DOI 10.37882/2223–2966.2021.10.14

ПРИМЕНЕНИЕ ПОДХОДОВ ОПТИМИЗАЦИИ АРХИТЕКТУРЫ ИНФОРМАЦИОННЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ОБРАТНОЙ СВЯЗИ ОТ ПОЛЬЗОВАТЕЛЕЙ В КОНТЕКСТЕ СООТВЕТСТВИЯ ТРЕБОВАНИЯМ СТАНДАРТА TOGAF

Насыров Наиль Фаизович
Аспирант, Университет ИТМО
pasdel@mail.ru
Лушников Даниил Михайлович
Университет ИТМО
MrLushnikov@yandex.ru
Тартынских Петр Сергеевич
Университет ИТМО
tartynskikh.ps@yandex.ru
Бережков Андрей Вячеславович
Преподаватель, Университет ИТМО
dead0343@gmail.com

Аннотация. В работе рассматриваются аспекты оптимизации архитектуры существующей информационной системы в соответствии с положениями методологии TOGAF. В работе перечислены факторы, которые обусловили необходимость оптимизации разработанного прототипа сервиса автоматизированного нормоконтроля. В результате проведенного GAP-анализа 4 уровней архитектуры, соответствующих методологии TOGAF, были сформулированы предложения, в результате реализации которых достигнуты такие показатели, как снижение количества обращений к серверу, уменьшение количества вычислительных операций на стороне сервера, повышена точность определения классов элементов текстовых документов, добавлена возможность анализировать результаты обратной связи пользователей. Оптимизированная архитектура сервиса предоставляет возможность масштабирования и дальнейшего развития сервиса автоматизированного нормоконтроля.

Ключевые слова: оптимизация программного обеспечения, архитектура информационной системы, TOGAF, The Open Group Architecture Framework, нормоконтроль, оформление документа.

APPLICATION OF APPROACHES TO OPTIMIZING THE ARCHITECTURE OF INFORMATION SYSTEMS USING USER FEEDBACK IN THE CONTEXT OF COMPLIANCE WITH THE REQUIREMENTS OF THE TOGAF STANDARD

N. Nasyrov
D. Lushnikov
P. Tartynskikh
A. Berezhkov

Summary. The article examines the aspects of optimization of the existing system's architecture in accordance with the TOGAF Standard. The factors that justified the necessity of the optimization of the developed prototype of the automated standard verification service are listed in the article. As a result of the GAP- analysis of the four levels of architecture according to the TOGAF Standard, a number of proposals was formulated. The proposed features' realization led to the decline in the number of server requests, the decline in the number of computational operations on the server side and the increase in the accuracy of the text documents' class prediction. The feature of the users' feedback analysis was also successfully implemented. The optimized architecture of the service provides opportunities for scaling and further development of the standard verification service.

Keywords: software optimization, information system architecture, TOGAF, The Open Group Architecture Framework, verification for compliance with regulations, document formatting.

Введение

Способность меняться является в настоящее время одним из ключевых факторов обеспечения эффективного управления информационной системой [1]. Для реализации возможности непрерывных изменений требуется четкое структурирование на уровне бизнес-процессов, информационных потоков, прикладных и технологических решений. Учет особенностей архитектур на каждом из уровней позволяет проектировать и создавать информационные системы, удовлетворяющие потребности организаций и предприятий различного уровня. В работе рассматриваются аспекты применения подходов оптимизации архитектуры информационных систем с использованием обратной связи от пользователей в контексте соответствия требованиям стандарта TOGAF.

Стандарт (методология) TOGAF (The Open Group Architecture Framework) представляет собой фрейм-

ворк, описывающий построение модели деятельности организации и, соответственно, инфокоммуникационных сервисов от абстракций более высокого уровня к более конкретным и детализированным аспектам.

Использование стандарта TOGAF помогает решать следующие задачи:

- описание модели архитектуры с учетом зависимостей между ИТ-сервисами и бизнес-процессами,
- оптимизация информационных потоков на предприятии/подразделении,
- снижение стоимости владения и сопровождения ИТ-инфраструктуры и др.

