ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.415.2

АРХИТЕКТУРА ИНТЕГРАЦИОННОЙ ТЕХНОЛОГИИ С РАЗДЕЛЕНИЕМ ПЛОСКОСТЕЙ УПРАВЛЕНИЯ И ДАННЫХ ДЛЯ ГЕТЕРОГЕННЫХ ИТ-ЛАНДШАФТОВ

Смирнов Антон Владимирович,
РТУ МИРЭА, г. Москва, Россия
Институт информационных технологий
Базовая кафедра № 232 - вычислительных систем реального времени
Магистрант, высшее образование степени бакалавр
E-mail: antonuniverse7@gmail.com

Аннотация

В статье рассматривается проблема информационной разобщенности в условиях гетерогенных корпоративных ИТ-ландшафтов, сочетающих унаследованные системы, современные облачные сервисы и распределенные компоненты. Существующие подходы к интеграции (ESB, EDA, API-led) демонстрируют фундаментальное противоречие: централизация управления ведет к образованию «узкого места» в канале передачи данных, тогда как децентрализация управления снижает прозрачность сквозных бизнес-процессов. Целью данной работы является разработка архитектуры и методов интеграционной технологии, основанной на универсальном модуле-контроллере. Научная новизна предлагаемого решения заключается в строгом разделении плоскости управления (Control Plane) и плоскости данных (Data Plane) на уровне корпоративной оркестрации. В работе вводится понятийный аппарат формализованных структур: реестра компонентов и реестра задач, обеспечивающих управление на основе желаемого состояния (Desired State Management). В статье представлен сравнительный анализ архитектурных подходов, обоснован выбор инструментальных средств моделирования (ArchiMate, BPMN, OpenAPI/AsyncAPI) и спроектированы ключевые сценарии функционирования модуля-контроллера, включая обработку отказов. Результаты исследования формируют методологическую основу для построения единого информационного пространства предприятия, снижая совокупную стоимость владения интеграционной инфраструктурой.

Ключевые слова: интеграционная технология, архитектура предприятия, оркестрация, гетерогенная ИТ-среда, модуль-контроллер, разделение плоскостей управления и данных, BPMN, реестр компонентов, управление состоянием, ESB.

ARCHITECTURE OF INTEGRATION TECHNOLOGY WITH SEPARATION OF CONTROL AND DATA PLANES FOR HETEROGENEOUS IT LANDSCAPES

Smirnov Anton Vladimirovich,
RTU MIREA, Moscow, Russia
Institute of Information Technology

Basic Department No. 232 - Real-Time Computing Systems
Master's Student, Bachelor of Engineering

ABSTRACT

The article addresses the issue of information fragmentation within heterogeneous corporate IT landscapes that combine legacy systems, modern cloud services, and distributed components. Existing integration approaches (ESB, EDA, API-led) reveal a fundamental contradiction: the centralization of management creates a bottleneck in the data transmission channel, whereas the decentralization of management reduces the transparency of end-to-end business processes. The objective of this work is to develop an architecture and methods for integration technology based on a universal controller module. The scientific novelty of the proposed solution lies in the strict separation of the Control Plane and the Data Plane at the enterprise orchestration level. The paper introduces the conceptual framework of formalized structures: a component registry and a task registry, which enable Desired State Management. The article provides a comparative analysis of architectural approaches, justifies the choice of modeling tools (ArchiMate, BPMN, OpenAPI/AsyncAPI), and designs key operational scenarios for the controller module, including failure handling. The research results form a methodological foundation for building a unified information space for an enterprise, reducing the total cost of ownership of the integration infrastructure.

Keywords: integration technology, enterprise architecture, orchestration, heterogeneous IT environment, controller module, separation of control and data planes, BPMN, component registry, state management, ESB.

Актуальность

Современный этап цифровой трансформации характеризуется экспоненциальным ростом сложности корпоративных информационных систем. Организации различных отраслей функционируют в условиях гетерогенного ИТ-ландшафта, в рамках которого одновременно сосуществуют унаследованные (legacy) системы с закрытыми протоколами обмена, современные прикладные платформы классов ERP и CRM с открытыми интерфейсами, облачные SaaS-сервисы, а также территориально распределенные мобильные и веб-компоненты [2, с. 45–47]. Подобная технологическая разнородность является объективной реальностью, обусловленной эволюционным характером развития ИТ-инфраструктуры и поэтапным внедрением решений от различных вендоров.

