Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский авиационный институт (национальный исследовательский университет)»

На правах рукописи

Никонов Юрий Юрьевич

РАЗРАБОТКА ПРОБЛЕМНО-ОРИЕНТИРОВАННОЙ СИСТЕМЫ ИНФОРМАЦИОННОЙ ПОДДЕРЖКИ РЕШЕНИЙ ДЛЯ КОНТРОЛЯ ПОДЛИННОСТИ ПАСПОРТИЗИРОВАННЫХ КОМПОНЕНТОВ ВОЗДУШНЫХ СУДОВ

Специальность

2.3.1 Системный анализ, управление и обработка информации, статистика

(технические науки)

Диссертация на соискание ученой степени

кандидата технических наук

Научный руководитель:

Доктор технических наук, старший научный сотрудник

Буряк Юрий Иванович

Москва – 2025

Оглавление

ВВЕДЕНИЕ

Актуальность темы исследования. Цифровая трансформация производственных отношений рождает ряд новых задач, в том числе в сфере интеграции разнородных источников данных предприятий – участников жизненного цикла (ЖЦ) сложной научно-технической продукции. Такие задачи возникают, например, в инспекционном контроле летной годности (ЛГ) воздушных судов (ВС), включающем комплексную проверку свойств ВС для обеспечения безопасности полетов с последующей выдачей или возобновлением сертификата летной годности (СЛГ). Снижение затрат на процедуры контроля летной годности при сохранении должного уровня безопасности полётов можно рассматривать в качестве важного условия конкурентоспособности отечественных ВС на мировом и российском рынке авиаперевозок.

Однако в силу многообразия типов ВС, а следовательно, и процессов их эксплуатации, а также видов информационных систем (ИС) сопровождения эксплуатации ВС, выполнение требований нормативных документов по согласованному сбору сведений для контроля летной годности приводит к возрастанию времени (сроков) проведения инспекционного контроля ВС, т.е. в конечном итоге к повышению затрат.

Повышение эффективности информационного сопровождения эксплуатации ВС невозможно без разработки гибкой информационной системы. В единое информационное пространство (ЕИП) данная система объединит все заинтересованные стороны, осуществляющие контроль летной годности ВС, а именно: источники информации о легальности производителей, ремонтных и эксплуатирующих организаций, а также данные о фактическом состоянии ВС.

В настоящее время задачи информационного обеспечения эксплуатации ВС уже решаются на основе создания ряда таких систем: АСУ ПЛГ ВС, CAMP, «my Boeing fleet», AirnavX, ИУС «Эрлан-3», AMOS, IBM Maximo for Aviation MRO, Ramco MRO Aviation, SAM, Veryon и др.

Функциональные возможности таких систем полностью отвечают требованиям информационного сопровождения эксплуатации ВС, однако не

поддерживают вопросы сбора актуальных, полных и разнородных сведений о текущем состоянии их паспортизированных компонентов состоящих из паспортов и изделия (далее компонентов) за минимальное время (критерий минимума данного параметра), например: заданной и фактической периодичности обслуживания, остаточного ресурса, корректности заполнения удостоверяющей документации, использование электронной и «бумажной» форм и пр.

Отметим, что сбор данных о паспортизированных компонентах включает ряд независимых операций в части паспорта и изделия с последующей интеграцией полученных сведений для подтверждения их подлинности, что ведет к необходимости углубления контроля и увеличения сроков проведения работ.

Дополнительные сложности вносит наличие ограничений в доступе по признаку «коммерческая тайна», а также необходимость проверки на достоверность, что требует организации дополнительных (перекрестных) запросов.

Свой вклад в развитие технологий интеграции и инструментов работы в ЕИП внесли как отечественные (М.В. Кулагин, В.А. Серебряков, Глушков М.В, Лебедев А.С, Ершов А.П., Брук И.С, Канторович Л.В.), так и зарубежные ученые (Делвин Г., Фукс-Китовски Ф., Винер Н., Кнут Д., Ритчи Д., Страуструп Б., Тордвальдс С.).

Однако их работы носят во многом теоретический характер, рассматривая общие вопросы минимизации времени взаимодействия разнородных источников путем оптимизации трафика Internet сетей, увеличения их пропускной способности, а также интеграции данных на семантическом уровне, интеграции метаданных и пр.

Специфические особенности рассматриваемой задачи (параметры разрабатываемой системы), обусловленные необходимостью интеграции структурированных/не структурированных источников данных самых разных форматов, в том числе баз данных (SQL/NoSQL), совокупности таблиц или просто файлов разных форматов (doc, xml, xls, pdf csv, json и пр) за минимальное время

обуславливают необходимость решения задачи синтеза проблемно-ориентированной системы (ПОС), способной формировать множественные запросы к территориально распределенным источникам разнородных данных, используя комбинацию ETL (extract, transform, load технология) и Web (RESTful, SOAP, JSON, GraphQL и прочих технологий обмена данными) сервисов.

Таким образом, актуальность работы определяется необходимостью решения задачи синтеза новой проблемно-ориентированной системы как нового инструмента создания ЕИП, обеспечивающей выполнение всех требований по качественным и количественным характеристикам данных (скорости передачи и объема) при контроле летной годности ВС, что, в конечном итоге, позволит снизить эксплуатационные затраты.

Целью диссертационной работы является повышение технико-экономической эффективности процессов интеграции разнородных информационных ресурсов для контроля подлинности паспортизированных компонентов ВС при эксплуатации авиационной техники за счет создания новой проблемно-ориентированной системы информационной поддержки решений.

Для достижения цели работы были поставлены и решены следующие **основные задачи:**

1. Анализ существующих подходов к контролю подлинности паспортизированных компонентов ВС, выявление направлений повышения эффективности процессов сбора, обработки и передачи данных.

2. Разработка модели описания проблемно-ориентированной системы информационной поддержки решений.

3. Разработка метода и алгоритмов синтеза проблемно-ориентированной системы информационной поддержки решений для существующих условий сбора, обработки и передачи данных, основными из которых являются:

- алгоритм оптимизации параметров проблемно-ориентированной системы, обеспечивающий организацию информационного взаимодействия разнородных источников/потребителей данных за минимальное время;

- разработка метода выбора характеристик проблемно-ориентированной системы с учетом параметров интегрируемых источников/потребителей данных.

4. Разработка информационной модели и программного комплекса, позволяющего на основе обработки информации об источниках/потребителях данных производить структурно-параметрический синтез проблемно-ориентированной системы.

5. Экспериментальное апробирование программного комплекса, реализующего предложенные модель, метода и алгоритмы.

При решении перечисленных выше задач **получены следующие новые научные результаты, выносимые на защиту:**

1. Алгоритм оптимизации параметров проблемно-ориентированной системы, обеспечивающий организацию информационного взаимодействия разнородных источников/потребителей данных за минимальное время.

2. Метод выбора характеристик проблемно-ориентированной системы с учетом структуры и параметров интегрируемых источников/потребителей данных.

3. Информационная модель, позволяющая на основе обработки информации об источниках/потребителях данных производить структурно-параметрический синтез проблемно-ориентированной системы.

4. Результаты применения программного комплекса, реализующего предложенные алгоритмы, метод и информационную модель.

Объект исследования – проблемно-ориентированная система информационной поддержки решений, состоящая из ETL- сервисов, участков передачи данных, различных видов баз данных и Web-сервисов для передачи данных во внешнюю среду.

Предмет исследования – характеристики баз данных, типов хранимых данных, а также технологий передачи данных в систему (ETL-сервисы) и из нее (Web-сервисы).

Научная новизна работы состоит в разработке оригинальных подходов к созданию новой проблемно-ориентированной системы, обеспечивающей организацию информационного взаимодействия разнородных источников/потребителей данных за минимальное время, а именно:

- разработан алгоритм оптимизации параметров проблемно-ориентированной системы, обеспечивающий организацию информационного взаимодействия разнородных источников и потребителей данных за минимальное время;
- разработан метод выбора характеристик проблемно-ориентированной системы на основе обработки информации о структуре и параметрах интегрируемых источников/потребителей данных;
- разработана информационная модель системы, позволяющая на основе обработки информации об источниках/потребителях данных производить структурно-параметрический синтез проблемно-ориентированной системы;
- разработан программный комплекс системы, позволяющий на основе обработки информации о структуре и параметрах интегрируемых источников/потребителей данных синтезировать проблемно-ориентированную систему.

Методы исследования. Для решения поставленных в работе задач использовались: методы системного анализа, методы динамического программирования, математической статистики.

Обоснованность и достоверность полученных результатов обеспечивается корректным использованием методов системного анализа, динамического программирования, также объектно-ориентированного анализа и проектирования информационных систем.

Теоретическая значимость состоит в совершенствовании метода и алгоритмов оптимизации параметров проблемно-ориентированной системы, обеспечивающей организацию информационного взаимодействия множественных разнородных источников/потребителей данных в режиме on-line.

Практическая значимость полученных научных результатов заключается в создании на их основе программного комплекса для реализации системы, обеспечивающей организацию информационного взаимодействия разнородных источников/потребителей данных на разных этапах жизненного цикла воздушных судов.

Практическая значимость подтверждается актом внедрения основных научных результатов диссертации в ООО «Научно-производственный центр «Бизнесавтоматика».

Достоверность полученных результатов обеспечивается корректным использованием: методов системного анализа, методов динамического программирования, математической статистики.

Апробация работы. Основные результаты диссертационной работы обсуждались и докладывались на следующих научных конференциях:

1. XLVI Международная молодёжная научная конференция «Гагаринские чтения» (Москва, МАИ, 2020)
2. 20-я Международная конференция «Авиация и космонавтика». (Москва, МАИ, 2021)
3. Программно-техническое обеспечение автоматизированных систем: Всероссийская молодежная научно-практическая конференция (Барнаул, 2021)
4. Цифровая трансформация социальных и экономических систем: международная научно-практическая конференция (Москва, 2022)
5. XLVIII Международная молодёжная научная конференция «Гагаринские чтения» (Москва, МАИ, 2022)
6. 22-я Международная конференция «Авиация и космонавтика». (Москва, МАИ, 2023).

Личный вклад. Автору принадлежит модель проблемно-ориентированной системы, а также ее техническое воплощение. Все исследования проведены самостоятельно. В основе работы лежит учет специфик различных типов систем хранения и передачи данных, их совместимость.

Публикации. Основные результаты работ представлены в десяти печатных работах, четыре из которых изданы в журналах, рекомендованных ВАК:

1. Никонов Ю. Ю., Столярчук В.А. Исследование применения технологий ESB и GraphQL в интеграции государственных автоматизированных информационных систем. Автоматизация. Современные технологии. 2022. Т. 76. № 8. С. 375-378.
2. Никонов Ю.Ю. Сравнение импорта данных из различных типов файлов в реляционные и нереляционные базы данных. Научно-технический вестник Поволжья. 2023. № 1. С. 81-84.
3. Никонов Ю. Ю. Исследование применения технологии etl в интеграции информационных систем. Научно-технический вестник Поволжья. 2023. № 10. С. 150-152.
4. Буряк Ю.И., Никонов Ю.Ю. Совершенствование процессов обеспечения летной годности ВС за счет создания высокоскоростной гетерогенной информационной системы. Вестник компьютерных и информационных технологий. 2024. Т. 21, № 6. С. 31 – 40.

Остальные шесть печатных работ размещены в тезисах докладов соответствующих научных конференций.

В первой главе проведен анализ существующих подходов к подтверждению подлинности компонентов воздушных судов. Показана необходимость создания новой системы, удовлетворяющей заданным требованиям для контроля летной годности ВС с обозначенным функционалом.

Во второй главе разработана модель описания проблемно-ориентированной системы. Разработан алгоритм оптимизации параметров проблемно-ориентированной системы, обеспечивающий организацию информационного взаимодействия разнородных источников и потребителей данных за минимальное время.

Предложенная модель отличается наличием дополнительных сущностей и атрибутов совместимости, необходимых для расчета оптимальной по критерию

минимального времени цепочки участков для прохождения сигнала с учетом совместимости сервисов.

В третьей главе автором была предложена структура программного комплекса и алгоритм его работы. Разработан программный комплекс системы, позволяющий на основе обработки информации о структуре и параметрах интегрируемых источников/потребителей данных синтезировать проблемно-ориентированную систему.

Отличием программного комплекса является возможность работы в двух режимах: настройки и эксплуатации. В первом происходит формирование характеристик системы, а во втором выполнение запросов на основании источников и выбранных ранее характеристик.

В заключении сформулированы выводы и основные результаты диссертационной работы.

Объем и структура работы. Работа состоит из оглавления, введения, трех глав, заключения, списка литературы из 105 позиций, 35 рисунков, 9 таблиц. Объем работы составляет 105 страниц машинописного текста.

1. АНАЛИЗ СУЩЕСТВУЮЩИХ ПОДХОДОВ К ПОДТВЕРЖДЕНИЮ ПОДЛИННОСТИ КОМПОНЕНТОВ ВОЗДУШНЫХ СУДОВ

1.1 Анализ развития подходов в области интеграции проблемно-ориентированных информационных систем

Разнообразие типов воздушных судов, а также связанных с ними процессов эксплуатации [1] и многочисленных участников, вовлеченных в этот процесс [2], приводит к сложности исполнения требований нормативных документов [3,4] по согласованию всех сторон, участвующих в информационном сопровождении эксплуатации воздушных судов. В свою очередь это ведет к увеличению сроков сертификации ВС и, следовательно, увеличению затрат. Схема организации запросов от участников согласования сертификационной документации представлена на рисунке 1.1 .

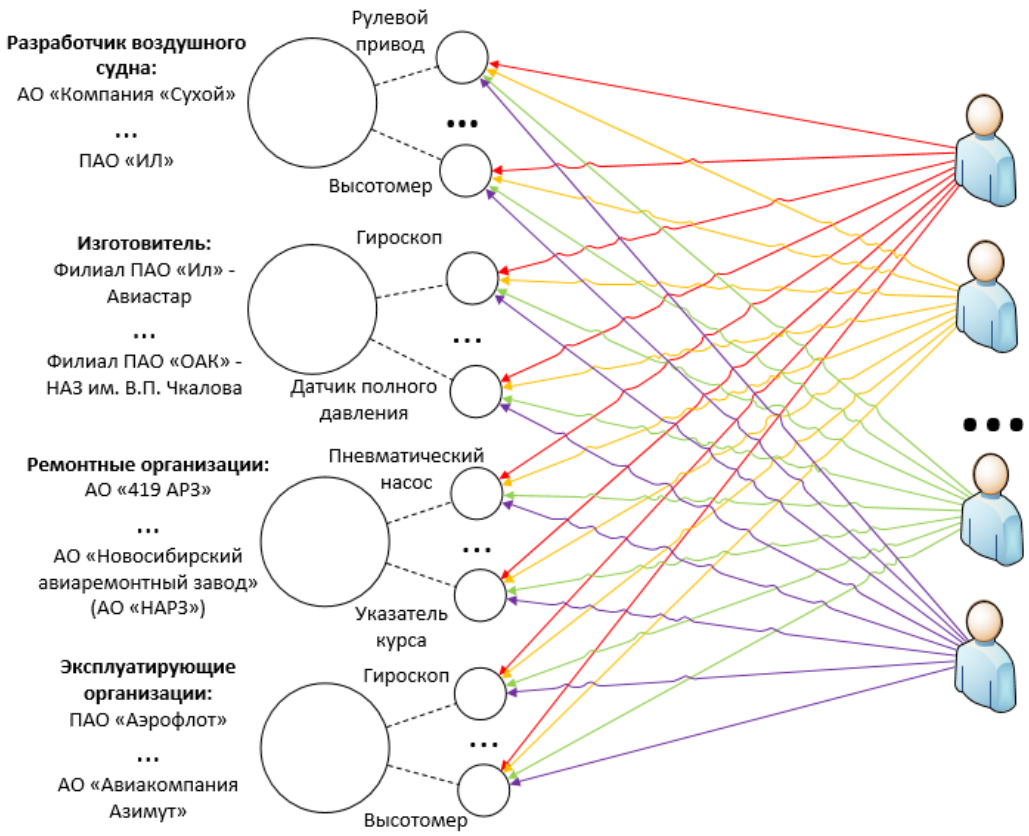

Рисунок 1.1. Текущая схема организации запросов к информационным системам

Описанные выше факторы являются основанием к созданию инфраструктуры информационных технологий (ИТ-инфраструктуры), которая должна обладать следующими характеристиками:

- множеством информационных систем, разработанных и созданных различными независимыми производителями с использованием разнообразных технологий;

- отсутствием единой модели данных и целостных процессов их обработки;

- значительным количеством баз данных с несвязанной и дублирующийся информацией;

- несовместимыми и нестандартизированными методами интеграции в рамках всей инфраструктуры.

В результате появляется проблема значительного увеличения трудоёмкости, стоимости и времени внедрения новых информационных систем, с сохранением средств, затраченных на создание существующей ИТ-инфраструктуры.

Для решения этих проблем требуется выбор стандартизированной и масштабируемой модели интеграции, позволяющей внедрять новые информационные системы с минимальными потерями производительности и изменениями в ИТ-инфраструктуре. Предлагается схема организации запросов к информационным системам. Она показана на рисунке 1.2.

Согласно статистике, до 62 % состава ИТ-руководителей назвали интеграцию устаревших систем самой большой проблемой при переходе к использованию облачными технологиям [5].

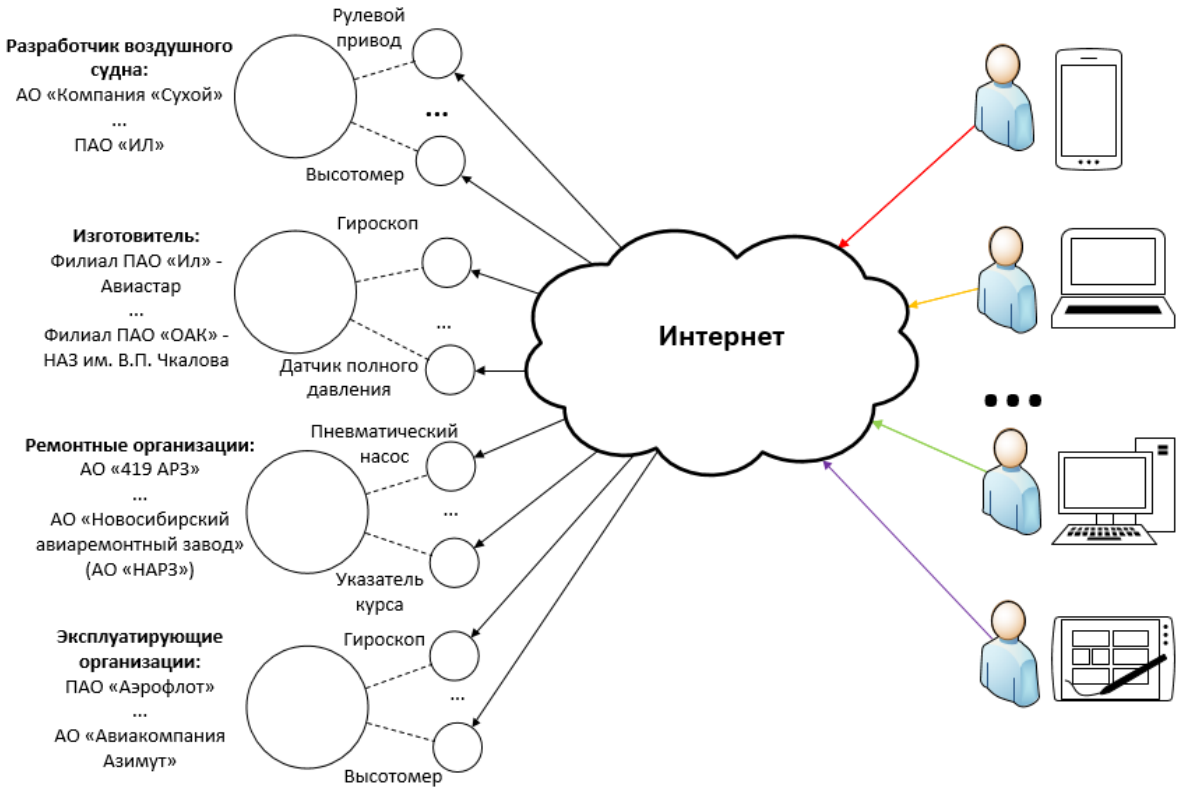

Рисунок 1.2. Предлагаемая схема организации запросов к информационным системам

Информационная система, основанная на устаревших технологиях и играющая ключевую роль в ежедневных операциях, называется «унаследованной системой» (legacy system)[6].

Обычно средний жизненный цикл большинства программных систем варьируется от 10 до 15 лет [7]. Однако, согласно данным счётной палаты правительства США, жизненный цикл десяти наиболее важных устаревших систем федерального уровня, которые требуют модернизации, меняется в диапазоне от 8 до 51 года. Их совокупные затраты на эксплуатацию и обслуживание составляют около 337 миллионов долларов в год [8].

В предшествующие несколько десятилетий компании активно использовали ряд прогрессивных, но ныне устаревших технологий (мейнфреймы, языки программирования Fortran и COBOL, операционные системы от Microsoft, а именно MSDOS, Windows 3.1, XP и т. д.). Эти технологии по-прежнему применяются лишь ограниченным числом компаний, что связано с ранее сделанными значительными инвестициями, опасением перед переменами,

необходимостью поддержания текущей деятельности и сложностями при внедрении новых решений. К тому же, эти системы по-прежнему отвечают потребностям компаний, и сотрудники их хорошо освоили [9, 10]. С момента, как системы внедрены в их хранилищах накопилось много важной бизнес-информации, в том числе в форматах, не поддерживающих или затрудняющих конвертацию [11].

Указанные типы систем существенно усложняют процесс адаптации организаций к глобальным изменениям, необходимым для их развития. Высокая стоимость является фактором усложнения технического обслуживания, а также нехватка квалифицированных специалистов, которые должны разбираться в этих системах. Еще одним фактором представляет собой недостаток надлежащей документации [12, 13]. Кроме того, устаревшие технологии [14] являются препятствием к масштабированию и интеграции с новыми информационными системами. В этом случае получается, что инновационные усилия организации сдерживаются из-за этих систем [15].

В настоящее время, в связи с активным внедрением современных цифровых технологий в современные производственные процессы, обновление устаревших систем становится критически важным для обеспечения пользователей конкурентоспособными и инновационными продуктами и сервисами [16]. В свою очередь, модернизация систем открывает возможности для применения новых и прорывных технологий, таких как искусственный интеллект, высокопроизводительные вычисления, облачные технологии, Интернет вещей, робототехника и большие данные [17].

К сожалению, решение о необходимости модернизации практически везде принимается руководителями или ведущими специалистами, а в расчет, при этом, принимается только стоимость такой модернизации.

На этапе эксплуатации любая система накапливает огромный объем данных и знаний [18], который необходим для эффективного и точного выполнения критически важных задач производственной деятельности. Из-за этого обслуживание унаследованных программных систем, а также миграция таких

данных и знаний является приоритетной задачей во многих организациях. И несмотря на сопутствующие трудозатраты компании продолжают эксплуатировать данные системы из принципа «лучшее – враг хорошего».

Тем не менее, устаревшие программные системы характеризуются рядом недостатков, среди которых:

• высокая стоимость обслуживания – эксплуатация унаследованных систем, размещенных на устаревшем оборудовании, приводит к существенным расходам на поддержание их функционирования. Например, ежегодные расходы бюджета США на ИТ превышают 100 миллиардов долларов, из которых 80% уходит на поддержку и обслуживание существующих ИТ-ресурсов, включая устаревшие системы;

• высокая стоимость обновлений – новая политика лицензирования от ведущих мировых брендов программного обеспечения предполагает продажу лицензий на срок всего один год или перевод платных сервисов в облачное пространство. На практике даже крупные компании вынуждены переходить на программное обеспечение с открытым кодом. Например, компания «Яндекс» в течении нескольких лет постепенно переходила на систему управления базами данных (СУБД) PostgreSQL [19] со свободной лицензией;

• программное обеспечение (ПО) не поддерживается разработчиком. Если программное обеспечение по-прежнему актуально и востребовано на рынке, его следует обновить для совместимости с новой версией операционной системы;

• проблемы с производительностью – длительное время загрузки, задержки в работе ПО и тому подобное. Это происходит из-за того, что производители больше не выпускают обновления, и нет реальных решений для устранения возникших проблем;

• устаревание документации по поддерживаемым бизнес-процессам – при изменении требований и доработке ПО в течение длительного времени никто не обновлял документацию по программному обеспечению. В итоге документация не соответствует фактическому состоянию и любые внесенные изменения в программу несут риски негативного воздействия на бизнес-процессы;

• старые и опытные сотрудники уходят вместе со своим багажом знания – в компании, использующей устаревшее программное обеспечение, может отсутствовать как полная, так и частичная документация, а также руководства для пользователей и администраторов системы. Новым сотрудникам не хватает необходимого опыта для работы с унаследованной системой, поскольку они не могут понять принципы ее работы;

• повышенный риск безопасности – устаревшие операционные системы без исправлений проблемы безопасности подвержены риску взлома, вирусам и другим вредоносным воздействиям. Надежность установленной защиты уменьшается с течением времени, потому что злоумышленники изучили данную защиту системы и могут её обходить. В отдельных случаях единственный способ защитить данную систему – это ее изоляция от остальной части предприятия, что может быть нецелесообразным, если есть потребность во взаимодействии с другими информационными системами. Например, атака программы-вымогателя WannaCry в 2017 г. дала обнаружить уязвимость старых систем, к которым больше не выпускаются обновления компанией Microsoft. В результате отсутствия актуальных обновлений безопасности более полумиллиона компьютеров, работающих на Windows XP, Windows 8 и Server 2003, были заражены в 200 странах по всему миру [20];

• проблема интеграции с внешними системами заключается в том, что различия в применяемых базовых технологиях затрудняют интеграцию с новыми сервисами, а также мешают своевременному масштабированию и гибкому совершенствованию ПО. В итоге развитие компаний замедляется, они могут становится неконкурентоспособными. Ярко выражена эта черта в государственных учреждениях, которые отличаются громоздкой и негибкой структурой, где любые изменения требуют длительного согласования. К примеру, чрезмерные затраты правительства США на ИТ-инфраструктуру в 2019 г. (более 70 миллиардов долларов) явились причиной внутреннего расследования. Оно показало, что главной проблемой является именно устаревшее ПО [21];

• удобство использования – пользователям безразлично, основана ли система на старом методе, технологии или использует она состоит из старого оборудования. Им необходимы надежные и безопасные системы с несложным интерфейсом. Данный интерфейс не требует дополнительных знаний системных команд для работы с ПО;

• дублирование и потеря данных – данная проблема появляется из-за трудностей интеграции устаревших систем. Их работа в организации происходит независимо друг от друга. Это приводит к изоляции информации, и сотрудникам приходится использовать множество различных инструментов для доступа к нужным данным;

• невозможность использования технологии больших данных – исторические данные хранятся в устаревшем формате. Для их интеграции в новую систему требуется использование специальных таблиц преобразования или специализированных хранилищ данных;

• политические причины – данные причины тоже имеют воздействие на принятие решения о переносе данных системы. Например, из-за введения антироссийских санкций большинство зарубежных компания со своим программным обеспечением ушли с российского рынка, что приводит к единственному выходу, а именно на переход отечественного ПО.

Существуют следующие виды возможных решений по сопровождению унаследованного программного обеспечения, рассмотрим каждый в отдельности.

Инкапсуляция (encapsulate) — это эффективный и быстрый способ повторного использования устаревших программных компонентов. Данный способ позволяет сохранить существующую среду, включая программный код и данные, предоставляя доступ к слоям через интерфейс программирования приложения (application programming interface, API). Таким образом создается новый интерфейс для данной устаревшей системы. Это помогает использовать приложение и расширять его возможности и ценность. К сожалению, данный подход не может решить проблемы обслуживания устаревшей системы. Этот вариант целесообразно применять, если устаревшее приложение обладает

качественным кодом, имеет значительную ценность для компании и выполняет необходимые функции [22], но интерфейсы приложения устарели (стали непривлекательными, не поддерживают современные методы обмена информацией и т. д.) или их использование стало неудобным.