Архитектура предприятия в модели TOGAF подразделяется на следующие уровни [2]:

1. *бизнес-архитектура* — описывает организационную структуру, бизнес-процессы и деятельность организации с учетом заданных показателей эффективности,
2. *информационная архитектура* — определяет информационные потоки процессов организации, методы сбора, обработки, предоставления и хранения данных,
3. *архитектура приложений* — определяет, какие приложения используются и должны использоваться для управления данными и поддержки бизнес-функций, реализует объекты информационной архитектуры на уровне прикладных систем,
4. *техническая архитектура* — описывает, какие аппаратные и программные средства необходимы для обеспечения работоспособности всей совокупности прикладных систем в соответствии с операционными требованиями (надежность, производительность и т.п.).

В контексте приведенных четырех уровней архитектуры необходимость оптимизации существующей архитектуры прототипа информационной системы была обусловлена, в частности, следующими факторами:

- при разработке прототипа не было предусмотрено, что пользователи сервиса могут выступать в роли асессоров, подтверждающих или опровергающих предлагаемый класс элемента (абзац, заголовки разных уровней, перечисления и т.д.) текстового документа, определенный с помощью алгоритмов машинного обучения,
- необходимостью добавления функций автоматизированного исправления обнаруженных ошибок оформления элементов текстового документа,
- необходимостью снижения вычислительной нагрузки на сервер за счет переноса части операций на сторону надстройки (add-in) для программного обеспечения Microsoft Word,
- выявленными особенностями и закономерностями в оформлении элементов текстовых документов пользователями. В частности, было отмечено, что в большинстве случаев у пользователя прослеживаются идентичные ошибки в оформлении элементов текстового документа одного класса.

Таким образом, стоит отметить, что перед началом работ по оптимизации архитектуры информационной системы с целью масштабирования и добавления новых функциональных возможностей необходимо предметно рассмотреть каждый из 4 уровней, описанных во фреймворке TOGAF.

Анализ исходного состояния архитектуры

В основу разработки системы автоматизированной проверки соответствия требованиям оформления была положена клиент-серверная архитектура. Приложение-надстройка для программного обеспечения Microsoft Word взаимодействует с сервером, предоставляющим функциональные возможности по классификации элементов текстовых документов и проверке их оформления посредством RESTful API. Подробные сведения о сервисе и его исходной архитектуре представлены в работах [3, 4].

Функциональные возможности по классификации элементов полученного docx-файла, отображению ошибок реализованы в едином программном модуле, что может создавать некоторые трудности [5]:

- увеличивается время на развертывание системы и внесение изменений,
- сильная связь кода делает разработчиков зависимыми друг от друга, сбой в работе одного компонента приводит к отказу всего модуля,
- затруднено масштабирование приложений и др.

Риски, связанные с существованием указанных недостатков, могут быть устранены путем деления приложений на более мелкие программные составляющие, что привело к возникновению способа проектирования и организации информационной архитектуры и бизнес-функциональности с использованием сервис-ориентированной архитектуры [6] и микросервисной архитектуры в частности [5].

В исходной версии прототипа сервиса предполагалось использовать его только для полной проверки всего docx-документа, а также вывода на экран результатов проверки. Схема проверки документа представлена на рисунке 1.

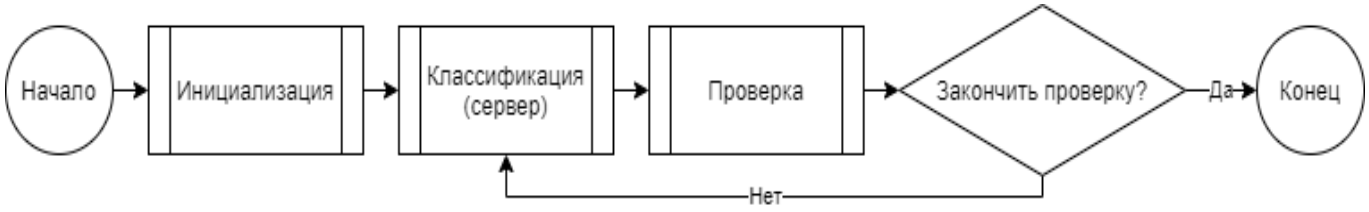

Рис. 1. Проверка документа (для прототипа)

Представленный алгоритм подтвердил свою работоспособность, однако он имеет ряд особенностей, которые могут ограничивать эксплуатацию сервиса, а также его дальнейшее масштабирование и адаптацию под новые требования пользователей:

- большая часть вычислительных операций осуществляется на стороне сервера,
- при каждой отправке документа на сервере проводилась классификация элементов текстового документа с использованием алгоритмов машинного обучения, что являлось избыточным в случае повторной проверки документа,
- при каждой новой классификации элементов docx-файлов не учитывались закономерности оформления соответствующих элементов, которые были классифицированы ранее,
- у пользователя отсутствует возможность проверить только необходимые ему фрагменты документа, что создает избыточную потребность в вычислительных операциях на стороне сервера.

