografiya. Kazan': Otechestvo. 2009. 256 s. (In Russian)

7. Borisov A. N., Krumberg O. A., Fedorov I. P. Prinyatie reshenij na osnove nechetkih modelej. Primery ispol'zovaniya. Riga: Zinantne, 1990. 184 s. (In Russian)

8. Dolzhenko A. I. Ocenka nefunkcional'nyh harakteristik kachestva informacionnoj sistemy na osnove teorii nechetkih chisel. Izvestiya vuzov. Severo-Kavkazskij region. Estestvennye nauki. Prilozhenie. 2006. № 8. S. 3–9. (In Russian)

9. Katasev A. S., Emaletdinova L. Yu. Nechetko-produkcionnaya kaskadnaya model' diagnostiki sostoyaniya slozhnogo ob"ekta. Programmnye sistemy i vychislitel'nye metody. 2013. № 1 (2). S. 69–81. (In Russian)

Received: 28.05.2024
Accepted: 17.06.2024

УДК 625.03

# Автоматизированный контроль перемещения тормозных башмаков на железнодорожном транспорте: применение RFID-технологии при закреплении подвижного состава

**Кагадий Ирина Геннадьевна** — магистрант 2-го курса направления 09.04.02 «Информационные системы и технологии». E-mail: irina.kagadiy@mail.ru

**Ермаков Сергей Геннадьевич** — доктор технических наук, профессор, заведующий кафедрой «Информационные и вычислительные системы». E-mail: ermakov@pgups.ru

Петербургский государственный университет путей сообщения Императора Александра I, Россия, Санкт-Петербург

**Для цитирования:** Кагадий И. Г., Ермаков С. Г. Автоматизированный контроль перемещения тормозных башмаков на железнодорожном транспорте: применение RFID-технологии при закреплении подвижного состава // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 77–83. DOI: 10.20295/2413-2527-2024-238-77-83

***Аннотация.*** *В статье рассматривается возможность внедрения RFID-технологии в работе железнодорожного транспорта как одного из аспектов цифровизации железной дороги. Описывается общий принцип работы данной технологии при закреплении подвижного состава. Проанализированы положительные стороны внедрения данной технологии.*

***Ключевые слова:*** *RFID, контроль перемещения, закрепление состава, тормозные башмаки, цифровизация.*

В данной статье применяются следующие термины, определения и сокращения:

ОАО «РЖД» — открытое акционерное общество «Российские железные дороги»;
АРМ ЖУТБ — автоматизированное рабочее место «Журнал учета тормозных башмаков»;
ДСП — дежурный по станции;
RFID — Radio Frequency Identification;
МРМ — мобильное рабочее место;
КП ЭДО — комплекс программных средств технологического электронного документооборота;
ИСУЖТ НС — подсистема нормативного обеспечения планировщика работы железнодорожных станций для ПТК ИСУЖТ сетевого уровня;
ТРА станции — техническо-распорядительные акты железнодорожной станции;
ПЭП — простая электронная подпись;
ЭД — документ, зафиксированный на электронном носителе (в виде набора символов, изображений) и предназначенный для передачи во времени и пространстве с использованием средств вычислительной техники и электросвязи с целью хранения и использования;
КП ЭДО — комплекс Программных средств технологического электронного документооборота;
ПО — программное обеспечение.

**Введение**

Анализ существующей системы контроля станционных тормозных башмаков на железнодорожной станции ОАО «РЖД» показал, что основной проблемой является слабая организация контроля в процессе их хранения и перемещения. Сотрудники визуально определяют местонахождение башмака и его номер. ДСП вводит данные о закреплении состава в АРМ ЖУТБ ручным способом по принятию информации о выполнении операции от составителя.

Также при хранении, выдаче и списывании тормозных башмаков фиксация информации производится в ручном режиме.

Таким образом, в рассматриваемой предметной области закрепления подвижного состава и сохранности инвентаря строгого учета (тормозных башмаков), где основным фактором является безопасность, объективно существует противоречие между необходимостью автоматизированного мониторинга сохранности инвентаря, факта выполнения норм ТРА и возможностью их выполнения с применением существующей системы контроля.