Сосуществование перечисленных классов систем в рамках единой организационной среды закономерно порождает комплекс барьеров интеграционного характера. Согласно классификации, принятой в современной литературе по архитектуре предприятия, выделяются платформенные барьеры (несовместимость операционных сред и аппаратных архитектур), синтаксические барьеры (различия в форматах сериализации данных и протоколах взаимодействия, таких как SOAP/XML и REST/JSON) и семантические барьеры (конфликты интерпретации одних и тех же бизнес-сущностей в разных учетных системах) [3, с. 45–48]. Следствием указанных барьеров становится возникновение разрывов в сквозных бизнес-процессах, вынужденная практика ручной передачи данных между системами силами операторов, отсутствие единой видимости состояния процессов в режиме реального времени и, как итог, высокие операционные затраты на поддержку «лоскутной» автоматизации [5].

Анализ существующих архитектурных подходов к решению задачи корпоративной интеграции демонстрирует наличие фундаментальной дилеммы. С одной стороны, классическая сервис-ориентированная архитектура, реализуемая посредством сервисной шины предприятия (Enterprise Service Bus, ESB), предлагает модель с централизованным управлением маршрутизацией и трансформацией данных. Однако в такой архитектуре шина выступает не только координатором, но и единой точкой прохождения всех потоков полезных данных, что создает узкое место по производительности и снижает отказоустойчивость всей системы [2, с. 312–315]. С другой стороны, современные децентрализованные подходы, включая событийно-ориентированную архитектуру (Event-Driven Architecture, EDA) и хореографию микросервисов, успешно решают проблему масштабирования передачи данных, но при этом приводят к потере централизованного контроля над сквозными процессами [3, с. 178–181]. В подобных системах логика исполнения бизнес-процесса оказывается распределенной по множеству независимых компонентов, что делает крайне затруднительными мониторинг состояния процесса, диагностику сбоев и автоматическое восстановление после отказов [7, с. 78–82].

Существующие на рынке гибридные интеграционные платформы и решения с элементами разделения плоскости управления (Control Plane) и плоскости данных (Data Plane), такие как TIBCO Control Plane или аналоги, делают шаг в направлении разрешения данного противоречия. Тем не менее, их функциональность преимущественно сфокусирована на управлении сетевым оборудованием и API-шлюзами, в то время как задача комплексной оркестрации гетерогенных программно-аппаратных ресурсов предприятия с формализацией желаемого состояния сквозных бизнес-процессов остается недостаточно проработанной [6, с. 142–145]. Более того, в классических подходах отсутствует явное выделение структур данных, описывающих целевое состояние системы (реестр компонентов и реестр задач), что вынуждает «зашивать» логику управления в конфигурационные файлы или программный код, усложняя эволюцию ИТ-ландшафта [8, с. 245–248].

Таким образом, актуальность настоящего исследования обусловлена необходимостью разработки нового архитектурного подхода к построению интеграционной технологии, который позволил бы преодолеть выявленное противоречие между управляемостью и масштабируемостью. Требуется решение, обеспечивающее централизованную оркестрацию и контроль состояния компонентов при децентрализованной, не затрагивающей управляющий контур передаче полезных данных через существующие транспортные шины и брокеры сообщений.

Цель исследования

Целью настоящего исследования является разработка архитектуры и методов интеграционной технологии, основанной на применении универсального модуля-контроллера и обеспечивающей построение единого информационного пространства в условиях гетерогенного ИТ-ландшафта предприятия.

Для достижения поставленной цели в работе необходимо решить следующие исследовательские задачи:

Выполнить анализ предметной области и классифицировать существующие интеграционные подходы (точечная интеграция, сервисная шина предприятия, API-led архитектура, событийно-ориентированная архитектура, гибридные платформы и решения с разделением плоскостей) с целью выявления их ограничений в контексте сочетания централизованного управления и децентрализованной передачи данных.

Разработать архитектуру универсального модуля-контроллера, включающую формализованное описание его структурных компонентов: оркестратора процессов, контроллера состояния компонентов, менеджера ресурсов и роутера, а также обосновать

принципы его функционирования, базирующиеся на разделении плоскости управления и плоскости данных.

Определить состав и структуру ключевых информационных объектов, поддерживаемых модулем-контроллером, а именно реестра компонентов (структурированного описания физических и программных элементов инфраструктуры) и реестра задач (формализованного описания желаемого состояния системы и сквозных бизнес-процессов), что обеспечивает реализацию модели управления на основе состояния (Desired State Management).

Спроектировать методы и стандарты взаимодействия между модулем-контроллером и плоскостью данных, включая обоснование выбора нотаций моделирования архитектуры (ArchiMate, C4 Model), средств спецификации интерфейсов (OpenAPI, AsyncAPI) и языка описания бизнес-процессов (BPMN), достаточных для формализации оркестрации без привязки к конкретным программным реализациям.