Анализ слоев бизнес-архитектуры и информационной архитектуры

Бизнес-архитектура, как правило, первична по отношению к ИТ-архитектуре. Она определяет требования к общей архитектуре, является основой для оценки ее эффективности. В то же время новые возможности информационных технологий, программного и аппаратного обеспечения, а также появляющиеся требования к бизнес-процессам могут стимулировать развитие как бизнес-архитектуры, так и архитектуры информационной, приложений, технической.

При интерпретации и трансляции аспектов бизнес-архитектуры на более низкие уровни необходимо учитывать следующие факторы:

- в ходе функционирования информационных систем в большинстве случаев происходит формирование новых требований к функциональным возможностям,
- в ряде случаев существуют объективные причины, по которым нецелесообразно реализовывать все функциональные возможности до завершения апробации разработанного прототипа,
- часть решений о дальнейшем развитии системы можно сформулировать, только имея данные, полученные эмпирическим путем в ходе функционирования системы.

С целью определения разрывов между состоянием созданного прототипа и спрогнозированным результатом был проведен GAP-анализ, который позволил выявить ряд специфических аспектов в контексте осуществления автоматизированной проверки оформления текстовых документов.

Было выявлено, что в большинстве случаев у пользователя прослеживаются идентичные ошибки в оформлении элементов текстового документа одного класса (заголовков одного уровня, подписи иллюстраций, абзацев, перечислений и т.д.). С учетом необходимости реализации функциональных возможностей по автоматизированному исправлению ошибок в соответствии с предъявляемыми к их оформлению требованиями отмечена необходимость оптимизации алгоритмов с целью уменьшения вычислительной нагрузки по классификации элементов и определению ошибок их оформления.

Классификация элементов docx-файлов осуществляется с использованием алгоритмов машинного обучения библиотеки CatBoost, для которой разработчиками сервиса был сформирован соответствующий датасет. В ходе анализа было выявлено, что пользователи сервиса могут выступать в роли асессоров, подтверждающих или опровергающих предлагаемый класс элемента текстового документа, который был определен с помощью алгоритмов машинного обучения CatBoost. Это дает возможность, с одной стороны, формировать датасет на основе анализа пользовательских данных для последующего переобучения моделей, а с другой стороны — формировать частные случаи некорректной классификации элементов для последующего анализа и оптимизации алгоритмов без изменения архитектуры информационной системы.

Также следует учитывать, что оформление элементов текстовых документов может удовлетворять предъявляемым требованиям, соответственно, опе-

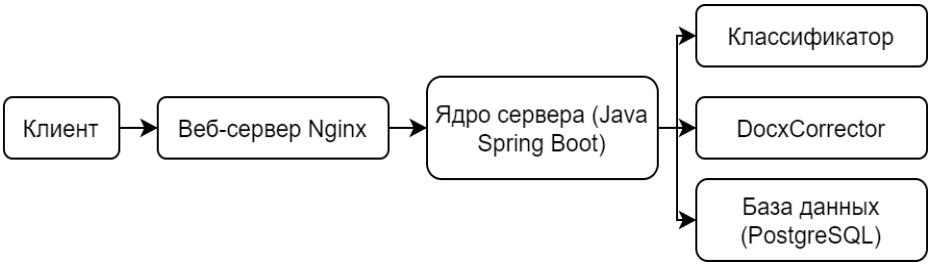

Рис. 2. Схема клиент-серверного приложения

рации по классификации таких элементов, проверке корректности оформления и исправлению ошибок избыточны.

