Архитектура программного прототипа системы радиационно-химического мониторинга жидких радиоактивных сред

А.П. Полянский, М.Г. Жабицкий, В.А.Сычев

***Аннотация*—В работе представлена реализация и экспериментальная проверка программного прототипа автоматизированной системы радиационно-химического мониторинга жидких радиоактивных сред, основанной на событийной модели передачи данных. Предложенная архитектура включает асинхронный транспортный контур на основе Apache Kafka, локальную буферизацию измерительных событий, раздельное хранение первичных и обработанных данных в хранилищах RAW и PROC, а также механизмы контроля целостности и прослеживаемости результатов обработки. Реализованный прототип обеспечивает сохранение первичных измерений в неизменном виде, возможность повторного выполнения обработки и независимость измерительного контура от аналитических подсистем. Для проверки архитектурных свойств разработана методика экспериментального тестирования, включающая модульные, функциональные и интеграционные испытания. В ходе экспериментальной проверки было сформировано и обработано 108000 измерительных событий, выполнены сценарии отказов транспортного контура, проверки контроля целостности данных, защиты хранилища RAW, прослеживаемости результатов обработки и воспроизводимости вычислений. Результаты испытаний подтвердили выполнение требований к сохранности данных, контролю целостности, неизменяемости первичной информации, прослеживаемости результатов обработки, воспроизводимости вычислений и изоляции измерительного контура. Полученные результаты подтверждают практическую реализуемость предложенного подхода для построения автоматизированных систем радиационно-химического мониторинга объектов использования атомной энергии.**

***Ключевые слова*— радиационно-химический мониторинг, событийная модель данных, Apache Kafka, RAW/PROC, прослеживаемость данных, воспроизводимость обработки, контроль целостности, автоматизированные системы мониторинга.**

I. Введение

Развитие цифровых технологий в промышленности и атомной энергетике сопровождается переходом от периодического контроля к непрерывному мониторингу технологических процессов.

Статья получена 27 июня 2026
А.П.Полянский – магистрант Высшей инжиниринговой школы НИЯУ МИФИ, (e-mail: pol.andrey2019@yandex.ru)
М.Г.Жабицкий – заместитель директора Высшей инжиниринговой школы НИЯУ МИФИ, (e-mail: jabitsky@mail.ru)
В.А.Сычев – ст. преподаватель Высшей инжиниринговой школы НИЯУ МИФИ, к.т.н. (e-mail: Ssytchov@mail.ru)

Это повышает требования к информационным системам, обеспечивающим сбор, передачу, хранение и обработку измерительной информации. Особую актуальность данные задачи приобретают при обращении с жидкими радиоактивными отходами и другими жидкими радиоактивными средами объектов использования атомной энергии, где радиационно-химические параметры используются для контроля технологических процессов, оценки эффективности очистки и обеспечения радиационной безопасности. В этих условиях особое значение приобретают сохранность данных, контроль их целостности, прослеживаемость происхождения результатов и возможность последующего анализа накопленной информации.

В предыдущей работе [1] была предложена архитектура автоматизированной системы радиационно-химического мониторинга, основанная на событийной модели данных, асинхронной передаче сообщений и разделении контуров хранения первичной и обработанной информации, обеспечивающих воспроизводимость обработки и прослеживаемость данных.

Однако вопросы программной реализации данной архитектуры и экспериментальной проверки её свойств ранее не рассматривались. Целью настоящей работы является разработка и экспериментальная проверка программного прототипа автоматизированной системы радиационно-химического мониторинга, направленная на подтверждение работоспособности событийной модели передачи данных, механизмов обеспечения сохранности информации, контроля целостности и воспроизводимости обработки.

Рабочая гипотеза исследования состоит в том, что событийно-ориентированная архитектура с локальной буферизацией измерительных событий, асинхронным транспортным контуром и раздельным хранением первичных и обработанных данных позволяет обеспечить сохранность, целостность, прослеживаемость и воспроизводимость обработки измерительной информации при временных отказах отдельных программных компонентов.

A. Проблемы традиционного радиационно-химического контроля

Несмотря на высокую точность лабораторных методов

радиационно-химического контроля, их применение сопровождается задержками получения результатов и ограниченными возможностями формирования непрерывных временных рядов данных. Кроме того, традиционные подходы затрудняют повторный анализ исторической информации и воспроизводимость вычислений, что делает актуальной разработку специализированных систем мониторинга и долговременного хранения измерительной информации [2–7].

B. Архитектурные предпосылки создания системы

В работе [1] была предложена событийно-ориентированная архитектура автоматизированной системы радиационно-химического мониторинга, основанная на асинхронной передаче данных и разделении контуров хранения первичной и обработанной информации. В рамках данной архитектуры исходные измерения сохраняются в хранилище RAW, а результаты обработки — в отдельном хранилище PROC, что обеспечивает прослеживаемость происхождения результатов, воспроизводимость вычислений и возможность независимого развития аналитических модулей [1, 7, 8].

Предложенная архитектура послужила основой для разработки программного прототипа и экспериментальной проверки выполнения требований к сохранности данных, контролю целостности, прослеживаемости результатов обработки, воспроизводимости вычислений и изоляции измерительного контура.

Таблица 1. Проверяемые требования и критерии их подтверждения

Требование	**Риск**	**Механизм реализации**	**Критерий проверки**
Сохранность данных	Потеря измерительной информации при временной недоступности транспортного контура	Локальная буферизация событий в SQLite и их повторная публикация после восстановления соединения	Число сформированных событий соответствует числу записей в хранилище RAW
Целостность данных	Искажение сообщений при передаче или хранении	Использование контрольных сумм SHA-256 и проверка их корректности в процессе обработки	Модифицированные события выявляются до выполнения обработки
Прослеживаемость результатов	Невозможность установить происхождение результата обработки	Связь записей PROC с исходными событиями RAW посредством идентификатора event_id	Каждый результат обработки однозначно связан с исходным событием
Воспроизводимость обработки	Невозможность повторного выполнения вычислений над ранее накопленными данными	Разделение хранилищ RAW и PROC, сохранение сведений о параметрах и версиях обработки	Повторная обработка выполняется без изменения первичных данных
Изоляция измерительного контура	Влияние аналитических модулей на процессы измерения и регистрации данных	Асинхронная передача сообщений через Apache Kafka	Отказ подсистем обработки не приводит к остановке сбора данных

II. ТРЕБОВАНИЯ К АВТОМАТИЗИРОВАННОЙ СИСТЕМЕ МОНИТОРИНГА

Особенности жизненного цикла измерительной информации предъявляют к системе требования по обеспечению сохранности данных, контролю их целостности, прослеживаемости результатов обработки, воспроизводимости вычислений и изоляции измерительного контура от аналитических подсистем [5–10]. Для проверки выполнения данных требований в рамках настоящего исследования были сформулированы критерии, представленные в таблице 1.