Выполнить проектирование ключевых сценариев функционирования предлагаемой технологии, охватывающих как штатные режимы выполнения сквозных процессов, так и исключительные ситуации, связанные с отказами отдельных компонентов, недоступностью интеграционных шин и сбоями самого модуля-контроллера, с целью верификации устойчивости архитектурного решения.

Провести сравнительную оценку предложенного архитектурного подхода с существующими альтернативными решениями по критериям управляемости, масштабируемости, гибкости, сложности внедрения и отказоустойчивости, а также определить научную новизну и практическую значимость полученных результатов.

Достижение сформулированной цели и решение перечисленных задач позволит создать целостное архитектурное и методологическое описание интеграционной технологии, которое может служить основой для последующей практической реализации на предприятиях различных отраслей, сталкивающихся с проблемой информационной разобщенности гетерогенных систем.

Материалы и методы исследования

Теоретической и методологической базой настоящего исследования послужили фундаментальные труды и прикладные разработки в области архитектуры предприятия, сервис-ориентированной архитектуры (SOA), событийно-управляемых систем, а также современные концепции построения распределенных информационных систем.

Материалы исследования. В качестве исходных материалов в работе использованы:

Научные публикации и монографии, охватывающие проблематику интеграции корпоративных приложений, включая работы Е. П. Зараменских [1], Ф. Нила и соавторов [2], А. Беллемара [3], а также зарубежные источники, посвященные микросервисным паттернам и программно-определяемым сетям [5; 6; 8].

Стандарты и спецификации в области моделирования бизнес-процессов и архитектуры систем, в частности нотация BPMN (Business Process Model and Notation) версии 2.0 [1, с. 98–101], язык описания архитектуры предприятия ArchiMate, модель визуализации архитектуры C4 Model, а также спецификации описания интерфейсов OpenAPI и AsyncAPI [8, с. 178–181].

Результаты предпроектного обследования типовых гетерогенных ИТ-ландшафтов, выполненного на уровне классов систем, что позволило выявить инвариантные характеристики разрывов в сквозных бизнес-процессах, практик ручной передачи данных и высоких затрат на поддержку точечных интеграций [6, с. 271–273].

Аналитические обзоры существующих интеграционных решений, включая сервисные шины предприятия (ESB), брокеры сообщений, API-шлюзы, гибридные

интеграционные платформы и решения с разделением плоскости управления и плоскости данных [5, с. 142–145; 7, с. 78–82].

Методы исследования. Для достижения поставленной цели и решения сформулированных задач в работе применен комплекс общенаучных и специальных методов, адекватных предмету исследования:

Метод системного анализа использован при исследовании объекта автоматизации — гетерогенного ИТ-ландшафта предприятия, что позволило декомпозировать его на характерные слои (унаследованные системы, современные прикладные платформы, облачные сервисы, распределенные компоненты) и выявить системные барьеры интеграции: платформенные, синтаксические и семантические [2, с. 45–47; 3, с. 45–48].

Метод классификации применен при анализе существующих вариантов решения проблемы исследования, в результате чего выделены и сопоставлены шесть классов интеграционных подходов: точечная интеграция, сервисная шина предприятия, API-led архитектура, событийно-ориентированная архитектура, гибридные интеграционные платформы и решения с разделением плоскостей управления и данных [3, с. 234–237; 6, с. 275–277].

Метод сравнительного анализа использован для оценки достоинств и ограничений каждого из перечисленных подходов в контексте ключевых критериев: управляемость, масштабируемость, гибкость, сложность внедрения и устойчивость к сбоям [5, с. 98–102]. Это позволило обосновать необходимость разработки нового архитектурного решения.

Метод архитектурного проектирования составил ядро исследования и был направлен на разработку структуры универсального модуля-контроллера. В рамках данного метода определены составные части модуля (оркестратор процессов, контроллер состояния компонентов, менеджер ресурсов, роутер), сформулированы принципы их взаимодействия и обосновано разделение плоскости управления и плоскости данных [4, с. 125–128].

Метод формализации применен при определении структур данных, поддерживаемых модулем-контроллером: реестра компонентов (описания физических и программных элементов инфраструктуры, их интерфейсов и текущего состояния) и реестра задач (спецификации желаемого состояния системы, сквозных бизнес-процессов и политик восстановления) [7, с. 203–206].