Перенос на другую платформу (rehosting и replatforming) — это процесс повторного развертывания устаревшего программного обеспечения на новой физической, виртуальной или облачной инфраструктуре с незначительными изменениями в коде или функциональности.

Рехостинг (rehosting) – самый популярный метод модернизации из-за его простоты и доступности. Под рехостингом мы будем понимать простое копирование и вставка локальных приложений в облако без внесения серьезных изменений сотрудником компании. Приложение переносится и в нем не проводят изменения. Все это позволяет ускорить процесс оптимизации с минимальными затратами ресурсов. Для переноса в облако рекомендуется использовать базовые облачные функции.

Реплатформинг (replatforming) подразумевает, что часть кода может быть переработана или изменен способ, которым приложение взаимодействует с базой данных. Все внесенные изменения необходимы для наибольшего соответствия приложения новой платформе (в том числе облачной среде) и максимального использования ее функций.

Миграция в облако подразумевает перенос устаревшего программного обеспечения в облачную среду, включая переход на сервис-ориентированную архитектуру (SOA) или микросервисную архитектуру. Это один из современных подходов к улучшению унаследованных систем [23]. Процесс предполагает миграцию цифровых активов организации (данные и приложения). Миграция происходит с персональных компьютеров на виртуальный хост. Он использует общие ресурсы. К числу преимуществ миграции устаревших систем в облако можно отнести снижение затрат, масштабируемость, мобильность и повышение безопасности.

Рефакторинг — это процесс при котором изменяются структуры программного кода, при этом разработчик не касается изначальной функциональности, аппаратного обеспечения, общей архитектуры приложения [24]. Основная цель рефакторинга заключается в том, чтобы сделать код более читаемым и понятным, а также улучшить его за счет внесения множества мелких изменений, не меняя при этом общее поведение программы. Методы рефакторинга более детально показаны в источнике [25]. Основное различие между рефакторингом и реинжинирингом заключается в том, что реинжиниринг фокусируется на увеличении функциональных возможностей программного обеспечения. Рефакторинг дает возможность более качественно поддерживать данный проект и уменьшить затраты на его модернизацию. Замечено, что крупные рефакторинги предваряют реинжиниринг ИС.

Еще один тип рефакторинга заключается в изменении архитектуры программной системы [26]. Для компаний, которые уже используют облачные технологии, это оптимальный способ перейти от монолитной архитектуры к микросервисной, что позволяет улучшить масштабируемость и изоляцию компонентов.

Согласно прогнозу IDC, к 2022 году 90% всех новых приложений будут построены на основе микросервисной архитектуры. Данный вид архитектуры улучшит возможности проектирования, отладки, обновления и использования стороннего кода [27].

Многие устаревшие приложения реализованы в виде контейнеров, характеризующихся высокой связностью между компонентами и модулями. В таких приложениях вся обработка данных выполняется централизованно на сервере.

Из-за перехода на микросервисную архитектуру происходит масштабируемость системы. Это преимущество особенно проявляется при переходе к микросервисной архитектуре для тех компонентов, которые активно используются большинством пользователей. Подробную проработку вопросов миграции унаследованных приложений на микросервисы можно найти в работах [28, 29]. Также существует возможность создания унифицированной модели

данных на основе онтологии, которая не только описывает сложные взаимосвязи между сущностями, но и облегчает процесс взаимодействия и интерпретации данных между микросервисами, тем самым упрощая интеграцию сервисов [30].

Редизайн (redesign) представляет собой переработку исключительно графических пользовательских интерфейсов (GUI) информационных систем, не затрагивая при этом их архитектуру. Чаще всего этот процесс применяется к web-сайтам, где важную роль играют эргономика web-интерфейса и удобство использования (юзабилити) [31, 32]. Данный процесс, направлен на улучшение работы с пользователем, в первую очередь путем переоценки существующей навигации, макетов, контента и стека технологий [33].

Перепроектирование (Rebuilding) представляет собой процесс создания компонентов приложения заново, при этом сохраняя их объем и характеристики. Этот метод считается более современным способом обновления устаревшей системы. Он позволяет создать новую архитектуру и внести изменения в код. Перепроектирование приложения дает возможность внедрять новые функции и процессы, используя возможности современных технологий и платформ. Решение проблем с производительностью и масштабируемостью ИС данный подход тоже удовлетворяет.

Замена (Replacing) подразумевает полное замещение приложения другой информационной системой. В этом случае повторное использование существующей бизнес-логики невозможно, поскольку она перестает соответствовать потребностям компании или её стратегиям. Этот подход используется в тех случаях, когда существующая система не в состоянии удовлетворить все требования бизнес-процесса.

1.2 Обзор существующих моделей и стандартов в области интеграции

1.2.1 Модель интеграции приложений на основании уровня приложений

Рассмотрим основные применяемые модели интеграций [34]:

- интеграция на уровне данных (рисунок 1.3) создается на базе общего хранилища данных (datawarehouses), доступ к которому осуществляется вне зависимости от бизнес-логики;

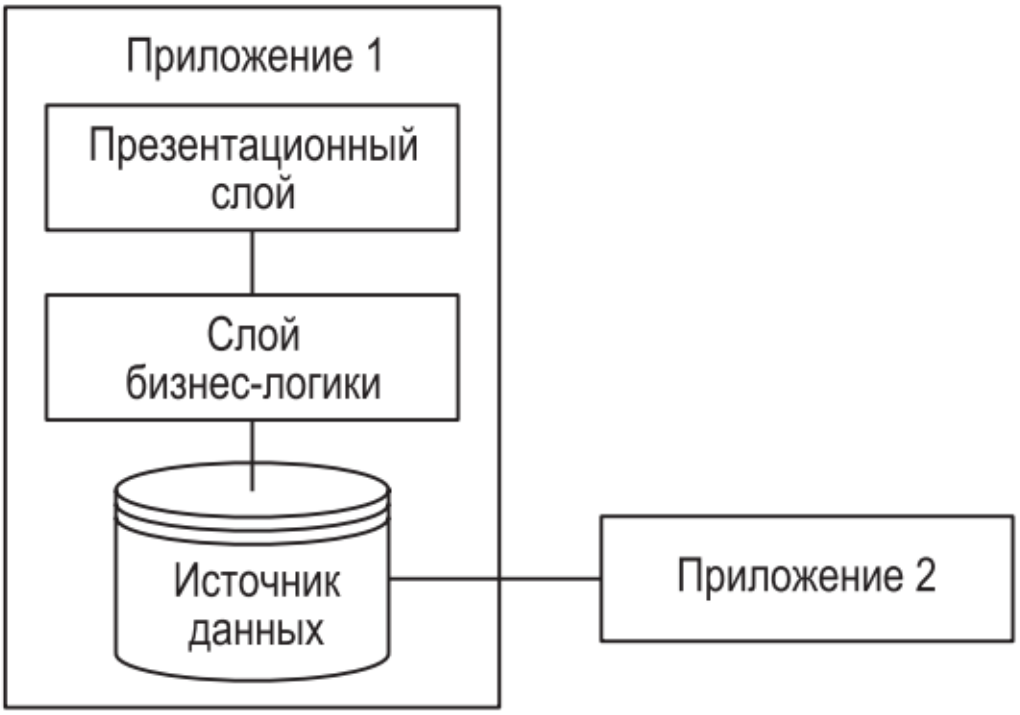

Рисунок 1.3. Схема модели интеграции на уровне данных

- интеграция на уровне бизнес-логики (рисунок 1.4.), которая управляет потоками данных и организует взаимодействие его компонентов;

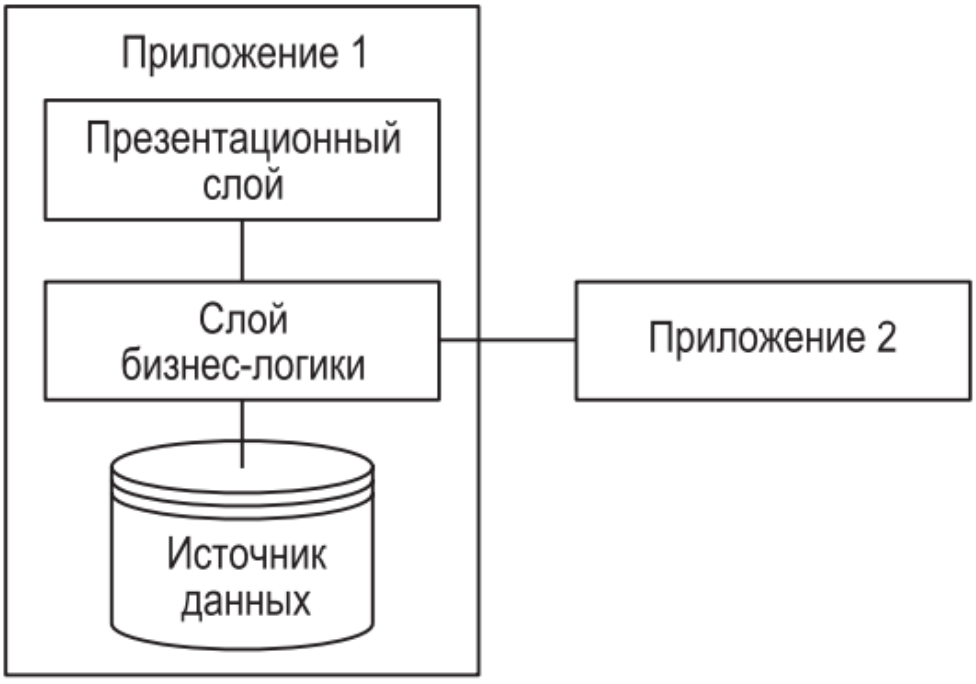

Рисунок 1.4. Схема модели интеграции на уровне бизнес-логики

- интеграция на уровне пользовательских интерфейсов (рисунок 1.5), где приложения взаимодействуют друг с другом через пользовательский интерфейс (screen scraping);

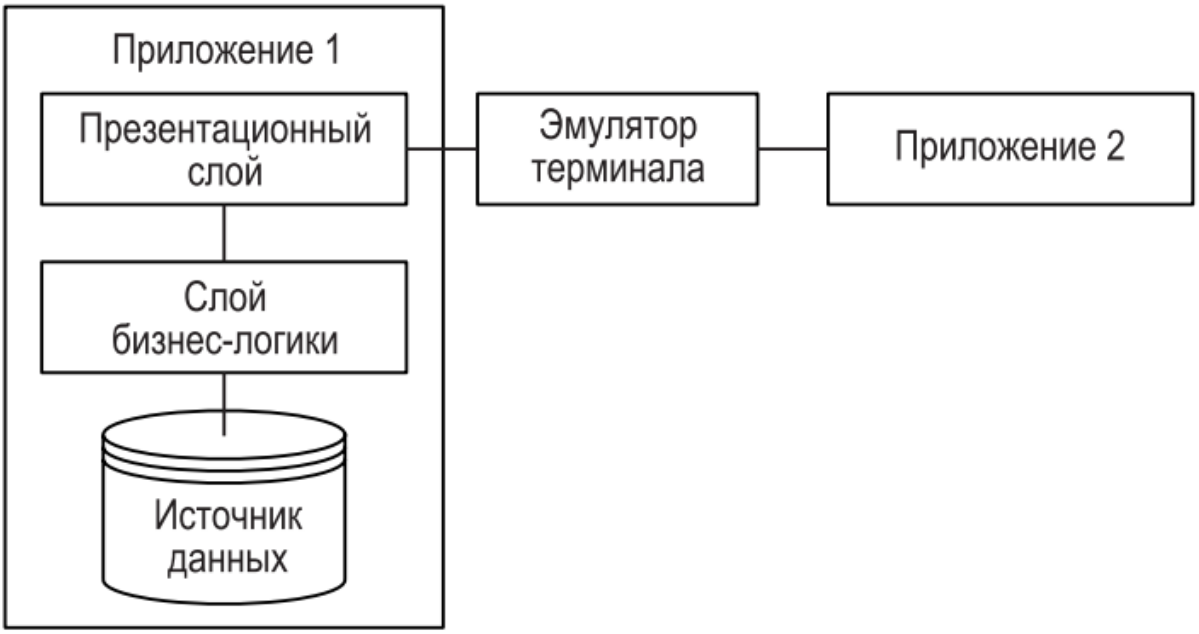

Рисунок 1.5. Схема модели интеграции на уровне данных пользовательских интерфейсов

1.2.2 Модель интеграции приложений на основании топологии связей

- интеграция типа «каждый с каждым» (полносвязная топология) (рисунок 1.6), где создаются интерфейсы обмена данных для каждой пары обменивающихся приложений;

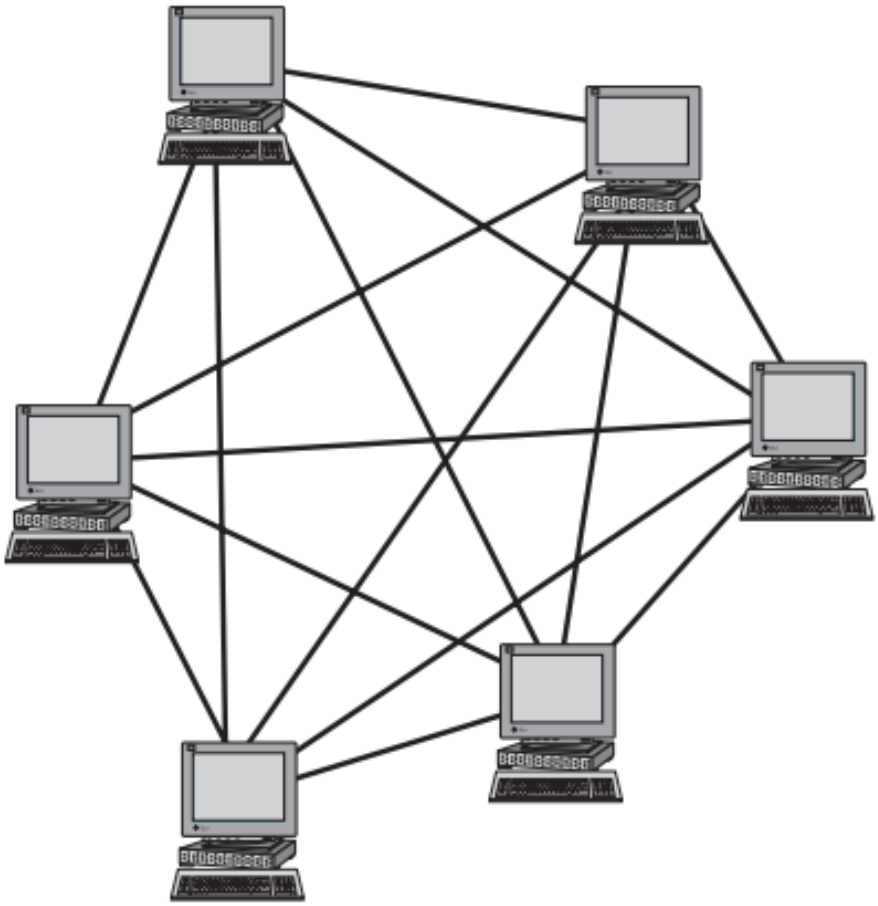

Рисунок 1.6. Схема модели интеграции типа «каждый с каждым»

- интеграция типа «звезда» (рисунок 1.7), где происходит путем обмена данными между приложениями через центральный компонент;

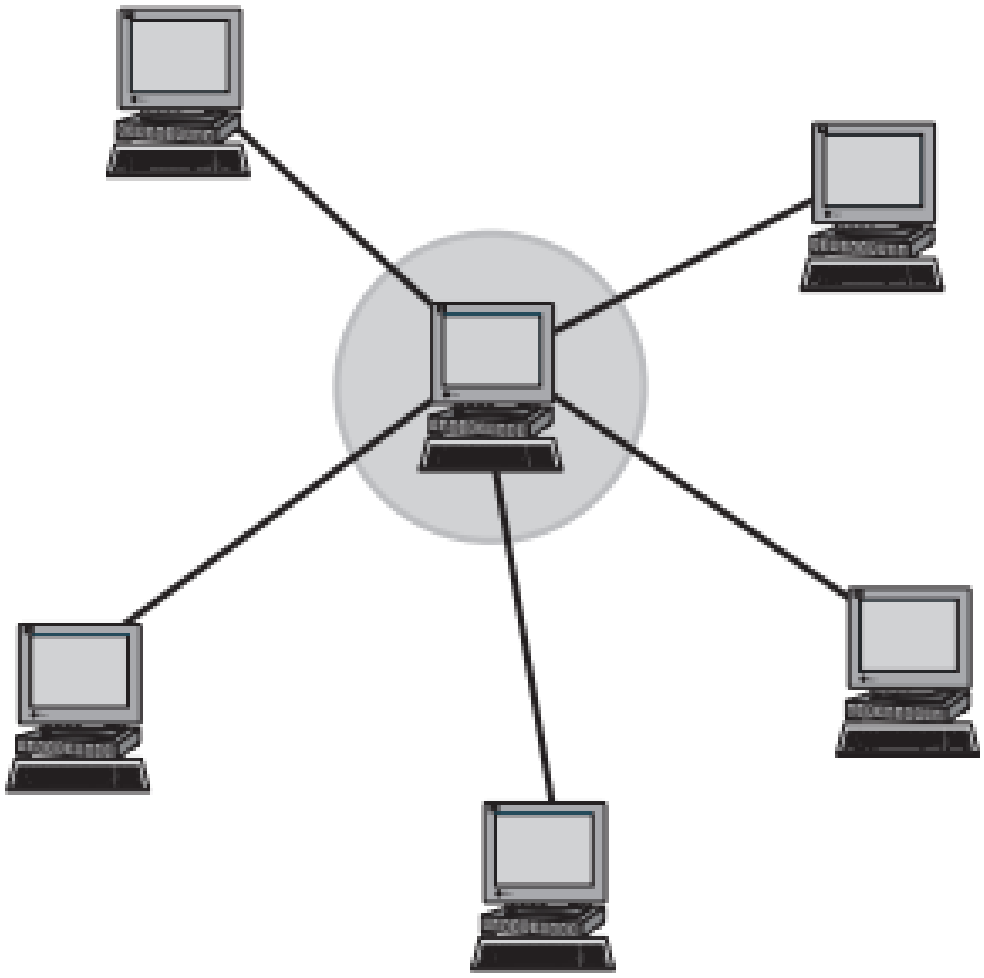

Рисунок 1.7. Схема модели интеграции типа «звезда»

- интеграция типа «дерево» (рисунок 1.8) построена по схеме строго двоичного дерева, где каждое приложение более высокого уровня взаимодействует с двумя приложениями более низкого уровня и далее по порядку;

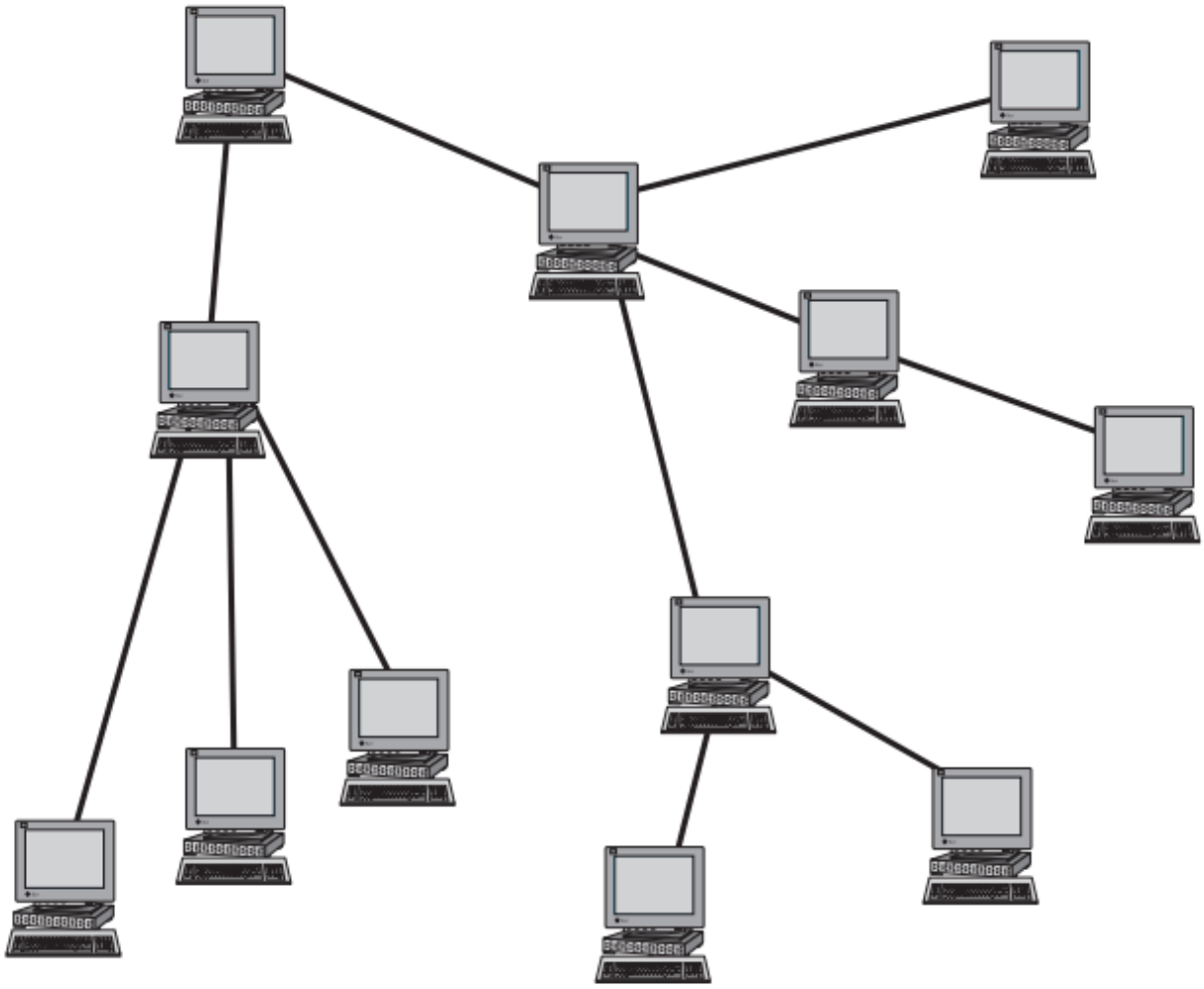

Рисунок 1.8. Схема модели интеграции типа «дерево»

- интеграция типа «линейная модель» (рисунок 1.9), где приложения, выстроенные в цепочку, взаимодействуют с двумя соседними кроме крайних (они взаимодействуют с одним приложением);

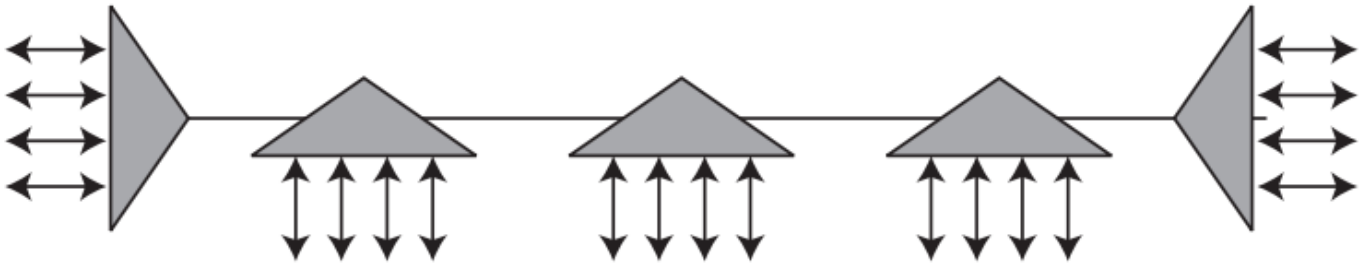

Рисунок 1.9. Схема модели интеграции типа «линейная модель»

- интеграция типа «кольцо» (рисунок 1.10), образуется путем замыкания линейного типа; в этом варианте каждое приложение взаимодействует с двумя другими.

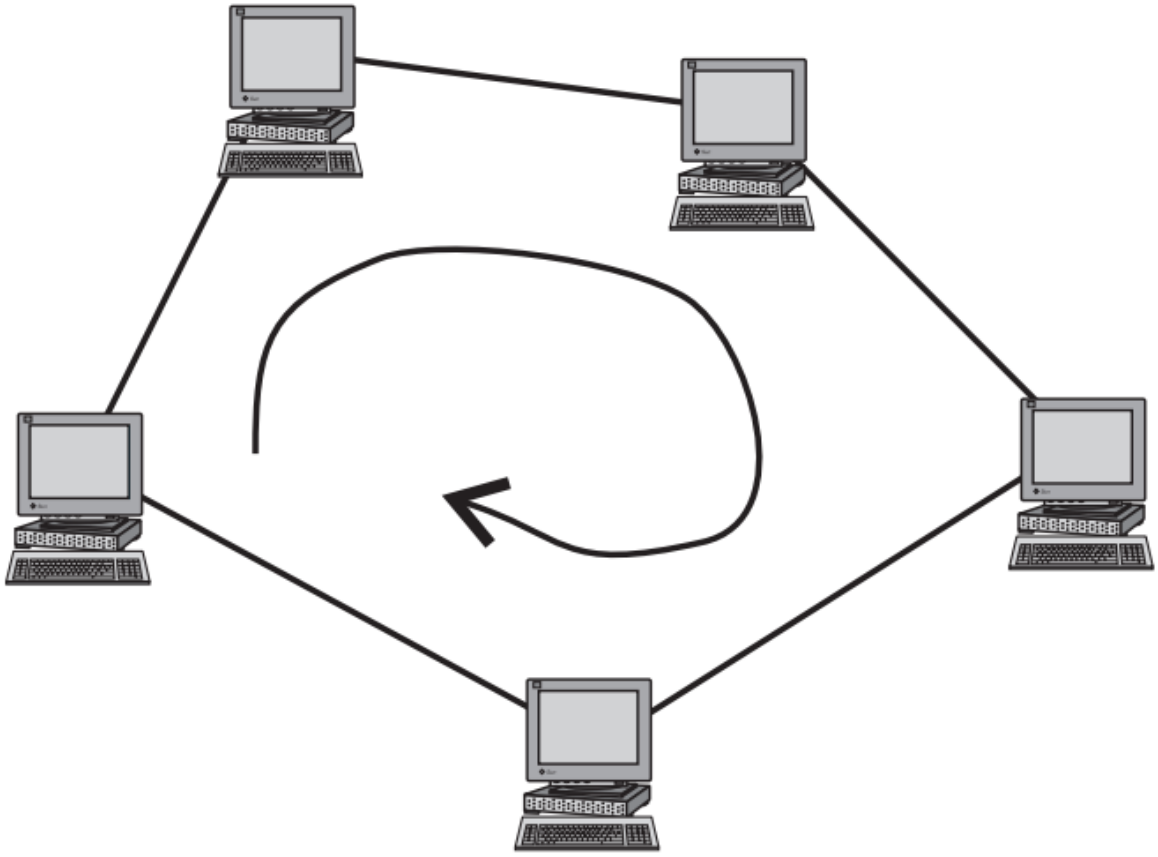

Рисунок 1.10. Схема модели интеграции типа «кольцо»

Более сложные топологии строятся на основе модификации описанных моделей, поэтому в данной работе рассматриваться не будут.

