

УДК 004.4:004.738.5

UDC 004.4:004.738.5

5.2.2. Математические, статистические и инструментальные методы в экономике

5.2.2. Mathematical, statistical and instrumental methods in economics

**АРХИТЕКТУРНЫЕ СРЕДСТВА
АВТОМАТИЗАЦИИ ПРОЦЕССА
УПРАВЛЕНИЯ ЗАКАЗАМИ В
ИНФОРМАЦИОННЫХ СИСТЕМАХ НА
ПРИМЕРЕ СЕРВИСА ИНТЕЛЛЕКТУАЛЬНОЙ
ОБРАБОТКИ МУЗЫКАЛЬНОГО КОНТЕНТА**

**ARCHITECTURAL APPROACHES TO
AUTOMATING ORDER MANAGEMENT
PROCESSES IN INFORMATION SYSTEMS: A
CASE STUDY OF AN INTELLIGENT MUSIC
CONTENT PROCESSING SERVICE**

Замотайлова Дарья Александровна
канд. экон. наук, доцент
*ФГБОУ ВО Кубанский государственный аграрный
университет имени И.Т. Трубилина, Краснодар,
Россия*

Zamotajlova Daria Aleksandrovna
Candidate of Economic Sciences, Associate Professor
*Kuban State Agrarian University named after I.T.
Trubilin, Krasnodar, Russia*

Ковыляев Роман Александрович
Студент
*ФГБОУ ВО Кубанский государственный аграрный
университет имени И.Т. Трубилина, Краснодар,
Россия*

Kovylyaev Roman Alexandrovich
student
*Kuban State Agrarian University named after I.T.
Trubilin, Krasnodar, Russia*

В статье рассматривается проблема автоматизации процесса управления заказами в современных информационных системах как комплексная архитектурная задача, требующая учета требований к масштабируемости, отказоустойчивости, интеграции разнородных данных и обработке больших объемов вычислительно интенсивных задач. На основе анализа эволюции архитектурных парадигм – от монолитных систем через сервис-ориентированную архитектуру к микросервисным и событийно-ориентированным решениям – обосновывается необходимость применения распределенных подходов для обеспечения эффективного жизненного цикла заказов в условиях высокой динамики бизнес-процессов и роста объемов данных. Особое внимание уделяется роли систем управления заказами (Order Management System, OMS) как центрального оркестратора, обеспечивающего согласованность состояний, интеграцию с внешними сервисами и автоматизацию workflow на основе стандартов жизненного цикла программных средств по ГОСТ Р ИСО/МЭК 12207

This article examines the automation of order management processes in modern information systems as a complex architectural challenge that requires consideration of scalability, fault tolerance, heterogeneous data integration, and the processing of computationally intensive workloads. Based on an analysis of the evolution of architectural paradigms—from monolithic systems through Service-Oriented Architecture (SOA) to microservices and event-driven solutions—the study substantiates the need for distributed approaches to support an efficient order lifecycle in environments characterized by rapidly changing business processes and continuously growing data volumes. Particular attention is paid to the role of Order Management Systems (OMS) as central orchestration components responsible for maintaining state consistency, integrating external services, and automating workflows in accordance with software life cycle standards defined by GOST R ISO/IEC 12207. The proposed approach is illustrated through the example of an intelligent music content processing service that employs artificial intelligence technologies for audio analysis and transformation

Ключевые слова: АРХИТЕКТУРА
ИНФОРМАЦИОННЫХ СИСТЕМ,
МИКРОСЕРВИСНАЯ АРХИТЕКТУРА,
УПРАВЛЕНИЕ ЗАКАЗАМИ, ORDER
MANAGEMENT SYSTEM, EVENT-DRIVEN

Keywords: INFORMATION SYSTEMS
ARCHITECTURE, MICROSERVICES
ARCHITECTURE, ORDER MANAGEMENT,
ORDER MANAGEMENT SYSTEM (OMS),
EVENT-DRIVEN ARCHITECTURE, AUDIO

АРХИТЕКТУРА, ОБРАБОТКА
АУДИОКОНТЕНТА, ИСКУССТВЕННЫЙ
ИНТЕЛЛЕКТ, BPM-СИСТЕМЫ, ИНТЕГРАЦИЯ
ДАННЫХ, КОНТЕЙНЕРИЗАЦИЯ,
РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ,
ИНТЕЛЛЕКТУАЛЬНАЯ ОБРАБОТКА
МУЗЫКАЛЬНОГО КОНТЕНТА