Нарушение безопасности движения подвижного состава, включая несанкционированное движение и сход вагонов, представляет серьезную угрозу с тяжелыми последствиями, такими как крушения и аварии. Главной причиной таких происшествий часто является неправильное выполнение процесса закрепления вагонов, связанное с человеческим фактором, а также отсутствие автоматизированных систем контроля станционных тормозных башмаков.

Для цифровизации процесса закрепления подвижного состава можно воспользоваться технологиями RFID, которые представляют собой автоматическую идентификацию объекта с помощью радиоприемника (считывателя) и радиопередатчика (метки).

Данные идентификации хранятся на RFID-метке, которая размещается на объекте. Эта информация передается считывателю при помощи радиочастотной связи. Основной задачей RFID-системы является хранение информации об объекте с возможностью ее чтения и использования. На RFID-метке могут содержаться различные данные, такие как тип объекта, стоимость, вес, температура, логистическая информация и другие параметры, которые могут быть представлены в цифровом формате [1].

RFID-технология обладает широким спектром применения и имеет значительный потенциал для улучшения операций и процессов в различных отраслях благодаря своей высокой эффективности, точности и автоматизации идентификации объектов.

Такая система может помочь улучшить управление логистикой, обеспечить требуемый уровень безопасности станционных транспортных процессов, повысить эффективность погрузочно-разгрузочных операций и сократить количество ошибок и задержек.

Учитывая все вышесказанное, использование RFID-технологии для контроля местоположения тормозных башмаков на станционных путях представляется целесообразным. Преимущества применения RFID-меток в данном контексте включают в себя следующие технические характеристики:

— большой рабочий диапазон температур и устойчивость к их изменению (от –40 до +70 °С), что обеспечивает надежную работу меток в различных условиях;

— влагостойкость, что позволяет использовать RFID-метки даже в условиях повышенной влажности;

— отсутствие негативного влияния внешних электромагнитных полей на работоспособность меток;

— маленькие размеры меток, которые позволяют удобно размещать их на опорной колодке тормозного башмака без закрытия его номера с минимизацией механических повреждений;

— высокая автономность работы меток, обеспечивающая их долгосрочное функционирование (до 10 лет);

— высокая дальность приема-передачи сигнала меток на антенну, что обеспечивает эффективное отслеживание и контроль за местоположением тормозных башмаков на дистанции до 300 м.

Эти характеристики делают применение RFID-технологии в качестве средства контроля тормозных башмаков на станционных путях эффективным и надежным решением.

Внедрение данной системы позволит:

1. Снизить риск несанкционированного движения вагонов.
2. Обеспечить контроль корректности установки тормозных башмаков исполнителем.
3. Следить за перемещениями тормозных башмаков по территории.
4. Производить удаленный мониторинг операций в любое время из любого места.
5. Ускорить поиск и идентификацию тормозных башмаков.
6. Обеспечить быструю идентификацию и учет.
7. Исключить утерю тормозных башмаков.

## Состав RFID-системы

Автоматизированная RFID-система контроля хранения и перемещения тормозных башмаков включает в себя следующие взаимодействующие функциональные компоненты:

*1. RFID-метки, установленные на станционные тормозные башмаки* (рис. 1).

Для установки на тормозные башмаки рекомендуется использовать пассивные, прочные RFID-метки в корпусе, с рабочей частотой UHF (865–868 Мгц), устойчивые к воздействию окружающей среды и адаптированные для использования на металле.

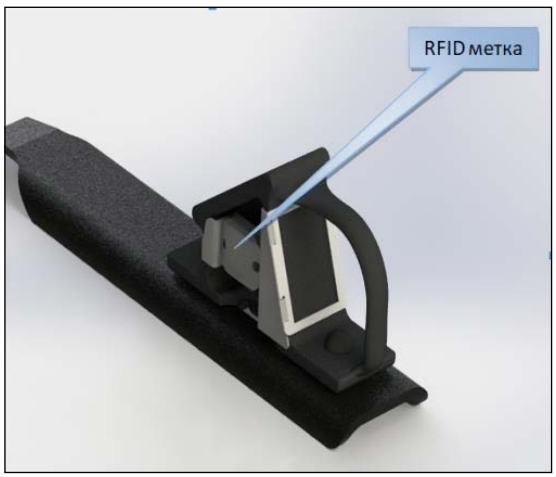



*Рис. 1.* Место установки RFID-метки на станционном тормозном башмаке

*2. RFID-ридеры:*

• в зоне стоек хранения устанавливаются стационарные считыватели с антеннами, диапазон считывания UHF (865–868 МГц);

• портативный RFID-считыватель — планшет составителя (рис. 2).