В ходе расширения функциональных возможностей сервиса необходимо учитывать, что решения об исправлении ошибок принимаются на стороне клиента. В силу этого необходимо процесс нормоконтроля декомпозировать на три этапа — определение класса элемента, определение ошибок оформления (по запросу, если пользователь согласен с определенным классом), исправление ошибки для одного или всех представителей этого класса (также по запросу), что снимает необходимость отправки данных на сервер.

Оптимизация архитектуры приложений и технической архитектуры

Основа разрабатываемого сервиса — клиент-серверный архитектурный стиль. Он позволяет разделять логику сервиса на две части, сторону которая будет находится у клиента (надстройка в Microsoft Word), и сторону сервера, где реализуется логика классификации элементов, проверки корректности оформления элементов, формирование перечня ошибок оформления для отображения на стороне клиента. Особенности реализации с учетом потребностей технической архитектуры представлены на рисунке 2.

Веб-сервер Nginx используется для распределения запросов от клиента и при необходимости исполняющий роль балансировщика, для реализации описанных ранее функциональных возможностей используются серверы микросервисов и сервер базы данных. Таким образом архитектура поддерживает возможность дальнейшего горизонтального масштабирования.

При проектировании микросервисной архитектуры информационной системы для организации взаимодействия микросервисов использовался подход «API Gateway». Для клиентов API Gateway является единой точкой взаимодействия с микросервисами, обеспечивая перенаправление запросов клиента к конкретному сервису. В рассматрвиаемом случае данным интерфейсом выступает модуль, написанный на Java Spring Boot. В рамках микросервисной архитектуры микросервисы не обмениваются сообщениями между собой, т.е. не взаимодействуют напрямую. Это исключает влияние качества обслуживания одного микросервиса на другой. Следует отметить, что даже при отказе одного или нескольких микросервисов другие микросервисы продолжают функционировать (это обеспечивается свойством слабой связанности микросервисов). При этом качество функционирования информационной системы снижается, но отказа всей системы не происходит [7].

На выбор описанных решений повлияли следующие преимущества микросервисного архитектурного стиля:

- независимость — каждый микросервис может быть развернут на отдельной физической или виртуальной машине, чтобы иметь собственную распределенную архитектуру,
- масштабируемость — благодаря независимости упрощается как горизонтальное, так и вертикальное масштабирование системы,
- каждый микросервис может обновляться и обслуживаться независимо,
- каждый микросервис может быть разработан в соответствии с языком программирования, знакомым команде разработчиков, и затем предоставлять API в соответствии с определённым протоколом (например, REST).

В связи с этим было решено оптимизировать архитектуру сервиса и разделить проверку всего документа на несколько функций проверки отдельных его частей, которые будут доступны пользователю. Нужно отметить, что функция полной проверки по прежнему подлежит реализации, однако в общем виде она будет представлять из себя последовательный вызов функций классификации, проверки корректности отдельных элементов документа. На рисунке 3 представлена актуализированная обобщённая диаграмма активности процесса проверки форматирования элементов docx-файла.

Рис. 3. Проверка и исправление форматирования элементов

Далее рассматриваются этапы, наиболее подвергшиеся изменениям с учетом реализации функции обратной связи от пользователей в контексте соответствия требованиям стандарта TOGAF.

На этапе 2 происходит первичная классификация элементов без использования алгоритмов машинного обучения.

Авторами были определены три группы шаблонов оформления (пресетов):

- пресеты классов элементов, оформленные с настройками «по умолчанию» средствами Microsoft Word,
- пресеты, соответствующие корректному оформлению элементов согласно требований ГОСТ 7.32–2017 и ГОСТ 7.0.11–2011,
- пресеты на основе анализа закономерностей оформления документов конкретным пользователем.

Так, например, в ходе исследования было отмечено, что элемент «обычный абзац» в 24–28% оформляется в соответствии с требованиями ГОСТ 7.32–2017, а в 6–8% случаев для него применяется стандартное оформление Microsoft Word. В рамках реализации архитектуры приложений пользователю предлагается дальнейшая работа с этими элементами без необходимости классификации средствами машинного обучения, что снижает вычислительную нагрузку на стороне сервера. Точность классификации элементов на основе пресетов достигала в ряде случаев 92 процентов.