CONTENT PROCESSING, ARTIFICIAL
INTELLIGENCE, BPM SYSTEMS, DATA
INTEGRATION, CONTAINERIZATION,
DISTRIBUTED SYSTEMS, INTELLIGENT MUSIC
CONTENT PROCESSING

<http://dx.doi.org/10.21515/1990-4665-221-008>

Введение

Управление заказами в современных информационных системах уже давно вышло за рамки простой автоматизации рутинных операций. Сегодня заказ стал центральным элементом всей системы – именно вокруг него выстраивается весь цикл взаимодействия с пользователем: от поступления запроса и его проверки до распределения задач, контроля их выполнения и выдачи результата.

С ростом объемов данных и усложнением бизнес-процессов традиционные подходы к построению информационных систем все чаще показывают свои слабые стороны. Их отличают жесткая связанность компонентов, они плохо масштабируются и медленно адаптируются к изменениям.

В связи с этим возникает потребность в архитектуре, которая способна не только автоматизировать стандартные операции, но и эффективно справляться со сложными вычислительными нагрузками. Особенно остро эта задача стоит в тех областях, где обработка заказов требует применения моделей искусственного интеллекта (ИИ) при работе с медиаконтентом.

В качестве примера рассматривается сервис интеллектуальной обработки музыкального контента с использованием ИИ. Здесь заказ – это уже не простая транзакция, а сложный многоэтапный запрос на обработку аудиофайлов, включающий шумоподавление, разделение на отдельные дорожки, мастеринг, генерацию метаданных и другие операции. Такая обработка требует асинхронного выполнения тяжелых вычислений.

<http://ej.kubagro.ru/2026/07/pdf/08.pdf>

В работе описывается авторская гибридная архитектурная модель, которая объединяет микросервисную структуру, событийно-ориентированное взаимодействие через очереди сообщений и BPM-оркестрацию состояний заказа. В модель входят пользовательский интерфейс, backend-оркестратор, специализированные AI-модули, распределенное хранилище и система выдачи результатов.

Научная новизна работы заключается в разработке интегрированной архитектуры, учитывающей специфику обработки медиаконтента с высоким использованием ИИ. Обосновывается выбор технологий контейнеризации и CI/CD, а также механизмов обеспечения конечной согласованности данных. Сравнительный анализ монолитных и микросервисных решений показал, что основные сложности при реализации таких систем связаны с задержками обработки, работой с большими объемами данных и обеспечением безопасности. Для каждой из этих проблем были предложены соответствующие архитектурные решения. Переход к распределенной архитектуре позволяет повысить устойчивость системы к росту нагрузки и снизить влияние отдельных сбоев на ее работу.

Материалы и методы

Развитие информационных систем прошло путь от монолитных приложений к сервис-ориентированным и микросервисным архитектурам. В микросервисной модели функциональность системы разделяется между независимыми компонентами, которые могут разрабатываться, обновляться и масштабироваться отдельно друг от друга. Такой подход упрощает развитие системы и позволяет использовать различные технологии для решения отдельных задач. Вместе с тем увеличивается количество взаимодействующих сервисов, усложняется контроль согласованности данных и возрастает роль средств мониторинга.

Обмен информацией между компонентами обеспечивается через программные интерфейсы (API). В зависимости от поставленной задачи взаимодействие может выполняться как в синхронном режиме, например с использованием REST или GraphQL, так и асинхронно через очереди сообщений и механизмы передачи событий.

В системах управления заказами (Order Management System, OMS) такие механизмы позволяют организовать полный жизненный цикл заказа – от создания до завершения. OMS обычно выступает центральным компонентом корпоративной информационной системы. На нее ложится задача приема заказов из разных каналов, контроля ресурсов, распределения задач и обеспечения прослеживаемости процесса. Поэтому при проектировании подобных систем важно ориентироваться на стандарты жизненного цикла программных средств, в частности на ГОСТ Р ИСО/МЭК 12207.

Особенности рассматриваемой архитектуры можно показать на примере сервиса интеллектуальной обработки музыкального контента. Здесь заказ представляет собой запрос пользователя на выполнение ряда операций по обработке аудиофайла, которые выполняются последовательно различными компонентами системы.