1.2.3 Модель интеграции приложений на основании используемого метода (технологии)

Были проанализированы следующие модели интеграции на основе топологии:

- интеграция на уровне информационных ресурсов с помощью ECM-технологии (Enterprise content management: управление корпоративным контентом). Данная интеграция может объединять отдельные информационные системы предприятия. Для этого она связывает их на уровне бизнес-процессов и потоков информации;

- интеграция на уровне корпоративных приложений (EAI, Enterprise Application Integration). Данная интеграция использует совместно исполняемый код корпоративных приложений компании, а не их внутренние данные. Схема интеграции показана на рисунке 1.11;

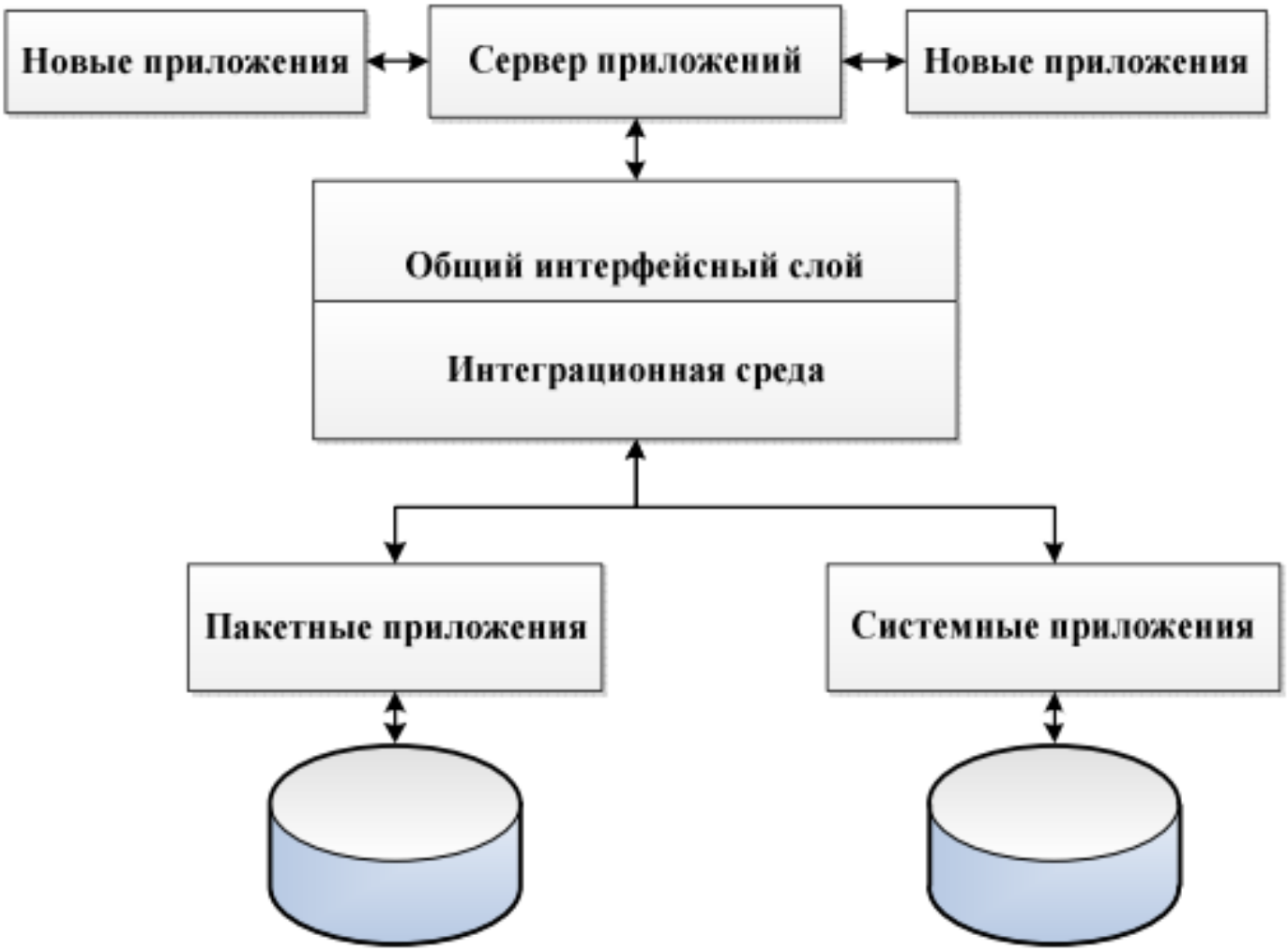

Рисунок 1.11. Схема модели интеграции на уровне корпоративных приложений

- интеграция на основе единой сервисной шины (рисунок 1.12), где взаимодействие всех приложений происходит через обмен сообщениями через единую сервисную шину (ESB) (Enterprise Service Bus);

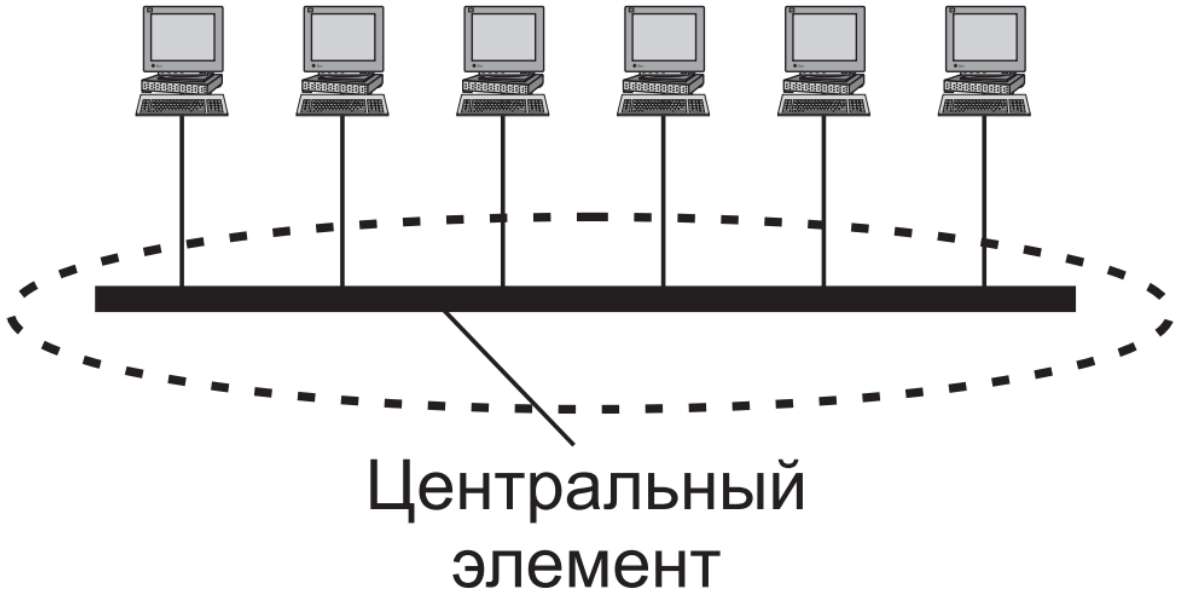

Рисунок 1.12. Схема модели интеграции на основе единой сервисной шины

- интеграция с помощью web-сервисов [35] (рисунок 1.13) обеспечивается стандартизированным для web-служб интерфейсом доступа к приложениям и данным.

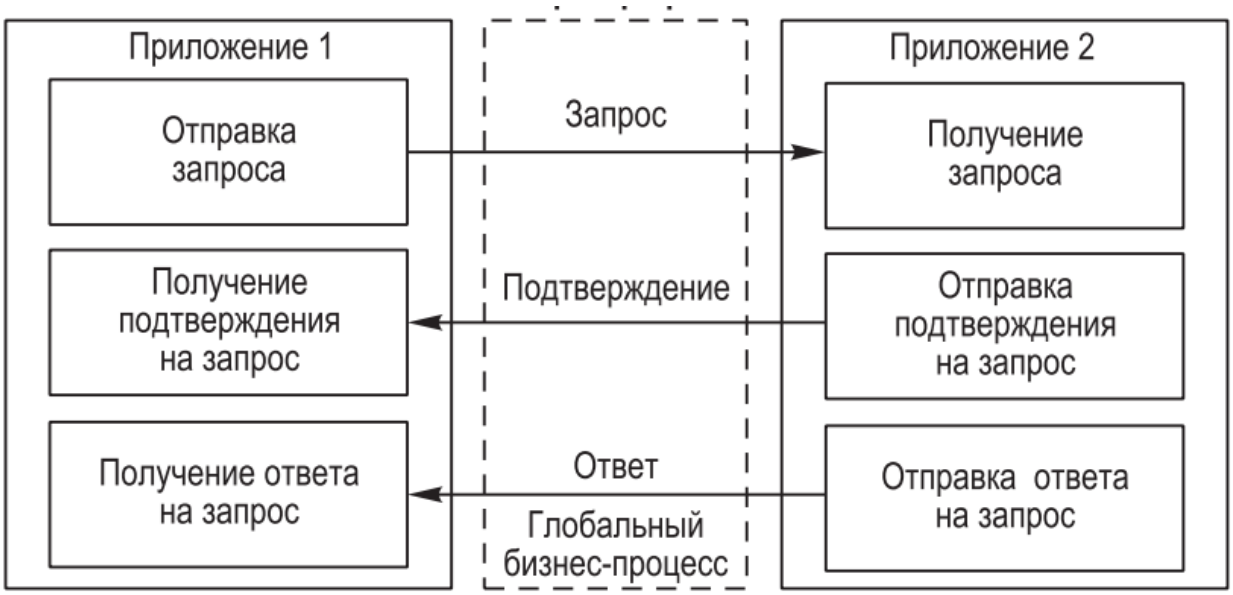

Рисунок 1.13. Схема модели интеграции с помощью технологии web-сервисов

Для осуществления запроса внешними системами из единой модели данных организации предлагается использовать наиболее эффективные и широко распространенные на текущий момент технологии: RESTful (Representational State Transfer — передача репрезентативного состояния), SOAP (Simple Object Access

Protocol — простой протокол доступа к объектам), JSON-Pure, GraphQL [36], Falcor [37].

Технологии RESTful и SOAP созданы в 1990-х г. и являются простыми, поэтому нашли использование для большинства web-сервисов.

Другие технологии были созданы в 2010-х г., причем технологии GraphQL и Falcor изначально разрабатывались крупными компаниями для своих нужд (Facebook и Netflix соответственно). В таблице 1.1 показано сравнение технологий web-сервисов.

Таблица 1.1. Сравнение технологий web-сервисов по множественным критериям

Технология	Производи-тельность	Простота	Множест-венность запросов	Стандар-тизация	Удобство отладки	Языковая поддержка
RESTful	Средняя	-	-	-	-	+
SOAP	Средняя	-	-	+	-	+
JSON-Pure	Высокая	+	-	-	+	+
GraphQL	Высокая	+	+	+	+	+
Falcor	Высокая	+	+	-	+	-

На основании данной таблицы можно сделать вывод, что информационное взаимодействие эффективнее внедрять с помощью технологии GraphQL [38]. По сравнению с остальными приведенными технологиями она позволяет проще масштабировать подключаемые сервисы, без необходимости полного обновления запросов. Также технология обеспечивает троекратное ускорение получения данных по сравнению с RESTful [39]. Разработка конкретных решений в области интеграции ПОС подробно рассматривается в разделе 1.2.4 .

1.2.4 Разработка решения в области интеграции проблемно-ориентированных информационных систем

В предыдущем разделе проведен анализ существующих подходов к выявлению направлений повышения эффективности процессов интеграции разнородных структурированных/ не структурированных источников данных самых разных форматов, в том числе баз данных (SQL/NoSQL), спроектированных на основе разных технологий, таблиц, совокупности таблиц или просто файлов (данные в котором разделены символами-разделителями) разных форматов (doc, xml, xls, pdf, csv, json и пр). По результатам анализа отечественных и зарубежных работ сделан вывод, что в современных условиях в качестве основного инструмента для извлечения данных из разнородных источников рассматривается комбинация ETL (extract, transform, load) и Web-сервисов (RESTful, SOAP, JSON, GraphQL и пр.), где ETL-сервисы обеспечивают извлечение информации из структурированных/неструктурированных источников, затем их преобразование в нужный формат с последующей загрузкой в место назначения, а Web-сервисы поддерживают эффективные запросы из разнообразных внешних систем в адрес единой базы данных. При этом, вопросы обеспечения конфиденциальности данных решаются путем построения виртуальных частных сетей (VPN) и использования криптографических протоколов (сертификат безопасности, СБ), позволяющих установить безопасное соединение между базами данных и устройствами пользователей.

1.3 Особенности процесса обеспечения летной годности воздушных судов

1.3.1 Контроль летной годности воздушных судов

Контроль летной годности ВС является многоуровневой задачей, одной из составных частей которой является процесс подтверждения подлинности компонентов ВС, который описан нормативными и методическими документами.

Компонентом ВС паспортизирован и состоит из паспорта и самого изделия. Сбор данных состоит из ряда независимых друг от друга действий, как с паспортом, так и с самим изделием. Данные действия могут происходить

синхронно или последовательно. В дальнейшем возникает необходимость полученные данные интегрировать, что дополнительно усложняет процесс контроля и ведет к увеличению его срока.

Согласно руководству по летной годности [40] очевидна необходимость обеспечения того, чтобы составные части (СЧ), установленные на ВС, отвечали проектным техническим требованиям и были работоспособными. Если установка какой-либо СЧ, не отвечает выдвинутым требованиям к конструкции, это приводит к снижению данных требований и, поэтому, к снижению летной годности.

Ключевым аспектом обеспечения летной годности является разработка системы контроля, которая будет гарантировать установку на конкретном воздушном судне только тех составных частей, которые соответствуют утвержденным конструкторским данным данного ВС [41].

Согласно рекомендательному циркуляру №РЦ-АП-145.А.42d [42] организуется информирование о сомнительных изделиях или неутвержденных составных частях.

На основании Приказа Минтранса России от 27.11.2020 №519 [43] результаты инспекционного контроля летной годности воздушных судов служат основанием для принятия решения уполномоченным органом гражданской авиации о приостановлении или возобновлении действия свидетельства о летной годности в рамках его срока действия. По итогам инспекционного контроля составляется акт, подтверждающий результаты проверки летной годности воздушного судна. Схема алгоритма инспекционного контроля летной годности ВС показана на рисунке 1.14.

Таким образом на основании нормативных и методических документов, описанных выше включая методику оценки подлинности компонентов ВС [44] можно выделить основные этапы подтверждения, приведенные на рисунке 1.15:

1. Сбор данных по изделиям и удостоверяющим документам ВС.
2. Сопоставительный анализ полученных данных.

3. Формирование запросов к участникам процесса инспекционного контроля ВС.

4. Оценка качественных характеристик данных и выдача решения о возобновлении (в случае приостановления) сертификата летной годности в пределах срока его действия, а также составление акта инспекционного контроля летной годности ВС.

Причем, необходимо подчеркнуть, что решение о подлинности компонента воздушного судна принимается на основе анализа текущих и исторических данных, включая удостоверяющие документы и, собственно, изделие.

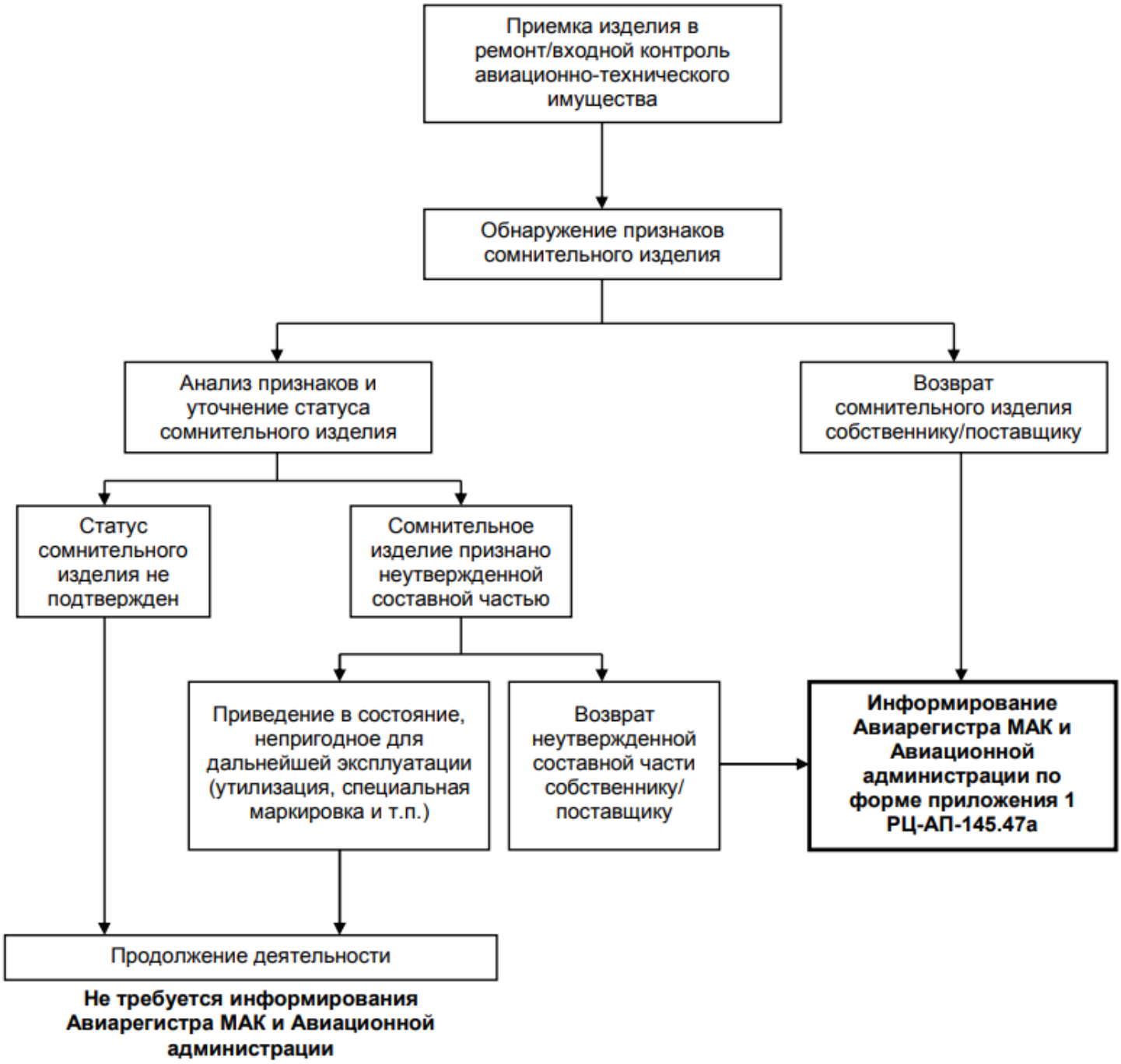

Рисунок 1.14. Схема алгоритма инспекционного контроля

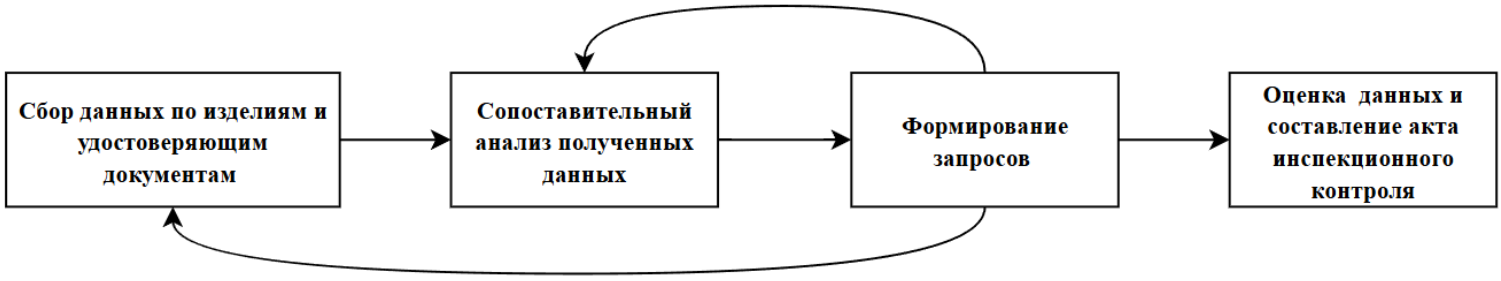

Рисунок 1.15. Схема процесса подтверждения подлинности компонентов ВС

В основу процесса положена организация множественных запросов к источникам информации (ВС включает до 6000 изделий с удостоверяющими документами). Полученные ответы, зачастую, характеризуются неполнотой, недостаточной достоверностью и актуальностью информации об изделиях/удостоверяющих документах, в том числе: отсутствие ключевых показателей (дата, производитель, наличие действующей лицензии, сроки проведения работ, ремонтное предприятие и пр.), подтверждающих подписей и печатей, наличие «двойников», изделий с «неустановленным происхождением», дубликатов удостоверяющих документов и пр., что предполагает формирование повторных уточняющих запросов.

1.3.2 Анализ существующих систем сопровождения эксплуатации воздушных судов

В рамках решения задачи подтверждения подлинности компонентов ВС рассмотрим возможности использования современных информационных систем сопровождения эксплуатации ВС, в том числе: АСУ ПЛГ ВС, CAMP, «my Boeing fleet», AirnavX, ИУС «Эрлан-3», AMOS, IBM Maximo for Aviation MRO, Ramco MRO Aviation, SAM, Veryon.

В АСУ ПЛГ ВС [45] показана на рисунке 1.16. Система обрабатывает основные потоки информации в зависимости от источников, а именно:

- предусмотрена возможность ввода данных о ВС и компонентах в базу данных двумя способами: непосредственно и путем загрузки из существующей информационной системы компании (если таковая имеется);
- создание БД электронной пономерной документации (электронных формуляров и паспортов) ВС и компонентов;
- формирование комплекта доказательной документации, содержащей достоверную информацию о ВС и компонентах, в соответствии с установленными нормативными требованиями;

- данные о ВС и компонентах загружаются в БД. Они могут быть получены из нее. Эти данные используются для прогнозирования ресурсного и технического состояния эксплуатируемых ВС и компонентов.

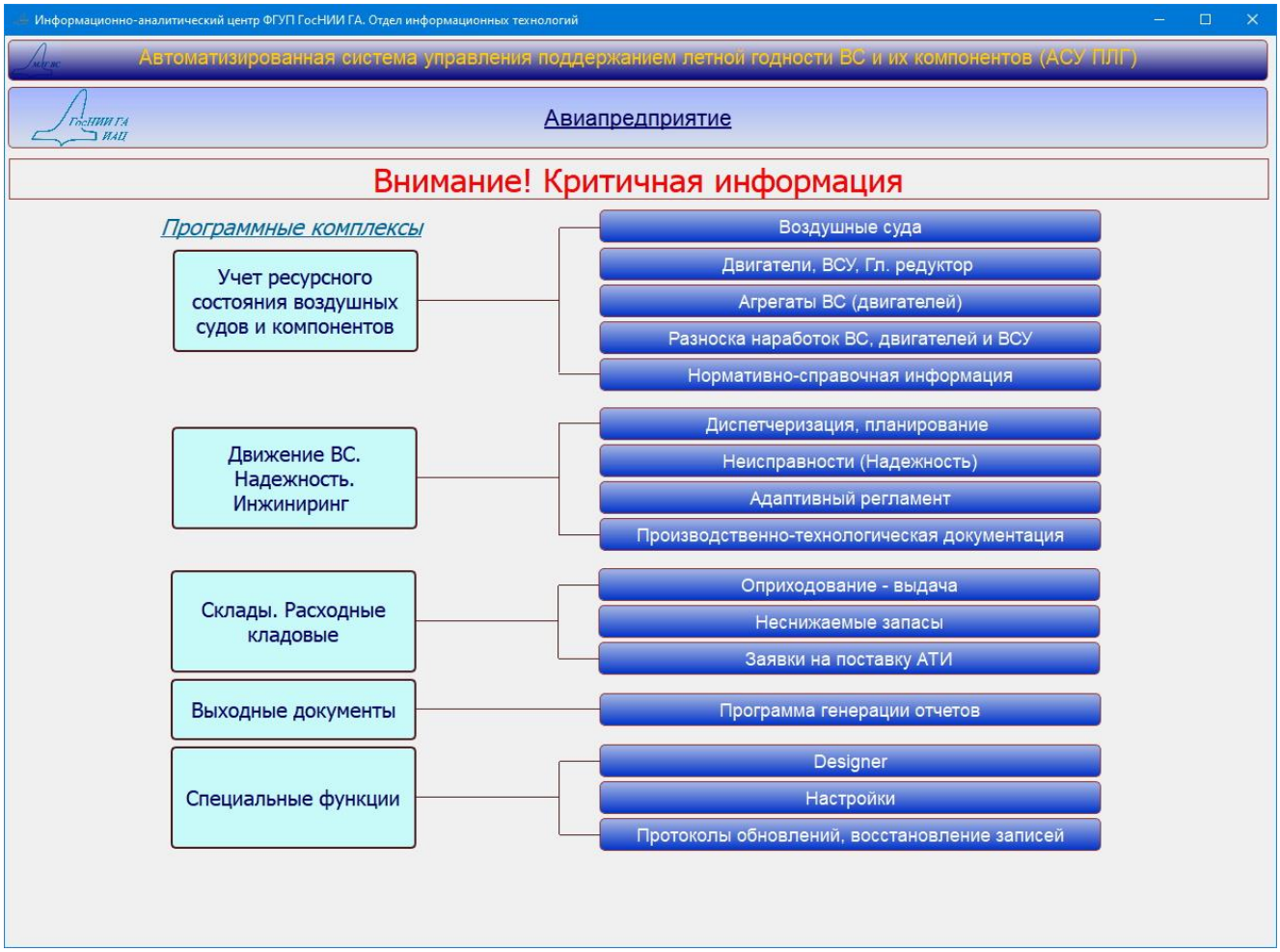

Рисунок 1.16. Модульная схема АСУ ПЛГ ВС [46]

ИУС «Эрлан-3» имеет функциональную структуру, которая состоит из 5 модулей [47]:

- учет технического состояния ВС и комплектующих изделий («Учёт»);
- управление техническим обслуживанием ВС («Управление ТО»), который показан на рисунке 1.17;
- планирование использования парка ВС («Планирование»);
- регистрация отказов и неисправностей («Надежность»);
- управление запасами комплектующих изделий («Управление запасами»);
- приложение для представления сводных данных руководителю - портал руководителя («Портал»);

- администрирование («Сервис»).

