МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОЙ БИЗНЕС-РАЗМЕТКИ РАСПРЕДЕЛЁННЫХ ТРЕЙСОВ НА ОСНОВЕ OPENTELEMETRY

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса ПИ-551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Сунчаляева Мунира Раисовича

Научный руководитель
доцент, к. ф.-м. н. _____ М. И. Сафрончик

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. В микросервисных системах обработка одного запроса проходит через десятки сервисов. Для наблюдения за такими вызовами применяется распределённая трассировка по стандарту OpenTelemetry: каждый сервис записывает спан с длительностью и атрибутами, а цепочка спанов с общим идентификатором описывает путь запроса по системе.

Однако трейс отвечает только на технический вопрос — какие сервисы и в каком порядке участвовали в обработке. Он не показывает, какому шагу бизнес-процесса соответствует каждый вызов: один и тот же спан $\mathrm{POST}\ /\mathrm{orders}$ может означать «оформление заказа» или «повторную проводку платежа». Без этой связи нельзя ни сверить исполнение с эталонной моделью процесса, ни обнаружить новые сценарии, а заполнять её вручную дорого.

Существующие платформы Process Mining (Celonis, Apromore) работают поверх готовых журналов событий и слабо интегрируются с трассировкой, а инструменты просмотра трейсов (Jaeger, Tempo) такой разметки не дают. Этот разрыв между технической наблюдаемостью и бизнес-аналитикой становится серьёзным барьером для автоматизации управления процессами, особенно в условиях растущей сложности систем. Поэтому создание сквозного решения, которое автоматически связывает технические спаны с шагами бизнес-процесса, сверяет выполнение с эталонной BPMN-моделью и обнаруживает новые или аномальные сценарии, является актуальной задачей.

Объект исследования — процесс распределённой трассировки запросов в микросервисных системах и его связь с моделями бизнес-процессов.

Предмет исследования — методы автоматического сопоставления технических спанов OpenTelemetry с шагами бизнес-процесса, сверки реального исполнения с эталонной BPMN-моделью и обнаружения новых сценариев.

Цель выпускной квалификационной работы — спроектировать и реализовать систему, которая автоматически связывает технические спаны с шагами бизнес-процесса по декларативным правилам разметки и сверяет результат с эталонной BPMN-моделью.

Поставленная цель определила следующие задачи:

— проанализировать существующие подходы к связыванию трейсов с бизнес-процессом и выявить их ограничения;

— предложить модель правила разметки и алгоритм сопоставления спана с

правилами;

- спроектировать архитектуру с минимальным вмешательством в наблюдаемые сервисы;
- реализовать прототип с пользовательским интерфейсом для разметки, сверки соответствия и обнаружения новых сценариев;
- провести тестирование решения и количественно оценить его ключевые свойства.

Методологические основы анализа процессов и сверки соответствия представлены в работах W. M. P. van der Aalst и A. Rozinat, A. Adriansyah, A. Berti, L. Reinkemeyer, A. H. Зуевой и И. Ю. Каневой; вопросы построения микросервисной архитектуры рассмотрены в работе А. Н. Баланова, а архитектурного описания программных систем — в работе S. Brown. В работе использованы методы контент-анализа научной и технической литературы, объектно-ориентированное моделирование, архитектурное проектирование по модели C4 и инструментальные эксперименты в контейнеризованной среде.

Теоретическая значимость работы состоит в формализации модели правила разметки в виде тройки «шаблон – контекст – действие» с приоритетом и схемы обогащения трейсов на чтение, которая разделяет техническую и бизнес-семантику наблюдаемости без модификации источников телеметрии.

Практическая значимость работы состоит в создании воспроизводимого, полностью контейнеризованного решения, которое встраивается в инфраструктуру OpenTelemetry без изменения наблюдаемых сервисов и предоставляет командам эксплуатации и бизнес-аналитикам единый инструмент разметки трейсов, сверки соответствия и обнаружения дрейфа процесса. Предложенная схема обогащения на чтение обеспечивает мгновенное действие правок разметки на всю историю трейсов. Демонстрационный стенд и конфигурации могут использоваться в учебном процессе для иллюстрации современных технологий наблюдаемости и анализа процессов.