*Рис. 2.* RFID-планшет

Для полной автоматизации закрепления подвижного состава на планшет составителя необходимо установить программное обеспечением «Мобильное рабочее место» с доступом к функционалу закрепления подвижного состава. После проверки рассчитанных норм, ДСП из АРМ ЖУТБ на планшет составителя с ПО МРМ будет отправлять наряд-задание на установку закрепления состава.

*3. Антенны.*

Для считывания данных RFID-меток на станции должны быть установлены специальные антенны. Целесообразно применять анизотропные антенны с круговой поляризацией, адаптированные к российскому климату и приспособленные для работы с частотами UHF, которые выделены для применения в Российской Федерации. Это полоса частот 865–868 МГц.

## Серверное аппаратное и программное обеспечение

В базу данных на сервере вносится информация о каждом пути станции, где может производиться закрепление согласно ТРА станции (диапазон координат широты и долготы), номерах чипированных башмаков.

Собранные данные, сгенерированные RFID и хранящиеся на сервере, должны интегрироваться в существующий и эксплуатируемый на всей сети дорог АРМ ЖУТБ.

Принцип работы RFID-технологии мониторинга тормозных башмаков

На рис. 3 представлена диаграмма компонентов эксплуатации тормозных башмаков на железнодорожной станции с применением RFID-системы:

1. Получение башмака с центрального склада ДМТО с установленной RFID-меткой (сканирование кладовщиком RFID-метки башмака для автоматического заполнения электронных форм отчетности).

2. Передача башмака на станцию и его распределение на пункты эксплуатации с автоматическим заполнением электронных форм отчетности с помощью RFID-системы.

3. Изъятие башмака на замену или обновление окраса (ремонт).

4. Оформление башмака, вышедшего из строя, в металлолом с помощью RFID-системы для заполнения электронных форм.