Пользователь формирует заказ, указывая необходимые операции с аудиофайлом или группой файлов: очистку от шумов, разделение на отдельные дорожки, автоматический мастеринг, генерацию метаданных и другие задачи. Такая сущность меняет свое состояние по мере прохождения этапов обработки, требует хранения промежуточных результатов и допускает возможность отмены или перезапуска отдельных операций.

Архитектура предлагаемой системы строится по многоуровневой схеме. Она начинается с пользовательского интерфейса, через который загружаются файлы, настраиваются параметры и отслеживается прогресс.

Далее работает backend-оркестратор, принимающий заказ, проверяющий данные и запускающий асинхронную обработку. Центральное место занимают AI-модули, реализованные как набор специализированных микросервисов. Каждый из них отвечает за конкретную задачу обработки аудио. Эти модули взаимодействуют с распределенным хранилищем, обеспечивая контроль за промежуточными результатами. Завершает цепочку компонент выдачи результата. Обобщенная архитектура системы представлена на рисунке 1.

Взаимодействие между компонентами системы организовано в основном через API и очереди сообщений. Такой подход позволяет эффективно обрабатывать большие аудиофайлы и сложные модели ИИ в асинхронном режиме. Когда поступает новый заказ, оркестратор отправляет соответствующее событие в очередь (например, в RabbitMQ или Apache Kafka). Нужный AI-сервис забирает это событие, выполняет свою часть работы и публикует новое событие о завершении этапа. Благодаря событийной модели компоненты оказываются слабо связанными между собой, что дает возможность выполнять разные этапы обработки параллельно.

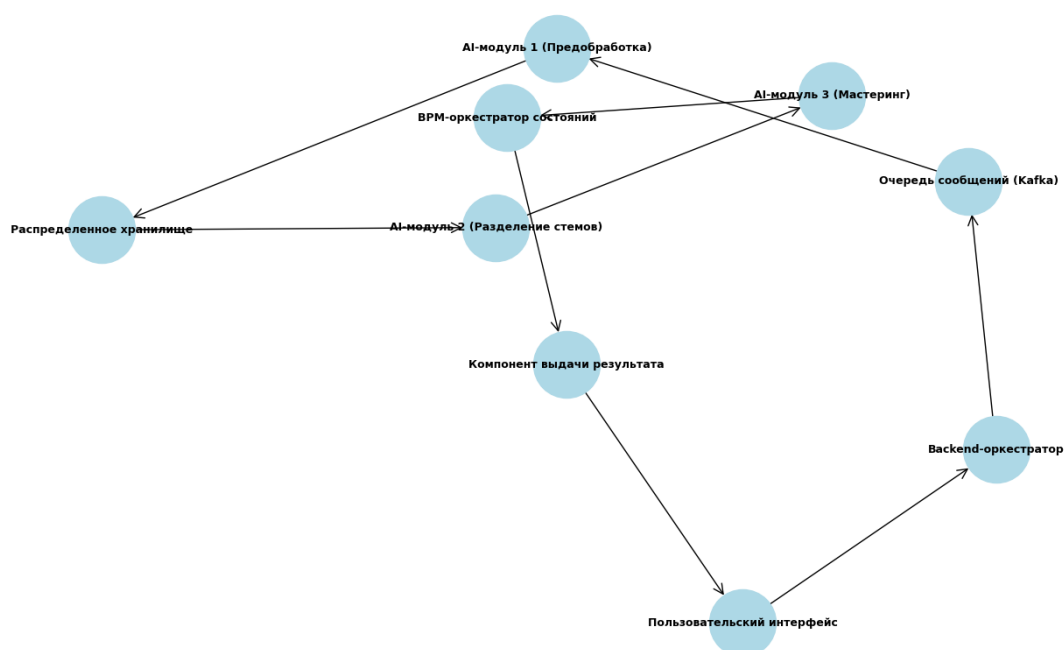


Рисунок 1 – Общая архитектура системы управления заказами в сервисе интеллектуальной обработки музыкального контента

Управление состоянием заказа реализовано в виде конвейера (рисунок 2). Каждое состояние заказа (`created` – создан, `validated` – валидирован, `processing` – в обработке, `completed` – завершен, `failed` – не выполнен) сохраняется в базе данных, а переходы между ними определяются правилами BPM (Business Process Management). Это позволяет задавать сложную логику обработки, включая ветвления, условные переходы и компенсационные действия, без необходимости прописывать ее жестко внутри каждого сервиса.