Структура и объём работы. Бакалаврская работа состоит из введения, четырёх разделов, заключения, списка использованных источников и четырёх приложений. Общий объём работы — 56 страниц, из них 51 страница — основное содержание, включающее 21 рисунок и 6 таблиц; список использованных источников информации насчитывает 26 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Анализ проблемы и обзор существующих подходов» посвящён постановке задачи. Вводится модель данных OpenTelemetry: трейс описывает выполнение одного запроса и идентифицируется $\mathrm{trace\_id}$, каждый шаг обработки оформляется как спан с атрибутами и ссылкой на родителя, а совокупность спанов одного трейса образует дерево, которое визуализируется «водопадом задержек». Показывается, что стандартный трейс фиксирует только техническую сторону выполнения — имена сервисов и операций — и не содержит бизнес-смысла шагов; соответствие технических полей спана и характеристик, интересующих бизнес-аналитика (идентификатор бизнес-кейса, название шага, бизнес-поток, факт совпадения с моделью), сведено в отдельную таблицу. Подчёркивается ключевая трудность: одно и то же техническое имя операции (например, $\mathrm{POST}\ /\mathrm{orders}$) в разных бизнес-сценариях может означать разные шаги процесса, что порождает ложные корреляции при автоматическом сопоставлении.

Рассматриваются базовые понятия анализа процессов (Process Mining): журнал событий, активность, кейс, вариант, эталонная модель в нотации BPMN и сверка соответствия с показателями *fitness* и *precision*; описаны два метода сверки — проигрывание по фишкам и выравнивание — и стандартная для Python библиотека PM4Py. Отмечается принципиальное наблюдение: журнал событий, требуемый для анализа процессов, — это бизнес-журнал, тогда как сырые трейсы содержат технические события; преобразование «трейс $\rightarrow$ бизнес-журнал» и составляет содержательное ядро работы.

Проанализированы пять подходов к связыванию трейсов с бизнес-процессом: ручное чтение трейсов (не масштабируется), запись бизнес-меток на стороне сервиса (требует вмешательства в код всех сервисов), коммерческие платформы анализа процессов (дороги, закрыты, не используют инфраструктуру OpenTelemetry), свободные библиотеки поверх логов (требуют собственных адаптеров) и обогащение на чтение. Подходы сопоставлены по критериям вмешательства в наблюдаемые сервисы, влияния правок, интеграции с OpenTelemetry и стоимости. На основе анализа сформулированы требования к системе: функциональные (обнаружение повторяющихся шаблонов, декларативная разметка, поддержка контекста потока, обогащение исторических трейсов, обнаружение дрейфа, сверка с BPMN, ведение версий моделей, ролевой интерфейс), нефункциональные (ми-

нимальное вмешательство, единая точка интеграции, накладные расходы не более 50 %, масштабируемость до 10^6 спанов, запуск одной командой) и ограничения прототипа. *Вывод по разделу:* ни один из существующих подходов не сочетает невмешательство в сервисы, мгновенное действие правок и естественную интеграцию с инфраструктурой OpenTelemetry; этим обоснован выбор схемы обогащения на чтение для системы, получившей рабочее название Business-Aware Tracing (BAT).

Второй раздел «Проектирование системы» содержит основной авторский вклад в части архитектуры. Сформулированы пять архитектурных принципов: невмешательство в наблюдаемые сервисы (единственная точка интеграции — OpenTelemetry Collector, к которому добавляется один экспортёр копии спанов); обогащение на чтение (сырые спаны не изменяются, бизнес-метки вычисляются на каждый запрос по текущему набору правил); разделение сигнатуры спана и контекста потока; контракт-первичность (HTTP-API описываются в OpenAPI, интерфейсы и DTO генерируются); выбор стека языков по задаче. Ключевые решения зафиксированы записями ADR с явным указанием размена (ClickHouse как источник истины для сырых спанов — быстрые агрегации ценой отсутствия транзакционности; Postgres для правил и моделей — транзакционность и индексы JSONB; сигнатура как хэш цепочки родителей — устойчивость к перепорядку спанов). Архитектура описана по модели C4 на уровнях контекста, контейнеров и компонентов; выделены две роли — бизнес-аналитик и разработчик-наблюдатель.