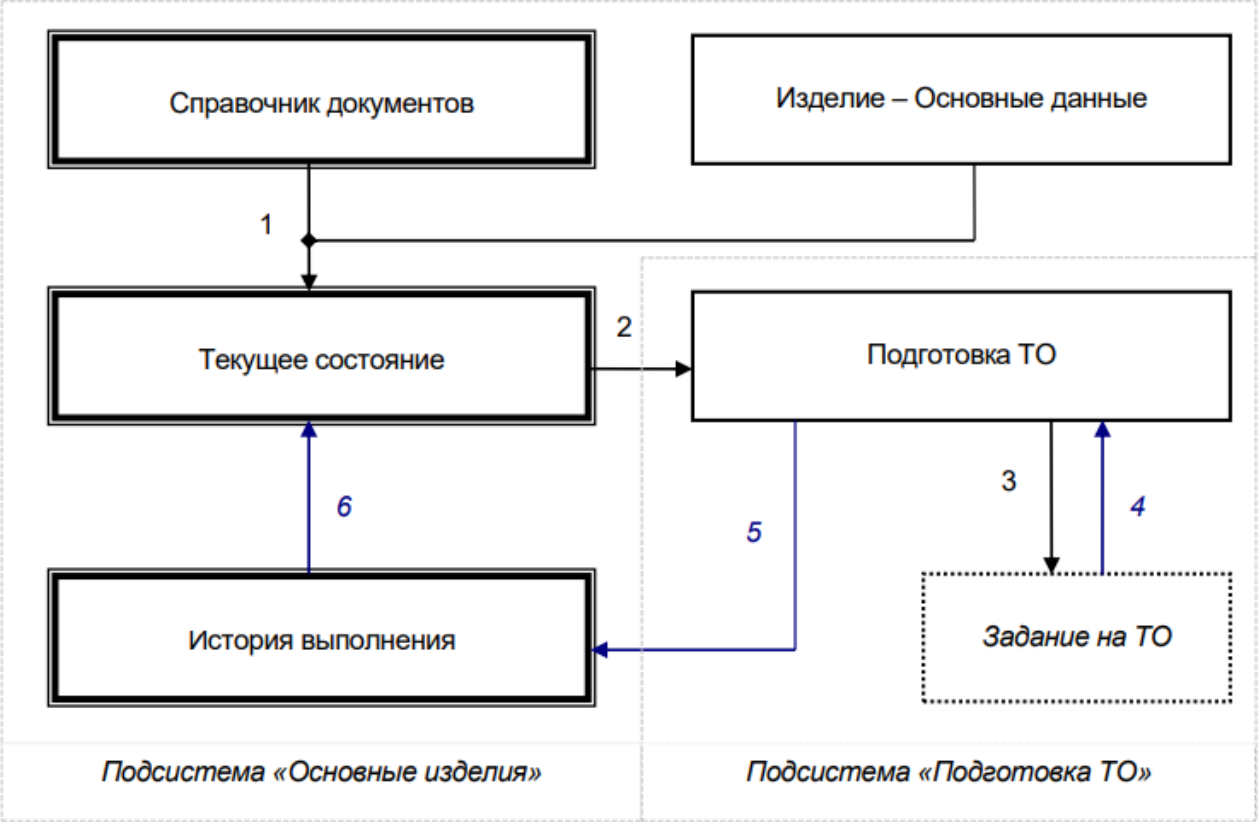

Рисунок 1.17. Подсистемы комплекса управления техническим обслуживанием ВС

AMOS (программное обеспечение MRO) состоит из основных модулей (их 8 штук), которые составляют ядро системы [48]. Интерфейс данного программного обеспечения изображен на рисунке 1.18. Описание модулей следующее:

- Модуль управления материальными потоками: отвечает за процессы управления запасами, заказами и приемкой, а также за наличие деталей, использование компонентов и контроль гарантий;
- Инженерный модуль: обеспечивает выполнение различных программ технического обслуживания и соблюдение применимых Директив летной годности (AD) и Сервисных бюллетеней (SB);
- Модуль планирования: управляет подготовкой как плановых, так и внеплановых задач по техническому обслуживанию;
- Производственный модуль: отвечает за фактическое выполнение задач по техническому обслуживанию;

• Модуль управления техническим обслуживанием: выполняет роль связующего звена между эксплуатацией и техническим обслуживанием;

- Модуль технического обслуживания компонентов: управление масштабными мероприятиями по техническому обслуживанию, ремонту и капитальному ремонту в условиях цеха;
- Коммерческий модуль: мониторинг взаимоотношений со всеми клиентами и отслеживание текущих предпродажных мероприятий с потенциальными и существующими клиентами;
- Модуль обеспечения качества: Соблюдение стандартов качества.

Помимо этих основных модулей, AMOS также включает два дополнительных функциональных набора:

- Модуль «Кадровые ресурсы»: Управление персоналом;
- Модуль финансового управления: контролирует расходы на техническое обслуживание, которые включают такие задачи, как контроль затрат и гарантий, а также генерация счетов. Финансовые функции предоставляют только область, связанную с контекстом обслуживания, и не заменяют полную финансовую систему.

AirnavX, показанная на рисунке 1.19, от компании Airbus имеет следующие функции [50]:

- «Поиск данных» — быстрое решение на основе ключевых слов для сканирования всех доступных технических данных по техническому обслуживанию;
- «Моя библиотека» — единая точка доступа для просмотра и загрузки технических руководств Airbus;
- «Устранение неисправностей», которая использует коды неисправностей и предупреждения централизованной электронной системы контроля состояния воздушного судна для определения и поиска необходимой документации.

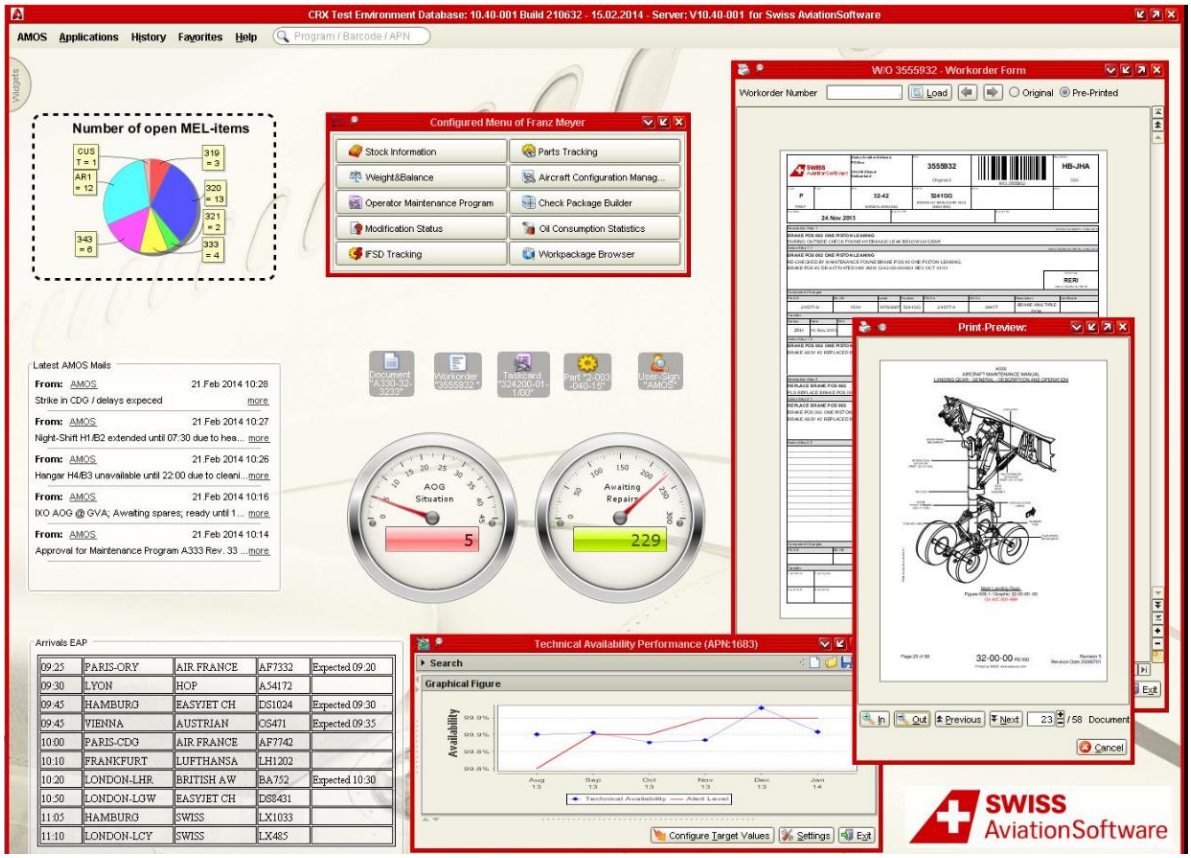

Рисунок 1.18. Интерфейс AMOS (программное обеспечение MRO) [49]

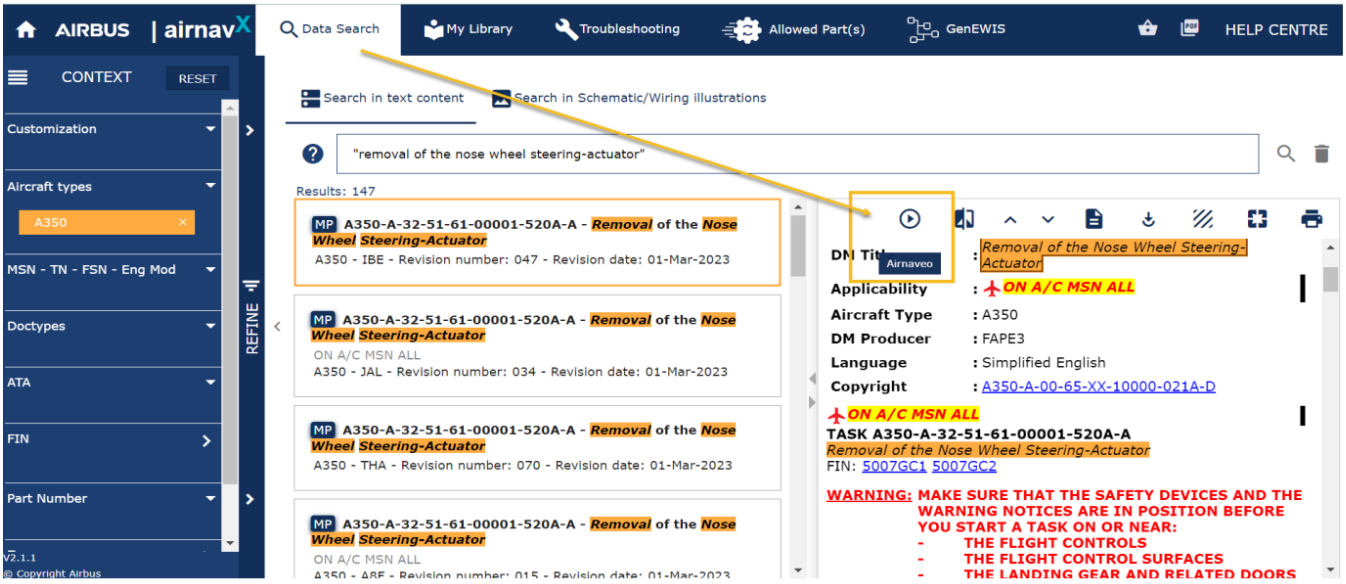

Рисунок 1.19. Интерфейс AirnavX

Анализ возможности использования современных информационных системы сопровождения эксплуатации ВС показал, что они, в основном, ориентированы на обработку данных уже введенных в систему и не включают

функционал по организации сбора и обработки разнородных данных о текущем состоянии паспортизированных компонентов в автоматизированном режиме.

В этой связи, рассмотрим возможности использование существующих технологических решений для подтверждения подлинности компонентов воздушных судов.

1.3.3 Проблемы использования существующих технологических решений для повышения эффективности процесса подтверждения подлинности компонентов воздушных судов

Для начала рассмотрим несколько использующихся на текущий момент технологий для автоматизации.

Цифровая телефония — это современная система. Она использует сети передачи данных для передачи голоса. Данную технологию можно масштабировать. Её можно использовать в локальных сетях компаний. Минусом технологии является то, что данные передаются не круглосуточно (производственный календарь сотрудника), возможны ошибки в передаче данных (человеческий фактор) [51], необходимо запрос по каждому агрегату производить отдельно и к каждому источнику.

Электронная почта – устаревшая система для передачи данных почти в любом формате (текстовый, графический и т.п.), которая может работать в локальных сетях. Минусом является то, что отчет на отправленное письмо, может быть не предоставлен. Также пользователю необходимо писать письма по каждому агрегату (возможно сгруппировать) и к каждому источнику отдельно (так как у каждого источника свой почтовой адрес).

Разовые удаленные запросы к источникам данных для поддержки инспекционного контроля. Минусом данной технологии, является само ее определение, каждый запрос по каждому агрегату к конкретному источнику посылается отдельно. Например, 20 ВС уже потребует формирования свыше 480 тыс. всех видов запросов к источникам данных (общий объем циркулирующей информации составляет более 5 терабайт данных).

Таким образом, можно отметить, что попытки автоматизации указанных процедур за счет использования существующих технологических решений не обеспечит существенного сокращения времени проведения работ (до 14 дней).

1.3.4 Требования к созданию новой проблемно-ориентированной информационной системы

Как показано выше (в главах 1.3.2, 1.3.3) существующие системы в основном, ориентированы на обработку данных, уже введенных в систему, и не включают функционал по организации сбора и обработки разнородных данных в автоматизированном режиме, а технологические решения не обеспечивают существенного сокращения времени проведения работ по подтверждению подлинности компонентов воздушных судов.

Поэтому возникает необходимость создания новой информационной системы, позволяющей обеспечить сбор полных и достоверных сведений о компонентах и пономерной документации ВС путем организации множественных запросов [52] к источникам оригинальной информации: производителям, ремонтным организация и эксплуатантам с учетом использования сетевой структуры современной коммуникационной среды, а также совместимых механизмов извлечения, передачи и обработки данных в режиме on-line.

В работе сформированы основные требования к такой системе, в том числе:

• реализация сбора и передачи данных полностью в электронной модели. Необходимо для работы без бумажных носителей информации;

• получение данных от разнотипных источников посредством формирования множественных запросов. Одноразовые запросы будут создавать очереди, так как для отправки нового запроса, необходимо будет дождаться предыдущего;

• обработка разнородных данных в режиме on-line. Данные разных типов должны обрабатываться сразу, иначе нарушится их актуальность;

• работа с файлами в форматах doc, xml (eXtensible Markup Language), xls, pdf, json (JavaScript Object Notation) и пр. Необходима работа с разными видами документов;

• организация доступа к данным с признаками коммерческой тайны. Необходима работа с данными по характеристикам, зачастую имеющими конфиденциальный характер, авиационной техники у соответствующих эксплуатантов;

• использование для работы виртуальные частные сети (VPN). Необходимо для взаимодействия между источником данных и базами данных ПОС;

• использование для работы сертификаты безопасности. Необходимо для взаимодействия между пользователями системы и базами данных ПОС.

В таблице 1.2 показано, что современные отечественные системы (ИУС «ЭРЛАН» и АСУ ПЛГ ВС) могут работать с виртуальными частными сетями. Иностранные системы (AMOS и Airbus airnavX) могут использовать для работы файлы xml. Однако, рассмотренные системы не удовлетворяют всем выдвинутым требованиям. Следовательно, необходимо создать новую систему, удовлетворяющую заданным выше требованиям для контроля летной годности ВС. Предлагаемый облик программного комплекса системы описан в разделе 1.3.5 .

Таблица 1.2. Сравнение рассмотренных систем согласно выбранным критериям

Системы \ Требования	Получение данных в режиме on-line	Множественные запросы к ИС поставщиков компонентов ВС	Обработка разнородных данных	Организация доступа к данным с признаками коммерческой тайны	Использование для работы виртуальные частные сети (VPN)	Работа с файлами в формате xml	Использование для работы сертификаты безопасности
ИУС «ЭРЛАН»	-	-	-	-	+	+	-
AMOS	-	-	-	-	+	+	-
Airbus airnavX	-	-	-	-	-	+	+
АСУ ПЛГ ВС	-	-	-	-	+	-	-
Новая система	+	+	+	+	+	+	+

1.3.5 Предлагаемый облик программного комплекса системы

Рассмотрим основные элементы комплекса, которые образуют его структуру. Так же необходимо показать этапы и функции. Всё вместе будет образовывать облик программного комплекса системы.

При подтверждении подлинности компонентов ВС огромную значимость имеют **данные.** Анализ данных выполняет роль выявления паттернов, корреляций и моделей, которые помогают использовать эти данные для принятия бизнес-решений. Паттерны могут формироваться на основе различных аналитических методов: от простых (например, расчет средних значений) до сложных (алгоритмы машинного обучения, обрабатывающие большие объемы данных). При этом точность прогнозов становится все более важной по мере увеличения объемов обрабатываемых данных. Это делает предпочтительным использование всех доступных данных. Аналитические запросы часто представляют собой сложные и длительные процессы, обрабатывающие большой объем информации из различных источников. К примеру, это может быть агрегация данных по всей таблице или объединение нескольких крупных таблиц в одну плоскую денормализованную таблицу для дальнейшей обработки. Масштаб изменений может быть значительным, включая одновременное добавление, удаление и обработку большого количества строк. Наличие множества источников данных в компании обуславливает необходимость создания единого хранилища данных (data warehouse) для проведения эффективного анализа. Процесс агрегации данных в таких хранилищах называется extract, transform and load (ETL) и является частью интерактивной аналитической обработки (OLAP). В данном контексте определение OLAP-запросов охватывает широкий спектр возможностей, от работы исследователей данных до бизнес-аналитиков. Традиционно аналитические задачи решались на уровне систем управления базами данных (СУБД).

Одним из видов СУБД являются **реляционные**. Данные системы управления базами данных предлагают множество возможностей для оптимизации доступа к данным и их анализа в хранилище. Благодаря

абстрагированию методов доступа и алгоритмов обработки, СУБД могут контролировать способ хранения информации и улучшать планы выполнения запросов. Разнообразные запросы к одному и тому же хранилищу могут негативно сказываться на общей производительности системы. Важно разделять процессы обработки запросов для обновления и анализа, особенно при повторяющихся запросах, таких как сбор данных для онлайн-визуализации или создания дашбордов, а также при переиспользовании данных для поиска новых паттернов и работы бизнес-аналитиков. Четко точная схема хранилищ данных и специализированная структура хранения (например, физическое дублирование или колончатая организация данных) содействует повышению производительности при выполнении вычислительно сложных аналитических запросов, из-за того, что данные запросы часто требуют доступа ко всем доступным данным. Объем данных требует распределенного хранения и вычислений, а сложность запросов подразумевает необходимость качественного планирования, включая выбор оптимальных алгоритмов и уровня параллелизма для выполняемых операций.

На протяжении более чем 50 лет развития систем управления базами данных (СУБД) было создано множество принципов и подходов к оптимизации запросов на различных уровнях — от архитектурных решений до конкретных алгоритмических методов [53-55]. Основной целью этих методов традиционно было повышение эффективности использования аппаратных ресурсов. Однако с развитием индустрии и резким увеличением объема данных акценты в оптимизации начали меняться [56, 57]. В классической СУБД от разработчиков требовалось особое внимания к взаимодействию с медленными дисковыми накопителями. Минимизация числа обращений и времени доступа привела к множеству оптимизаций, таких как методы хранения данных (сжатие), способы доступа к данным (индексирование) и улучшение точности ответов. Также была введена модель выполнения операторов «производитель/потребитель», позволяющая максимально долго удерживать записи внутри процессора — в кэшах и регистрах. С другой стороны, увеличение объема оперативной памяти

(ОП) до нескольких терабайт и ее удешевление способствовали появлению баз данных в основной памяти. Алгоритмы, оптимизированные для классических СУБД, демонстрируют иную производительность при работе с данными, хранящимися в ОП. Фокус на оптимизации доступа к данным на диске смещается в сторону исполнения запросов [58]. СУБД начинают учитывать и использовать предвыборку команд (prefetching). Актуальность вторичных индексов снижается [59]. Необходимое число инструкций для выполнения запроса, будет являться одним из важнейших факторов наравне с производительностью подсистемы памяти [60]. Достаточные скорости доступа позволяют задействовать физический параллелизм на уровне множества ядер и векторных инструкций [61]. Наконец, исчерпание ресурсов этих техник [62] привело индустрию к исследованию возможностей повышения эффективности с помощью специализированных аппаратных модулей.

Одним из свойств системы является **гетерогенность**. Суть гетерогенности заключается в возможности совместного выполнения задачи устройствами, обладающими разной архитектурой. Это могут быть ядра центральное процессорное устройство ЦПУ с разными возможностями (например, наличие дополнительных векторных исполнителей или FP ALU), а также устройства с различными моделями программирования, такие как сочетание ЦПУ с графическими процессорами, тензорными процессорами, ускорителями ИИ, NPU, MIC и другими. Различия в наборах инструкций и микроархитектурных параметрах используются для оптимизации оборудования под специфические типы задач. Например, графический процессор обладает большей пропускной способностью памяти по сравнению с ЦПУ и способен одновременно обрабатывать значительно больше потоков. Тем не менее, из-за относительной простоты каждого ядра выполнение сложных арифметических операций, таких как деление, и сложных потоков исполнения, например, ветвления, оказывается менее эффективным. Разнообразие наборов инструкций делает механизмы генерации исполняемого кода и поддержку среды выполнения необходимыми условиями для интеграции различных устройств в единую систему. В

гетерогенных системах модели программирования могут быть разными. Например, центральный процессор управляет выполнением кода на графическом, который в свою очередь выполняет специализированные программы, называемые ядрами (kernel). Оптимизация запросов с использованием гетерогенных систем привела к ускорению работы как отдельных классов операторов [63], так и целых запросов [64, 65].

Однако переход на графический ускоритель в качестве основного устройства для вычисления запросов не гарантирует автоматического увеличения производительности для всех типов запросов. Наоборот, во многих случаях графический процессор уступает обычному ЦПУ, даже если данные находятся близко к нему [66]. В связи с тем, что производительность сильно варьируется в зависимости от нагрузки, необходимо уделять особое внимание подбору оптимальных устройств для выполнения запроса, целиком или по частям. Высокоскоростные сети в дата-центрах, появление новых типов быстрой неволатильной памяти, а также механизмы передачи данных между устройствами создают новый класс задержек с характерным временем, измеряемым в микросекундах [67]. При снижении времени отклика операций ввода-вывода и обращений к нелокальной памяти пропускная способность вычислительных устройств может снижаться. Современные способы уменьшения задержек памяти используют параллелизм инструкций, а для ввода-вывода полагаются на функции операционной системы, например, смену контекста. Оба подхода плохо масштабируются для операций в микросекундном диапазоне: параллелизм уровня инструкций ограничен, а смена контекста по времени сопоставима с передачей данных из нелокальной памяти, что приводит к значительным задержкам. Поэтому необходимо разработать новые алгоритмы для эффективного использования гетерогенных систем. Дополнительные сложности возникают в распределенных системах.

В связи с этим, разработчику СУБД с использованием гетерогенных устройств предстоит решить нижеперечисленные задачи:

1) Создать эффективную модель для использования доступного параллелизма и векторизации. Чтобы СУБД работала эффективно, её архитектура должна учитывать различия в программировании процессоров и грамотно планировать запросы.

2) Оптимизировать передачу данных между устройствами. Необходимо из-за сниженной производительности. Требуется механизм, оптимизирующий конфигурацию устройств и размещение данных в памяти для минимизации затрат на их перемещение.

3) Увеличивается пространство поиска оптимального плана. Это происходит из-за появления неоднородностей ресурсов. Эффективная работа оптимизатора невозможна без использования производительных алгоритмов поиска и новых эвристик.

4) Различия в архитектуре устройств. А именно процессоры и системы хранения данных, которые делают переиспользование моделей затрат для ЦПУ сложной задачей. Каждое устройство имеет свои уникальные характеристики, которые влияют на производительность выполнения запросов. Время, необходимое для доступа к данным, может значительно отличаться в зависимости от их расположения: кэш, оперативная память или диск. Поэтому требуется разработка единой модели затрат, которая учитывала бы не только удаленность данных, но и специфику каждого устройства. Это требует детального анализа архитектурных и микроархитектурных свойств аппаратуры, чтобы оптимально планировать исполнение запросов и эффективно загружать устройства.

5) С увеличением количества платформ, на которых работают системы управления базами данных (СУБД), возникают значительные сложности в их поддержке и разработке. Разные архитектуры требуют различных подходов к оптимизации, что приводит к усложнению исходного кода и увеличению времени на его поддержку. Для решения этих проблем необходимо разрабатывать методы, снижающие сложность и объем исходного кода. Это может включать использование абстракций, модульного проектирования и внедрение стандартных интерфейсов, которые позволят разработчикам легче адаптировать СУБД под

разные платформы. Прежде чем углубляться в эти решения, важно рассмотреть классические механизмы и подходы к оптимизации запросов, которые служат основой для дальнейших улучшений.

Увеличение производительности достигается разными подходами. Выбор подхода зависит от типа решаемой задачи. Ключевые параметры оптимизации могут включать среднее время обработки одной транзакции (время отклика), общую пропускную способность, измеряемую в транзакциях в секунду, количество задействованных ресурсов, таких как сервера, узлы выполнения и потоки, объем используемой памяти, а также уровень потребления энергии. В сценариях OLAP оптимизация направлена на два основных параметра:

• Время выполнения отдельного запроса — это метрика эффективности оптимизатора, определяющая «время ответа» базы данных. Минимизация этого показателя крайне важна для всех типов СУБД;

• Стоимость выполнения — это мера объема ресурсов, необходимых для обработки запроса. Оптимизация аналитического запроса начинается с построения логического плана на основе реляционных операторов. Полученный план представляет собой направленный ациклический граф (DAG).

Процесс оптимизации плана включает в себя:

• Поиск эффективных реализаций операторов логического плана с использованием физических операторов, реализованных (либо напрямую, либо сгенерированных) в СУБД;

• Выбор метода доступа к данным (например, использование вторичного индекса, если это необходимо, и способ хранения промежуточных результатов);

• Определение алгоритма и порядка соединения таблиц (join) среди семантически эквивалентных физических планов;

• Определение уровня параллелизма. После этого физический план передается на исполнение, а его эффективность зависит от модели выполнения.

Существуют несколько **моделей выполнения** физического плана:

• Итеративная модель (volcano или pipeline, tuple-at-a-time) — это классическая реализация, при которой каждый физический оператор выполняет

функцию перехода к следующему оператору в графе «next», вызываемую для каждой записи. Оператор «next» получает записи от дочернего оператора и рекурсивно спускается по графу, пока не достигнет листового оператора. Данный оператор получает табличную запись. Запись идет через цепочку операторов пока это возможно, прежде чем начнется обработка следующего кортежа (поздняя материализация [68]). Это позволяет минимизировать доступ к относительно медленному диску.

• Полная материализация (operator-at-a-time) [69] — каждый оператор сразу материализует все записи в памяти. Такой подход эффективен для OLTP-запросов, поскольку промежуточные результаты, как правило, небольшие и легко помещаются в кэш.

• Векторная обработка (vector-at-a-time) [70] — аналог итеративной модели, но вызов «next» возвращает не одну запись, а блок записей. Это очень эффективно при использовании векторной аппаратуры, например, одиночных и множественных потоков команд (ОКМД), и подходит для OLAP-запросов.

• Компиляция и конвейерная обработка (обычно operator-at-a-time/morsel) [71] — реализована в таких системах, как Hekaton [72], MemSQL [73], SparkImpala. Представлены три подхода к динамической компиляции: компиляция отдельных выражений, полных запросов и автоматическая специализация интерпретатора.

Качество генерируемого кода во многом зависит от выразительности промежуточных представлений, то есть способности этих представлений отражать детали алгоритма в форме, удобной для эффективного исполнения на аппаратуре. Генерация кода запроса, будь то исходный код или промежуточное представление, позволяет существенно уменьшить количество вызовов функций, что повышает эффективность. Кроме того, оптимизируется работа с кэшами путем повышения локальности, а также применяются компиляторские оптимизации для сокращения количества инструкций, необходимых для выполнения запроса. Однако из-за дополнительных затрат на материализацию промежуточных результатов, обусловленных ограничениями пропускной

способности памяти, полная материализация и итеративная модель гораздо сильно уступают в производительности двум другим моделям. [74].

Наиболее полное исследование сравнений векторного и скомпилированного исполнения для ЦПУ показывает, что нет однозначного лидера ни по производительности, ни по удобству использования. Векторный подход обеспечивает удобство отладки и профилирования, но требует ручной оптимизации примитивов. Компиляция избавляет от низкоуровневых деталей и позволяет использовать все преимущества существующих компиляторских оптимизаций, что может значительно сократить количество машинных инструкций по сравнению с векторизацией. К сожалению, анализ созданного кода конвейера операторов получается значительно сложнее, чем для заранее определенных примитивов. В гетерогенных системах векторное исполнение может быть неэффективным; например, для графического процессора накладные расходы на доступ к глобальной памяти могут превышать выгоду от векторизации, поскольку исполнение оператора часто не требует дополнительного параллелизма. С другой стороны, компиляция больших объемов кода конвейеров создает нагрузку на регистры и может привести к высокой конкуренции за ресурсы. Благодаря сгенерированному коду запроса автоматическая специализация интерпретатора может эффективно взаимодействовать с другими компонентами базы данных. Тем не менее, модель программирования графического процессора накладывает ограничения на такие оптимизации из-за специфики работы ускорителя: выполнение кода на графическом процессоре в режиме интерпретации оказывается крайне неэффективным. Одним из ключевых аспектов оптимизации является рациональное использование доступного многоуровневого параллелизма (многопроцессорного, многопоточного, ОКМД), который предлагает современное оборудование.