Масштабируемость и распределенность системы обеспечиваются за счет контейнеризации. Компоненты упаковываются в Docker-контейнеры и управляются с помощью Kubernetes. Такой подход позволяет динамически выделять вычислительные ресурсы под наиболее нагруженные AI-модули в зависимости от текущего потока заказов. Кроме того, практики DevOps и конвейеры непрерывной интеграции и доставки (CI/CD) дают возможность обновлять модели искусственного интеллекта и бизнес-логику без остановки работы сервиса.

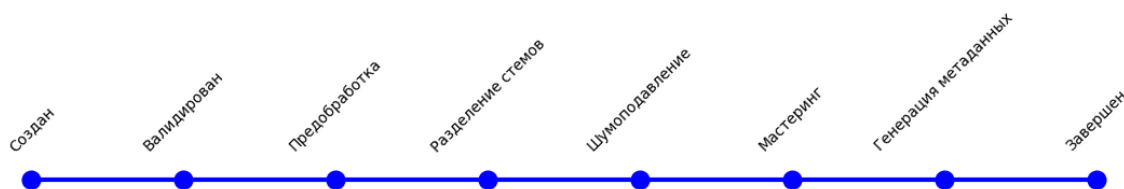


Рисунок 2 – Линейный поток обработки заказа

Задержки удастся сократить за счет асинхронной обработки, приоритизации очередей и горизонтального масштабирования GPU-

кластеров. Для работы с большими объемами данных применяются распределенные файловые системы и оптимизированные форматы хранения.

Таблица 1 – Среднее время обработки заказов по этапам пайплайна

Этап обработки	Среднее время (секунды)	Стандартное отклонение
Предобработка аудио	12,4	2,1
Разделение на стемы	45,8	8,3
Шумоподавление	28,6	4,7
Автоматический мастеринг	67,2	12,5
Генерация метаданных	8,9	1,4

Таблица 2 – Распределение нагрузки по компонентам системы

Компонент системы	Доля нагрузки (%)	Основная причина нагрузки
Backend-оркестратор	15	Валидация и маршрутизация заказов
AI-модули (GPU)	65	Вычислительно интенсивная обработка аудио
Распределенное хранилище	10	Чтение/запись больших аудиофайлов
ВРМ-оркестратор состояний	5	Управление переходами состояний
Очереди сообщений	5	Асинхронная передача событий

Структура системы основана на распределении функций между специализированными сервисами, каждый из которых выполняет отдельный этап обработки аудиоданных. Первый модуль осуществляет предварительную обработку аудиофайла, включая нормализацию уровня громкости, приведение данных к единому формату и извлечение характеристик аудиосигнала. Следующий модуль отвечает за разделение аудиозаписи на отдельные дорожки, что позволяет независимо обрабатывать вокальные и инструментальные партии. Отдельный сервис выполняет подавление шумов и улучшение качества звучания. Завершающий модуль осуществляет автоматический мастеринг и формирование метаданных, включая определение темпа, тональности и жанровых особенностей музыкального произведения.

Обмен данными между компонентами системы организован по событийному принципу. После загрузки аудиофайла серверный оркестратор сохраняет его в распределенном файловом хранилище, создает запись о заказе в базе данных и формирует событие «order.created» («заказ создан»). Далее вычислительные модули получают информацию о новом задании, обрабатывают аудиофайл на графических ускорителях и по завершении этапа формируют событие «stage.completed» («этап завершен»).

Результаты исследования

Центральный оркестратор, взаимодействующий с системой управления бизнес-процессами (Business Process Management, BPM), отвечает за контроль выполнения заказа. На основе событий, поступающих от других компонентов, он отслеживает изменение состояний заказа, определяет последовательность дальнейших действий и обеспечивает взаимодействие между сервисами.

Техническое взаимодействие между компонентами построено на двух уровнях. Для операций, где нужен быстрый ответ, используются обычные синхронные запросы через API (прямые вызовы, при которых клиент ждет ответа). А для длительных этапов обработки данные передаются через брокер сообщений Apache Kafka (система для асинхронного обмена сообщениями между сервисами). Такой подход помогает разделить этапы между разными сервисами и не перегружать отдельные части системы. Сервисы искусственного интеллекта сделаны в виде независимых модулей, которые не сохраняют состояние между запросами. Благодаря этому модули искусственного интеллекта можно легко масштабировать горизонтально при увеличении нагрузки.