Предложена *модель правила* в виде тройки $\langle match,\ context,\ assign \rangle$ с целочисленным приоритетом, а *сигнатура спана* определена как

$$signature(s) = \mathrm{SHA256}\big(parent\_chain(s) \parallel service(s) \parallel span\_name(s)\big),$$

где $parent\_chain(s)$ — упорядоченная от корня цепочка предков спана s. Разработан алгоритм сопоставления: отбор кандидатов по сигнатуре, проверка контекстных предикатов, при пустом множестве — метка $\mathrm{UNMATCHED}$, иначе побеждает правило с максимальным приоритетом (при равенстве — более специфичное, при равной специфичности возвращается метка $\mathrm{AMBIGUOUS}$). Принципиальное свойство модели — разделение шаблона и контекста — позволяет одной сигнатуре в разных потоках соответствовать разным активностям, что снимает основной источник ложных корреляций.

Жизненный цикл данных описан шестью сценариями, из которых подробно разобраны три. При обогащении на чтение запрос GET /traces/{id}/enriched читает все спаны трейса из ClickHouse одним запросом, строит в памяти снимок активных правил и сопоставляет каждый спан без дополнительных обращений к базам (сложность $O(N + M)$ на чтение и $O(N \cdot k)$ на сопоставление). Обнаружение дрейфа выполняет планировщик, который агрегирует сигнатуры за окно, отсекает покрытые правилами и записывает оставшиеся в журнал нераспознанных. Сверка с эталоном собирает обогащённые трейсы, отфильтровывает неразмеченные спаны, строит журнал событий, конвертирует BPMN-модель в сеть Петри и выполняет проигрывание по фишкам в PM4Py. Модель данных разделена между двумя хранилищами: в Postgres определены четыре таблицы (rules, bpmn_models, unmatched_signatures, flow_contexts) с индексами, в том числе инвертированным GIN-индексом по контексту; в ClickHouse — единственная таблица сырых спанов с партиционированием по дню. *Вывод по разделу:* спроектирована архитектура, удовлетворяющая требованию минимального вмешательства, а формализованная модель правила даёт детерминированный и расширяемый механизм разметки.

Третий раздел «Реализация системы» описывает построенный автором прототип; выбор каждой технологии обоснован. Rules Service реализован на Java 21 и Spring Boot 3.4 и организован по слоям (контроллеры, сервисы, репозитории, сущности, мапперы): контроллеры реализуют сгенерированные из OpenAPI интерфейсы, доступ к Postgres — через Spring Data JPA, к ClickHouse — через собственный SpanReader на основе JdbcTemplate, JSONB-поля сериализуются Hibernate, преобразование сущностей в DTO выполняет MapStruct, а агрегатор нераспознанных сигнатур запускается по расписанию. Ядро сервиса — метод обогащения трейса, использующий единый снимок правил для всех спанов. Variants Service реализован на Python 3.12, FastAPI и PM4Py и предоставляет точки /variants (кластеризация трейсов по структурной сигнатуре), /variants/{id}/canonical (дерево спанов образцового трейса) и /conformance (сверка соответствия). Подробно описан шестишаговый алгоритм выделения похожих трейсов (отбор окна, загрузка спанов, сборка сигнатуры трейса, группировка, фильтрация, выбор топ-N) со сложностью $O(M \log M + N)$ и обоснован его сознательно консервативный характер: два трейса попадают в один вариант лишь при полном совпадении последовательности пар «сервис,

имя операции».