Одним из методов оптимизации является **параллелизм.** Центральный процессор, как и другие вычислительные устройства, развивается в направлении увеличения ресурсов для параллельного выполнения. Это касается как уровня

инструкций с глубокой конвейерной обработкой и изменением порядка команд (out-of-order), так и векторных вычислений. Применение параллелизма в OLAP-системах является более чем оправданным, поскольку запросы к базе данных независимы: каждый запрос может выполняться в отдельном потоке — это называется «interquery» параллелизмом. Большинство операторов в рамках одного запроса также могут выполняться параллельно, что способствует более эффективному использованию ресурсов и сокращению времени доступа к данным.

В рамках одного запроса обычным способом управления и применения параллелизма представляет собой введение дополнительного оператора (Exchange) в граф физического плана выполнения запроса. К основным видам внутреннего параллелизма относятся:

• Горизонтальный параллелизм — один оператор может выполняться одновременно в нескольких потоках. Например, оператор scan-filter может заранее разделить память (части одной таблицы) на сегменты и быть выполнен в нескольких потоках, каждый из которых работает с отдельным участком таблицы (partition). Полученные результаты затем необходимо агрегировать для обеспечения единообразия интерфейса операторов.

• Вертикальный параллелизм — в отличие от горизонтального, основывается на идее конвейеризации выполнения последовательных операторов в графе физического плана. Если следующий оператор не требует полной материализации результатов предыдущего, он может начать выполнение без ожидания завершения работы дочернего оператора — это типичная модель производитель/потребитель.

Существует альтернатива варианту вертикального параллелизма — параллельное выполнение независимых операторов.

Таблица 1.3. Требования и функции ПОС

Требования	**Функции ПОС**
реализация сбора и передачи данных полностью в электронной модели	Формирование множественных запросов к территориально-распределенным источникам данных посредством Web и ETL сервисов в режиме on-line. Настройка характеристик системы под каждого пользователя системы.
получение данных от источников в режиме on-line	
множественные запроса к ИС поставщиков компонентов ВС	
обработка разнородных данных	Преобразование данных ETL-сервисом, хранение в реляционной и нереляционной БД, выдача данных пользователям посредством Web-сервисов
работа с файлами в формате xml	
организация доступа к данным с признаками коммерческой тайны	Использование сертификата безопасности на устройствах пользователей и VPN между БД и источниками данных
использование для работы виртуальные частные сети (VPN)	
использование для работы сертификаты безопасности	

Архитектура NUMA позволяет разделять и размещать данные оптимальным образом. Она особенно важна в распределенных системах, где время передачи данных может значительно варьироваться [75]. Исследования показывают, что явное адаптивное управление размещением «горячих» данных и планирование выполнения способны значительно увеличить пропускную способность системы.

Модель исполнения играет ключевую роль в производительности реализации стратегии обработки запроса. При одинаковой стратегии семантически эквивалентные планы могут отличаться по эффективности. СУБД необходимо определить наиболее оптимальный план исполнения.

Разрабатываемая система ориентирована на решение конкретных задач; структура комплекса, представленная на рисунке 1.20, формируется на основе использования ETL-сервисов для сбора данных из различных источников, каналов передачи данных, разнообразных баз данных и Web-сервисов для передачи информации во внешнюю среду. В таблице 1.3 показаны требования и функции к ПОС.

При этом, вопросы обеспечения конфиденциальности решаются путем организации передачи данные безопасным соединением в зашифрованном виде, а именно: посредством формирования VPN от источника до баз данных, а СБ от баз данных к потребителям. В исходном состоянии комплекс настраивается на адресное пространство источников данных каждого конкретного пользователя. Причем, каждый новый пользователь имеет возможность создавать собственное адресное пространство и работать независимо от других пользователей. В процессе работы пользователь выбирает адреса необходимых источников, а ПО комплекса выполняет множественные запросы и осуществляет поиск цепочек прохождения сигнала минимального времени.

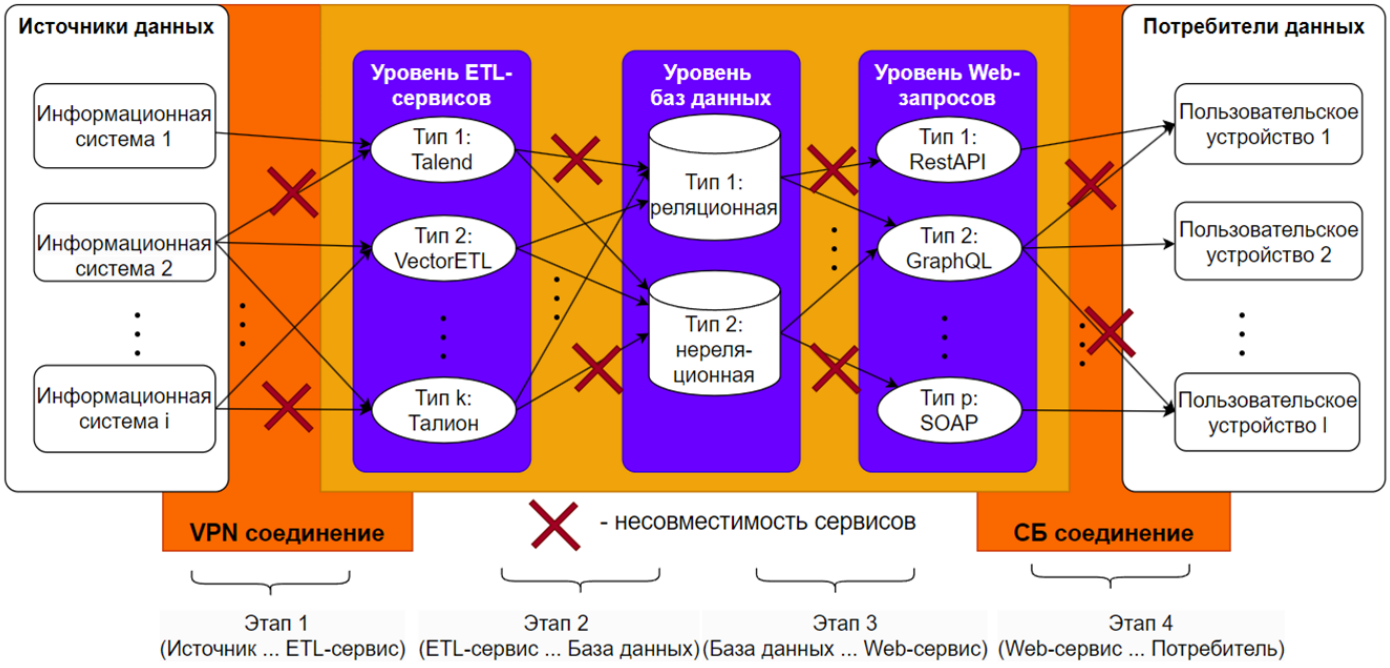

Рисунок 1.20. Структура комплекса и этапы прохождения сигнала в нём

1.3.6 Постановка задачи синтеза проблемно-ориентированной системы информационной поддержки решений для контроля подлинности паспортизированных компонентов воздушных судов

Как показано в разделе 1.3.5 развитие информационных технологий позволяет по-новому поставить и решить задачу синтеза проблемно-ориентированной системы информационной поддержки решений для контроля подлинности паспортизированных компонентов воздушных судов.

Входными данными в систему будут являться:

- адреса информационных систем, откуда получают данные о паспортизируемом компоненте, то есть о самом изделии и об его паспорте;
- паспортизируемый компонент, данные о котором запрашиваются;
- адреса устройств потребителей, на которые данные передаются.

Выходными данными из системы будут интегрированные данные о паспортизируемом компоненте из источников данных на разных этапах его жизненного цикла.

В общем виде задача синтеза оптимальной системы передачи данных в проблемно-ориентированной информационной системе сводится к комбинаторной задаче поиска связей множества внешних систем источников/потребителей данных с множеством хранилищ данных

Задача представляет собой отыскание отображения:

$\varphi : I \to L = I \to J \cap J \to L$, где (1.1)

$I = \{i1 \ldots ib\}$: множество внешних систем, откуда данные передаются в хранилище данных;

$L = \{l_1 \ldots l_h\}$: множество внешних систем, куда данные передаются из хранилища данных;

$J = \{j1 \ldots js\}$: множество хранилищ данных.

Для целочисленного описания отображения φ введем матрицу двоичных переменных

$U = (u_{il}),\ i \in I, l \in L,$ (1.2)

где $u_{i\mathrm{l}} = 1$ означает, что выбрана i – тая связь внешней системы с хранилищем вместе с l-ой связью хранилища и внешней системой, при этом

$$\sum u_{i\mathrm{l}} = 1 \ \forall i \in I \text{ и } \forall l \in L \quad (1.3)$$

Тогда постановку задачи можно записать следующим образом:

Минимизировать $\sum_{m=1}^{M} Cm(U, \mathrm{J}) \rightarrow \min$ (1.4)

где:

M- количество всех возможных связей для передачи данных между внешними системами I и L;

I – количество внешних систем, откуда данные передаются в хранилище данных;

J –множество хранилищ данных;

L – количество внешних систем, куда данные передаются из систем хранения данных.

При ограничениях общего вида:

$x_{ij}^{k} \geq 0$, $\mathrm{c}_{ij}^{k} \geq 0$, $y_{jl}^{p} \geq 0$, $q_{jl}^{p} \geq 0$,

и ограничениях на переменные:

$k>0, i>0, j>0, p>0, l>0 \in N$.

Таким образом, для решения задачи разработки проблемно-ориентированной системы информационной поддержки решений для контроля подлинности паспортизированных компонентов воздушных судов необходимо:

- разработать модель описания проблемно-ориентированной системы информационной поддержки решений;
- разработать алгоритм оптимизации параметров проблемно-ориентированной системы, обеспечивающий организацию информационного взаимодействия разнородных источников/потребителей данных за минимальное время;
- разработать метод выбора характеристик проблемно-ориентированной системы с учетом параметров интегрируемых источников/потребителей данных.

- дополнить существующую информационную модель службы обмена сообщениями сущностями и атрибутами совместимости, необходимых для расчета минимального времени
- разработать программный комплекс, реализующий предложенные модель, методы и алгоритмы.

1.4 Выводы к главе

Выполненные в данной главе исследования и разработки позволяют сформулировать следующие выводы:

1) С увеличением сложности изделий авиационной техники и требованиями по сокращению сроков проведения работ по подтверждению подлинности компонентов (в составе паспортов и собственно изделий) воздушных судов становится предпочтительным организация соответствующих работ в рамках проблемно-ориентированной информационной системы, реализующей необходимые функции по оценке подлинности компонента на текущий момент времени и в течение жизненного цикла.

2) Ключевыми технологиями, нацеленными на формирование эффективной системы ПОС являются ETL-сервисы, Web-сервисы, базы данных, VPN и сертификаты безопасности. Результаты расчета времени прохождения сигнала по цепочкам передачи данных с учетом совместимости сервисов используются для синтеза ПОС

3) Несмотря на наличие проработанных методик и инструментальных средств, современный уровень развитие новых технологий преобразования и защиты данных позволяет по-новому поставить и решить задачу синтеза объектно-ориентированной системы с учетом осуществления поиска цепочек прохождения сигнала за минимальное время. Эта задача может быть решена комплексно – включая разработку математической модели системы, информационной модели, дополненной сущностями и атрибутами совместимости, необходимых для расчета минимального времени, метода и алгоритм синтеза ПОС, программного комплекса.

2. РАЗРАБОТКА МОДЕЛИ ОПИСАНИЯ ПРОБЛЕМНО-ОРИЕНТИРОВАННОЙ СИСТЕМЫ ИНФОРМАЦИОННОЙ ПОДДЕРЖКИ РЕШЕНИЙ ДЛЯ КОНТРОЛЯ ПОДЛИННОСТИ ПАСПОРТИЗИРОВАННЫХ КОМПОНЕНТОВ ВОЗДУШНЫХ СУДОВ

2.1 Модель описания проблемно-ориентированной системы

Для описания структуры модели ПОС определим следующие множества:

I – внешних систем, откуда данные передаются в хранилище данных;

L – внешних систем, куда данные передаются из хранилища данных;

J – хранилищ данных.

Внешние системы и хранилища связаны множественными связями (цепочками прохождения сигнала), характеризуемыми следующими множествами:

K – ETL-сервисом между внешними системами, откуда данные передаются в хранилище данных и самими хранилищами данных;

P – web-сервисов между системами хранения данных и внешними системами, куда данные передаются.

Входящими данными в модели являются:

W – множество запрашиваемых данных о компонентах ВС, формируемых пользователями.

Исходящими данными будут результаты этих запросов, используемые пользователями для дальнейших вычислений.

Для оценки совместимости сервисов и последующего прохождения сигнала от источника до потребителя автором предложено ввести переменные модели:

$a[i, j, k, w]$=1 участок модели, где сигнал проходит , $i \in I, j \in J, k \in K\text{-}\ k', w \in W$;

$a[i, j, k, w]$=0 участок модели, где сигнал не проходит , $i \in I, j \in J, k \in K - k'$, $w \in W$;

$b\ [j, l, p, w]$=1 участок модели, где сигнал проходит , $j \in J$, $l \in L, p \in P - p'$, $w \in W$;

b [j, l, p, w]=0 участок модели, где сигнал не проходит, $j \in J$, $l \in L$, $p \in P - p'$, $w \in W$.

Перепишем в язык алгебры логики:

a[i, j, k, w] → b[j, l, 1, w] b[j, l, 2, w] b[j, l, 3, w] …;

A → B ↔ ⌐A B;

⌐a[i, j, k, w] b [j, l, 1, w] b[j, l, 2, w] b [j, l, 3, w] …

Перепишем в уравнение неравенства:

(1 - a[i, j, k, w]) + b[j, l, 1, w] + b[j, l, 2, w] + b[j, l, 3, w] + +… >=1

При этом: a[i, j, k, w]>0 и b [j, l, p, w]>0 .

Введем следующие обозначения:

x_{ij}^{kw} - объем k-ых входящих данных из i-ой внешней системы в j-ю систему хранения данных (все топологические варианты связей);

c_{ij}^{kw} - скорость передачи данных ETL-сервисом (все топологические варианты связей);

y_{jl}^{pw} - объем p-ых исходящих данных из j-ой системы хранения данных в l-ю внешнюю систему (все топологические варианты связей);

q_{jl}^{pw} - скорость передачи данных web-сервисами (все топологические варианты связей).

Введем ограничения общего вида: $x_{ij}^{kw} \geq 0$, $c_{ij}^{kw} \geq 0$, $y_{jl}^{pw} \geq 0$, $q_{jl}^{pw} \geq 0$, и ограничениях на переменные: k>0, i>0, j>0, p>0, l>0,w>0 ∈N

Исходя из этого задачу целочисленного линейного программирования (**ЦЛП**) с двоичными переменными (2.1) можем записать следующим образом:

$$X_{ilw} = \{(k,j,p) \in C_{ilw} | \frac{x_{ij}^{kw}}{c_{ij}^{kw}} + \frac{y_{jl}^{pw}}{q_{jl}^{pw}} = m_{ilw}\} \subset C_{ilw}, \text{ где}$$

$$m_{ilw} = \min_{(k,j,p) \in C_{ilw}} \left(\frac{x_{ij}^{kw}}{c_{ij}^{kw}} + \frac{y_{jl}^{pw}}{q_{jl}^{pw}} \right) \quad (2.1)$$

После описания ПОС необходимо сформировать информационную модель, которая рассматривается в разделе 2.2 .

2.2 Информационная модель проблемно-ориентированной системы

За основу разработки информационной модели ПОС была взята модель службы обмена сообщениями [76], которая показан на рисунке 2.1 .

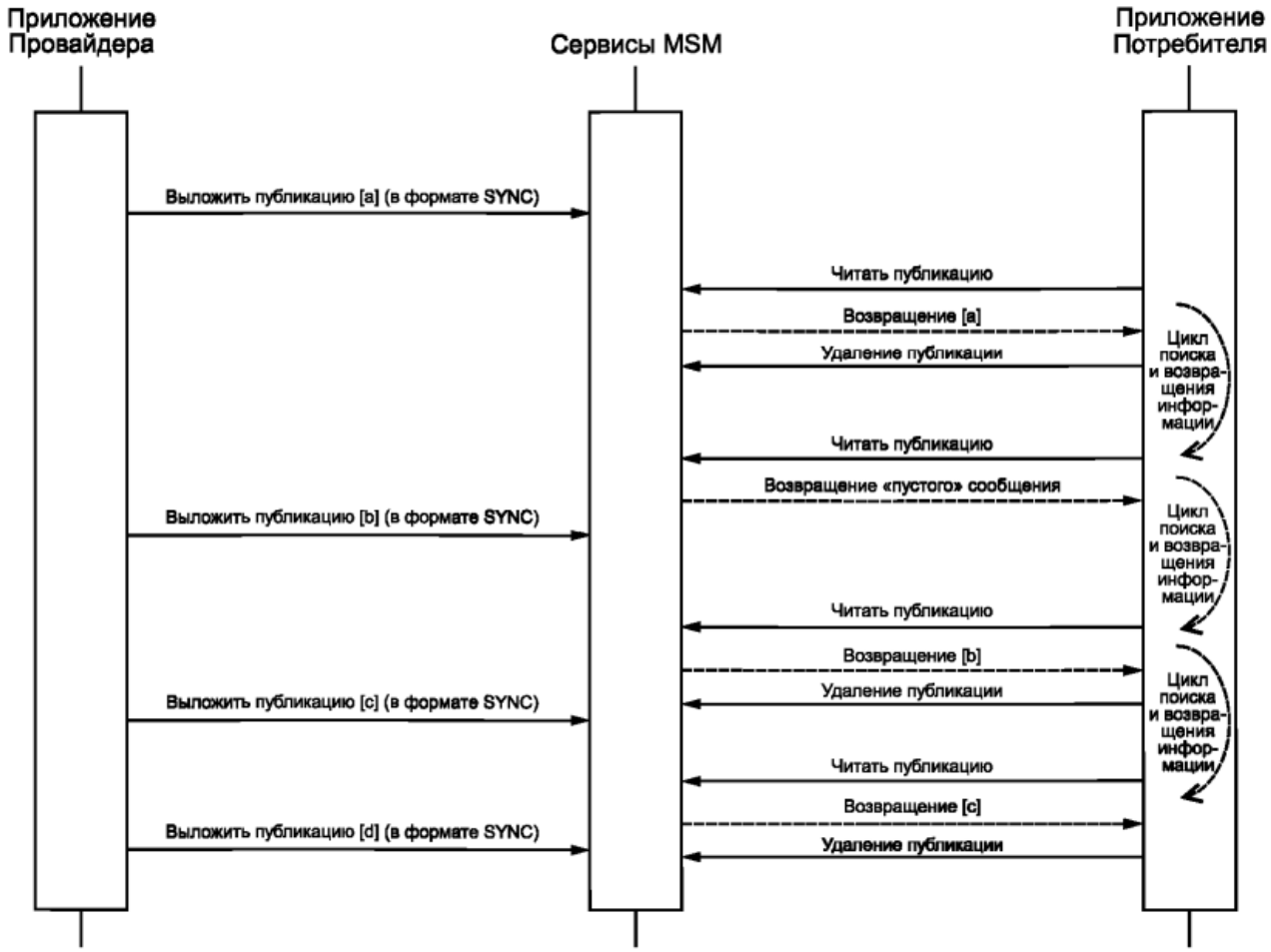

Рисунок 2.1. Модель службы сообщения

В данной модели описывается сценарий публикации и подписки с несколькими сообщениями между приложением провайдера и приложением потребителя через сервисы.

В этом сценарии приложение провайдера начинает сеанс связи для публикации службы обмена сообщениями (MSM) в этом канале. Допустим, что приложение провайдера принимает решение о публикации данных, тогда будут происходить следующие действия:

- выкладывание публикация;
- указывание темы сообщения;
- сообщение считается непросроченным.

Приложение потребителя устанавливает соединение с каналом публикаций MSM. В этом канале указывается список тем, а идентификатор соединения сохраняется для последующего использования (в случае, если приложение пользователя неожиданно завершает работу и затем перезапускается).

Приложение потребителя начинает передачу информации по MSM каналу для публикаций одним из двух способов: периодически или по наступлению некоторого события. На то, что новая публикация возвращена, указывает возвращение информации после прочтения. Из следующего вызова процедуры ReadPublication (чтение публикации) происходит возвращение либо следующей публикации из очереди на подписку, либо пустого сообщения (в случае если публикаций больше нет).

Разработанная в данной работе модель ПОС, которая показана на рисунке 2.2, основана на модели службы обменом сообщений и включает в себя описания источников данных, ETL-сервисов [77], Web-сервисов. Эта модель отличается от известной включением дополнительных сущностей и атрибутов совместимости, которые необходимы для расчета минимального времени, а именно:

- Источник данных: входит тип системы (документооборот, система управления данными об изделии и т.д.), тип данных (например: xml, json и т.д.), атрибуты данных (различные виды физических и геометрических величин и т.д.);

- ETL-сервис, который на основе типа данных и типа системы поддерживает взаимодействие с источником данных и, проведя преобразования, сохраняет извлеченные атрибуты данных в базы данных;

- Web-сервис, который на основе типа базы данных (например: SQL, NoSQL и т.д.) и типов данных обеспечивает взаимодействие с базами данных с передачей информации на устройства потребителей.

Таким образом информационная модель позволяет производить структурно-параметрический синтез оптимальной ПОС.

После описания информационной модели ПОС необходимо оптимизировать параметры системы. Алгоритм оптимизации параметров ПОС описан в разделе 2.3 .

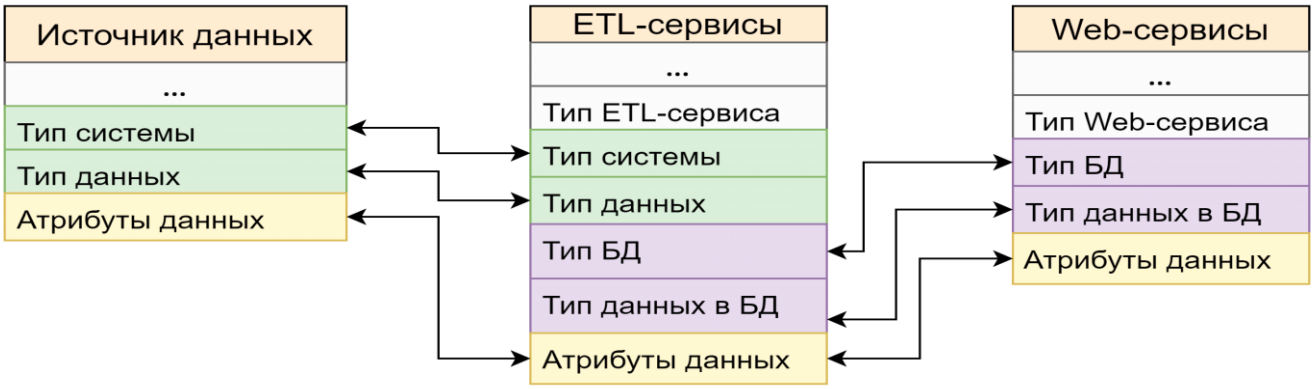

Рисунок 19.2. Информационная модель ПОС

2.3 Алгоритм оптимизации параметров проблемно-ориентированной системы

В основу предложенного автором алгоритма оптимизации параметров системы положена структурная схема (рис. 2.3), включающая следующие элементы:

- источники данных разного типа от 1 до n;
- хранилища данных (базы данных) от 1 до j, j=2;
- Web-сервисы всего 1 … p;
- потребители данных общим количеством от 1 до l.

При отношении несовместимости на этапе t =+∞ . Множество цепочек прохождения сигнала от источника к потребителю:

$C_{11w} = \{(k, j, p) \in K \times J \times t_1[1, j, k, w] = 1, t_4[j, 1, p, w] = 1\}$, так как

$t_1 \gg t_2$ и $t_4 \gg t_3$, то $t_2 = t_3 = 0$ (2.2) .

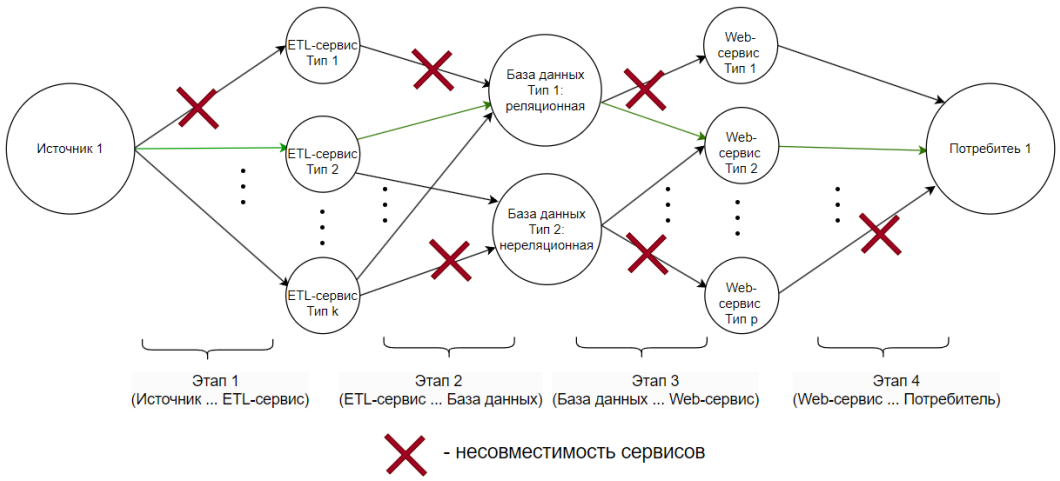

Рисунок 20.3. Схема структуры проблемно-ориентированной системы

Необходимо найти цепочки участков прохождения сигнала от источника до потребителя за минимальное время с учетом совместимости сервисов, то есть

$X_{i1w}=\{(k,j,p) \in C_{i1w} | \frac{x_{1j}^{kw}}{c_{1j}^{kw}}+\frac{y_{j1}^{pw}}{q_{j1}^{pw}}=m_{i1w}\} \subset C_{11w}$, **для** $m_{11w} =$

$= \underset{(k,j,p)\in C_{11w}}{\boldsymbol{min}} \left(\frac{x_{1j}^{kw}}{c_{1j}^{kw}}+\frac{y_{j1}^{pw}}{q_{j1}^{pw}}\right)$ (2.3) , **при ограничениях:**

$t_1[1, j, k, w]$=1 участок модели, где сигнал проходит , $j \in J$, $k \in K$- k', $w \in W$;

$t_1[1, j, k, w]$=0 участок модели, где сигнал не проходит , $j \in J$, $k \in K$ - k' , $w \in W$;

$t_4[j, 1, p, w]$=1 участок модели, где сигнал проходит , $j \in J$, $p \in P$ - p', $w \in W$;

$t_4[j, 1, p, w]$=0 участок модели, где сигнал не проходит, $j \in J$, $p \in P$ - p', $w \in W$.

Для решения данной задачи автор предложил использовать метод динамического программирования Беллмана-Форда.

Введем состояние (f_r) на r этапе, которое определяет время прохождение сигнала по цепочке участков затраченное к r этапу: $f_r= \sum_{t=1}^{4} t$.

Введем управление (u_r) на r этапе – время прохождения сигнала на r этапе в зависимости от выбора соответствующего сервиса: $u_1=\left(\frac{x_{1j}^{kw}}{c_{1j}^{kw}}\right)$, $u_4=\left(\frac{y_{j1}^{pw}}{q_{j1}^{pw}}\right)$. То есть на первом этапе выбор типа ETL, а на четвертом – выбор типа Web.

Введем управление связи $\vec{f}_r = \vec{f}_{r-1} + \vec{u}_r$, определяющее изменение состояния процесса под действием управления, на множестве допустимых управлений: $U'_{\text{доп}} = \bigcap\limits_{r} U'^{r}_{\text{доп}}$.