На этапе 3 пользователю предоставляется возможность самостоятельно определить класс или предоставить возможность определить класс с помощью классификатора с использованием алгоритмов машинного обучения CatBoost [3].

Наиболее существенные изменения в архитектуре сервиса были связаны с проектированием и реализацией интерактивно-диалогового режима с пользователем (этап 4), что предоставило возможность логирования ответов пользователя, касающихся корректности определения классов элементов текстовых документов, отображения ошибок оформления, автоматизированного исправления как текущего элемента, так и последующих, имеющих идентичное оформление. Общая схема взаимодействия с пользователем представлена на рисунке 4.

В результате были добавлены следующие возможности, влияющие на общую производительность и эффективность системы:

- указание вручную класса элемента docx-файла, как в результате классификации элементов, так и на усмотрение пользователя,
- автоматическое исправление оформления текущего и последующих элементов с идентичным оформлением,
- игнорирование ошибок оформления на усмотрение пользователя, что, с одной стороны, добавило гибкости системе, с другой — уменьшило количество обращений к серверу повторных вопросов на стороне клиента.

Заключение

В результате анализа исходной архитектуры сервиса в контексте ее оптимизации в соответствии с положениями методологии TOGAF, предметного рассмотрения уровней архитектуры и последующей реализации разработанных алгоритмов были получены следующие результаты:

- количество элементов, классифицируемых на сервере, уменьшилось на 64%,
- точность определения классов элементов увеличилась для некоторых элементов docx-файлов до 92–94%,
- Результаты взаимодействия пользователя с системой посредством разработанного интерактивно-диалогового режима, позволяют увеличивать набор данных для последующего формирования датасета для переобучения классификатора, а также фиксировать случаи некорректной работы классификатора,
- реализована возможность автоматизированного исправления ошибок оформления элементов текстовых документов.