Сформулированные требования определили структуру реализованного программного прототипа и использовались при разработке методики экспериментальной проверки архитектурных свойств системы. В последующих разделах рассматриваются механизмы реализации указанных требований и результаты их экспериментальной проверки.

Для реализации требований, представленных в таблице 1, был разработан программный прототип автоматизированной системы радиационно-химического мониторинга, включающий модули сбора данных,

передачи сообщений, хранения первичной информации, предварительной обработки и предоставления результатов пользователям.

Реализация программного прототипа выполнена рамках разработанных авторами в работе [1] подходов. В программном комплексе реализованы механизмы асинхронной передачи сообщений, локальной буферизации, раздельного хранения первичных и обработанных данных, контроля целостности и предоставления результатов обработки внешним приложениям.

В настоящем разделе рассматриваются структура реализованного прототипа и назначение его основных компонентов.

A. Общая структура системы

В разработанном прототипе реализован набор программных модулей, обеспечивающих полный цикл обработки измерительной информации — от формирования события до предоставления результатов пользователю.

Модуль сбора данных (DAM) реализован на языке Python и выполняет асинхронный опрос измерительных каналов, формирование событий и их подготовку к передаче в транспортный контур. При временной недоступности транспортного контура события сохраняются в локальном буфере SQLite и автоматически передаются после восстановления соединения [8, 10].

Первичные измерения сохраняются в хранилище RAW на основе PostgreSQL. Для защиты данных от изменения реализованы ограничения и триггеры, поддерживающие append-only модель хранения [5, 7, 11, 12].

Их последующая обработка выполняется модулем предварительной обработки (PPM), а результаты размещаются в отдельном хранилище PROC, что обеспечивает независимость первичных данных от алгоритмов обработки и возможность повторного выполнения вычислений [1, 11, 13].

Доступ к результатам обработки осуществляется через подсистему внешнего доступа API/UI, реализующую взаимодействие с внешними приложениями и сервисами без непосредственного доступа к хранилищу первичных данных. Асинхронное взаимодействие компонентов повышает отказоустойчивость системы и позволяет независимо масштабировать её отдельные подсистемы [14 – 16].

Общий поток данных в системе организован следующим образом (рис. 1):

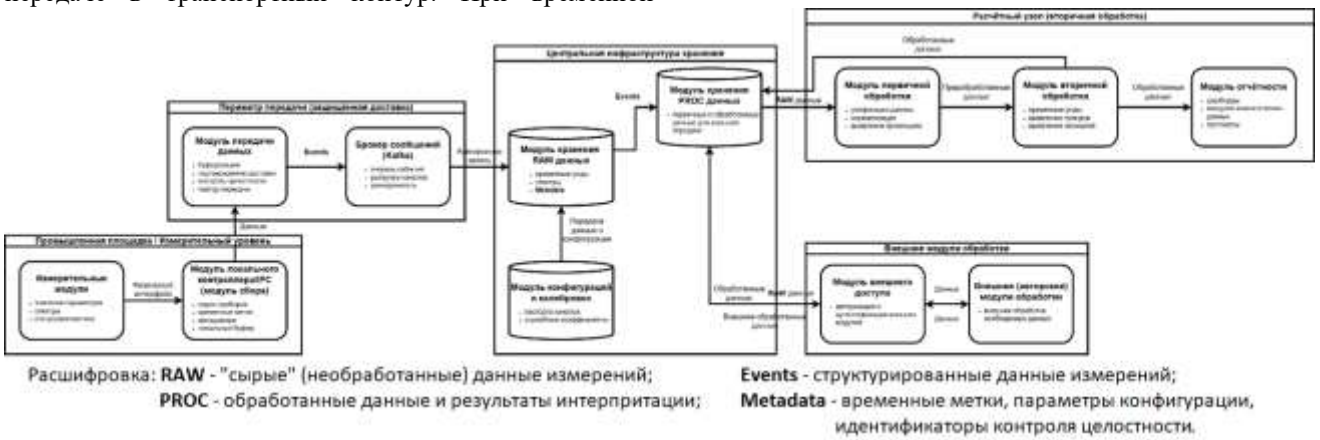

Рис. 1. Технологическая реализация модульной архитектуры и механизмы интеграции компонентов.

B. Событийная модель представления измерительной информации

В реализованном прототипе каждое измерение представляется отдельным событием, содержащим измеренное значение и набор метаданных. События используются как единый формат обмена между компонентами системы и сохраняются в неизменном виде в хранилище RAW, что обеспечивает возможность последующей проверки происхождения данных и повторной обработки результатов [1, 7, 17].

Событие содержит сведения об измеряемом параметре, времени получения данных, источнике измерения, показателях качества и контрольной сумме. Для однозначной идентификации используются поля event_id, timestamp, source_id и channel_id, обеспечивающие привязку измерений к результатам последующей обработки [17, 18].

Структура события измерения представлена в таблице 2. Поле payload содержит результаты измерения, quality_flag используется для оценки корректности данных, а checksum обеспечивает контроль целостности при передаче и хранении информации.

Выбранный состав полей обеспечивает не только передачу измерительного значения, но и сохранение сведений об источнике данных, измерительном канале, качестве информации и конфигурации системы, необходимых для последующего аудита происхождения результатов и повторной обработки накопленных данных.

Использование единой структуры события обеспечивает совместимость модулей системы и позволяет реализовать механизмы контроля целостности, прослеживаемости результатов, воспроизводимой обработки и хранения нескольких версий результатов анализа.

Таблица 2. Структура события измерения

Поле	**Назначение**
event_id	Уникальный идентификатор события
timestamp	Время формирования события
source_id	Идентификатор источника данных
channel_id	Идентификатор измерительного канала
measurement_type	Тип измеряемого параметра
sequence_no	Порядковый номер сообщения
producer_id	Идентификатор модуля-отправителя
schema_version	Версия схемы события
payload	Измерительное значение и единицы измерения
quality_flag	Признак качества данных
config_version	Версия конфигурации измерительного канала
checksum	Контрольная сумма SHA-256

C.Логическая схема потоков данных

В программном прототипе обработка данных организована как последовательность операций формирования, передачи, хранения и обработки событий [15 – 16. 19].

После получения измерительного значения модуль DAM формирует событие установленной структуры, вычисляет контрольную сумму SHA-256 и публикует сообщение в Apache Kafka [19 - 21]. Брокер сообщений обеспечивает асинхронную передачу данных между компонентами системы и снижает влияние отказов отдельных подсистем на работу измерительного контура [8, 10, 20].

После поступления сообщения из Kafka данные сохраняются в хранилище RAW, проходят процедуры контроля целостности и качества в модуле PPM, после чего результаты обработки записываются в PROC и становятся доступны через REST API [1, 11, 18].