Тогда основное динамическое уравнение Беллмана будет иметь вид:

$$Q_r^*(\vec{f}_{r-1})=\min\limits_{\vec{u}_r\in U'_{\text{доп}}} \{\vec{f}_r(\vec{u}_r)+Q_{r+1}^*(\vec{f}_{r-1}+\vec{u}_r)\} \min\limits_{\vec{u}_r\in U'_{\text{доп}}} \{t_r{+}{+}Q_{r+1}^*(\vec{f}_{r-1}+\vec{u}_r)\} \quad (2.4)$$

$$\vec{u}^*_r(\vec{f}_{r-1})=\arg(\min\limits_{\vec{u}_r\in U'_{\text{доп}}} \{t_r+Q_{r+1}^*(\vec{f}_{r-1}+\vec{u}_r)\}) \quad (2.5)$$

Где $Q_r^*(\vec{f}_{r-1})$ – функция Беллмана (2.4), а $\vec{u}_r^*(\vec{f}_{r-1})$ (2.5) – условно оптимальное управление на r –ом этапе. Параллельно независимо находится управление и для источника №2, и последующих источников. Алгоритм решения реализован с использованием пакета npm bellman-ford [78].

На рисунке 2.3 зеленым цветом показан пример нахождения оптимальной цепочки участков сети с минимальным временем прохождения сигнала.

Нахождение оптимальной цепочки является только частью режима настройки, который описан в разделе 2.4 .

2.4 Метод выбора характеристик проблемно-ориентированной системы

Созданная автором ПОС работает в двух режимах: настройка и эксплуатация.

В режиме настройки автор предложил использовать следующий метод выбора характеристик:

1) Формируем исходные данные: адрес устройства потребителя данных l=1 (подмножество L). Web-адреса множества источников данных (множество I). Список запрашиваемых объектов в системе (множество W).
2) Запускаем множественные сетевые запросы и получаем выходные данные (технически совместимые варианты связей):

$x_{i-ij}^{k-k'w}$ - объем k-k` входящих данных из i-ой внешней системы в j-ю систему хранения данных;

$c_{i-ij}^{k-k'w}$ - скорость передачи данных ETL-сервисом данных;

$y_{j1}^{p-p'w}$ объем p-up` исходящих данных из j-ой системы хранения данных во внешнюю систему пользователя;

$q_{j1}^{p-p'w}$ - скорость передачи данных web-сервисами.

3) После получения и обработки данных вычисляем необходимые характеристики: скорость от источника к хранилищу, объем принятых данных от источника к хранилищу, скорость от хранилища к потребителю, объем принятых данных от хранилища к потребителю.

4) Методом последовательного анализа Вальда определяем количество замеров характеристик.

Таблица 2.1. Блок характеристик ПОС

Источник данных	**Установление связи**	**Скорость от источника к хранилищу, (мб/с)**	**Объем принятых данных от источника к хранилищу, (мб)**	**Скорость от хранилища к потребителю, (мб/с)**	**Объем принятых данных от хранилища к потребителю, (мб)**	**Количество замеров характеристик**
Разработчик ВС	Не установлена	-	-	-	-	5
Разработчик ВС	Установлена	2	1	1	1	8
Эксплуатирующая организация 1	Установлена	1	1	0,5	1	6
Эксплуатирующая организация 2	Установлена	1,5	1	1,5	1	6
Ремонтная организация 1	Установлена	0,5	1	2	1	7
Изготовитель 1	Установлена	1	1	1	1	6

5) Выбираем наилучшие варианты связей и формируем блок характеристик объектно-ориентированной системы, показанной на таблице 2.1 .

Когда ПОС настроена и находится в режиме эксплуатации, то пользователь может запрашивает данные.

2.5 Выводы к главе

Выполненные в данной главе исследования и разработки позволяют сформулировать следующие выводы:

1) В поставленной в главе 1 задаче критерием оптимизации является минимальное время прохождения сигнала по цепочкам передачи данных. Приведенные в разделах 2.1 и 2.3 математическая модель и алгоритм расчета охватывают все необходимые исходные данные и ограничения, включая перечень цепочек, временные характеристики прохождения сигнала по каждой цепочке,

признаки совместимости сервисов на участках от источников данных до баз данных и от баз данных до потребителя.

2) В силу большого количества исходных параметров для расчета минимального времени все исходные данные и результаты расчетов должны храниться в структурированном хранилище информации в составе базы данных Приведенная в 2.2 разделе информационная модель обеспечивает такое хранение данных, а также возможность автоматизированной обработки этой информации с помощью программных средств.

3) Определение оптимальной структуры системы может осуществляться на основе алгоритма, одновременно обеспечивающего выбор цепочек прохождения данных и учет ограничений, вызванных несовместимостью сервисов. Автором предложен в разделе 2.3 алгоритм и его реализация методом динамического программирования Беллмана-Форда на основе пакета npm bellman-ford.

3. ПРОГРАММНЫЙ КОМПЛЕКС СИНТЕЗА ПРОБЛЕМНО-ОРИЕНТИРОВАННОЙ СИСТЕМЫ ИНФОРМАЦИОННОЙ ПОДДЕРЖКИ РЕШЕНИЙ ДЛЯ КОНТРОЛЯ ПОДЛИННОСТИ ПАСПОРТИЗИРОВАННЫХ КОМПОНЕНТОВ ВОЗДУШНЫХ СУДОВ.

3.1 Структура программного комплекса

Для проверки предложенной модели, метода и алгоритма был разработан программный комплекс, который включает следующие компоненты:

- Модуль ETL-сервисов;
- Модуль Web-сервисов;
- Модуль баз данных (SQL и NoSQL);
- Таблица адресов источников данных;
- Модуль VPN, через которые взаимодействуют ETL-сервисы с источниками данных;
- Модуль сертификатов безопасности, через которые потребители подключаются через Web сервисы;
- Алгоритм подбора ETL-сервисов к источникам;
- Алгоритм подбора Web-сервисов для потребителей.

Рассмотрим каждый модуль отдельно.

Технология ETL. ETL (Extract, Transform, Load: извлечение, трансформация, загрузка) является системой, которая может решать следующий набор задач:

1) извлечение данных из различных источников данных (также использование Web-сервисов);

2) преобразование и форматирование данных;

3) загрузка в хранилище данных.

ETL имеет расширенный функционал (относится к преимуществам данной технологии) [79]. Сравнение ETL-сервисов показано в таблице 3.1 .

Извлечение (Extract). Существуют адаптеры к различным источникам данных: ручная загрузка, Web-сервисы, базы данных. Базы данных делятся на

NoSQL (not only SQL — не только SQL) и SQL (structured query language — язык структурированных запросов).

Преобразование (Transform). Оно состоит из фильтрации данных и конвейера (переход от одной схемы к другой).

Таблица 3.1. Таблица сравнения ETL-сервисов

Название	**Наличие бесплатного (пробного) использования**	**Исходные данные**	**Полученные данные**	**Наличие API**
Hevo Data	Есть	Данные реляционной базы данных, данные нереляционной базы данных	Данные реляционной базы данных	RestAPI
Pentaho	Есть	Данные реляционной базы данных, csv, xml	Данные реляционной базы данных	RestAPI
Talend	Есть	Данные реляционной базы данных, csv	Данные реляционной базы данных, csv	RestAPI
AWS Glue	Есть	Данные реляционной базы данных, csv	Данные реляционной базы данных, csv	RestAPI
Informatica PowerCenter	Есть	Данные реляционной базы данных, xml	Данные реляционной базы данных, xml	RestAPI

Выгрузка данных (Load, загрузка в систему полученных данных). Адаптеры, работающие на запись (аналогичны адаптерам для извлечения).

Наличие визуальных средств для описания конвейера (редакторы для создания схем преобразования данных и т.п.);

Наличие возможности масштабирования (не во всех платформах заложено).

В ETL существуют недостатки, как и в любой системе, а именно:

- ограничение масштабирования процесса;

- нельзя, используя одну операцию, производить интеграцию данных из множества источников [80].

Продукт предоставляет компания Talend, а именно Talend Open Studio [81]. Платформа предлагает интеграцию с реляционными базами данных [82], а именно MySQL, MSSQL и PostgreSQL [83], показанного на рисунке 3.1 Существует возможность загрузки csv файла, а также использование технологии RestAPI [84].

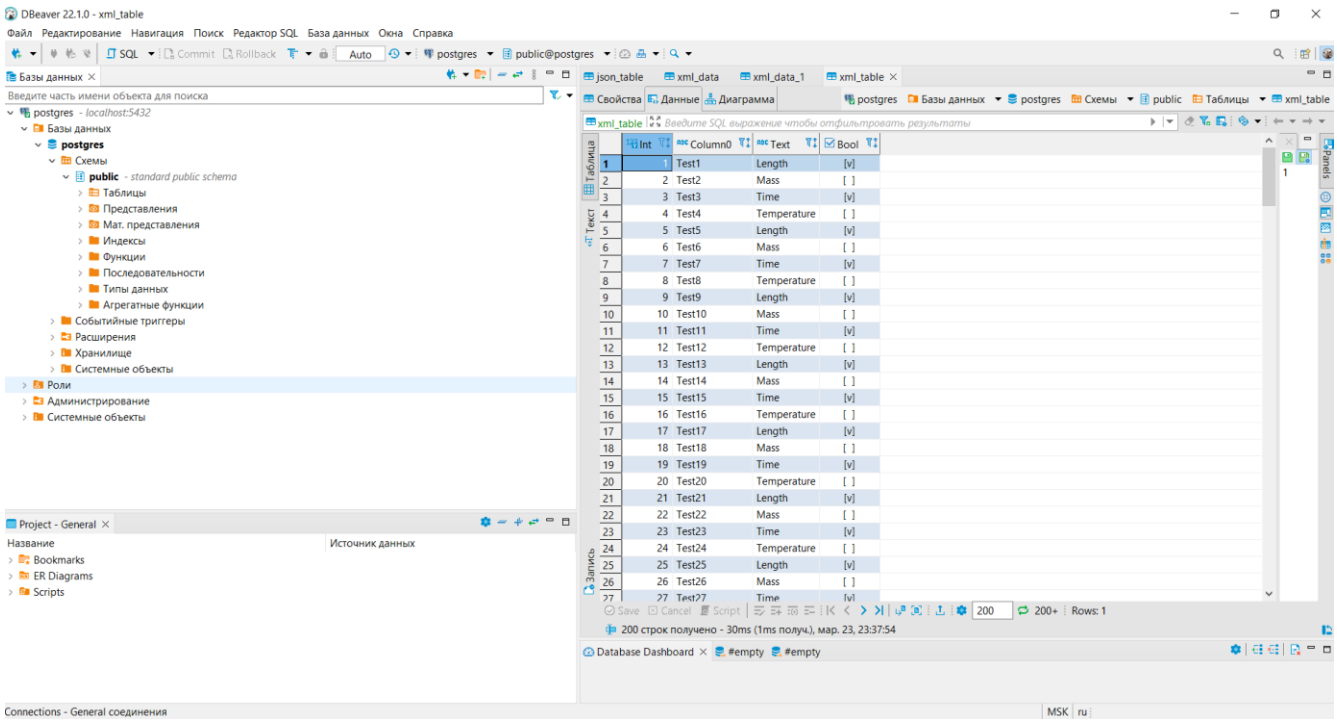

Рисунок 3.1. Таблица баз данных в PostgreSQL

Процесс состоит из создания бизнес-процесса, показанном на рисунке 3.2 с выстраиванием соответствующих графических элементов в рабочем пространстве программы.

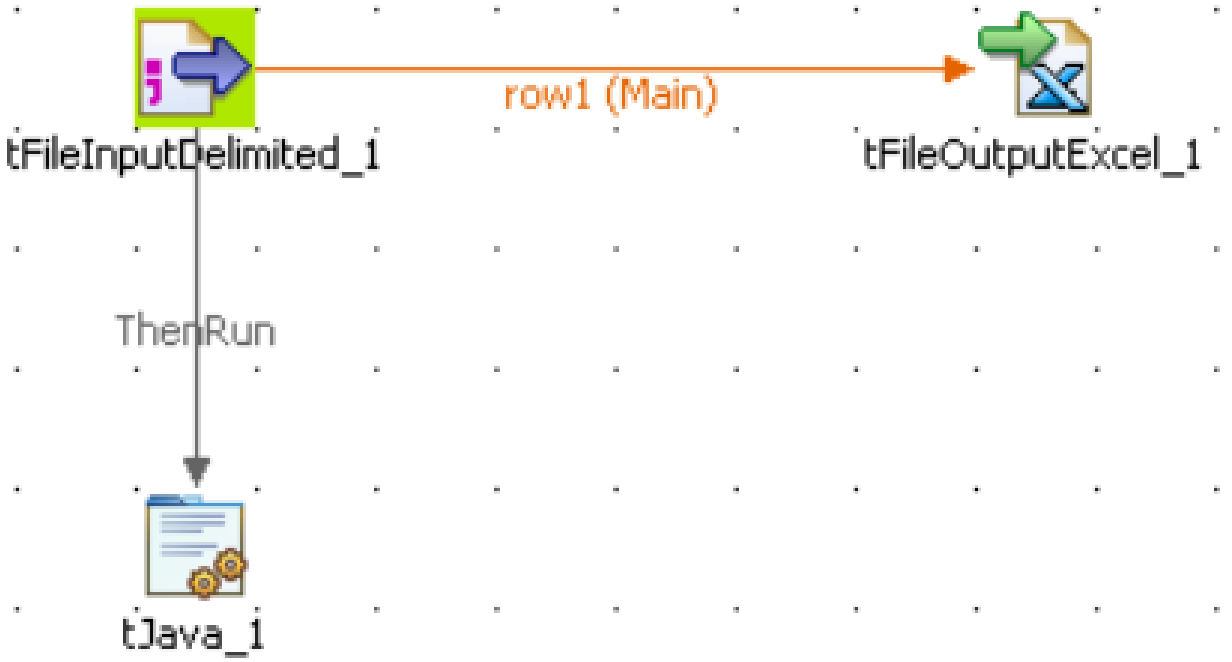

Рисунок 3.2. Схема бизнес-процесса в Talend Open Studio

Преобразования происходят с помощью графического элемента tJava, показанному на рисунке 3.3 . Данный элемент использует язык программирования Java [85] для преобразования данных из одной структуры в другую.

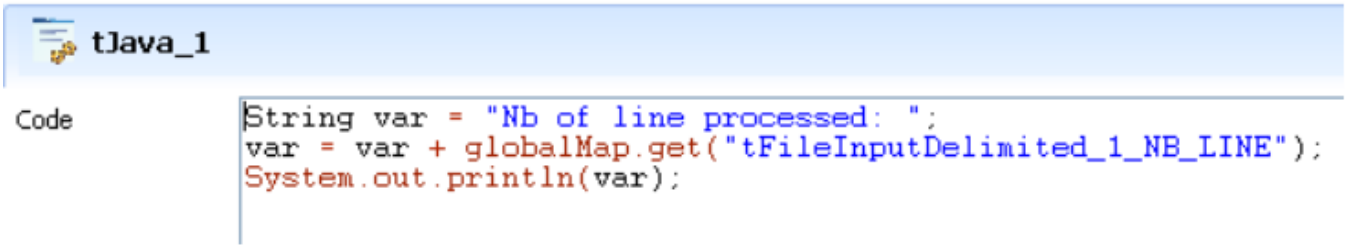

Рисунок 3.3. Интерфейс элемента Java

Финальный вариант схемы интеграции в системе Talend Open Studio показан на рисунке 3.4 .

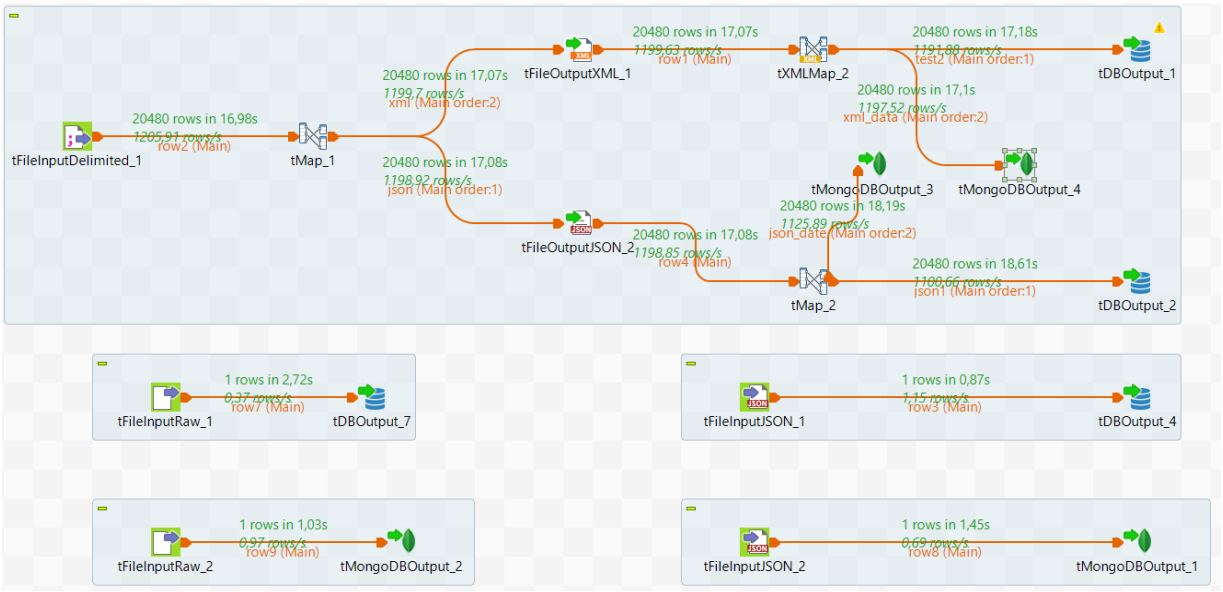

Рисунок 3.4. Схема интеграции в системе Talend Open Studio [86]

Рассмотрим **модуль web-сервисов** в программном комплексе для которого используется web-сервисы RestAPI и GraphQL [87]. При возникновении потребности, есть возможность добавлять новые web-сервисы. Функционал RestAPI позволяет работать с различными форматами данных. В системе добавлены возможности использования форматов xml и json. Статистика использования технологий для клиент-серверной архитектуры показана на рисунке 3.5 .

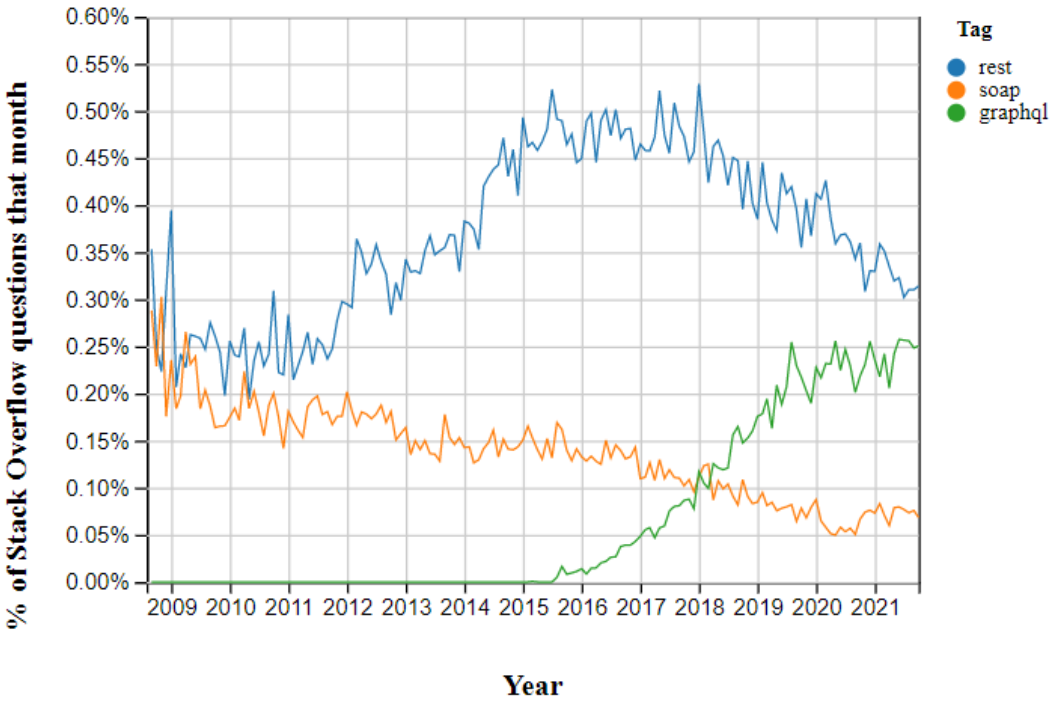

Рисунок 3.5. Статистическая информация об использовании технологий для клиент-серверной архитектуры

XML — это стандартизированный текстовый формат. Он используется для хранения данных и был разработан в 1998 году [88].

К его преимуществам можно отнести:

- возможность работы с различными типами данных и отображения информации благодаря языку разметки;
- поддержку различных кодировок;
- использование пространств имен.

Среди недостатков выделяются:

- отсутствие явного указания типов данных;
- все данные представлены в виде строк;
- низкая читаемость для человека;
- сложность кросс-браузерного анализа XML.

JSON— это стандартизированный текстовый формат. Он используется для хранения данных и был разработан в 2001 году [89]. Преимуществами данного формата является:

- наличие типов для объектов;
- поддержку различных типов данных (строки, числа, массивы, логические значения);
- удобочитаемость для человека;
- совместимость с большинством браузеров.

Однако у JSON есть и недостатки:

- поддержка ограничена только текстовыми и числовыми данными;
- отсутствие возможностей для отображения данных;
- поддержка только кодировки UTF-8;
- отсутствие поддержки пространств имен.

В реляционных базах данных, как и в нереляционных может храниться формат данных XML, так и формат данных JSON [90].

Для проведения оценки частоты использования выбранных форматов, была использована статистика данных сайта Stack Overflow в период с 2009 по 2021 год [91]. Частота использования формата json возрастает, а в формате xml

уменьшается. Можно отметить, что в 2009 году XML был более востребован, чем JSON примерно в 4 раза, а в 2021 году произошел полный поворот и JSON примерно в 3 раза более востребован, чем XML. Итоговый результат анализа показан на рисунке 3.6 .

Рисунок 3.6. Частота использования форматов xml и json [92]

Рассмотрим **модули баз данных** в программном комплексе. Реляционные базы данных используют реляционную модель в качестве базовой концепции. Она была предложена Эдгаром Коддом, который работал в компании IBM во второй половине двадцатого века. Структуры данных состоят из таблиц. Столбцы обозначают типы данных. В ячейках записаны значения. Строки представляют собой набор связанных значений, они относятся к одной сущности. Связь между таблицами происходит благодаря ключам [93].

Впервые стандарт языка структурированных запросов был одобрен в 1987 году Международной организацией по стандартизации (ISO) под названием SQL/86. На текущий момент последний стандарт: SQL/2016 [94].

Существуют различные реляционные (SQL) базы данных. К базам данных с открытым исходным кодом можно отнести MySQL, PostgreSQL и т.д. Напротив, к базам данных с закрытым исходным кодом относятся Microsoft SQL Server, Oracle Database и т.д.

Хранение данных в нереляционных, то есть NoSQL (not only SQL — не только SQL) базах происходит иначе. Различаются несколько типов:

- Key/value store (хранилища пар «ключ-значение») – является хэш-таблицей, которая содержит значения и советующие им ключи;
- Документо-ориентированные базы – используется для хранения иерархических систем данных;
- Колоночные базы данных – данные хранятся в строках, а не в колонках (как в реляционных базах данных). Колонка является отдельной таблицей. Она может хранить только свои значения (чаще всего в отсортированном виде). Каждая колонка хранится в отдельном файле;
- Графовые базы данных – используются для обработки графов.

Примерами нереляционных баз данных (NoSQL) можно назвать следующие системы: MongoDB, Redis, Elasticsearch [95].

Как реляционные базы данных, так и NoSQL базы на текущий момент могут хранить распространённые типы данных для взаимодействия с Web-сервисами: xml и json [96]. Если в реляционных СУБД (система управления базами данных) сначала появилась поддержка xml, а потом json, то в NoSQL базах было все наоборот.

Сравнивая производительность реляционных и NoSQL СУБД, можно выделить следующие факты (сравнивались реляционная СУБД PostgreSQL и NoSQL MongoDB) [97]:

1. MongoDB отстает от PostgreSQL в операциях вставки данных (использовались JSON-файлы и распределенное окружение), что идет вразрез с существующим заблуждением среди специалистов о преимуществе MongoDB в задачах логирования информации;

2. В подавляющем большинстве случаев MongoDB выполняет задачи быстрее PostgreSQL по работе с JSON-файлами. Разницу по времени можно считать незначительной, так как сравнивались реляционная СУБД и СУБД, для которой JSON-файл является почти единственным для использования;
3. Результаты экспериментов показали, что MongoDB занимает ведущую позицию в индексированном поиске. Это указывает на то, что использование этой СУБД является оптимальным решением для хранения данных, которые редко изменяются, но часто запрашиваются

В разрабатываемой системе источниками данных станут адреса, записанные в таблице. Она состоит из названия источника и самого адреса. Адрес представлен в формате интернет-протокола четвертной версии (IPv4) [98], которые показаны в таблице 3.2 .

Таблица 3.2. Пример адресов источников данных в формате IPv4.

Источник данных	**Адреса источников**
Разработчик ВС	10.0.10.10
Разработчик ВС	11.0.10.11
Эксплуатирующая организация 1	120.12.11
Эксплуатирующая организация 2	14. 0.14.11
Ремонтная организация 1	15.0.15.11
Ремонтная организация 2	16.16.14.11
Изготовитель 1	17.16.17.11
Изготовитель 2	18.16.17.18

Рассмотрим **модуль виртуальных приватных сетей (VPN) в** программном комплексе**.** Раньше наиболее распространенным способом соединения компьютеров между разными офисами было использование выделенной линии. Выделенные линии, такие как ISDN (интегрированная цифровая сеть услуг с пропускной способностью 128 Кбит/с), представляют собой частные сетевые соединения. Телекоммуникационная компания может сдавать такие соединения в аренду своим клиентам. Используя выделенные линии, компании получают возможность расширить свою частную сеть за пределы своей непосредственной географической зоны. Данные соединения создают единую глобальную сеть (WAN) для бизнеса. Хотя выделенные линии надежны и безопасны, аренда обходится дорого, и расходы растут по мере увеличения расстояния между офисами.

Благодаря глубокому проникновению Интернета поставщики интернет-услуг (ISP) продолжают разрабатывать более быстрые и надежные услуги с ценами ниже, чем выделенные линии. Чтобы воспользоваться этим, большинство предприятий заменили выделенные линии новыми технологиями, которые используют интернет-подключения без ущерба для производительности и безопасности.

Основные функции VPN [99]:

Безопасность — VPN должен защищать данные, пока они передаются по публичной сети. В случае попытки злоумышленников перехватить данные, они не смогут их прочитать или использовать;

Надежность — сотрудники должны иметь возможность подключаться к VPN без проблем в любое время (если только часы работы не ограничены), а VPN должна обеспечивать одинаковое качество соединения для каждого пользователя, даже если она обрабатывает максимальное количество одновременных подключений;

Масштабируемость — по мере роста количества пользователей у него должна быть возможность расширять свои VPN-сервисы, чтобы справиться с этим ростом, не заменяя полностью технологию VPN.

Туннелирование позволяет большинству VPN обеспечить безопасное соединение в частной сети, передавая данные через общедоступный Internet. Туннелирование — это процесс помещения целого пакета в другой пакет перед его передачей через Internet. Этот внешний пакет защищает содержимое от публичного просмотра и гарантирует, что пакет перемещается внутри виртуального туннеля.

Наложение пакетов называется инкапсуляцией. Компьютеры или другие сетевые устройства на обоих концах туннеля, называемые туннельными интерфейсами, могут инкапсулировать исходящие пакеты и повторно открывать входящие пакеты. Пользователи (на одном конце туннеля) и ИТ-персонал (на одном или обоих концах туннеля) настраивают туннельные интерфейсы, за которые они несут ответственность, для того, чтобы использовать протокол туннелирования. Протокол туннелирования, также называемый протоколом инкапсуляции, представляет собой стандартизированный способ инкапсуляции пакетов.