Метод сценарного моделирования использован для проектирования ключевых сценариев функционирования предлагаемой интеграционной технологии, охватывающих как штатные режимы выполнения процессов, так и исключительные ситуации, включая сбои отдельных серверов, недоступность интеграционных шин и отказы самого модуля-контроллера.

Метод качественной оценки применен при анализе факторов, влияющих на стоимость внедрения предлагаемого решения, включая сложность первоначальной настройки, необходимость модернизации унаследованных систем и объем изменений в существующей интеграционной инфраструктуре [2, с. 398–401].

Совокупность перечисленных материалов и методов обеспечила необходимую полноту и достоверность полученных результатов, позволив перейти от теоретического анализа предметной области к проектированию конкретных архитектурных и методологических решений, пригодных для последующей практической реализации.

Результаты и их обсуждение

В ходе выполнения исследования получены следующие научные и практические результаты, которые могут быть структурированы по основным направлениям работы.

1. Результаты анализа существующих интеграционных подходов.

Проведенный сравнительный анализ шести классов интеграционных решений позволил выявить их принципиальные ограничения в контексте задачи построения единого информационного пространства гетерогенного ИТ-ландшафта. Установлено, что подходы на основе точечной интеграции характеризуются квадратичным ростом числа связей при увеличении количества систем, что приводит к формированию «спагетти-архитектуры» и делает поддержку такой инфраструктуры экономически нецелесообразной [5]. Решения класса ESB, несмотря на обеспечение централизованного управления, создают узкое место по производительности и единую точку отказа, поскольку через шину проходят все потоки полезных данных [2, с. 312–315]. Событийно-ориентированные архитектуры успешно решают проблему масштабирования, но приводят к потере прозрачности сквозных процессов, поскольку логика их выполнения оказывается распределенной по множеству независимых компонентов [3, с. 178–181].

Особого внимания заслуживает анализ решений с разделением плоскости управления и плоскости данных (TIBCO Control Plane, Tufin, VAST Polaris и аналоги). Установлено, что данные решения реализуют искомый принцип разделения, однако их функциональность преимущественно ограничена сетевым уровнем и управлением API-шлюзами, в то время как задача оркестрации гетерогенных программно-аппаратных ресурсов предприятия и управления сквозными бизнес-процессами остается за рамками их применимости [6, с. 142–145]. Кроме того, ни один из рассмотренных подходов не предусматривает явного выделения формализованных структур данных, описывающих желаемое состояние системы, что вынуждает распределять логику управления по конфигурационным файлам и программному коду, усложняя эволюцию ИТ-ландшафта [8, с. 245–248].

2. Архитектура универсального модуля-контроллера.

Основным результатом проектной части исследования является разработанная архитектура универсального модуля-контроллера, обеспечивающая централизованное управление сквозными бизнес-процессами при децентрализованной передаче данных. В составе модуля выделены четыре функциональных компонента:

Оркестратор процессов является компонентом, отвечающим за интерпретацию формализованных описаний бизнес-процессов (в нотации BPMN), управление последовательностью выполнения шагов, обработку условных переходов и вызов внешних сервисов. В качестве технологической основы для реализации оркестратора могут выступать зрелые BPM-движки с открытым исходным кодом, поддерживающие спецификацию BPMN 2.0 [1, с. 156–159].

Контроллер состояния компонентов является компонентом, осуществляющим мониторинг доступности и работоспособности физических серверов, прикладных систем, интеграционных шин и иных элементов инфраструктуры. Контроллер поддерживает актуальное представление о состоянии компонентов в реестре компонентов и инициирует процедуры восстановления при обнаружении отклонений от желаемого состояния [7, с. 203–206].

Менеджер ресурсов является компонентом, реализующим функции распределения нагрузки между доступными экземплярами сервисов, управления пулами соединений и перераспределения ресурсов при изменении конфигурации инфраструктуры.

Роутер является компонентом, осуществляющим маршрутизацию управляющих команд к соответствующим элементам плоскости данных на основе информации, содержащейся в реестре компонентов.

Ключевым архитектурным принципом, отличающим предлагаемое решение от классических подходов, является строгое разделение плоскости управления и плоскости данных [4, с. 125–128]. Плоскость управления представлена самим модулем-контроллером и

отвечает за принятие решений об оркестрации, контроль состояния и распределение ресурсов. Плоскость данных образована существующими в организации интеграционными шинами, брокерами сообщений и API-шлюзами, которые выполняют команды модуля-контроллера по передаче полезных данных между системами, но не принимают самостоятельных решений о маршрутизации процессов. Такое разделение позволяет избежать возникновения узкого места, характерного для ESB, при сохранении централизованного контроля, свойственного сервис-ориентированным архитектурам.