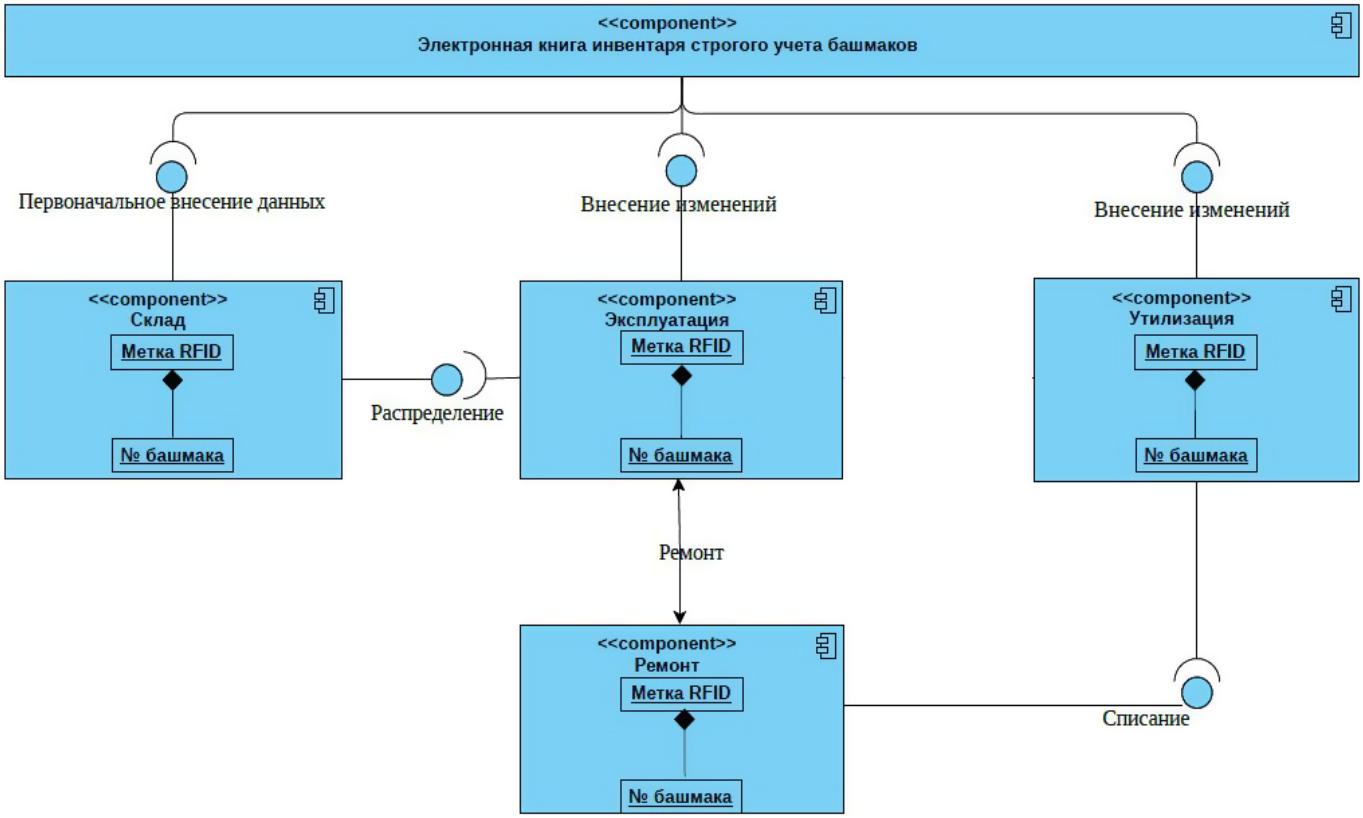



*Рис. 3.* Диаграмма компонентов эксплуатации тормозного башмака на железнодорожной станции

На рис. 4 рассмотрим диаграмму алгоритма действия ДСП и составителя при закреплении подвижного состава на железнодорожной станции с применением RFID-технологии.

В АРМ ЖУТБ по регламенту из АСОУП поступает информация о фактическом местоположении поездов и группы вагонов на железнодорожной станции; выполняется автоматический расчет необходимого количества тормозных башмаков на основании данных, получаемых из ТРА ИСУЖТ НС в зависимости от порядка закрепления (группа вагонов, состав поезда, место установки подвижного состава). ДСП выполняет контроль рассчитанной нормы закрепления с четной и нечетной стороны и передает команду о закреплении состава на планшет составителя. Далее составитель подтверждает полученную команду и выполняет операцию закрепление состава. Считывает RFID-метку с установленного башмака с помощью RFID-ридера и подтверждает операцию о закреплении состава.