Целью протокола туннелирования является добавление уровня безопасности, который защищает каждый пакет на его пути через Internet. Пакет перемещается с тем же транспортным протоколом, который он использовал бы без туннеля; этот протокол определяет, как каждый компьютер отправляет и получает данные через своего интернет-провайдера. Каждый внутренний пакет по-прежнему поддерживает пассажирский протокол, такой как интернет-протокол (IP), который определяет, как он перемещается по локальным сетям на каждом конце туннеля. Протокол туннелирования, используемый для инкапсуляции, добавляет уровень безопасности для защиты пакета на его пути через Internet.

Рассмотрим **модуль сертификатов** в программном комплексе**.** При создании защищенного соединения с применением TLS/SSL, например, через HTTPS [100] (по умолчанию используется порт 443), осуществляется обмен сообщениями между клиентом, который всегда выступает инициатором соединения, и сервером. Первоначальный обмен сообщений называется протоколом рукопожатия (Handshake Protocol), после завершения которого обе

стороны переходят к протоколу записи (Record Protocol). В процессе протокола рукопожатия достигаются следующие цели:

1. Определяется, какой протокол будет использоваться из поддерживаемых вариантов. В зависимости от реализации это может быть: SSLv3, TLSv1, TLSv1.1, TLSv1.2. При этом всегда выбирается самый последний доступный вариант, то есть TLSv1 будет иметь приоритет перед SSLv3, если обе стороны поддерживают эти версии.

2. Отправляются аутентификационные данные. Обычно сервер отправляет аутентификационные данные в форме сертификата X.509 [101] (включенного в сертификат), однако протокол также допускает использование других методов.

3. Устанавливается идентификатор сессии, что позволяет при необходимости перезапустить сессию.

Также происходит согласование набора шифров, который включает алгоритм обмена ключами, тип алгоритма шифрования объемных данных и тип аутентификационного кода сообщения (MAC), которые будут использоваться в дальнейшем обмене данными (протокол записи). Обычно для обмена ключами применяется асимметричный алгоритм (с закрытым и открытым ключом), такой как RSA, DSA или ECC (шифр на основе эллиптических кривых, см. RFC 5289). Асимметричные алгоритмы требуют значительных ресурсов процессора. Из-за этого используются симметричные шифры для последующего шифрования объемных данных. Задача алгоритма обмена ключами – безопасная передача данных, достаточных для того, чтобы обе стороны могли независимо вычислить одинаковый сеансовый ключ для симметричного шифрования. Применение MAC обеспечивает целостность данных, которые передаются в процессе записи протокола.

Это упрощенная схема, и в процессе установления соединения может происходить обмен другими данными. Например, в случае взаимной аутентификации может потребоваться сертификат X.509 от клиента. Однако описанный процесс является наиболее распространенным случаем, который проиллюстрирован на рисунке 3.7 .

У каждого работающего через TLS/SSL сайта имеется X.509-сертификат, подтверждающий его identity, алгоритм которого показан на рисунке 30. Когда браузер устанавливает первое защищённое соединение с Web-сервером, они обмениваются информацией об используемых криптоалгоритмах, в рамках этого обмена Web-сервер отдаёт браузеру набор сертификатов, в котором есть обязательно сертификат собственно сайта, а остальные — промежуточные сертификаты для построения цепочки доверия.

Рисунок 3.7. Последовательность обмена сообщениями протоколов TLS/SSL

Браузер сначала определяет, какой из переданных сертификатов является сайтовым, после чего пытается построить цепочку доверия из остальных полученных сертификатов и одного из доверенных, которые доступны браузеру. В цепочке каждый сертификат, начиная с сайтового, подписан следующим перед ним сертификатом. Если итоговая цепочка завершается доверенным, браузер

начинает доверять первоначальному сертификату (сертификат сайта) и далее происходит безопасное подключение с его использованием.

Здесь важный момент: браузер получает от сайта *несколько* сертификатов, чтобы это произошло, владелец сервера при его конфигурации добавляет не только сертификат сайта, но и промежуточные сертификаты, которые он получил от удостоверяющего центра.

Концепт CSR. Подписание сертификата удостоверяющим центром.

Здесь и дальше слово *заявитель* / *applicant* для обозначения персоны или организации, которая хочет выпустить цифровой сертификат. Не используется для этого контекста слово *клиент*, так как оно слишком общее и обычно применяется в контексте типа *клиент/пользователь приложения*.

Под *личными данными заявителя* подразумевается его имя, название организации, адрес, город и прочую информацию, которую в англоязычной терминологии принято называть *identity*.

Чтобы удостоверяющий центр создал сертификат с открытым ключом и личными данными заявителя, ему нужно убедиться, как минимум в двух вещах:

1. Заявитель является именно тем, за кого себя выдаёт, то есть предоставленные им личные данные точно его или его организации, уполномоченным представителем которой он является;
2. У заявителя есть закрытый ключ. Он нужен для открытого, который он предоставил удостоверяющему центру для подписи вместе с личными данными.

Первый пункт — это процедура, которая проводится административными, а не криптографическими методами. Второй пункт отлично решается в автоматическом режиме криптографически, достаточно заявителю подписать собственным закрытым ключом блок с информационными данными (куда также записан собственно этот же открытый ключ).

В этом случае удостоверяющий центр берёт открытый ключ из информационного блока и верифицирует им цифровую подпись всего блока.

Вот этот вот информационный блок (состоящий из личных данных и открытого ключа) плюс цифровая подпись для него формируют запрос на подпись сертификата (certificate signing request, CSR). Для X.509 формат данных CSR определён в спецификации PKCS#10 [102], он достаточно простой и по сути представляет собой линейный список информационных полей.

В удостоверяющем центре после верификации CSR выделяют из всего блока личных данных нужные, дополняют их данными удостоверяющих центров (УЦ) [103], после чего получившийся новый блок подписывают закрытым ключом УЦ и получается X.509-сертификат. Все эти шаги показаны на рисунке 3.8 .

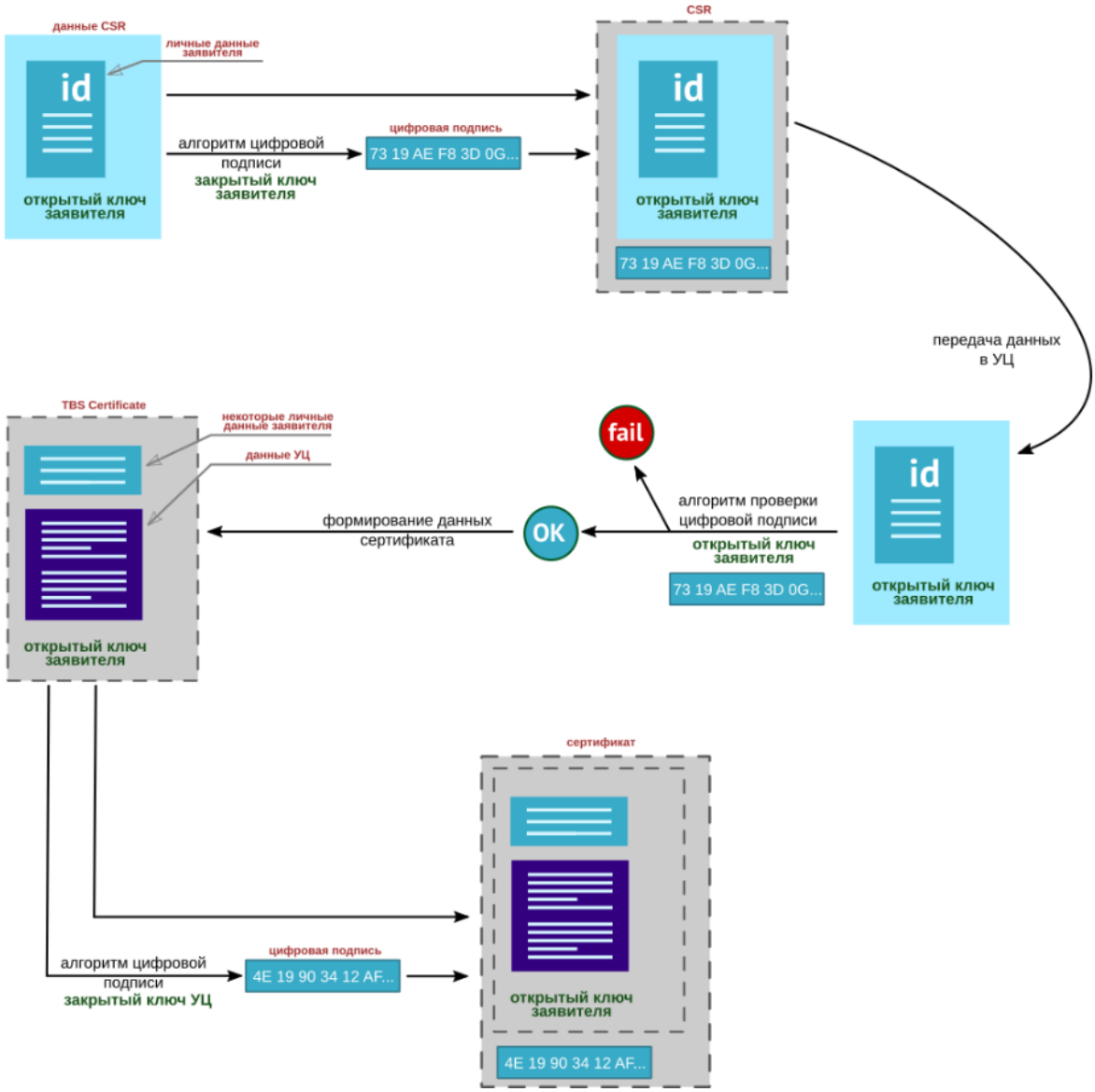

Рисунок 3.8. Концепт CSR

Описанная инфраструктура определяется как инфраструктура открытых ключей (PKI). Алгоритм работы программного комплекса описан в разделе 3.2 .

3.2 Алгоритм работы программного комплекса

Программный комплекс поддерживает многопользовательский режим работы и по запросам пользователей обеспечивает подключение пользовательских устройств к выбранным источникам данных. На первом шаге для заданного источника в модуле ETL - сервисов производится выбор соответствующего сервиса и типа базы данных. В качестве критерия используется тип данных, а процесс подбора основан на методе последовательного перебора из списка возможных решений. На втором шаге для известных значений типа базы данных и перечня поддерживаемых устройством сервисов в модуле Web-сервисов методом последовательного перебора производится выбор удовлетворяющего решения. В случае отсутствия подходящего решения производится возврат к первому шагу с уточнением ETL – сервиса и типа базы данных. После этого процесс совершается на втором шаге. Формируется блок характеристик ПОС. На третьем шаге производится выбор наилучшего сетевого решения – выбора комбинации каналов, обеспечивающих передачу данных за минимальное время. В процессе расчетов используются временные характеристики каналов. Комплекс готов к работе. После того, как проверены сертификаты безопасности он обеспечивает передачу данных на терминальные устройства пользователя. Для ограничения количества запросов к источникам данных использовался метод последовательного анализа Вальда, а для выбора наилучшей цепочки участков прохождения сигнала – алгоритм Беллмана—Форда. Сравнение методов динамического программирования показано в таблице 3.3 . После этого в режиме эксплуатации выполняется запрос и передача данных. На последнем шаге происходит оценка полученных данных и в случае необходимости добавление новых источников. Перед окончанием работы системы происходит сброс настройки характеристик ПОС к конкретному пользователю системы. Полное описание алгоритма работы программного комплекса изображено на рисунке 3.9 .

Для определения количества запросов к конкретному источнику используется метод Вальда. Он основан на максиминном критерии (получить максимальный выигрыш при наихудших условиях). Выбранные с его помощью варианты полностью устраняют риск. Это означает, что принимающий решение по данному варианту не столкнется с результатом хуже, чем тот, к которому он стремился.

На основании сложности метода и отсутствия эвристической информации был выбран метод Беллмана — Форда.

Основные преимущества данного метода при решении рассматриваемой задачи:

1)Получаем точное решение;

2)Может работать с отрицательными ребрами, поэтому отсутствие связи k′ и p′, обозначим через отрицательную скорость ETL-сервиса (c_{ij}^{kw}) и отрицательную скорость web-сервиса (q_{jl}^{pw});

3)Время выполнения алгоритма составляет O((I+J+L)*(K-k′+P-p′)).

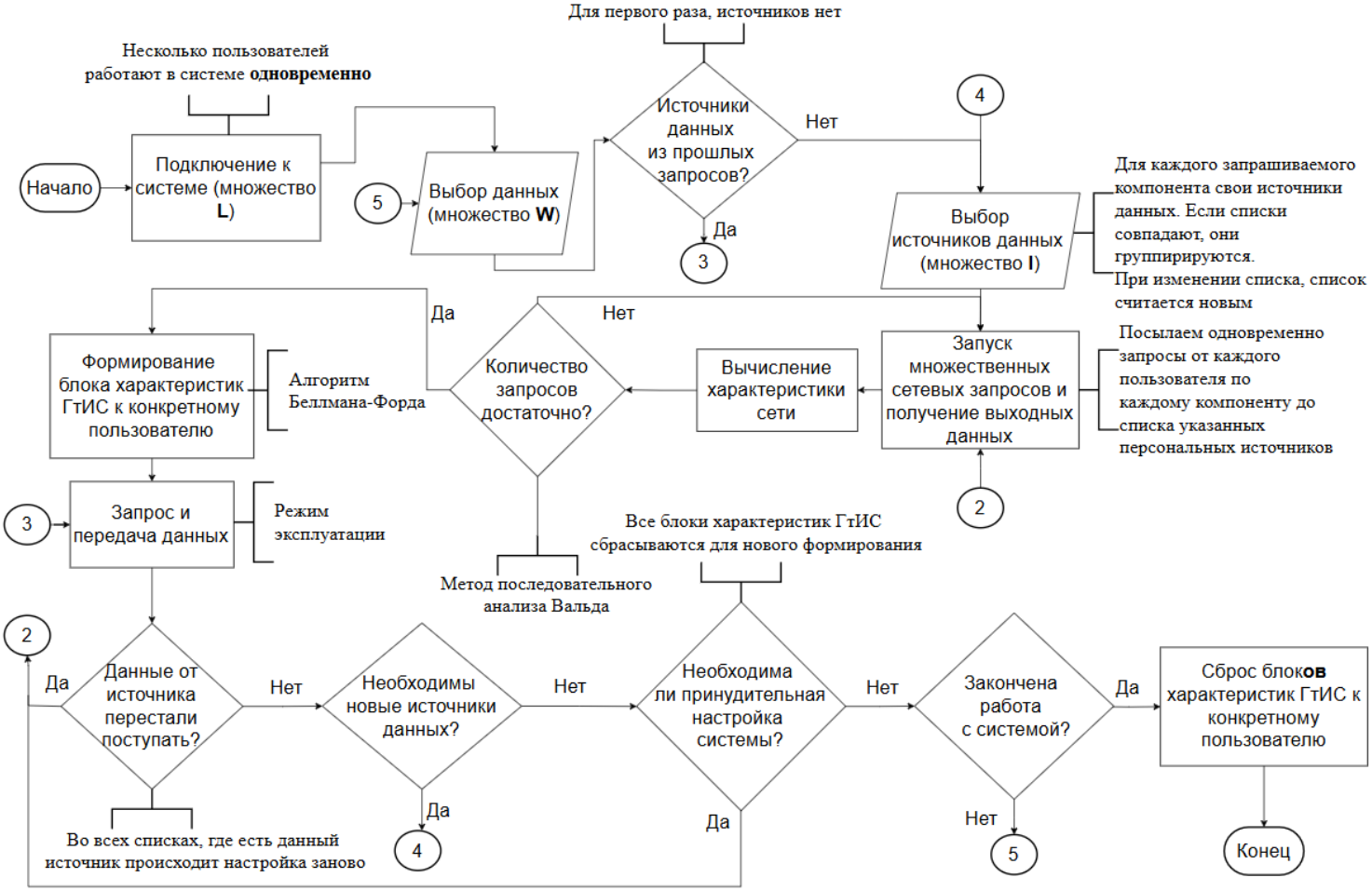

Рисунок 3.9. Алгоритм работы программного комплекса

Таблица 3.3. Сравнение методов динамического программирования

Критерии \ Метод	Левита	Джонсона	**Беллмана — Форда**	Флойда — Уоршелла	Алгоритм A*
Отрицательные ребра	+	+	+	+	+
Сложность	O(V2M), M- ребра	O(VE+VD), VD алгоритм Дейкстры	O(VE), E - обходы	O(V3)	O(E), необходима эвристическая информация

3.3 Алгоритм принятия решения о подлинности изделия

Алгоритм работы программного комплекса, описанный в разделе 3.2 является только частью работы алгоритма принятия решения о подлинности

изделия с помощью разработанного программного комплекса, который показан на рисунке 3.10 .

С использованием разработанного программного комплекса была проведена оценка эффективности предложенных методов и алгоритмов на примере характеристик отдельных блоков ближнемагистрального пассажирского самолёта типа Ил-114-300. Оценка проводилась для следующих составных частей: рулевой привод, пневматический насос, гироскоп, высотометр. При проведении расчетов, показанные в таблице 9, были использованы следующие программные компоненты: Talend Open Studio(ETL - сервис), Postgres, Mongo DB (хранилища данных), RestAPI, GraphQL (Web-сервисы), а для обеспечения безопасной передачи данных – VPN-сервер NGate от компании КриптоПро [104] и сертификаты МинЦифры РФ соответственно [105].

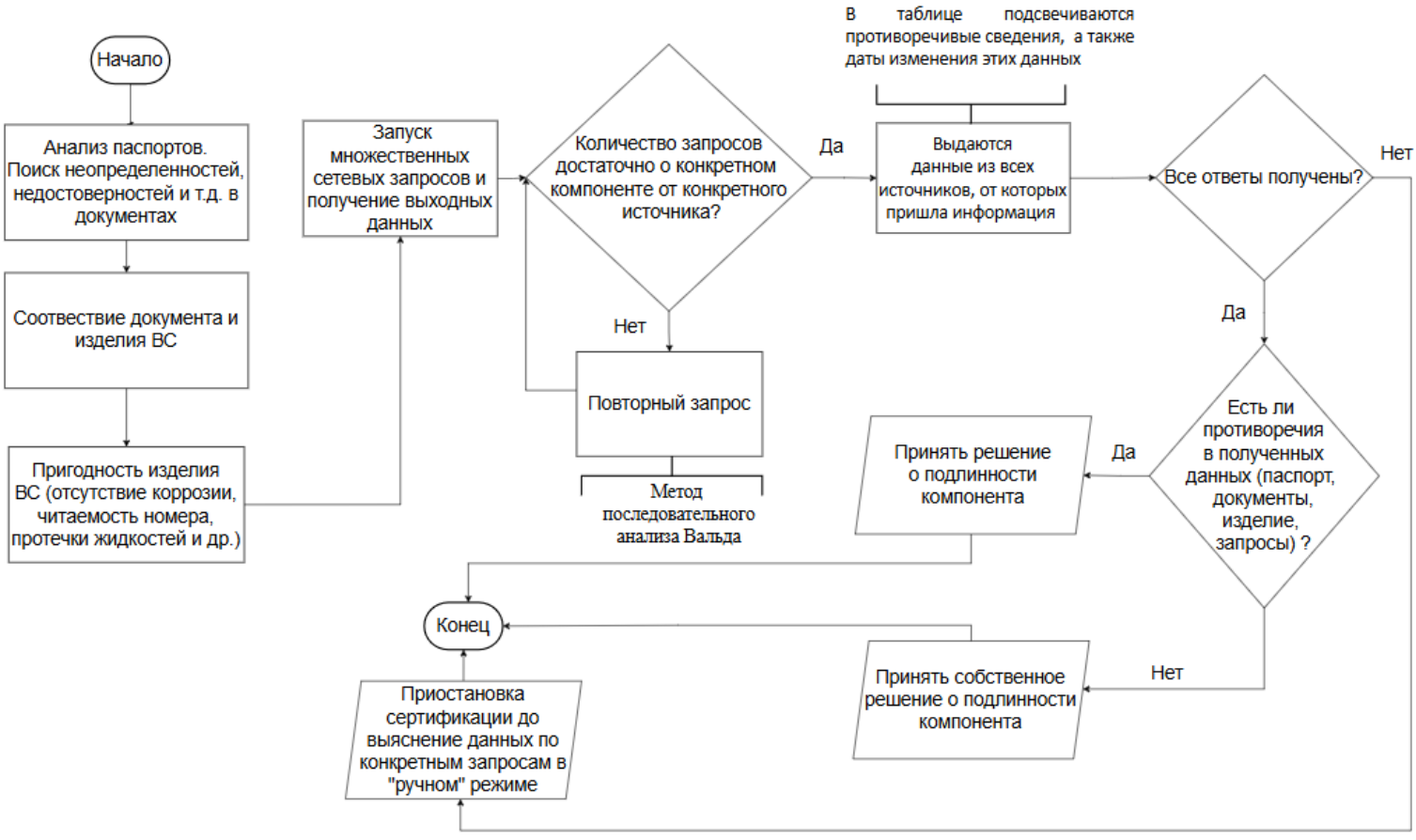

Рисунок 3.10. Схема алгоритма принятия решения о подлинности изделия

3.4 Результаты апробации программного комплекса на примере характеристик отдельных блоков ближнемагистрального пассажирского самолёта типа Ил-114-300

В результате апробации программного комплекса получены следующие результаты на примере характеристик отдельных блоков ближнемагистрального пассажирского самолёта типа Ил-114-300, который показан на рисунке 3.11:

- Проведен выбор наилучшей цепочки участков для прохождения сигнала за **минимальное время** (показано в таблице 3.4).
- Показана возможность повышения скорости отработки на **19-20 %** для одного запроса данных.
- Показана возможность сокращения времени сбора и анализа удостоверяющих документов одного ВС до **2,5** рабочих дней.
- Повышение достоверности результатов проверок благодаря ведению журнала записей.

Расчет возможности сокращения времени сбора и анализа удостоверяющих документов одного ВС:

Разница времени отработки одного запроса данных Х Количество компонентов Х Количество источников / С. в м / М. в ч. / Рабочий день, то есть 3 Х 6000 Х 4 / 60 / 60 / 8 = 2,5 .

Рисунок 3.11. Самолёт Ил-114-300

Схема реализованного программного комплекса показан на рисунке 3.12.

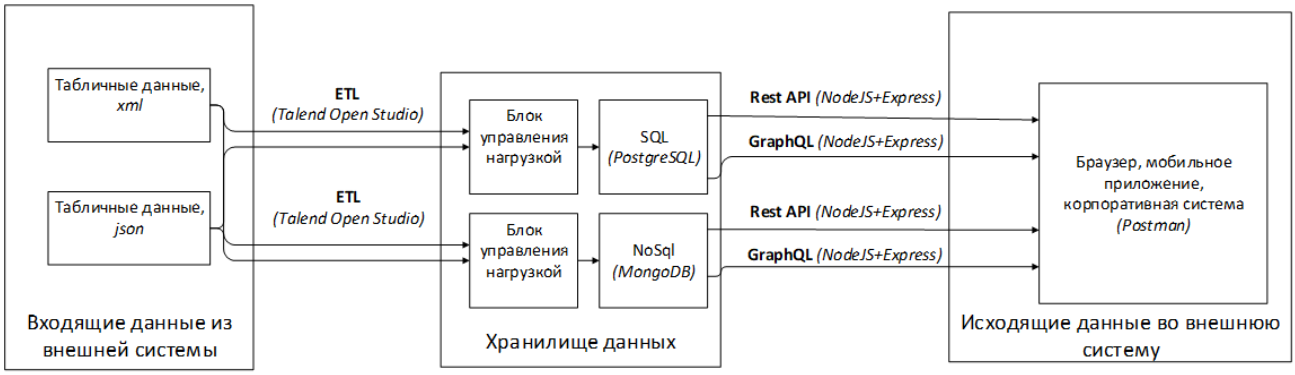

Рисунок 3.12. Схема структуры реализованного программного комплекса

Таблица 3.4. Примеры реализованных цепочек участков для прохождения сигнала

	Тип исходных данных	Тип базы данных	Тип сервиса
Пример 1	xml	Реляционная	Rest API (xml)
Пример 2	xml	Нереляционная	Rest API (xml)
Пример 3	json	Реляционная	Rest API (xml)
Пример 4	json	Нереляционная	Rest API (xml)
Пример 5	xml	Реляционная	Rest API (json)
Пример 6	xml	Нереляционная	Rest API (json)
Пример 7	json	Реляционная	Rest API (json)
Пример 8	json	Нереляционная	Rest API (json)
Пример 9	xml	Реляционная	GraphQL (json)
Пример 10	xml	Нереляционная	GraphQL (json)
Пример 11	json	Реляционная	GraphQL (json)
Пример 12	json	Нереляционная	GraphQL (json)

Таким образом: проведено экспериментальное апробирование программного комплекса, реализующего предложенные модель, метод и алгоритмы.

3.5 Выводы к главе

Выполненные в данной главе исследования и разработки позволяют сформулировать следующие выводы:

1) Разработан программный комплекс системы, обрабатывающий информацию в базе данных и реализующий предложенный метод и алгоритмы.

Для определения эффективности предложенных решений разработанный программный комплекс применен для синтеза ПОС на реальных данных по изделию авиационной техники.

2) Для применения программного комплекса были использованы паспортизированные компоненты (состоящие из паспортов и изделия) отдельных блоков ближнемагистрального пассажирского самолёта типа Ил-114-300. Исходные данные были структурированы в соответствии с предложенной в главе 2 информационной модели.

3) На основе разработанного программного комплекса, реализующего описанные в главе 2 алгоритм, была показана возможность повышения скорости отработки на 19-20 % для одного запроса данных. Так же повысилась достоверность результатов проверок благодаря ведению журнала записей.

4. ЗАКЛЮЧЕНИЕ

Проведенные исследования позволяют сформулировать следующие основные выводы и результаты работы:

1. Разработана модель описания проблемно-ориентированной системы, отражающая особенности ее функционирования при организации информационного взаимодействия разнородных источников/потребителей.

Отличием модели является описание в информационной системе характеристик баз данных, типов хранимых данных, а также технологий передачи данных в систему и из нее.

2. Разработан алгоритм оптимизации параметров проблемно-ориентированной системы, обеспечивающий организацию информационного взаимодействия разнородных источников и потребителей данных за минимальное время.

Алгоритм отличается параллельным поиском в пространстве сервисов цепочки участков прохождения сигнала через совместимые сервисы ETL и Web от источников к потребителю.

3. Разработан метод выбора характеристик проблемно-ориентированной системы на основе обработки информации о структуре и параметрах интегрируемых источников/потребителей данных.

Отличием метода является возможность для каждого типа устройства потребителей выбрать наилучшие варианты множества цепочек и сформировать блок характеристик проблемно-ориентированной системы.

4. Разработана информационная модель системы, позволяющая на основе обработки информации об источниках/потребителях данных производить структурно-параметрический синтез проблемно-ориентированной системы.

Предложенная модель отличается наличием дополнительных сущностей и атрибутов совместимости, необходимых для расчета оптимальной по критерию минимального времени цепочки участков для прохождения сигнала с учетом совместимости сервисов.

5. Разработан программный комплекс системы, позволяющий на основе обработки информации о структуре и параметрах интегрируемых источников/потребителей данных синтезировать проблемно-ориентированную систему.

Отличием программного комплекса является возможность работы в двух режимах: настройки и эксплуатации. В первом происходит формирование характеристик системы, а во втором выполнение запросов на основании источников и выбранных ранее характеристик.

6. Внедрение и экспериментально-промышленное апробирование результатов работ в ООО «НПЦ «БизнесАвтоматика» позволило провести детальный анализ интеграции на этапе их разработки и сократить время исследования, а также повысить уровень информативности руководителей проектов о прогнозируемой величине расходов на интеграцию как разрабатываемых систем, так уже и эксплуатируемых.