3. Формализованные структуры данных.

В рамках исследования введены и формализованы две ключевые структуры данных, обеспечивающие реализацию управления на основе состояния (Desired State Management).

Реестр компонентов — структурированное описание всех элементов ИТ-ландшафта, участвующих в интеграционном взаимодействии. Для каждого компонента фиксируются: идентификатор, тип (физический сервер, прикладная система, интеграционная шина, брокер сообщений), сетевой адрес, поддерживаемые протоколы и форматы данных, интерфейсы управления, текущее состояние (доступен, недоступен, degraded), а также метаданные (версия, время последнего опроса, накопленная статистика отказов) [7, с. 78–82].

Реестр задач — формализованное описание желаемого состояния системы, включающее: перечень сервисов, которые должны быть запущены и доступны; спецификации сквозных бизнес-процессов в нотации BPMN; маппинг шагов процессов на компоненты из реестра компонентов; политики восстановления при сбоях (стратегии повторных попыток, таймауты, альтернативные маршруты выполнения); приоритеты процессов и квоты на использование ресурсов.

Введение указанных структур данных является одним из элементов научной новизны работы, поскольку в классических интеграционных решениях подобная формализация отсутствует, а информация о компонентах и логика управления распределены по множеству конфигурационных артефактов, что затрудняет автоматическое восстановление после сбоев и усложняет внесение изменений.

4. Методы и стандарты взаимодействия.

Для обеспечения однозначности и формализуемости взаимодействия между модулем-контроллером и плоскостью данных определен следующий набор стандартов и нотаций:

Для описания статической архитектуры предлагаемого решения рекомендовано использование нотации ArchiMate (на уровнях бизнес-слоя, прикладного слоя и технологического слоя) и модели C4 (на уровнях контейнеров и компонентов), что позволяет представить архитектуру в виде, понятном как специалистам по интеграции, так и более широкой аудитории [2, с. 412–415].

Для спецификации синхронных управляющих интерфейсов модуля-контроллера предложено использование стандарта OpenAPI, обеспечивающего формальное описание эндпоинтов, форматов запросов и ответов, кодов состояния и механизмов аутентификации [8, с. 178–181].

Для описания асинхронных взаимодействий (например, при получении статусных уведомлений от плоскости данных) рекомендовано применение спецификации AsyncAPI, позволяющей формализовать топики, типы событий и схемы передаваемых данных.

Для моделирования сквозных бизнес-процессов, исполняемых под управлением оркестратора, в качестве базовой нотации выбран BPMN 2.0, предоставляющий развитые средства для описания последовательных и параллельных ветвлений, событий, таймеров и обработки исключительных ситуаций [1, с. 98–101].

5. Ключевые сценарии функционирования.

В ходе проектирования разработаны описания ключевых сценариев работы предлагаемой интеграционной технологии, охватывающие как штатные, так и исключительные ситуации.

Штатный сценарий предполагает, что все компоненты ИТ-ландшафта функционируют в пределах заданных эксплуатационных параметров. При поступлении запроса на выполнение сквозного бизнес-процесса оркестратор модуля-контроллера интерпретирует соответствующую BPMN-модель из реестра задач, определяет последовательность шагов и для каждого шага, используя данные реестра компонентов, формирует управляющую команду, адресованную соответствующему элементу плоскости данных. Плоскость данных (интеграционная шина или брокер сообщений) выполняет команду, осуществляя передачу полезных данных между системами, и по завершении отправляет статусное уведомление модулю-контроллеру. Оркестратор, получив подтверждение успешного выполнения шага, переходит к следующему шагу процесса. Таким образом, модуль-контроллер не участвует в передаче данных, но сохраняет полную информацию о состоянии каждого экземпляра процесса.

Сценарий отказа отдельного сервера или прикладной системы предусматривает, что контроллер состояния компонентов периодически опрашивает элементы инфраструктуры и фиксирует их доступность в реестре компонентов. При обнаружении недоступности компонента контроллер инициирует процедуру восстановления в соответствии с политиками, определенными в реестре задач. Если для отказавшего компонента предусмотрен резервный экземпляр, менеджер ресурсов перенаправляет нагрузку на него, обновляя соответствующие записи в реестре компонентов. Если резервный экземпляр отсутствует, оркестратор приостанавливает выполнение процессов, зависящих от отказавшего компонента, и инициирует эскалацию проблемы администратору.