Такая последовательность обеспечивает разделение первичных и обработанных данных, а использование идентификатора event_id позволяет сохранять однозначную связь между исходным событием и результатами его обработки, обеспечивая прослеживаемость происхождения результатов и возможность их повторного получения при изменении алгоритмов обработки.

III. РЕАЛИЗАЦИЯ ПРОГРАММНОГО КОМПЛЕКСА

На основе предложенной архитектуры был реализован программный прототип системы радиационно-химического мониторинга. В прототипе реализованы механизмы локальной буферизации измерительных событий, асинхронной передачи сообщений, контроля целостности данных, неизменяемого хранения первичных измерений и прослеживаемой обработки исторической информации.

Далее приведены особенности реализации программных компонентов, обеспечивающих выполнение требований, сформулированных в таблице 1.

A.Реализация модуля сбора данных (DAM)

В соответствии с рассмотренной схемой потоков данных начальным элементом системы является модуль сбора данных (DAM), выполняющий получение измерительной информации, формирование событий и первичный контроль корректности данных [15, 19].

Сбор данных реализован посредством асинхронного опроса измерительных каналов, для каждого из которых используется независимый механизм получения информации [1, 14]. После получения измерительного значения формируется событие, содержащее идентификатор, временную метку и необходимые метаданные для последующей обработки и прослеживаемости [11, 14].

В реализованном прототипе модуль DAM разработан на языке Python 3.12 с использованием библиотеки AsyncIO. Каждый измерительный канал обслуживается отдельной асинхронной задачей, что позволяет выполнять независимый опрос источников данных без блокировки остальных каналов.

Для контроля целостности данных вычисляется контрольная сумма SHA-256, сохраняемая в поле checksum [20]. При временной недоступности брокера сообщений события сохраняются в локальной базе SQLite. После восстановления доступности транспортного контура выполняется автоматическая повторная публикация накопленных сообщений в порядке их формирования [8, 9, 22].

B.Реализация подсистемы передачи сообщений

Передача данных между компонентами системы реализована с использованием брокера сообщений Apache Kafka. Данный компонент выполняет функции промежуточного слоя обмена данными и обеспечивает асинхронное взаимодействие между модулем сбора данных и последующими подсистемами хранения и обработки информации [23 - 24].

После формирования события модуль DAM публикует его в соответствующий топик Kafka. Дальнейшая доставка сообщений осуществляется независимо от состояния источника данных, что позволяет исключить прямую зависимость процессов измерения от процессов хранения и обработки информации [8, 23].

В прототипе Kafka используется в качестве транспортного контура между DAM, RAW и модулями обработки. Такая схема позволяет продолжать формирование событий независимо от состояния подсистем хранения и обработки, а также обеспечивает

возможность подключения дополнительных потребителей данных без модификации измерительного контура.

C. Реализация хранилища первичных данных

Хранилище RAW реализовано на основе системы управления базами данных PostgreSQL и предназначено для долговременного хранения первичной измерительной информации [25]. Структура базы данных включает таблицы событий измерений и связанных с ними метаданных, необходимых для последующей обработки и анализа данных.

Основным принципом функционирования хранилища является модель append-only, при которой новые события могут только добавляться в базу данных без изменения ранее сохранённых записей 17, 26]. Такой подход позволяет сохранить полный журнал измерений и обеспечить воспроизводимость последующей обработки данных.

Для реализации append-only модели хранения в базе данных созданы триггеры, запрещающие выполнение операций UPDATE и DELETE для таблиц первичных измерений. Изменение или исправление данных осуществляется только путём добавления новых записей, при этом ранее сохранённые события остаются неизменными.

Реализованный механизм обеспечивает выполнение требований к сохранности первичных данных и исключает возможность изменения ранее зарегистрированных измерений без формирования новой записи.

D. Реализация модуля предварительной обработки данных (PPM)

Модуль предварительной обработки данных (Primary Processing Module, PPM) предназначен для контроля корректности событий, поступающих из хранилища RAW, и подготовки информации к дальнейшей аналитической обработке [1].

На этапе обработки выполняется проверка контрольных сумм, позволяющая выявлять повреждённые или изменённые сообщения. Дополнительно анализируются признаки качества данных, содержащиеся в поле quality_flag, что позволяет обнаруживать измерения, требующие дополнительной проверки [17, 27].

Для обеспечения трассируемости обработки в модуле реализовано журналирование выполненных операций. Для каждого запуска сохраняются сведения о времени обработки, обработанных событиях и результатах выполнения процедур контроля данных, что позволяет восстановить историю обработки и провести аудит вычислений [17, 18].

Помимо сведений о времени обработки, журнал содержит идентификаторы обработанных событий, версию алгоритма и параметры выполнения процедуры обработки. Это позволяет воспроизводить результаты вычислений, сопоставлять результаты различных версий алгоритмов и обеспечивать прослеживаемость изменений процедуры обработки данных.

E. Реализация хранилища результатов PROC

Хранилище PROC предназначено для хранения результатов обработки и аналитической интерпретации измерительной информации. В отличие от RAW, данное хранилище содержит не исходные измерения, а данные, полученные в результате выполнения алгоритмов обработки [1, 11].

Помимо результата обработки, запись PROC содержит идентификатор исходного события RAW, сведения о версии алгоритма обработки и времени выполнения вычислений.

Каждая запись в PROC содержит ссылку на соответствующее событие в хранилище RAW, что обеспечивает возможность восстановления происхождения результата и последующей проверки корректности выполненных вычислений [18, 17].

Разделение первичных данных и результатов обработки позволяет повторно выполнять вычисления при изменении алгоритмов анализа без модификации исходной информации. Такой подход обеспечивает воспроизводимость результатов, упрощает сравнение различных версий алгоритмов и соответствует современным принципам построения информационно-аналитических систем [11, 28, 29].

F. Реализация подсистемы внешнего доступа

После выполнения обработки данные становятся доступны пользователям и внешним информационным системам через REST API, реализованный на основе фреймворка FastAPI [24, 28].

Подсистема обеспечивает доступ к данным из хранилища PROC посредством HTTP-запросов и предоставляет унифицированный интерфейс для интеграции с пользовательскими приложениями и внешними сервисами. Использование REST-подхода позволяет отделить клиентские приложения от внутренней структуры хранения и обработки данных [30].

Реализованный интерфейс поддерживает подключение средств визуализации, мониторинга и аналитических сервисов без непосредственного доступа к измерительному контуру и хранилищу первичных данных, что соответствует требованиям к изоляции компонентов системы [11, 17, 31].

G. Реализация модуля вторичной обработки данных (SPM)

Модуль вторичной обработки данных (Secondary Processing Module, SPM) предназначен для выполнения аналитических операций над результатами, сформированными на предыдущих этапах обработки. В отличие от PPM, работающего с первичными измерениями, SPM использует данные из хранилища PROC и не взаимодействует непосредственно с измерительным контуром.