Рисунок 3 – Потоки данных и событий в системе управления заказами

Проектирование системы ведется в соответствии со стандартом ГОСТ Р ИСО/МЭК 12207–2010, который регламентирует процессы жизненного цикла программных средств. Для понимания того, как система ведет себя в реальных условиях, используются средства мониторинга. Они позволяют отслеживать время выполнения операций, выявлять наиболее нагруженные компоненты и оперативно перераспределять ресурсы.

Одна из главных сложностей при обработке аудиоданных – это длительное время работы моделей искусственного интеллекта, особенно на длинных треках. Чтобы ускорить обработку, файл обычно разбивают на фрагменты и обрабатывают их параллельно. Кроме того, применяют различные методы оптимизации вычислений, которые помогают повысить скорость работы моделей без существенного ухудшения качества результата.

Для хранения данных используется многоуровневая система. Активные заказы размещаются на быстрых накопителях, а архивные материалы переносятся в более медленное, но надежное хранилище. Надежность работы системы достигается за счет резервирования сервисов, автоматического восстановления после сбоев и дублирования наиболее важных компонентов.

По сравнению с монолитными решениями распределенная архитектура лучше подходит для систем с высокой вычислительной нагрузкой. В монолитном приложении все функции работают в рамках единого процесса, поэтому масштабирование обычно затрагивает всю систему. В распределенной архитектуре отдельные компоненты могут работать и масштабироваться независимо друг от друга. Это особенно

важно для сервисов обработки аудиоданных, где основная нагрузка приходится на модули искусственного интеллекта.

Практический результат работы заключается в разработке архитектурной модели управления заказами для сервиса интеллектуальной обработки музыкального контента. В основе модели лежит разделение системы на независимые сервисы, обмен данными через события и использование средств управления бизнес-процессами. Такой подход позволяет распределять вычислительную нагрузку между специализированными модулями обработки и упрощает дальнейшее развитие системы.

Заключение

Разработка системы выполнялась поэтапно. По мере уточнения требований корректировались отдельные архитектурные решения и структура взаимодействия компонентов. Для проверки работоспособности использовались модульное тестирование, проверка обмена данными между сервисами и тестирование основных пользовательских сценариев. Это позволило выявлять ошибки на ранних этапах разработки и снизить риск их появления при эксплуатации системы.

В работе рассмотрена архитектура сервиса интеллектуальной обработки музыкального контента, основанная на использовании микросервисов, событийного взаимодействия и средств управления бизнес-процессами. Предложенный подход обеспечивает независимое развитие отдельных компонентов системы, упрощает распределение вычислительной нагрузки и позволяет организовать обработку ресурсоемких задач в условиях роста количества пользователей и объема данных.

Список литературы

1. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. – М.: Стандартинформ, 2012.
2. ISO/IEC 12207:2008. Systems and software engineering – Software life cycle processes.
3. Newman S. Building Microservices: Designing Fine-Grained Systems. – 2nd ed. – O'Reilly Media, 2021.
4. Зараменских Е. П. Управление жизненным циклом информационных систем : монография. – Новосибирск : Изд-во ЦРНС, 2014. – 270 с.
5. ГОСТ Р ИСО/МЭК ТО 15271-2002. Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207.
6. Зараменских Е. П. Управление жизненным циклом информационных систем : монография. – Новосибирск : Изд-во ЦРНС, 2014. – 270 с.

References

1. GOST R ISO/MJeK 12207-2010. Informacionnaja tehnologija. Sistemnaja i programmaja inzhenerija. Processy zhiznennogo cikla programmnyh sredstv. – M.: Standartinform, 2012.
2. ISO/IEC 12207:2008. Systems and software engineering – Software life cycle processes.
3. Newman S. Building Microservices: Designing Fine-Grained Systems. – 2nd ed. – O'Reilly Media, 2021.
4. Zaramenskih E. P. Upravlenie zhiznennym ciklom informacionnyh sistem : monografija. – Novosibirsk : Izd-vo CRNS, 2014. – 270 s.
5. GOST R ISO/MJeK TO 15271-2002. Informacionnaja tehnologija. Rukovodstvo po primeneniju GOST R ISO/MJeK 12207.
6. Zaramenskih E. P. Upravlenie zhiznennym ciklom informacionnyh sistem : monografija. – Novosibirsk : Izd-vo CRNS, 2014. – 270 s.