Сценарий недоступности интеграционной шины рассматривает ситуацию, когда элемент плоскости данных, ответственный за передачу полезных данных, оказывается недоступен. Модуль-контроллер, обнаружив отсутствие статусных уведомлений от шины в течение заданного таймаута, фиксирует ее состояние как «недоступно» и приостанавливает отправку новых команд. При этом оркестратор сохраняет состояние всех незавершенных процессов в персистентном хранилище. После восстановления доступности шины выполнение процессов возобновляется с точки последнего зафиксированного состояния, что исключает потерю данных и нарушение целостности процессов.

Сценарий отказа самого модуля-контроллера является наиболее критическим и требует реализации кластеризации. В кластерной конфигурации несколько экземпляров модуля-контроллера работают в режиме active-passive или active-active, используя распределенный координатор для выбора лидера и синхронизации состояния [4, с. 178–181]. При отказе активного экземпляра координатор инициирует процедуру выбора нового лидера, который принимает на себя управление, используя актуальное состояние из реестра компонентов и реестра задач, сохраненное в распределенном хранилище. Время восстановления в такой конфигурации определяется параметрами координатора и может составлять от нескольких секунд до десятков секунд, что приемлемо для большинства корпоративных сценариев.

6. Сравнительная оценка предложенного решения.

Для верификации полученных результатов проведена сравнительная оценка предлагаемой архитектуры с существующими подходами по пяти критериям: управляемость, масштабируемость, гибкость, сложность внедрения и устойчивость к сбоям [5]. Результаты сравнения представлены в таблице 1.

Таблица 1 — Сравнительная оценка интеграционных подходов

Критерий / Подход	Точечная интеграция	ESB	EDA	Предлагаемое решение
Управляемость	Низкая	Высокая	Низкая	Высокая
Масштабируемость	Низкая	Средняя	Высокая	Высокая
Гибкость	Низкая	Средняя	Высокая	Высокая
Сложность внедрения	Низкая (на старте)	Высокая	Средняя	Средняя
Устойчивость к сбоям	Низкая	Средняя	Высокая	Высокая

Как следует из таблицы, предлагаемое решение сочетает высокую управляемость, свойственную ESB, с высокой масштабируемостью и устойчивостью к сбоям, характерными для EDA, что подтверждает достижение поставленной цели исследования. При этом сложность внедрения оценивается как средняя, что обусловлено необходимостью первоначальной формализации реестров компонентов и задач, однако в долгосрочной перспективе эти затраты компенсируются снижением операционных расходов на поддержку интеграционной инфраструктуры.

Обсуждение результатов. Полученные результаты демонстрируют, что предложенный архитектурный подход позволяет преодолеть фундаментальное противоречие между централизацией управления и децентрализацией передачи данных, характерное для существующих интеграционных решений. Разделение плоскости управления и плоскости данных, реализованное на уровне корпоративной оркестрации, а не только на сетевом уровне или уровне API-шлюзов, расширяет область применимости данного принципа и создает основу для построения единого информационного пространства в гетерогенных ИТ-ландшафтах.

Введение формализованных структур данных — реестра компонентов и реестра задач — обеспечивает переход от императивного управления (при котором модуль-контроллер должен «знать», какие именно команды и в каком порядке следует отдавать для достижения цели) к декларативному управлению на основе желаемого состояния (при котором модуль-контроллер сравнивает текущее состояние системы с желаемым и инициирует действия по устранению расхождений). Такой подход зарекомендовал себя в области оркестрации контейнеризованных приложений (Kubernetes) и в рамках данного исследования впервые систематически применен к задаче интеграции корпоративных приложений [4, с. 125–128].

Следует отметить, что предложенная архитектура не требует отказа от уже развернутых в организации интеграционных решений. Напротив, существующие ESB, брокеры сообщений и API-шлюзы могут быть задействованы в качестве плоскости данных при условии наличия у них интерфейсов управления, соответствующих спецификациям OpenAPI или AsyncAPI. Это снижает порог входа и уменьшает начальные инвестиции, необходимые для внедрения предлагаемой технологии.

Ограничением настоящего исследования является его архитектурно-методологический характер, предполагающий отсутствие программной реализации и количественных оценок производительности. Дальнейшие исследования могут быть направлены на создание референтной реализации модуля-контроллера с использованием открытого программного обеспечения, проведение нагрузочного тестирования и разработку методики количественной оценки экономической эффективности внедрения предлагаемого решения в конкретных организационных контекстах.

Выводы

Проведенное исследование, посвященное разработке архитектуры и методов интеграционной технологии для построения единого информационного пространства в условиях гетерогенного ИТ-ландшафта, позволяет сформулировать следующие выводы.