Web UI реализован на React 18, TypeScript и Vite с пятью маршрутами и разделением на режим аналитика (/setup: BPMN-редактор и разметка спанов перетаскиванием на полотне с логикой «сначала деактивировать, затем создать») и режим разработчика (/monitoring: показатели, отчёт сверки, список нераспознанных сигнатур, поиск трейса по идентификатору); приведены снимки реализованных экранов. Вся инфраструктура описана одним файлом docker-compose.yml и состоит из семи контейнеров (Kafka в режиме KRaft, ClickHouse, Postgres, OpenTelemetry Collector, Rules Service, Variants Service, Web UI), поднимаемых командой docker compose up; принципиальная особенность конфигурации коллектора — параллельный экспорт одного потока спанов в Kafka и ClickHouse. Для проверки на реалистичной нагрузке создан демонстрационный стенд из семи микросервисов (bank-gateway, users, accounts, fraud, ledger, notifications, audit), имитирующий услугу P2P-перевода в банке; полная цепочка порождает от 9 до 17 спанов, причём сервисы сознательно *не* проставляют бизнес-атрибуты. *Вывод по разделу:* получен работоспособный прототип, в котором бизнес-смысл «открывается» системой через обнаружение шаблонов, а не задаётся заранее в коде сервисов.

Четвёртый раздел «Тестирование» посвящён количественной проверке системы. Эксперимент понимается в инженерном смысле: для каждого критического свойства формулируется проверяемая гипотеза с числовым критерием приёмки, описывается воспроизводимая процедура измерения и фиксируется фактическое значение. Инструментарий наблюдения — Prometheus (опрос метрик с интервалом 15 секунд) и Grafana; помимо стандартной гистограммы задержек HTTP-запросов Spring Boot объявлены четыре собственные метрики системы (число известных сигнатур с разбивкой на покрытые и непокрытые, длина журнала нераспознанных, счётчик прошедших сверку трейсов, число активных правил), а каждый эксперимент реализован отдельным Python-сценарием, который запускает нагрузку, опрашивает HTTP API Prometheus и сохраняет результаты.

Проверены пять групп свойств. *E1* (задержка обогащения на чтение): p_{95} с правилами составил $30{,}5$ мс против $20{,}0$ мс без правил, накладные расходы $+52{,}7\,\%$ ($1{,}53\times$ при пороге $1{,}5\times$) — гипотеза не принята с минимальным промахом; основная причина (избыточные запросы к Postgres) разобрана и устранена

снимком правил, остаток обусловлен задержкой сетевого стека WSL2. *E2* (обнаружение дрейфа): медиана задержки $27{,}6$ с при интервале запуска агрегатора 30 с, все значения в пределах порога — принята. *E3* (точность сверки): точность $1{,}0$, полнота $0{,}69$, $F_1 = 0{,}82$ — не принята; все ложноотрицательные срабатывания приходятся на шумовой класс деформаций, что является сознательным следствием обогащения на чтение (незнакомые активности видны через список нераспознанных, а не через сверку), тогда как на классах «пропуск», «перестановка» и «дубликат» активности $F_1 = 1{,}0$. *E4* (покрытие правилами): при распределении сигнатур по закону Ципфа покрытие достигает $0{,}69$ на 5 правилах, $0{,}86$ на 10 и $1{,}00$ на 20 — принята с большим запасом, что подтверждает дешевизну разметки «длинного хвоста». *E5* (масштабирование): при 25-кратном росте объёма данных задержка чтения выросла лишь в $1{,}86$ раза — принята; узким местом названо отсутствие ключа сортировки по $\mathrm{TraceId}$ в схеме, создаваемой экспортёром. Отдельно разобраны угрозы валидности результатов (объём выборок, шум локального Docker, синтетичность данных, простота движка сверки). *Вывод по разделу:* из пяти гипотез приняты три, две не приняты с разобранными причинами; результаты подтверждают пригодность предложенной архитектуры для целевых сценариев, а непринятые гипотезы носят характер близкого к порогу промаха и сознательного архитектурного ограничения, для которых указаны направления доработки.

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе спроектирована и реализована система BAT, выполняющая автоматическую бизнес-разметку распределённых трейсов на основе OpenTelemetry.

Проанализированы пять подходов к связыванию трейсов с бизнес-процессом и выявлены их ограничения. Предложена схема обогащения на чтение: сырые спаны не модифицируются, а бизнес-метки вычисляются по правилам в момент запроса. Модель правила формализована тройкой $\langle match,\ context,\ assign \rangle$ с приоритетом; разделение сигнатуры и контекста потока устраняет коллизии имён между сценариями. Архитектура описана диаграммами C4 и шестью сценариями работы системы.