Результаты работы SPM также сохраняются в PROC и сохраняют связь с исходными событиями RAW через результаты предварительной обработки, что обеспечивает прослеживаемость всей цепочки вычислений.

Основной задачей модуля является формирование производных показателей, агрегация данных и

выполнение аналитических процедур, требующих обработки накопленных результатов измерений. Такое разделение позволяет независимо развивать аналитические алгоритмы не изменяя процессы сбора данных и хранения первичной информации [28 - 29].

Работа SPM организована на основе данных, содержащихся в PROC, что обеспечивает сохранение воспроизводимости вычислений и исключает воздействие аналитических процедур на исходные измерения. При необходимости новые алгоритмы могут быть применены к уже накопленным данным без изменения содержимого хранилища RAW [1, 11].

Выделение вторичной обработки в отдельный модуль обеспечивает дополнительное разделение ответственности между компонентами системы и соответствует принципам построения масштабируемых информационно-аналитических платформ [14 - 16, 23].

H. Технологический стек системы

Выбор используемых технологий определялся требованиями к асинхронной обработке событий, локальной буферизации, долговременному хранению данных и предоставлению результатов внешним потребителям. Используемый набор программных компонентов (представлен в таблице 3) позволил реализовать все механизмы, проверяемые в рамках экспериментальных испытаний прототипа.

Как следует из таблицы 3, каждая подсистема реализует один или несколько механизмов, используемых для обеспечения сохранности данных, контроля целостности, прослеживаемости результатов обработки, воспроизводимости вычислений и изоляции измерительного контура. Проверка выполнения указанных свойств рассматривается в следующем разделе.

IV. МЕТОДИКА ТЕСТИРОВАНИЯ

Для проверки рабочей гипотезы исследования была выполнена программно-экспериментальная проверка разработанного прототипа с использованием стенда, включающего программные эмуляторы измерительных каналов, транспортный контур Apache Kafka, хранилища RAW и PROC, а также средства мониторинга и журналирования. Испытания проводились как в штатных режимах функционирования, так и при моделировании отказов отдельных компонентов системы.

В связи с этим после завершения разработки была проведена серия экспериментальных испытаний, направленных на проверку ключевых функциональных и архитектурных свойств системы.

Тестирование проводилось с использованием многоуровневого подхода, включающего проверку отдельных модулей, их функций и взаимодействия между компонентами системы. Такой подход позволил оценить работоспособность как отдельных программных компонентов, так и всего информационного контура в целом.

Таблица 3. Программные компоненты реализованного прототипа и используемые технологии

Подсистема	**Технология**	**Назначение**	**Проверяемое свойство**
DAM	Python 3.12, AsyncIO	Сбор данных и формирование событий	Корректность формирования событий
Локальная буферизация	SQLite	Временное хранение данных при недоступности брокера	Сохранность данных
Передача сообщений	Apache Kafka 2.6	Асинхронный обмен сообщениями между компонентами	Изоляция измерительного контура
RAW	PostgreSQL	Хранение первичных измерений	Неизменяемость и сохранность данных
PPM	Python 3.12	Контроль целостности и качества данных	Целостность данных
PROC	PostgreSQL	Хранение результатов обработки	Прослеживаемость и воспроизводимость обработки
SPM	Python 3.12	Вторичная обработка и аналитические процедуры	Воспроизводимость обработки
API	FastAPI	Предоставление данных внешним пользователям и сервисам	Изоляция измерительного контура
Мониторинг	Prometheus	Сбор эксплуатационных метрик	Контроль работоспособности компонентов
Визуализация	Grafana	Отображение состояния системы и результатов мониторинга	Контроль работоспособности компонентов
Конфигурация	MongoDB	Хранение конфигурационных параметров системы	Воспроизводимость обработки

В последующих подразделах рассматриваются используемая методика испытаний, структура тестового покрытия и результаты проверки основных архитектурных решений, реализованных в разработанном прототипе.

A. Подход к организации испытаний

Экспериментальная проверка прототипа выполнялась на трёх уровнях тестирования: модульном, функциональном и интеграционном [20, 23, 31]. Такой подход позволил оценить как корректность работы отдельных компонентов системы, так и функционирование полного контура обработки данных.

Для каждого проверяемого свойства были заранее определены критерии успешности. Сохранность данных считалась подтверждённой при отсутствии потерь событий после восстановления отказавшего компонента. Целостность данных считалась подтверждённой при выявлении всех преднамеренно модифицированных тестовых сообщений. Прослеживаемость подтверждалась возможностью однозначного сопоставления результата обработки с исходным событием RAW. Воспроизводимость подтверждалась повторным выполнением обработки без изменения исходных данных. Изоляция измерительного контура подтверждалась сохранением формирования событий при отказе подсистем обработки.

Модульное тестирование было направлено на проверку отдельных программных компонентов, включая формирование событий, работу механизмов буферизации, контроль целостности данных и операции хранения информации. Функциональное тестирование использовалось для подтверждения выполнения основных требований к системе, включая доставку сообщений с локальным сохранением и повторной отправкой при восстановлении доступности транспортного контура, защиту данных и корректность обработки измерительной информации.

Интеграционное тестирование проводилось для проверки взаимодействия между компонентами архитектуры и прохождения данных по полному маршруту от источника измерений до пользовательского интерфейса. Дополнительно оценивалась работоспособность системы при возникновении отказов отдельных компонентов инфраструктуры [8, 21].

Структура тестового покрытия, использованная при проведении испытаний, представлена в следующем подразделе.

B. Матрица покрытия тестов

Для обеспечения полноты проверки использовалась матрица тестирования, объединяющая программные модули, типы тестирования, проверяемые функции и применяемые инструменты [29, 31].

Таблица 4 Матрица тестирования программного комплекса

Модуль	**Тип тестирования**	**Проверяемая функция**	**Средства проверки**	**Проверяемое свойство**
DAM	Модульное	Формирование событий	Автоматизированные программные тесты	Корректность формирования событий
DAM	Модульное	Расчёт контрольных сумм	Автоматизированные программные тесты	Целостность данных
Kafka	Функциональное	Передача сообщений	Средства мониторинга брокера сообщений	Изоляция измерительного контура
DAM + Kafka	Интеграционное	Буферизация при отказе брокера	Имитация отказа и анализ восстановления	Сохранность данных
RAW	Функциональное	Запись событий	SQL-запросы и анализ содержимого БД	Сохранность данных
RAW	Функциональное	Защита append-only	Попытка изменения и удаления записей	Неизменяемость первичных данных
PPM	Модульное	Проверка контрольных сумм	Автоматизированные программные тесты	Целостность данных
PPM	Модульное	Контроль качества данных	Автоматизированные программные тесты	Целостность данных
PROC	Функциональное	Сохранение результатов обработки	SQL-запросы и анализ содержимого БД	Прослеживаемость и воспроизводимость обработки
API	Функциональное	Получение данных через REST API	Тестовые HTTP-запросы	Изоляция измерительного контура
Комплекс в целом	Интеграционное	Сквозная обработка данных	Сквозной сценарий прохождения данных	Сохранность данных, прослеживаемость и воспроизводимость

				обработки

Такой подход позволил систематизировать испытания и обеспечить контроль основных архитектурных требований системы (матрица тестирования приведена в таблице 4).