В ходе анализа предметной области установлено, что современные организационные среды характеризуются одновременным присутствием унаследованных систем с закрытыми протоколами, современных прикладных платформ, облачных сервисов и распределенных компонентов, что порождает комплекс платформенных, синтаксических и семантических барьеров интеграции [2, с. 45–47; 3, с. 45–48]. Следствием указанной разнородности становятся разрывы в сквозных бизнес-процессах, вынужденная практика ручной передачи данных, отсутствие единой видимости состояния процессов и высокие операционные затраты на поддержку точечных интеграций [6, с. 271–273]. Сравнительный анализ существующих интеграционных подходов — точечной интеграции, сервисной шины предприятия, API-led архитектуры, событийно-ориентированной архитектуры и гибридных платформ — выявил их принципиальную ограниченность: подходы, централизующие управление, неизбежно создают узкое место по производительности и единую точку отказа, тогда как децентрализованные подходы, обеспечивая высокую масштабируемость, приводят к потере прозрачности и управляемости сквозных процессов [2, с. 312–315; 3, с. 178–181]. Решения с разделением плоскости управления и плоскости данных, существующие на рынке, делают важный шаг в направлении разрешения данного противоречия, однако их функциональность преимущественно ограничена сетевым уровнем и управлением API-шлюзами, в то время как задача комплексной оркестрации гетерогенных программно-аппаратных ресурсов предприятия остается за их рамками [6, с. 142–145].

Основным научным результатом работы является разработанная архитектура универсального модуля-контроллера, в которой впервые на уровне корпоративной интеграции реализовано строгое разделение плоскости управления и плоскости данных. Плоскость управления, представленная модулем-контроллером, выполняет функции оркестрации бизнес-процессов, контроля состояния компонентов и распределения ресурсов, в то время как плоскость данных, образования существующими интеграционными шинами и брокерами сообщений, отвечает исключительно за передачу полезных данных между системами, не принимая самостоятельных решений о маршрутизации. Такое разделение позволяет преодолеть фундаментальное противоречие между управляемостью и масштабируемостью, характерное для классических подходов [4, с. 125–128]. В составе модуля-контроллера выделены четыре функциональных компонента: оркестратор процессов, контроллер состояния компонентов, менеджер ресурсов и роутер, каждый из которых имеет четко определенную зону ответственности и интерфейсы взаимодействия.

Элементом научной новизны обладает введение и формализация двух ключевых структур данных — реестра компонентов и реестра задач, — которые обеспечивают переход от императивного управления к декларативному управлению на основе желаемого состояния. Реестр компонентов содержит структурированное описание всех физических и программных элементов инфраструктуры, их интерфейсов, поддерживаемых протоколов и текущего состояния доступности. Реестр задач включает формализованное описание желаемого состояния системы, спецификации сквозных бизнес-процессов в нотации BPMN, маппинг шагов процессов на компоненты и политики восстановления при сбоях [7, с. 78–82, 203–206]. В отличие от классических интеграционных решений, где логика управления распределена по множеству конфигурационных артефактов или «зашита» в программный код, предлагаемый подход обеспечивает централизованное представление о целевом

состоянии системы и механизмы автоматического приведения текущего состояния к желаемому.

В работе определены и обоснованы методы и стандарты, обеспечивающие формализацию и однозначность описания предлагаемой архитектуры. Для моделирования статической архитектуры рекомендовано использование нотаций ArchiMate и C4 Model, позволяющих представить решение на различных уровнях детализации — от контекста системы до отдельных компонентов [2, с. 412–415]. Для спецификации интерфейсов взаимодействия между модулем-контроллером и плоскостью данных предложено применение стандартов OpenAPI для синхронных вызовов и AsyncAPI для асинхронных уведомлений, что обеспечивает независимость архитектурного описания от конкретных программных реализаций [8, с. 178–181]. В качестве нотации моделирования сквозных бизнес-процессов выбран BPMN 2.0, предоставляющий развитые средства для описания последовательных и параллельных ветвлений, обработки событий и исключительных ситуаций [1, с. 98–101].

Спроектированные ключевые сценарии функционирования предлагаемой технологии, охватывающие как штатные режимы выполнения процессов, так и исключительные ситуации (отказы отдельных серверов, недоступность интеграционных шин, сбои самого модуля-контроллера), подтверждают работоспособность и отказоустойчивость архитектурного решения. Показано, что кластеризация модуля-контроллера с использованием распределенного координатора обеспечивает автоматическое восстановление управления при отказе активного экземпляра, а наличие резервных мощностей в плоскости данных позволяет перенаправлять потоки данных при недоступности отдельных компонентов [4, с. 178–181].