Построен прототип из четырёх компонентов: Rules Service на Spring Boot, Variants Service на FastAPI и PM4Py, Web UI на React и OpenTelemetry Collector. Стек разворачивается одной командой $\mathrm{docker\ compose\ up}$. Для демонстрации создан стенд из семи микросервисов услуги перевода. Выполнено пять экспериментов; из пяти гипотез приняты три, две не приняты с разобранной причиной.

Перспективы развития: кэш правил в памяти, поддержка BPMN со шлюзами и параллельными ветвями, расширение правил регулярными выражениями, добавление авторизации, калибровка на реальных трейсах. Цель работы достигнута: система соответствует принципу минимального вмешательства в наблюдаемую инфраструктуру и подтверждена экспериментально.

ОСНОВНЫЕ ИСТОЧНИКИ ИНФОРМАЦИИ

1. *van der Aalst W. M. P., Rozinat A.* Conformance checking of processes based on monitoring real behavior [Электронный ресурс] // Information Systems. — 2008. — Vol. 33, no. 1. — Pp. 64–95. — URL: https://doi.org/10.1016/j.is.2007.07.001 (дата обращения: 25.04.2026). Загл. с экр. Яз. англ.
2. *Adriansyah A.* Conformance checking using cost-based fitness analysis [Электронный ресурс] // Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC). — 2011. — Pp. 55–64. — URL: https://doi.org/10.1109/EDOC.2011.12 (дата обращения: 25.04.2026). Загл. с экр. Яз. англ.
3. *Berti A.* Process mining for Python (PM4Py): Bridging the gap between process- and data science [Электронный ресурс] // Proceedings of the ICPM Demo Track 2019, CEUR Workshop Proceedings. — 2019. — Vol. 2374. — Pp. 13–16. — URL: https://ceur-ws.org/Vol-2374/paper4.pdf (дата обращения: 25.04.2026). Загл. с экр. Яз. англ.
4. Process Intelligence in Action: Taking Process Mining to the Next Level [Электронный ресурс] / Ed. by L. Reinkemeyer. — Cham, Switzerland: Springer, 2024. — 248 p. — URL: https://doi.org/10.1007/978-3-031-61343-2 (дата обращения: 25.04.2026). Загл. с экр. Яз. англ.
5. *Зуева А. Н., Канева И. Ю.* Бизнес-процессы. Анализ, моделирование, управление: учебное пособие для вузов [Электронный ресурс]. — СПб.: Лань, 2023. — 160 с. — URL: https://lanbook.com/catalog/informatika/biznes-protsessy-analiz-modelirovanie-upravlenie/ (дата обращения: 25.04.2026). Загл. с экр.
6. Business Process Model and Notation (BPMN), Version 2.0.2 [Электронный ресурс] / Object Management Group. — 2014. — URL: https://www.omg.org/spec/BPMN/2.0.2/ (дата обращения: 25.04.2026). Загл. с экр. Яз. англ.
7. OpenTelemetry Specification [Электронный ресурс] / Cloud Native Computing Foundation. — 2025. — URL: https://opentelemetry.io/docs/specs/otel/ (дата обращения: 25.04.2026). Загл. с экр. Яз. англ.
8. W3C Trace Context, Level 2 [Электронный ресурс] / World Wide Web Consortium. — 2025. — URL: https://www.w3.org/TR/trace-context-2/ (дата обращения: 25.04.2026). Загл. с экр. Яз. англ.
9. *Brown S.* The C4 model for visualising software architecture [Электрон-

ный ресурс]. — 2025. — URL: https://c4model.com/ (дата обращения: 25.04.2026). Загл. с экр. Яз. англ.

10. *Баланов А. Н.* Построение микросервисной архитектуры и разработка высоконагруженных приложений: учебное пособие для вузов [Электронный ресурс]. — СПб.: Лань, 2025. — 244 с. — URL: https://lanbook.com/catalog/informatika/postroenie-mikroservisnoy-arkhitektury-i-razrabotka-vysokonagruzhennykh-prilozheniy/ (дата обращения: 25.04.2026). Загл. с экр.