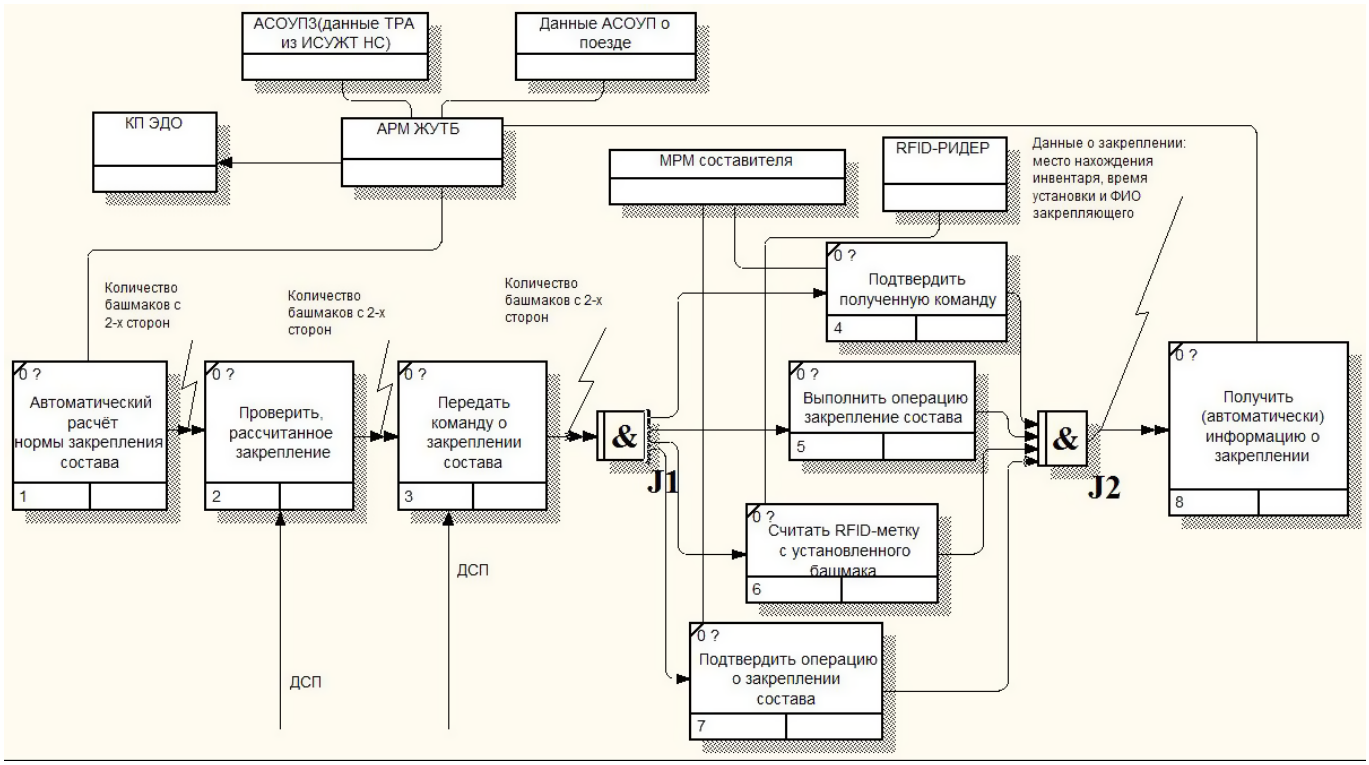



*Рис. 4.* Диаграмма декомпозиции действия «Закрепление подвижного состава на железнодорожной станции с применением RFID-технологии» в нотации IDEF3

В АРМ ЖУТБ по регламенту из АСОУП поступает информация о фактическом местоположении поездов и группы вагонов на железнодорожной станции; выполняется автоматический расчет необходимого количества тормозных башмаков на основании данных, получаемых из ТРА ИСУЖТ НС в зависимости от порядка закрепления (группа вагонов, состав поезда, место установки подвижного состава). ДСП выполняет контроль рассчитанной нормы закрепления с четной и нечетной стороны и передает команду о закреплении состава на планшет составителя. Далее составитель подтверждает полученную команду и выполняет операцию закрепление состава. Считывает RFID-метку с установленного башмака с помощью RFID-ридера и подтверждает операцию о закреплении состава.

Таким образом, в АРМ ЖУТБ автоматически поступает информация о закреплении (место нахождения инвентаря, время установки и ФИО закрепляющего); статус установленных башмаков в ПО меняется на «занят». В конце смены формируется электронный документ ЖУТБ; его подписывают принимающий смену ДСП и сдающий с применением ПЭП. Далее ЭД поступает на хранение в КП ЭДО.

## Заключение

Внедрение RFID-технологий полностью автоматизирует процесс контроля хранения и перемещения станционных тормозных башмаков.

Задачи, решаемые при внедрении RFID-системы контроля хранения и перемещения станционных тормозных башмаков:

- автоматизация и цифровизация станционных технологических процессов;
- обеспечение требуемого уровня безопасности движения поездов;
- предотвращение и выявление прецедентов нарушения порядка и норм закрепления вагонов;
- обеспечение контроля сохранности инвентаря строгого учета;
- снижение трудозатрат и накладных расходов;
- снижение риска несанкционированного движения вагонов;
- оперативность передачи информации.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Григорьев П. В. Особенности технологии RFID и ее применение // Молодой ученый. 2016. № 11 (115). С. 317–322 [Электронный ресурс]. URL: https://moluch.ru/archive/115/30692/ (дата обращения: 12.03.2024).