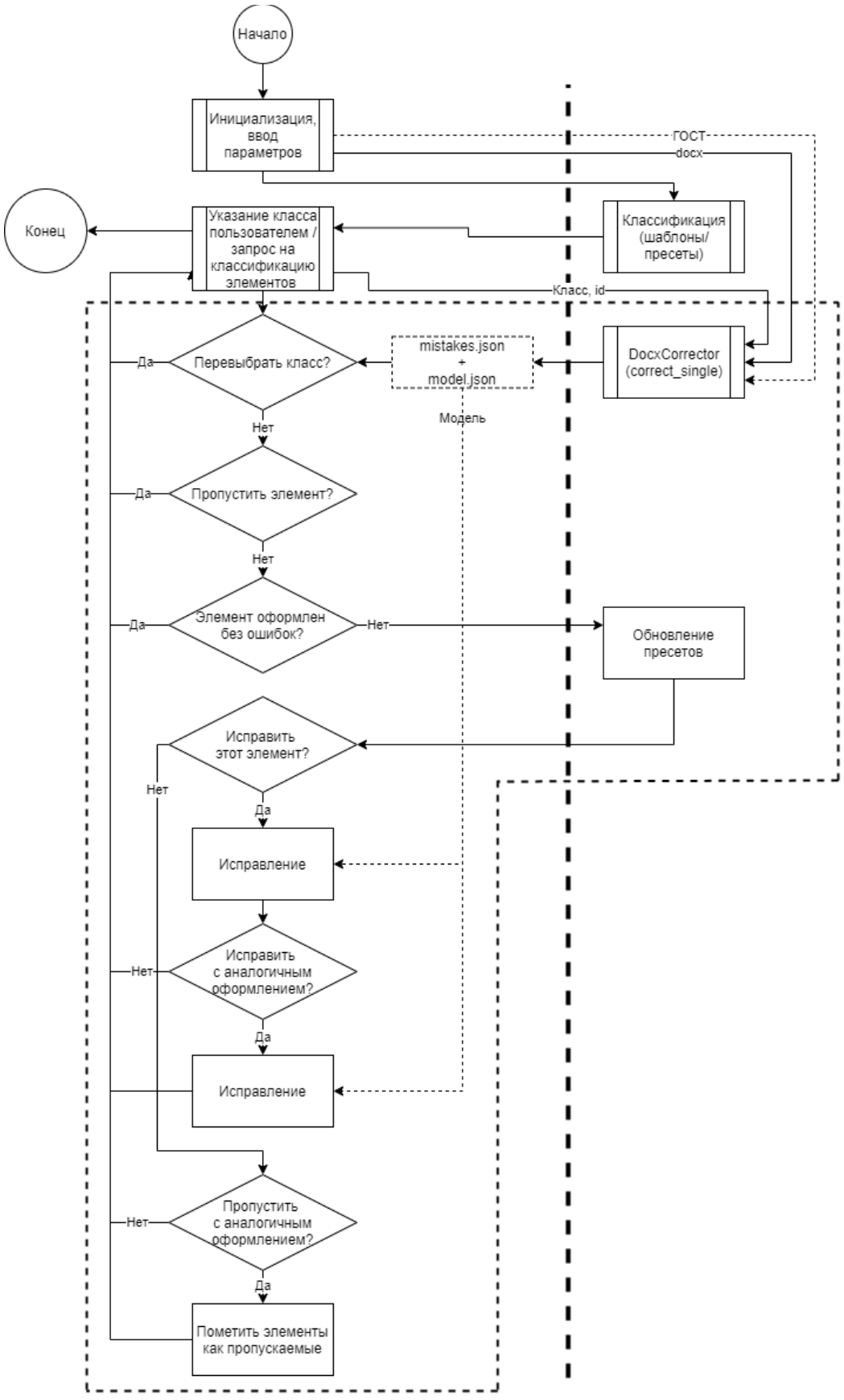

Рис. 4. Проверка и исправление элементов текстовых документов

Положения, рассмотренные в исследовании, соответствуют результатам, полученным в ходе оптимизации существующей архитектуры прототипа сервиса автоматизированного нормоконтроля путем применения принципов методологии TOGAF. Практическая значимость исследования определяется эмпирическими выводами, полученными в ходе реализации проекта, готового к масштабированию и дальнейшему сопровождению.

ЛИТЕРАТУРА

1. Кунцман А.А. Построение эффективной архитектуры предприятия как необходимое условие адаптации к цифровой экономике // Вопросы инновационной экономики. — 2018. — Том 8. — № 4. — С. 753–770. doi: 10.18334/vinec.8.4.39477.
2. Логиновский, О.В. Применение методов архитектурного подхода в развитии информационной системы крупного вуза / О.В. Логиновский, М.И. Нестеров, А.Л. Шестаков // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. — 2013. — Т. 13. — № 4. — С. 123–128.
3. Nail Nasyrov, Mikhail Komarov, Petr Tartynskikh, Nataliya Gorlushkina. Automated formatting verification technique of paperwork based on the gradient boosting on decision trees // Procedia Computer Science, Volume 178, 2020, pp. 365–374, ISSN1877–0509. https://doi.org/10.1016/j.procs.2020.11.038.
4. Кобец Е.А., Насыров Н.Ф., Тартынских П.С., Горлушкина Н.Н. Построение объектной модели эталонного текстового документа для сервиса автоматизированного нормоконтроля // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. — 2021. — № 8. С. 57–63. DOI 10.37882/2223–2966.2021.08.14.
5. Гольчевский, Ю.В. Актуальность использования микросервисов при разработке информационных систем / Ю.В. Гольчевский, А.В. Ермоленко // Вестник Сыктывкарского университета. Серия 1: Математика. Механика. Информатика. — 2020. — № 2(35). — С. 25–36.
6. Платонов, Ю.Г. Анализ перспектив перехода информационных систем на сервисно-ориентированную архитектуру / Ю.Г. Платонов // Проблемы информатики. — 2011. — № 4(12). — С. 56–65.
7. Долженко, А.И. Нечеткая продукционная сеть для анализа качества микросервисной архитектуры / А.И. Долженко, И.Ю. Шполянская, С.А. Глушенко // Бизнес-информатика. — 2020. — Т. 14. — № 4. — С. 36–46. — DOI 10.17323/2587–814X.2020.4.36.46.

© Насыров Наиль Фаизович (pasdel@mail.ru), Лушников Даниил Михайлович (MrLushnikov@yandex.ru),
Тартынских Петр Сергеевич (tartynskikh.ps@yandex.ru), Бережков Андрей Вячеславович (dead0343@gmail.com).
Журнал «Современная наука: актуальные проблемы теории и практики»

ИТМО