Таким образом, достигнута поставленная цель и решена важная задача повышения технико-экономической эффективности процессов интеграции разнородных информационных ресурсов для контроля подлинности паспортизированных компонентов ВС при эксплуатации авиационной техники за счет создания новой проблемно-ориентированной системы информационной поддержки решений. В качестве направления дальнейших исследований могут быть рассмотрены методы совершенствования программного комплекса на основе более полной информации о источниках и потребителях, а также предпочтительных сервисов интеграции, выбранных по умолчанию.

СПИСОК СОКРАЩЕНИЙ

AD Airworthiness Directives (Директива лётной годности)

API Application programming interface (интерфейс программирования приложения)

CSR Certificate signing request (Запрос на подпись сертификата)

DAG Directed Acyclic Graphs (направленные ациклические графы)

EAI Enterprise application integration (интеграция на уровне корпоративных приложений)

ESB Enterprise service bus (единая сервисная шина)

ETL Extract, Transform, Load

IPv4 Internet Protocol version 4 (интернет-протокол четвертной версии)

ISDN Integrated Services Digital Network (интегрированная цифровая сеть услуг)

GUI Graphical User Interface (Графический пользовательский интерфейс)

JSON JavaScript Object Notation

MAC Message authentication code (Код аутентификации послания)

MSM Messaging service model (Служба обмена сообщениями)

NUMA Non-uniform memory access (Неоднородный доступ к памяти)

OLAP Online analytical processing (Интерактивной аналитической обработки)

PKI Public key infrastructure (инфраструктура открытых ключей)

RESTful Representational State Transfer

SB Service Bulletin (Сервисные бюллетени)

SOA Service–Oriented architecture (Сервис-ориентированная архитектура)

SOAP Simple Object Access Protocol

VPN Virtual private network (Виртуальная частная сеть)

WAN Wide area network (Глобальная вычислительная сеть)

XML eXtensible Markup Language

АСУ ПЛГ ВС Автоматизированная система

поддержания лётной годности воздушных судов

БД Базы данных

ВС Воздушное судно

ЕИП Единое информационное пространство

ЖЦ Жизненный цикл

ИС Информационная система

ИТ Информационные технологии

ЛГ Летная годность

ОП Оперативная память

ОКМД Одиночные и множественные потоки команд

ПО Программное обеспечение

ПОС Проблемно-ориентированной системы

СБ Сертификат безопасности

СЛГ Сертификат летной годности

СУБД Системы управления базами данных

СЧ Составные части

УЦ Удостоверяющие центр

ЦЛП Целочисленное линейное программирование

ЦПУ Центральное процессорное устройство

СПИСОК ЛИТЕРАТУРЫ

1. Буряк Ю.И., Никонов Ю.Ю. Совершенствование процессов обеспечения летной годности воздушных судов за счет создания высокоскоростной гетерогенной информационной системы. Вестник компьютерных и информационных технологий. 2024. Т. 21, № 6. С. 31 – 40.
2. Брусникин В. Ю., Глухов Г.Е., Черников П.Е. Жизненный цикл авиационной техники на этапе эксплуатации в информационно-аналитической системе мониторинга летной годности воздушных судов // Научный вестник ГосНИИ ГА. 2016. № 15. С. 33-39.
3. ГОСТ Р 57907-2017 «Воздушный транспорт. Техника авиационная гражданская. Ремонт по техническому состоянию. Общие требования».
4. Приказ Минтранса РФ №519 от 27.11.2020 Об утверждении Федеральных авиационных правил.
5. 62 % of IT leaders say legacy systems are biggest roadblock to multi-cloud success // URL: https://www.techrepublic.com/ article/62-of-it-leaders-say-legacy-systems-are-biggest-roadblock-to-multi-cloud-success/ (дата обращения 15.03.2022)/
6. Gartner Glossary. Legacy Application Or System, URL: https://www.gartner.com/en/information-technology/glossary/legacyapplication-or-system (дата обращения 21.03.2022).
7. Mallidi, Ravi Kiran & Sharma, Manmohan & Singh, Jagjit. (2021). Legacy Digital Transformation: TCO and ROI Analysis. International journal of electrical and computer engineering systems. 12. 163-170. 10.32985/ijeces.12.3.5.
8. GAO, Information Technology: Agencies Need to Develop and Implement Modernization Plans for Critical Legacy Systems, 2021, https://www.gao.gov/products/gao-21-524t (дата обращения 24.03.2022).
9. Jha, S.; Jha, M.; O'Brien, L.; Wells, M. Supporting Decision Making with Big Data Integrating Legacy Systems and Data. In Proceedings of the 2017 4th Asia-Pacific World Congress on Computer Science and Engineering, APWC, Mana Island, Fiji, 11–13 December 2017; pp. 120–128.

10. Abu Bakar, H.; Razali, R.; Jambari, D.I. Legacy Systems Modernisation for Citizen-Centric Digital Government: A Conceptual
Model. Sustainability 2021, 13, 13112. https://doi.org/10.3390/su132313112.
11. Seetharamatantry, H.; Murulidhar, N.; Chandrasekaran, K. Implications of Legacy Software System Modernization—A Survey In A Changed Scenario. Int. J. Adv. Res. Comput. Sci. 2017, 8, 1002–1008.
12. D. Beach, Legacy systems push up costs of digital transformation, 2019, https://www.theglobaltreasurer.com/
2018/09/27/legacy- systems-push- up- costs- of- digital- transformation/.
13. Crotty, J.; Horrocks, I. Managing Legacy System Costs: A Case Study of a Meta-Assessment Model to Identify Solutions in a Large
Financial Services Company. Appl. Comput. Inform. 2017, 13, 175–183.
14. Srinivas, M.; Ramakrishna, G.; Rajasekhara Rao, K.; Suresh Babu, E. Analysis of Legacy System in Software Application
Development: A Comparative Survey. Int. J. Electr. Comput. Eng. 2016, 6, 292–297.
15. Assunção, Wesley. (2021). Contemporary Software Modernization: Perspectives and Challenges to Deal with Legacy Systems. 10.13140/RG.2.2.25176.42243.
16. D. Wolfart, W. K. G. Assunc, ˜ao, I. F. da Silva, D. C. P. Domingos, E. Schmeing, G. L. D. Villaca, and D. d. N. Paza, “Modernizing legacy
systems with microservices: A roadmap,” in 25th Evaluation and Assessment in Software Engineering (EASE). ACM, 2021, p. 149–159.
17. P. L. Leon and F. E. A. Horita, “On the modernization of systems for supporting digital transformation: A research agenda,” in XVII Brazilian Symposium on Information Systems, 2021, pp. 1–8.
18. Зрячев С. А., Ларин С. Н. Разработка базы знаний послепродажного обслуживания авиационной техники // Известия Самарского научного центра Российской академии наук. 2020. Т. 22, № 5. С. 48–53.
19. «Яндекс»: Переход с Oracle на открытое ПО, URL: https://www.tadviser.ru/index.php/%D0%9F%D1%80%D0%BE%D0%B5%D0

%BA%D1%82:%D0%AF%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_(%D0%BC%D0%B8%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F_%D1%81_Oracle_%D0%BD%D0%B0_PostgreSQL) (дата обращения 17.03.2022).

20. Интерфакс, Создатели вируса WannaCry получили в качестве выкупа $42 тысячи, URL: https://www.interfax.ru/world/562284, (дата обращения 24.03.2022).
21. U.S. Government Accountability Office (GAO), Information Technology: Agencies Need to Develop Modernization Plans for Critical Legacy Systems, URL: https://www.gao.gov/products/gao-19-471#summary (дата обращения 17.03.2022).
22. Legacy System Modernization: When and How to Do It Right, URL: https://euristiq.com/legacy-system-modernization-whenand-how-to-do-it-right/, (дата обращения 24.02.2022).
23. Abdellatif, Manel & Shatnawi, Anas & Mili, Hafedh & Moha, Naouel & El-Boussaidi, Ghizlane & Hecht, Geoffrey & Privat, Jean & Guéhéneuc, Yann-Gaël. (2020). A taxonomy of service identification approaches for legacy software systems modernization. Journal of Systems and Software. 10.1016/j.jss.2020.110868.
24. Фаулер М. Рефакторинг. Улучшение существующего кода. С-пб., 2009.
25. Басов А.С. Методы рефакторинга кода // Вестник науки. 2020. №8 (29). URL: https://cyberleninka.ru/article/n/metodyrefaktoringa-koda (дата обращения: 21.03.2022).
26. Ксензов М. Рефакторинг архитектуры программного обеспечения: выделение слоев // Труды ИСП РАН. 2004. №1. URL: https://cyberleninka.ru/article/n/refaktoring-arhitektury-programmnogo-obespecheniya-vydelenie-sloev (дата обращения: 21.03.2022).
27. IDC Top 10 Predictions For Worldwide IT, 2019 URL: https://www.forbes.com/sites/louiscolumbus/2018/11/04/idc-top10-predictions-for-worldwide-it-2019/?sh=82f41db7b962, (дата обращения: 21.03.2022).

28. Brent Frye 8 Steps for Migrating Existing Applications to Microservices, URL: https://insights.sei.cmu.edu/blog/8-steps-formigrating-existing-applications-to-microservices/, (дата обращения: 22.03.2022).
29. Радостев Д.К., Никитина Е.Ю. Стратегия миграции программного кода из монолитной архитектуры в микросервисы // Вестник Пермского университета. Серия: Математика. Механика. Информатика. 2021. №2 (53). URL: https://cyberleninka.ru/article/n/strategiya-migratsii-programmnogo-koda-iz-monolitnoy-arhitektury-v-mikroservisy (дата обращения: 31.03.2022).
30. Балес А.И. Унифицированная модель данных и её применение в микросервисной архитектуре / DOI 10.25559/SITITO.16.202002.416-425 // Современные информационные технологии и ИТ-образование. – 2020. –Т. 16, № 2. – С. 416-425.
31. Слинкин Д.А. Современные подходы к модернизации веб-ресурсов образовательнойорганизации // Вестник Шадринского государственного педагогического университета. 2019. №2 (42). URL: https://cyberleninka.ru/article/n/ sovremennye-podhody-k-modernizatsii-veb-resursov-obrazovatelnoi-organizatsii (дата обращения: 22.03.2022).
32. Flavian, Carlos & Gurrea, Raquel & Orús, Carlos. (2009). Web design: A key factor for the website success. J. Systems and IT. 11. 168-184. 10.1108/13287260910955129.
33. Website Redesign Strategy – How to Redesign a Website in 2022, URL: https://uxhacks.com/website-redesign/, (дата обращения: 21.03.2022).
34. Морозова О.А. Интеграция корпоративных информационных систем: М80 учебное пособие. — М.: Финансовый университет, 2014. — 140 с
35. Никонов Ю. Ю. Исследование интеграции государственных автоматизированных информационных систем с помощью GraphQL. «Гагаринские чтения – 2020»: Сборник тезисов докладов. — М.: МАИ, 2020. — 1731 с.
36. GraphQL [Электронный ресурс]. Режим доступа: https://spec.graphql.org/October2021// (дата обращения: 23.04.2025).

37. Гаранин С.А., А. Ю. Коньков, В. Ю. Брусникин, А. Н. Шарыпов, С. В. Коваль. Системы искусственного интеллекта в рамках современных задач гражданской авиации // Научный вестник ГосНИИ ГА. 2022. № 38. С. 84-91.
38. Воронина Н.В. Анализ существующих подходов в реализации API веб-сервиса // Сб. ст. по мат. XXII Международной научно-практической конференции (Россия, Москва, 1-2 февраля, 2018). Москва. Изд. «Проблемы науки», 2018. С. 21-24.
39. Bc. Dominik Hanák. GraphQL as modern access to jBPM process engine / Master’s Thesis, Masaryk University, Brno, Czech Republic, 2019. 80 p.
40. Doc 9760 Руководство по летной годности [Электронный ресурс]. Режим доступа: https://аэростандарт.рф/ru/icao/book/документ-9760-руководство-по-летной-годности-из-0-ру-12193 (дата обращения: 05.06.2025).
41. Гаранин С.А., Глухов Г.Е., Брусникин В.Ю., Черников П.Е., Карапетян А.Г., Коваль С.В. Об алгоритмах обработки информации при мониторинге жизненного цикла и оценке аутентичности компонентов воздушного судна // Научный вестник ГосНИИ ГА. 2020. № 31. С. 30-40.
42. Рекомендательный циркуляр № РЦ-АП-145.А.42d о создании условий по обеспечению гарантии подлинности составных частей воздушного судна и его компонентов, используемых при техническом обслуживании и ремонте Введен в действие с 01.12.2024 .
43. Приказа Минтранса России от 27.11.2020 №519 Об утверждении федеральных авиационных правил «требования к летной годности гражданских воздушных судов. форма и порядок оформления сертификата летной годности гражданского воздушного судна. Порядок приостановления действия и аннулирования сертификата летной годности гражданского воздушного судна»
44. Автоматизированная система поддержания лётной годности воздушных судов [Электронный ресурс]: URL: https://www.mlgvs.ru/asuplgvs.html (дата обращения: 16.04.2025) .

45. Автоматизированная система поддержания лётной годности воздушных и их компонентов (АСУ ПЛГ) судов [Электронный ресурс]: URL: https://www.mlgvs.ru/images/slides/asu0.png (дата обращения: 16.04.2025) .
46. «Эрлан-3» документация [Электронный ресурс]: URL: https://airlan.ru/products1/Airlan-3-dokumentatsiya.php (дата обращения: 16.04.2025) .
47. Авдеева Е. С., Панюшкина Л. В., Денисов Д. Д. Развитие ремонта и послепродажного обслуживания отечественной авиационной техники на мировом рынке // Вестник Института экономических исследований. 2017. № 2 (6). С. 25–33.
48. Aviation MRO Software (AMOS & RAMCO) [Электронный ресурс]: URL: https://www.aviationhunt.com/aviation-mro-software/ (дата обращения: 16.04.2025) .
49. Software Spotlights At MRO Americas [Электронный ресурс]: URL: https://aviationweek.com/sites/default/files/styles/crop_freeform/public/Swiss%20AviationSoftware-00_Main_Screen_busy_2.jpg?itok=UBhA_L2s (дата обращения: 16.04.2025) .
50. New-generation technical data management with airnavX [Электронный ресурс]: URL: https://www.airbus.com/en/newsroom/news/2018-02-new-generation-technical-data-management-with-airnavx (дата обращения: 16.04.2025).
51. Быкова В.В., Глухов Г.Е., Шарыпов А.Н., Карапетян А. Г., Ладыгина Н.Н., Коваль С.В. Проблемы уязвимости информационных систем предприятий авиационной отрасли: ошибки, связанные с человеческим фактором // Научный вестник ГосНИИ ГА. 2020. № 32. С. 88-99.
52. Никонов Ю. Ю. Разработка технологии интеграции гетерогенных информационных систем при помощи веб-сервисов. «Гагаринские чтения – 2022»: Сборник тезисов докладов. — М.: МАИ, 2022. — 835 с.

53. Barthels C., Müller I., Schneider T., Alonso G., Hoefler T. Distributed Join Algorithms on Thousands of Cores. Proceedings of the VLDB Endowment. 2017; 10(5):517-528. (In Eng.) doi: https://doi.org/10.14778/3055540.3055545
54. Graefe G. Encapsulation of Parallelism in the Volcano Query Processing System. Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data (Atlantic City, New Jersey, USA) (SIGMOD'90). Association for Computing Machinery, New York, NY, USA; 199. p. 102-111. (In Eng.) doi: https://doi.org/10.1145/93597.98720
55. Shaikhha A., Klonatos Ya., Parreaux L., Brown L., Dashti M., Koch C. How to Architect a Query Compiler. Proceedings of the 2016 International Conference on Management of Data (San Francisco, California, USA) (SIGMOD'16). Association for Computing Machinery, New York, NY, USA; 2016. p. 1907-1922. (In Eng.) doi: https://doi.org/10.1145/2882903.2915244
56. DeWitt D.J., Katz R.H., Olken F., Shapiro L.D., Stonebraker M.R., Wood D.A. Implementation Techniques for Main Memory Database Systems. ACM SIGMOD Record. 1984; 14(2):1-8. (In Eng.) doi: https://doi.org/10.1145/971697.602261
57. Stonebraker M., Cetintemel U. "One Size Fits All": An Idea Whose Time Has Come and Gone. Proceedings of the 21st International Conference on Data Engineering (ICDE'05). IEEE Computer Society, USA; 2005. p. 2-11. (In Eng.) doi: https://doi.org/10.1109/ ICDE.2005.1
58. Ailamaki A., DeWitt D.J., Hill M.D., Wood D.A. DBMSs on a Modern Processor: Where Does Time Go? Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA; 1999. p. 266-277. Available at: https://www.vldb.org/conf/1999/P28.pdf (accessed 13.10.2021). (In Eng.)
59. Kester M.S., Athanassoulis M., Idreos S. Access Path Selection in Main-Memory Optimized Data Systems: Should I Scan or Should I Probe? Proceedings of the 2017 ACM International Conference on Management of Data (Chicago, Illinois,

USA) (SIGMOD’17). Association for Computing Machinery, New York, NY, USA; 2017. p. 715-730. (In Eng.) doi: https://doi.org/10.1145/3035918.3064049

60. Neumann T. Efficiently Compiling Efficient Query Plans for Modern Hardware. Proceedings of the VLDB Endowment. 2011; 4(9):539-550. (In Eng.) doi: https://doi.org/10.14778/2002938.2002940
61. Kersten T., Leis V., Kemper A., Neumann T., Pavlo A., Boncz P. Everything You Always Wanted to Know about Compiled and Vectorized Queries but Were Afraid to Ask. Proceedings of the VLDB Endowment. 2018; 11(13):2209-2222. (In Eng.) doi: https://doi.org/10.14778/3275366.3284966
62. Lottarini A., Ramirez A., Coburn J., Kim M.A., Ranganathan P., Stodolsky D., Wachsler M. Vbench: Benchmarking Video Transcoding in the Cloud. ACM SIGPLAN Notices. 2018; 53(2):797-809. (In Eng.) doi: https://doi.org/10.1145/3296957.3173207
63. Kara K., Giceva J., Alonso G. FPGA-Based Data Partitioning. Proceedings of the 2017 ACM International Conference on Management of Data (Chicago, Illinois, USA) (SIGMOD’17). Association for Computing Machinery, New York, NY, USA; 2017. p. 433-445. (In Eng.) doi: https://doi.org/10.1145/3035918.3035946
64. Karnagel T., Habich D., Schlegel B., Lehner W. Heterogeneity-Aware Operator Placement in Column-Store DBMS. Datenbank-Spektrum. 2014; 14(3):211-221. (In Eng.) doi: https://doi.org/10.1007/s13222-014-0167-9
65. Müller M., Leich T., Pionteck T., Saake G., Teubner J., Spinczyk O. He..ro DB: A Concept for Parallel Data Processing on Heterogeneous Hardware. In: Brinkmann A., Karl W., Lankes S., Tomforde S., Pionteck T., Trinitis C. (eds.) Architecture of Computing Systems – ARCS 2020. ARCS 2020. Lecture Notes in Computer Science. Vol. 12155. Springer, Cham; 2020. p. 82096. (In Eng.) doi: https://doi. org/10.1007/978-3-030-52794-5_7
66. Zhang Ya., Zhang Yu, Lu J., Wang Sh., Liu Z., Han R. One size does not fit all: accelerating OLAP workloads with GPUs. Distributed and Parallel Databases. 2020; 38(4):995-1037. (In Eng.) doi: https://doi.org/10.1007/s10619-020-07304-z

67. Barroso L., Marty M., Patterson D., Ranganathan P. Attack of the Killer Microseconds. Communications of the ACM. 2017; 60(4):48-54. (In Eng.) doi: https://doi.org/10.1145/3015146
68. Chernishev G.A., Galaktionov V.A., Grigorev V.D., Klyuchikov E.S., Smirnov K.K. PosDB: An Architecture Overview. Programming and Computer Software. 2018; 44(1):62-74. (In Eng.) doi: https://doi.org/10.1134/S0361768818010024
69. Funke H., Breß S., Noll S., Markl V., Teubner J. Pipelined Query Processing in Coprocessor Environments. Proceedings of the 2018 International Conference on Management of Data (Houston, TX, USA) (SIGMOD'18). Association for Computing Machinery, New York, NY, USA; 2018. p. 1603-1618. (In Eng.) doi: https://doi.org/10.1145/3183713.3183734
70. Răducanu B., Boncz P., Zukowski M. Micro Adaptivity in Vectorwise. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (New York, New York, USA) (SIGMOD'13). Association for Computing Machinery, New York, NY, USA; 2013. p. 1231-1242. (In Eng.) doi: https://doi.org/10.1145/2463676.2465292
71. Leis V., Boncz P., Kemper A., Neumann T. Morsel-Driven Parallelism: A NUMA-Aware Query Evaluation Framework for the Many-Core Age. Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (Snowbird, Utah, USA) (SIGMOD'14). Association for Computing Machinery, New York, NY, USA; 2014. p. 743-754. (In Eng.) doi: https://doi.org/10.1145/2588555.2610507
72. Diaconu C., Freedman C., Ismert E., Larson P.-A., Mittal P., Stonecipher R., Verma N., Zwilling M. Hekaton: SQL Server's Memory-Optimized OLTP Engine. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (New York, New York, USA) (SIGMOD'13). Association for Computing Machinery, New York, NY, USA; 2013. p. 1243-1254. (In Eng.) doi: https://doi.org/10.1145/2463676.2463710
73. Chen J., Jindel S., Walzer R., Sen R., Jimsheleishvilli N., Andrews M. The MemSQL Query Optimizer: A Modern Optimizer for Real-Time Analytics in a

Distributed Database. Proceedings of the VLDB Endowment. 2016; 9(13):1401-1412. (In Eng.) doi: https://doi.org/10.14778/3007263.3007277

74. Boncz P., Zukowski M., Nes N. MonetDB/X100: Hyper-Pipelining Query Execution. Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR'05). Asilomar, CA, USA; 2005. Available at: http://cidrdb.org/cidr2005/papers/P19.pdf (accessed 13.10.2021). (In Eng.)
75. Psaroudakis I., Scheuer T., May N., Sellami A., Ailamaki A. Scaling up Concurrent Main-Memory Column-Store Scans: Towards Adaptive NUMA-Aware Data and Task Placement. Proceedings of the VLDB Endowment. 2015; 8(12):1442-1453. (In Eng.) doi: https://doi.org/10.14778/2824032.2824043
76. ГОСТ Р 58546-2019 Интеграция систем управления предприятием. Часть 6. Модель службы обмена сообщениями. Москва. Стандартинформ. 2019.
77. Никонов Ю. Ю. Исследование интеграции автоматизированных информационных систем используя технологию ETL. 20-я Международная конференция «Авиация и космонавтика». Москва. Тезисы. – М.: Издательство «Перо», 2021 – 730 с.
78. Gferrin / bellman-ford [Электронный ресурс]: URL: https://github.com/gferrin/bellman-ford (дата обращения: 21.04.2025).
79. Никонов Ю.Ю. Исследование применения технологии ETL в интеграции информационных систем. Научно-технический вестник Поволжья. 2023. № 10. С. 150-152.
80. Лебедев С. В. Модель-ориентированный подход к построению связанных данных на основе разнородных источников. // Онтология проектирования. 2019. Т. 9. №1(31). 104 с
81. Talend Open Studio Cookbook [Электронный ресурс]. – Режим доступа: https://www.programmer-books.com/wp-content/uploads/2019/08/Talend-Open-Studio-Cookbook.pdf (дата обращения 23.12.2022).
82. Базы данных: проектирование, реализация и сопровождение. Теория и практика / Томас Коннолли, Каролин Бегг ; [перевод с английского Р. Г.

Имамутдиновой, К. А. Птицына]. - 3-е изд. - Москва [и др.] : Вильямс, 2018. - 1439 с

83. PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/ (дата обращения 22.04.2025).
84. Representational State Transfer (REST) [Электронный ресурс]. – Режим доступа: https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (дата обращения 22.04.2025).
85. Java [Электронный ресурс]. – Режим доступа: https://www.java.com/ru/ (дата обращения 23.04.2025).
86. Никонов Ю.Ю. Сравнение импорта данных из различных типов файлов в реляционные и нереляционные базы данных. Научно-технический вестник Поволжья. 2023. № 1. С. 81-84
87. Никонов Ю.Ю., Столярчук В.А. Исследование применения технологий ESB и GraphQL в интеграции государственных автоматизированных информационных систем. Автоматизация. Современные технологии. 2022. Т. 76. № 8. С. 375-378.
88. Extensible Markup Language (XML) 1.0 (Fifth Edition) [Электронный ресурс]. – Режим доступа https://www.w3.org/TR/xml/ (дата обращения 23.04.2025) .
89. The JavaScript Object Notation (JSON) Data Interchange Format) [Электронный ресурс]. – Режим доступа https://datatracker.ietf.org/doc/html/rfc8259 (дата обращения 23.04.2025)
90. Никонов Ю. Ю. Анализ видов баз данных: реляционная, NoSQL. Цифровая трансформация социальных и экономических систем: материалы международной научно-практической конференции / отв. ред. И.А. Королькова; Моск. ун-т им. С.Ю. Витте. – Москва: изд. ЧОУВО «МУ им. С.Ю. Витте», 2022 – 937 с.
91. Stack Overflow Trends// Stack Overflow [Электронный ресурс]. – Режим доступа Stack Overflow Trends// Stack Overflow [Электронный ресурс]. –

Режим доступа https://insights.stackoverflow.com/trends (дата обращения 24.11.2021).

92. Никонов Ю. Ю. Анализ и сравнение форматов данных xml, json. Программно-техническое обеспечение автоматизированных систем: материалы Всероссийской молодежной научно-практической конференции (9 декабря 2021., г. Барнаул) / Алтайский государственный технический университет им. И. И. Ползунова ; под ред. А. Г. Якунина. – Барнаул : АлтГТУ, 2021. 94 с.
93. Савоськин И. В., Фирсов А. О. Исследование способов применения NoSQL и реляционных баз данных // E-Scio. 2019. № 6 (33). С. 101-108.
94. ISO/IEC 9075:2016 Information technology – Database languages – SQL
95. Никонов Ю. Ю. Интеграция гетерогенных информационных систем с помощью ETL-сервисов и веб-сервисов. 22-я Международная конференция «Авиация и космонавтика». 20-24 ноября 2023 года. Москва. Тезисы. — 413с.
96. JSON Functions and Operators [Электронный ресурс].– Режим доступа: https://www.postgresql.org/docs/9.5/functions-json.html, свободный. – (дата обращения: 14.08.2021)
97. Новиков Б.А., Левин М.Ю. Сравнительный анализ производительности SQL и NoSQL СУБД // Компьютерные инструменты в образовании. 2017. № 4. С. 48-63.
98. RFC 791 [Электронный ресурс].– Режим доступа: https://datatracker.ietf.org/doc/html/rfc791 (дата обращения: 24.04.2025)
99. RFC 3809 [Электронный ресурс].– Режим доступа: https://datatracker.ietf.org/doc/html/rfc3809 (дата обращения: 24.04.2025)
100. RFC 2818 [Электронный ресурс].– Режим доступа: https://datatracker.ietf.org/doc/html/rfc2818 (дата обращения: 24.04.2025).
101. RFC 5280 [Электронный ресурс].– Режим доступа: https://datatracker.ietf.org/doc/html/rfc5280 (дата обращения: 24.04.2025)

102. RFC 2986 [Электронный ресурс].– Режим доступа: https://www.ietf.org/rfc/rfc2986.txt (дата обращения: 15.06.2025)
103. What is a certificate authority (CA) [Электронный ресурс].– Режим доступа: https://www.techtarget.com/searchsecurity/definition/certificate-authority (дата обращения: 24.04.2025)
104. VPN-сервер NGate [Электронный ресурс].– Режим доступа: https://www.cryptopro.ru/products/ngate/vpn (дата обращения: 24.04.2025)
105. Поддержка работы сайтов с российскими сертификатами [Электронный ресурс].– Режим доступа: https://www.gosuslugi.ru/crt (дата обращения: 24.04.2025)