Представленная матрица охватывает все основные компоненты разработанного прототипа и позволяет оценить корректность реализации отдельных функций, работоспособность полного контура обработки измерительной информации. В последующих подразделах рассматриваются наиболее значимые испытания, подтверждающие выполнение ключевых архитектурных требований системы.

C. Испытательный стенд и параметры моделирования

Испытания выполнялись на программном стенде, включающем модули DAM, PPM, PROC и API, брокер сообщений Apache Kafka, локальный буфер SQLite, хранилища RAW и PROC на основе PostgreSQL. Конфигурация программной среды - в таблице 5.

Таблица 5 Конфигурация испытательного стенда

Параметр	**Значение**
Язык программирования	Python 3.12
Apache Kafka	2.6
PostgreSQL	17
SQLite	3.53
FastAPI	0.136
Prometheus	3.13
Grafana	13.1
MongoDB	8.2

Для имитации работы измерительного оборудования использовались программные эмуляторы измерительных каналов. Параметры формирования измерительных событий задавались в соответствии со сценарием испытаний и приведены в Таблице 6.

Таблица 6 Параметры моделирования каналов

Параметр	**Значение**
Количество каналов	15
Частота формирования событий	1 Гц
Продолжительность испытания	2 часа (7200 с.)
Размер события	350 Байт/событие
Общее число событий	108000
Объём сгенерированных событий	37.8 МБ

За время испытаний было сформировано и обработано 108000 событий измерений общим объёмом около 37.8 МБ, что позволило выполнить проверку механизмов передачи, хранения и обработки данных в условиях непрерывного потока сообщений и смоделировать режим эксплуатации системы мониторинга.

D. Средства тестирования и контроля результатов

В качестве источников данных использовались программные эмуляторы измерительных каналов, формирующие последовательности измерительных событий в диапазонах значений для радиометрических, химических и спектральных измерений.

Для проверки механизмов хранения данных использовалась система управления базами данных PostgreSQL, в которой были развёрнуты хранилища RAW и PROC [25]. Передача сообщений между компонентами системы осуществлялась посредством брокера Apache Kafka, обеспечивающего реализацию событийной модели обмена данными [21, 27].

Контроль выполнения тестов осуществлялся с использованием встроенных средств журналирования программных модулей, журналов PostgreSQL и Apache Kafka, а также средств мониторинга состояния системы. Анализ журналов использовался для подтверждения корректности прохождения сообщений между компонентами, выявления ошибок передачи данных и контроля выполнения процедур обработки [21, 24].

Применение указанных средств позволило обеспечить воспроизводимость испытаний и получить объективные данные о работоспособности разработанного программного комплекса.

E. Сценарии испытаний

Проверка архитектурных свойств прототипа выполнялась в рамках набора штатных и отказных сценариев. Для каждого сценария были определены проверяемое свойство и критерий успешности испытания (таблица 7).

Представленные сценарии охватывают требования, сформулированные в таблице 1, и позволяют выполнить экспериментальную проверку сохранности данных, контроля целостности, прослеживаемости результатов обработки, воспроизводимости вычислений и изоляции измерительного контура. Результаты выполнения указанных испытаний приведены в следующем разделе.

Таблица 7. Сценарии испытаний

Сценарий	Проверяемое свойство	Критерий успешности
Штатная работа	Сохранность данных	Число записей в RAW соответствует числу сформированных событий
Отказ Kafka	Сохранность данных	После восстановления отсутствуют потери событий
Нарушение контрольной суммы	Целостность данных	Все модифицированные события выявлены до обработки
Попытка UPDATE/DELETE	Неизменяемость первичных данных	Операции отклоняются механизмами защиты RAW
Повторная обработка	Воспроизводимость обработки	Повторный запуск выполняется без изменения RAW
Отключение PPM	Изоляция измерительного контура	DAM продолжает формирование событий
Отключение API	Изоляция измерительного контура	DAM и PPM продолжают работу независимо от внешнего доступа
Сопоставление RAW и PROC	Прослеживаемость результатов обработки	Для каждого результата PROC определяется исходное событие RAW

.

V.РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Разработанная методика испытаний была применена для проверки ключевых архитектурных свойств реализованного прототипа. Целью испытаний являлось подтверждение выполнения требований к сохранности данных, контролю целостности измерительной информации, воспроизводимости обработки и изоляции измерительного контура. Особое внимание уделялось сценариям, связанным с отказами отдельных компонентов системы и сохранением работоспособности информационного контура в условиях возникновения нештатных ситуаций [8-9, 16, 31].

В данном разделе представлены результаты наиболее значимых испытаний, позволяющих оценить практическую реализуемость предложенных архитектурных решений.

A. Проверка буферизации при отказе брокера

Проверка механизма буферизации выполнялась путём моделирования отказа брокера сообщений Apache Kafka в процессе работы системы. В ходе испытаний модуль DAM продолжал получать данные от эмуляторов измерительных каналов и формировать события измерений, однако передача сообщений в брокер была недоступна [21, 27].

При отсутствии соединения события временно сохранялись в локальном буфере SQLite до восстановления доступности брокера сообщений [32].

Для проверки механизма буферизации была выполнена серия из пяти последовательных отказов брокера Apache Kafka (таблица 8).

Таблица 8 Параметры сценария отказов Kafka

№ отказа	Длительность отказа, с	Штатная работа после отказа, с
1	5	300
2	10	300
3	15	300
4	20	300
5	25	300

Продолжительность отказов составляла 5, 10, 15, 20 и 25 секунд соответственно. Между отказами система функционировала в штатном режиме в течение 5 минут, обеспечивая передачу и обработку поступающих сообщений. Количественные результаты испытания приведены в таблице 9.

Таблица 9 Результаты проверки механизма буферизации

Параметр	Значение
Общее число событий испытания	108000
Событий, накопленных в SQLite	1125
Событий, переданных после восстановления	1125
Записано в RAW	108000
Потери данных	0

Таким образом, критерий успешности, сформулированный для сценария отказа Kafka в таблице 7, выполнен.

Число записей в хранилище RAW совпало с числом сформированных событий (108000). Все 1125 сообщений, накопленных в локальном буфере SQLite в период недоступности брокера, были успешно переданы после восстановления соединения. Потери данных не зафиксированы.