2. Гудин М., Зайцев В. Технология RFID: реалии и перспективы // Компоненты и технологии. 2003. № 4. С. 42–44.

3. Бхуптани М., Морадпур Ш. RFID-технологии на службе вашего бизнеса // Альпина Диджитал, 2011. 350 с.

4. Финкенцеллер К. RFID-технологии. М.: ДМК Пресс, Додэка XXI, Hanser Publishers, 2016. 490 с.

5. Власов М. RFID: 1 технология – 1000 решений. Практические примеры использования RFID в различных областях. М.: Альпина Паблишер, 2015. 218 с.

6. RFID – радиочастотная идентификация // ДатаКрат [Электронный ресурс]. URL: http://www.datakrat.ru/technology/7942.html (дата обращения: 11.01.2024).

7. Распоряжение ОАО «РЖД» от 06.06.2020 № 1440-р «Об утверждении Порядка по оформлению и подписанию Журнала учета тормозных башмаков, применяемых для закрепления железнодорожного подвижного состава, подписанных электронной подписью». Утверждено зам. генерального директора Р. Ф. Сайбаталовым. 6 с.

Дата поступления: 20.03.2024
Решение о публикации: 03.06.2024

# Automated Control of the Movement of Brake Shoes in Railway Transport: the Use of RFID Technology in Securing Rolling Stoc

**Irina G. Kagady** — Master's Degree Student.
E-mail: irina.kagadiy74@mail.ru

**Sergey G.Ermakov** — Doctor of Technical Sciences, Professor, Head of the Department Information and Computing Systems.
E-mail: ermakov@pgups.ru

Emperor Alexander I Petersburg State Transport University, Saint Petersburg, Russia

**For citation:** Kagady I. G., Ermakov S. G. Automated control of the movement of brake shoes in railway transport: the use of RFID technology in securing rolling Stoc // Intelligent technologies on transport. 2024. No. 2 (38). P. 77–83. (In Russian). DOI: 10.20295/2413-2527-2024-238-77-83

***Abstract.*** *The possibility of introducing RFID technology in the work of railway transport is considered as one of the aspects of digitalization of the railway. The general principle of operation of this technology in securing rolling stock is described. The positive aspects of the introduction of this technology are analyzed.*

***Keywords:*** *RFID, movement control, composition fixation, brake shoes, digitalization.*

**REFERENCES**

1. Grigor'ev P. V. Osobennosti tehnologii RFID i ee primenenie // Molodoj uchenyj. 2016. № 11 (115). S. 317–322 [Jelektronnyj resurs]. URL: https://moluch.ru/archive/115/30692/ (data obrashhenija: 12.03.2024). (In Russian)
2. Gudin M., Zajcev V. Tehnologija RFID: realii i perspektivy // Komponenty i tehnologii. 2003. № 4. S. 42–44. (In Russian)
3. Bhuptani M., Moradpur Sh. RFID-tehnologii na sluzhbe vashego biznesa // Al'pina Didzhital, 2011. 350 s. (In Russian)
4. Finkenceller K. RFID-tehnologii. M.: DMK Press, Dodjeka XXI, Hanser Publishers, 2016. 490 c. (In Russian)
5. Vlasov M. RFID: 1 tehnologija – 1000 reshenij. Prakticheskie primery ispol'zovanija RFID v razlichnyh oblastjah. M.: Al'pina Pablisher, 2015. 218 c. (In Russian)
6. RFID – radiochastotnaja identifikacija // DataKrat [Jelektronnyj resurs]. URL: http://www.datakrat.ru/technology/7942.html (data obrashhenija: 11.01.2024). (In Russian)
7. Rasporjazhenie OAO "RZhD" ot 06.06.2020 № 1440-r "Ob utverzhdenii Porjadka po oformleniju i podpisaniju Zhurnala ucheta tormoznyh bashmakov, primenjaemyh dlja zakreplenija zheleznodorozhnogo podvizhnogo sostava, podpisannyh jelektronnoj podpis'ju". Utverzhdeno zam. general'nogo direktora R. F. Sajbatalovym. 6 s. (In Russian)

Received: 20.03.2024
Accepted: 03.06.2024