Сравнительная оценка предложенного архитектурного подхода с существующими альтернативами по критериям управляемости, масштабируемости, гибкости, сложности внедрения и устойчивости к сбоям демонстрирует, что разработанное решение сочетает высокую управляемость, свойственную сервисной шине предприятия, с высокой масштабируемостью и отказоустойчивостью, характерными для событийно-ориентированных архитектур. При этом сложность внедрения оценивается как средняя, что обусловлено необходимостью первоначальной формализации реестров компонентов и задач, однако в долгосрочной перспективе эти инвестиции компенсируются снижением операционных затрат на поддержку и развитие интеграционной инфраструктуры [5].

Практическая значимость полученных результатов определяется тем, что разработанное архитектурное и методологическое описание может служить основой для проектирования и внедрения интеграционных решений на предприятиях различных отраслей, сталкивающихся с проблемой объединения гетерогенных информационных систем в единое информационное пространство. Предложенная архитектура не требует отказа от уже развернутых интеграционных компонентов и может быть реализована с использованием открытого программного обеспечения, что снижает зависимость от конкретных вендоров и уменьшает совокупную стоимость владения. Перспективы дальнейших исследований связаны с созданием референтной программной реализации модуля-контроллера, проведением нагрузочного тестирования для получения количественных оценок производительности и отказоустойчивости, а также с разработкой методики оценки экономической эффективности внедрения предлагаемого решения в конкретных организационных контекстах.

Список литературы:

1. Зараменских Е. П. Архитектура предприятия: учебник для вузов / Е. П. Зараменских, Д. В. Кудрявцев, М. Ю. Арзуманян ; под редакцией Е. П. Зараменских. — 2-е изд., перераб. и доп. — Москва : Юрайт, 2025. — 433 с.
2. Нил Ф. Современный подход к программной архитектуре. Сложные компромиссы / Ф. Нил, Р. Марк, С. Прамод, Д. Жамак ; пер. с англ. А. Н. Киселева. — Санкт-Петербург : Питер, 2023. — 480 с. — (Серия «Для профессионалов»).
3. Беллемар А. Создание событийно-управляемых микросервисов / А. Беллемар ; пер. с англ. А. Н. Киселева. — Санкт-Петербург : БХВ-Петербург, 2024. — 320 с.
4. Баланов А. Н. DevOps: Интеграция и автоматизация : учебное пособие для вузов / А. Н. Баланов. — 2-е изд., стер. — Санкт-Петербург : Лань, 2025. — 240 с.
5. Hohpe G. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions / G. Hohpe, B. Woolf. — Boston : Addison-Wesley Professional, 2003. — 736 p.
6. Bannour F. Software-Defined Networking 2: Extending SDN Control to Large-Scale Networks / F. Bannour, S. Souihi, A. Mellouk. — London : ISTE Ltd, 2023. — 160 p.
7. Johnson R. Service-Oriented Architecture Design and Patterns: Definitive Reference for Developers and Engineers / R. Johnson. — New York : Nextory, 2025. — 350 p.
8. Richardson C. Microservices Patterns / C. Richardson. — New York : Manning Publications, 2025. — 500 p.

References:

1. Zaramenskikh E. P., Kudryavtsev D. V., Arzumanyan M. Yu. Enterprise Architecture: Textbook for Universities. 2nd ed., rev. and enl. Moscow: Yurait Publ., 2025. 433 p. (In Russ.)
2. Neal F., Mark R., Pramod S., Jhamak D. Modern Approach to Software Architecture: Complex Trade-offs. Transl. from Engl. by A. N. Kiselev. St. Petersburg: Piter Publ., 2023. 480 p. (In Russ.)
3. Bellemare A. Building Event-Driven Microservices. Transl. from Engl. by A. N. Kiselev. St. Petersburg: BHV-Petersburg Publ., 2024. 320 p. (In Russ.)
4. Balanov A. N. DevOps: Integration and Automation: Textbook for Universities. 2nd ed., ster. St. Petersburg: Lan' Publ., 2025. 240 p. (In Russ.)
5. Hohpe G., Woolf B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Boston: Addison-Wesley Professional, 2003. 736 p.
6. Bannour F., Souihi S., Mellouk A. Software-Defined Networking 2: Extending SDN Control to Large-Scale Networks. London: ISTE Ltd, 2023. 160 p.
7. Johnson R. Service-Oriented Architecture Design and Patterns: Definitive Reference for Developers and Engineers. New York: Nextory, 2025. 350 p.
8. Richardson C. Microservices Patterns. New York: Manning Publications, 2025. 500 p.