B. Проверка контроля целостности измерительной информации

Испытание проводилось путём преднамеренного изменения содержимого сформированных событий после вычисления контрольной суммы. В качестве критерия успешности использовалось обнаружение несоответствия между сохранённым и вычисленным

значениями контрольной суммы [20]. Количественные результаты испытания приведены в таблице 10.

Таблица 10 Результаты проверки контроля целостности

Параметр	Значение
Преднамеренно модифицированных сообщений	100
Выявлено нарушений контрольной суммы	100
Необнаруженных нарушений	0
Повреждённых сообщений, допущенных к обработке	0

Таким образом, критерий успешности, сформулированный для сценария нарушения контрольной суммы в таблице 7, выполнен. Все 100 преднамеренно модифицированных сообщений были выявлены механизмом контроля целостности, зарегистрированы в журнале обработки и исключены из дальнейшей обработки. Необнаруженных нарушений не зафиксировано.

C. Проверка неизменяемости хранилища RAW

Для проверки реализации принципа append-only была выполнена попытка изменения ранее сохранённых записей в хранилище RAW посредством выполнения SQL-операций UPDATE и DELETE [17, 26].

Для проверки реализации принципа append-only выполнялась серия попыток изменения и удаления ранее сохранённых записей. Количественные результаты испытания приведены в таблице 11.

Таблица 11 Результаты проверки неизменяемости хранилища RAW

Операция	Количество попыток	Успешно выполнено
UPDATE	100	0
DELETE	100	0

Таким образом, критерий успешности, сформулированный для сценария проверки неизменяемости первичных данных в таблице 7, выполнен. Все попытки выполнения операций UPDATE и DELETE были отклонены средствами защиты базы данных. Изменений содержимого хранилища RAW не зафиксировано.

D. Проверка прослеживаемости и воспроизводимости обработки

Для проверки прослеживаемости выполнялось сопоставление результатов обработки, сохранённых в хранилище PROC, с исходными событиями RAW посредством идентификатора event_id. Для проверки воспроизводимости обработка исторических данных выполнялась повторно без изменения содержимого хранилища RAW (таблица 12).

Таблица 12 Результаты проверки прослеживаемости и воспроизводимости обработки

Параметр	Значение
Проверено записей PROC	108000
Корректно связанных с RAW	108000
Ошибок сопоставления	0
Повторных запусков обработки	100
Изменений в RAW	0

E. Сквозное тестирование системы

Завершающим этапом экспериментальной проверки являлось сквозное тестирование полного контура обработки данных. В рамках испытаний контролировалось прохождение измерительной информации через все основные компоненты системы: от формирования события до получения результата пользователем. Маршрут прохождения данных представлен на рисунке 2. Для проверки полного контура обработки анализировалось прохождение сформированных событий через транспортный контур Apache Kafka, хранилище RAW, модуль предварительной обработки PPM, хранилище результатов PROC и подсистему внешнего доступа API. Количественные результаты испытания приведены в таблице 13.

Таким образом, критерий успешности формулированный для сценария сквозной обработки данных в таблице 4, выполнен. Все 108000 сформированных событий были успешно переданы через транспортный контур, сохранены в хранилище RAW, обработаны модулем PPM, записаны в PROC и предоставлены пользователям через REST API. Потери данных на этапах передачи, хранения и обработки не зафиксированы.

Полученные результаты подтверждают работоспособность реализованной архитектуры и возможность практического применения предложенного подхода при построении автоматизированных систем радиационно-химического мониторинга.

VI. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Проведённые испытания позволили оценить соответствие реализованного прототипа архитектурным требованиям, сформулированным на этапе проектирования системы. Основное внимание уделялось проверке сохранности измерительной информации, контролю её целостности, реализации принципа неизменяемости первичных данных и разделению контуров хранения и обработки [1, 5, 17].

A. Анализ выполнения архитектурных требований

Результаты испытаний показали, что все сформулированные на этапе проектирования архитектурные требования были выполнены. В ходе проверки механизма буферизации при отказе брокера сообщений потери данных не зафиксированы: число записей в RAW совпало с числом сформированных событий (108000), включая 1125 событий, временно

сохранённых в локальном буфере SQLite. Проверка контрольных сумм показала обнаружение всех 100 преднамеренно модифицированных сообщений.

Попытки изменения содержимого RAW посредством операций UPDATE и DELETE были отклонены во всех случаях. Для всех 108000 записей PROC была подтверждена однозначная связь с исходными событиями RAW посредством идентификатора event_id. Повторное выполнение обработки исторических данных не приводило к изменению содержимого хранилища RAW и обеспечивало получение результатов на основе неизменного набора исходных измерений, что подтвердило выполнение требования к воспроизводимости обработки.

Таблица 13 Результаты сквозного тестирования системы

Параметр	**Значение**
Сформировано событий	108000
Передано через Apache Kafka	108000
Записано в RAW	108000
Обработано модулем PPM	108000
Записано в PROC	108000
Получено через REST API	108000
Потери данных	0

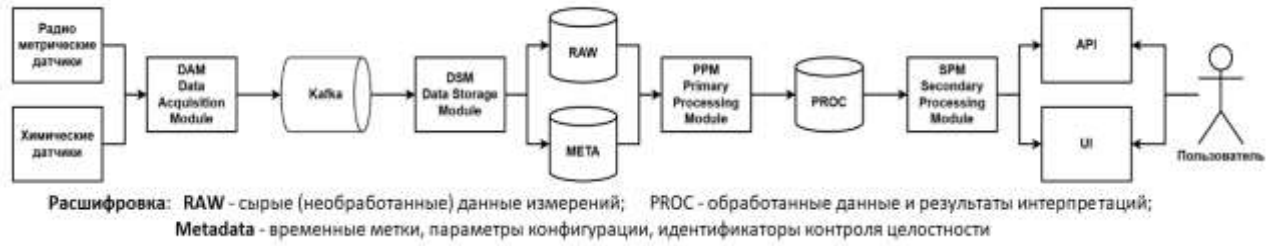

Рис. 2. Маршрут прохождения данных через систему

B. Значение полученных результатов для систем радиационно-химического мониторинга

Полученные результаты показывают, что использование событийной модели передачи данных совместно с разделением хранилищ RAW и PROC позволяет реализовать требования, характерные для систем радиационно-химического мониторинга. В отличие от традиционных схем хранения текущего состояния объекта, реализованный подход обеспечивает сохранение полной истории измерений, возможность последующего аудита результатов обработки и повторного применения новых алгоритмов к ранее накопленным данным [1, 5, 7, 17].

C. Ограничения текущего этапа исследования

Проведённая экспериментальная проверка была выполнена на программном стенде с использованием эмуляторов измерительных каналов. В рамках исследования не рассматривались вопросы масштабирования системы при существенно более высоких скоростях поступления данных, длительной эксплуатации в непрерывном режиме, а также интеграции с реальными измерительными приборами. Кроме того, испытания проводились для ограниченного набора сценариев отказов и не охватывали все возможные варианты нарушения работоспособности инфраструктурных компонентов.

D. Итоговая оценка выполнения требований

Обобщённые результаты проверки требований, сформулированных в разделе II, представлены в таблице 14.

Результаты экспериментальной проверки показывают, что разработанный прототип обеспечивает выполнение требований к сохранности данных, контролю целостности, прослеживаемости результатов обработки, воспроизводимости вычислений и изоляции измерительного контура. Полученные результаты подтверждают практическую реализуемость предложенной архитектуры автоматизированной системы радиационно-химического мониторинга и возможность её дальнейшего развития для использования в задачах контроля жидких радиоактивных сред.

VII. ЗАКЛЮЧЕНИЕ

В работе разработан и реализован программный прототип автоматизированной системы радиационно-химического мониторинга жидких радиоактивных сред, основанный на событийной модели передачи данных, асинхронном взаимодействии компонентов и разделении контуров хранения первичной и обработанной информации.

В состав прототипа вошли модуль сбора данных DAM, подсистема передачи сообщений на основе Apache Kafka, хранилища RAW и PROC, модуль предварительной обработки PPM, модуль вторичной обработки SPM и подсистема внешнего доступа через REST API. Реализованная архитектура обеспечивает сохранение первичных данных в неизменном виде, контроль целостности сообщений, прослеживаемость результатов обработки и возможность повторного выполнения вычислений.

В ходе экспериментальной проверки было обработано 108000 событий измерений. Результаты испытаний подтвердили выполнение требований к сохранности данных, контролю целостности, неизменяемости первичной информации, прослеживаемости результатов обработки, воспроизводимости вычислений и изоляции измерительного контура. Потери данных при моделировании отказов не зафиксированы, а все преднамеренно модифицированные сообщения были выявлены средствами контроля целостности.

Полученные результаты подтверждают практическую реализуемость предложенного подхода к построению автоматизированных систем радиационно-химического мониторинга. Дальнейшее развитие работы связано с интеграцией прототипа с реальными средствами измерений и исследованием возможностей использования накопленных исторических данных для интеллектуальной обработки и поддержки принятия решений.

Таблица 14 Итоговая оценка выполнения архитектурных требований

Требование	**Критерий успешности**	**Результат**
Сохранность данных	Потери отсутствуют	Выполнено
Целостность данных	Все нарушения выявлены	Выполнено
Неизменяемость RAW	UPDATE/DELETE отклоняются	Выполнено
Прослеживаемость	PROC связан с RAW	Выполнено
Воспроизводимость	RAW не изменяется	Выполнено
Изоляция контура	DAM работает при отказах	Выполнено

Библиография

[1] А. П. Полянский, М. Г. Жабицкий, «Автоматизация радиационно-химического контроля жидких радиоактивных отходов и сред», International Journal of Open Information Technologies, т. 14, № 5, с. 77–89, 2026.

[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," IEEE Industrial Electronics Magazine, vol. 11, no. 1, pp. 17–27, 2017.

[3] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," IEEE Transactions on Industrial Informatics, vol. 10, no. 4, pp. 2233–2243, 2014.

[4] Мартынова О. И., Живилова Л. М., Рогацкин Б. С., Субботина Н. П. Химический контроль на тепловых и атомных электростанциях / Под ред. О. И. Мартыновой. – М.: Энергия, 1980. – 320 с.

[5] H. M. Gladney, Preserving Digital Information. Berlin, Germany: Springer, 2007.

[6] OSIsoft LLC, PI System Architecture, Planning and Implementation Course. San Leandro, CA, USA: OSIsoft LLC, 2020.

[7] M. Fowler, Patterns of Enterprise Application Architecture. Boston, MA, USA: Addison-Wesley, 2003.

[8] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A Distributed Messaging System for Log Processing," presented at the NetDB Workshop, Athens, Greece, Jun. 2011.

[9] M. Kleppmann, Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. Sebastopol, CA, USA: O'Reilly Media, 2017.

[10] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Boston, MA, USA: Addison-Wesley, 2003.

[11] International Electrotechnical Commission, IEC 62264-1:2013 Enterprise-Control System Integration — Part 1: Models and Terminology. Geneva, Switzerland: IEC, 2013.

[12] International Society of Automation, ISA-95.00.01-2010 Enterprise-Control System Integration — Part 1: Models and Terminology. Research Triangle Park, NC, USA: ISA, 2010.

[13] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The Industrial Internet of Things (IIoT): An Analysis Framework," Computers in Industry, vol. 101, pp. 1–12, 2018.

[14] S. Newman, Building Microservices: Designing Fine-Grained Systems, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2021.

[15] M. Richards and N. Ford, Fundamentals of Software Architecture: An Engineering Approach. Sebastopol, CA, USA: O'Reilly Media, 2020.

[16] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 4th ed. Boston, MA, USA: Addison-Wesley, 2021.

[17] M. Fowler, "Event Sourcing," 2005. [Online]. Available: https://martinfowler.com/eaaDev/EventSourcing.html . Accessed: Jul. 15, 2026.

[18] International Organization for Standardization, ISO 8000-61:2016 Data Quality — Part 61: Data Quality Management: Process Reference Model. Geneva, Switzerland: ISO, 2016.

[19] Apache Software Foundation, "Record, Event and Message Concepts," Apache Kafka Documentation, 2025. [Online]. Available: https://kafka.apache.org/documentation/

[20] D. Eastlake and T. Hansen, US Secure Hash Algorithms (SHA and SHA-Based HMAC and HKDF), RFC 6234. Internet Engineering Task Force, 2011.

[21] Apache Software Foundation, "Introduction to Apache Kafka," Apache Kafka Documentation, 2025. [Online]. Available: https://kafka.apache.org

[22] J. Cheney, L. Chiticariu, and W. C. Tan, "Provenance in Databases: Why, How, and Where," Foundations and Trends in Databases, vol. 1, no. 4, pp. 379–474, 2009.

[23] C. Richardson, Microservices Patterns. Shelter Island, NY, USA: Manning Publications, 2018.

[24] R. T. Fielding, Architectural Styles and the Design of Network-Based Software Architectures, Ph.D. dissertation, University of California, Irvine, CA, USA, 2000.

[25] PostgreSQL Global Development Group, "PostgreSQL Documentation," Version 17, 2025. [Online]. Available: https://www.postgresql.org/docs/

[26] C. J. Date, An Introduction to Database Systems, 8th ed. Boston, MA, USA: Addison-Wesley, 2004.

[27] J. Kreps, I Heart Logs: Event Data, Stream Processing, and Data Integration. Sebastopol, CA, USA: O'Reilly Media, 2014.

[28] FastAPI, "FastAPI Documentation," 2025. [Online]. Available: https://fastapi.tiangolo.com

[29] P. Ammann and J. Offutt, Introduction to Software Testing, 2nd ed. Cambridge, U.K.: Cambridge University Press, 2016.

[30] M. Fowler, "Microservices Resource Guide." [Online]. Available: https://martinfowler.com/microservices/

[31] G. J. Myers, C. Sandler, and T. Badgett, The Art of Software Testing, 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, 2011.

[32] SQLite Consortium, "About SQLite," SQLite Documentation, 2025. [Online]. Available: https://www.sqlite.org

Software Architecture Validation of a Radiation and Chemical Monitoring System for Liquid Radioactive Media

A.P. Polyanskiy, M.G. Zhabitskii, V.A. Sychev

***Abstract* - This paper presents the implementation and validation of a software prototype for an automated radiation and chemical monitoring system for liquid radioactive media based on an event-driven data model. The proposed architecture incorporates an asynchronous messaging layer based on Apache Kafka, local buffering of measurement events, separate storage of primary and processed data in RAW and PROC repositories, and mechanisms for data integrity verification and processing traceability. The developed prototype preserves primary measurement data in an immutable form, supports repeated processing of historical data, and ensures the independence of the measurement subsystem from analytical components. To validate the proposed architecture, a testing methodology including unit, functional, and integration tests was developed. During validation, 108,000 measurement events were generated and processed. Test scenarios covered transport layer failures, data integrity verification, protection of the RAW repository, traceability of processing results, and reproducibility of data processing. The results confirmed compliance with the requirements for data preservation, integrity verification, immutability of primary data, traceability of processing results, reproducibility of processing, and isolation of the measurement subsystem. The proposed approach demonstrates its technical feasibility as a basis for developing automated radiation and chemical monitoring systems for nuclear facilities.**

***Keywords: radiation and chemical monitoring, event-driven architecture, Apache Kafka, RAW and PROC repositories, data traceability, reproducible data processing, data integrity, automated monitoring systems*.**

REFERENCES

[1]A.P. Poljanskii, M.G. Zhabitskii «Liquid Radioactive Waste and Mixtures Radiation and Chemical Monitoring Automation», International Journal of Open Information Technologies, т. 14, № 5, с. 77–89, 2026.
[2]M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," IEEE Industrial Electronics Magazine, vol. 11, no. 1, pp. 17–27, 2017.
[3]L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," IEEE Transactions on Industrial Informatics, vol. 10, no. 4, pp. 2233–2243, 2014.
[4]O. I. Martynova, L. M. Zhivilova, B. S. Rogatskin, and N. P. Subbotina, Chemical Control at Thermal and Nuclear Power Plants, Moscow: Energiya, 1980, 320 p.
[5]H. M. Gladney, Preserving Digital Information. Berlin, Germany: Springer, 2007.
[6]OSIsoft LLC, PI System Architecture, Planning and Implementation Course. San Leandro, CA, USA: OSIsoft LLC, 2020.
[7]M. Fowler, Patterns of Enterprise Application Architecture. Boston, MA, USA: Addison-Wesley, 2003.
[8]J. Kreps, N. Narkhede, and J. Rao, "Kafka: A Distributed Messaging System for Log Processing," presented at the NetDB Workshop, Athens, Greece, Jun. 2011.
[9]M. Kleppmann, Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. Sebastopol, CA, USA: O'Reilly Media, 2017.
[10] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Boston, MA, USA: Addison-Wesley, 2003.
[11] International Electrotechnical Commission, IEC 62264-1:2013 Enterprise-Control System Integration — Part 1: Models and Terminology. Geneva, Switzerland: IEC, 2013.
[12] International Society of Automation, ISA-95.00.01-2010 Enterprise-Control System Integration — Part 1: Models and Terminology. Research Triangle Park, NC, USA: ISA, 2010.
[13] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The Industrial Internet of Things (IIoT): An Analysis Framework," Computers in Industry, vol. 101, pp. 1–12, 2018.
[14] S. Newman, Building Microservices: Designing Fine-Grained Systems, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2021.
[15] M. Richards and N. Ford, Fundamentals of Software Architecture: An Engineering Approach. Sebastopol, CA, USA: O'Reilly Media, 2020.
[16] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 4th ed. Boston, MA, USA: Addison-Wesley, 2021.
[17] M. Fowler, "Event Sourcing," 2005. [Online]. Available: https://martinfowler.com/eaaDev/EventSourcing.html . Accessed: Jul. 15, 2026.
[18] International Organization for Standardization, ISO 8000-61:2016 Data Quality — Part 61: Data Quality Management: Process Reference Model. Geneva, Switzerland: ISO, 2016.
[19] Apache Software Foundation, "Record, Event and Message Concepts," Apache Kafka Documentation, 2025. [Online]. Available: https://kafka.apache.org/documentation/
[20] D. Eastlake and T. Hansen, US Secure Hash Algorithms (SHA and SHA-Based HMAC and HKDF), RFC 6234. Internet Engineering Task Force, 2011.
[21] Apache Software Foundation, "Introduction to Apache Kafka," Apache Kafka Documentation, 2025. [Online]. Available: https://kafka.apache.org
[22] J. Cheney, L. Chiticariu, and W. C. Tan, "Provenance in Databases: Why, How, and Where," Foundations and Trends in Databases, vol. 1, no. 4, pp. 379–474, 2009.
[23] C. Richardson, Microservices Patterns. Shelter Island, NY, USA: Manning Publications, 2018.
[24] R. T. Fielding, Architectural Styles and the Design of Network-Based Software Architectures, Ph.D. dissertation, University of California, Irvine, CA, USA, 2000.
[25] PostgreSQL Global Development Group, "PostgreSQL Documentation," Version 17, 2025. [Online]. Available: https://www.postgresql.org/docs/
[26] C. J. Date, An Introduction to Database Systems, 8th ed. Boston, MA, USA: Addison-Wesley, 2004.
[27] J. Kreps, I Heart Logs: Event Data, Stream Processing, and Data Integration. Sebastopol, CA, USA: O'Reilly Media, 2014.
[28] FastAPI, "FastAPI Documentation," 2025. [Online]. Available: https://fastapi.tiangolo.com
[29] P. Ammann and J. Offutt, Introduction to Software Testing, 2nd ed. Cambridge, U.K.: Cambridge University Press, 2016.
[30] M. Fowler, "Microservices Resource Guide." [Online]. Available: https://martinfowler.com/microservices/
[31] G. J. Myers, C. Sandler, and T. Badgett, The Art of Software Testing, 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, 2011.
[32] SQLite Consortium, "About SQLite," SQLite Documentation, 2025. [Online]. Available: https://www.sqlite.org