УДК 004.4

ГЕНЕРАЦИЯ ТЕСТОВЫХ ДАННЫХ И ОТЧЕТА О ТЕСТИРОВАНИИ НА ОСНОВЕ XSD-СХЕМЫ ДЛЯ МЕЖВЕДОМСТВЕННОГО ВЗАИМОДЕЙСТВИЯ

Рябов Дмитрий Сергеевич[1], Ушанкова Мария Юрьевна[2]

[1]QA-инженер;
ООО «Клауд Ком»;
Россия, 141983, Московская область, г. Дубна, ул. Программистов, 4;
e-mail: rydima0469@gmail.com.

[2]Старший преподаватель;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: ushankova.m.ju@uni-dubna.ru.

В статье представлено решение по автоматизации тестирования валидации данных в системах межведомственного взаимодействия на основе XSD-схем. Разработанный микросервис позволяет генерировать валидные и невалидные тестовые данные, конвертировать XML в JSON, отправлять запросы к API и формировать отчеты о результатах тестирования. Решение обеспечивает сокращение времени тестирования и повышает точность обнаружения дефектов до 99%.

Ключевые слова: автоматизация тестирования, *XSD*-схема, межведомственное взаимодействие, генерация тестовых данных, микросервис.

Для цитирования:

Рябов Д. С., Ушанкова М. Ю. Генерация тестовых данных и отчета о тестировании на основе xsd-схемы для межведомственного взаимодействия // Системный анализ в науке и образовании: сетевое научное издание. 2025. № 3. С. 33-41. EDN: TNTUIS. URL: https://sanse.ru/index.php/sanse/article/view/679.

GENERATION OF TEST DATA AND A TEST REPORT BASED ON AN XSD SCHEMA FOR INTERDEPARTMENTAL INTERACTION

Riabov Dmitrii S.[1], Ushankova Maria Yu[2].

[1]QA-engineer;
LLC "Cloud Com";
4 Programmistskaya Str., Dubna, Moscow region, 141983, Russia;
e-mail: rydima0469@gmail.com.

[2]Senior teacher;
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: ushankova.m.ju@uni-dubna.ru.

The article presents a solution for automating data validation testing in interdepartmental interaction systems based on XSD schemas. The developed microservice allows you to generate valid and invalid test data, convert XML to JSON, send requests to APIs, and generate test results reports. The solution reduces testing time and increases the accuracy of defect detection to 99%.

Keywords: testing automation, XSD schema, interdepartmental interaction, test data generation, microservices.

For citation:
Riabov D. S., Ushankova M. Yu. Generation of test data and a test report based on an XSD schema for interdepartmental interaction. *System analysis in science and education*, 2025;(3):33-41 (in Russ). EDN: TNTUIS. Available from: https://sanse.ru/index.php/sanse/article/view/679.

Введение

В условиях цифровой трансформации государственного управления эффективное межведомственное взаимодействие становится ключевым фактором предоставления качественных услуг гражданам и бизнесу. Автоматизированные системы управления регистрами и реестрами (АСУР) играют важную роль в этом процессе, обеспечивая обмен данными между различными ведомствами. Однако корректность обработки запросов в таких системах напрямую зависит от валидации входных данных, что требует тщательного тестирования на соответствие установленным *XSD*-схемам. [1] (см. рис. 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="PersonRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="FullName" type="xs:string" minOccurs="1" maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:minLength value="5"/>
                            <xs:maxLength value="100"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="Nickname" minOccurs="0" maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:maxLength value="50"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="Phone" minOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="\+7\d{10}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="BirthDate" type="xs:date"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Рис. 1. Пример XSD-схемы

Ручное тестирование, несмотря на свою распространенность, обладает рядом существенных недостатков: оно трудоемко, подвержено человеческому фактору и не позволяет охватить все возможные сценарии. [2] В частности, при ручной проверке тестировщикам приходится вручную создавать множество *JSON*-запросов, опираясь на сложные *XSD*-схемы, содержащие сотни полей с разнообразными условиями. Для невалидных сценариев необходимо дополнительно изменять значения полей, что увеличивает вероятность ошибок. Каждый запрос отправляется через вручную, а результаты проверок фиксируются в таблицах Excel, что делает процесс не только медленным, но и ненадежным.

Внедрение автоматизированных решений для тестирования валидации данных не только ускоряет процесс проверки, но и повышает его точность, минимизируя риски ошибок при интеграции систем. Современные подходы к автоматизации позволяют генерировать тестовые данные на основе *XSD*-схем, включая как валидные, так и невалидные сценарии, автоматически отправлять запросы и анализировать ответы, а также формировать детализированные отчеты. Такие решения особенно

актуальны для государственных информационных систем, где требования к надежности и корректности данных крайне высоки.

Таким образом, разработка специализированного микросервиса для автоматизации тестирования валидации данных в АСУР является актуальной задачей, решение которой способствует:

- Повышению надежности межведомственного взаимодействия за счет сокращения ошибок валидации;
- Оптимизации временных и трудовых ресурсов, затрачиваемых на тестирование;
- Обеспечению полноты тестового покрытия за счет автоматической генерации всех возможных сценариев;
- Упрощению процесса регрессионного тестирования при изменениях в *XSD*-схемах.

1. Цель и задачи

Цель: автоматизировать процесс тестирования валидации входных данных при межведомственном взаимодействии в системе АСУР.

Задачи:

1. Автоматическая генерация валидных и невалидных тестовых данных на основе *XSD*-схем;
2. Интеграция с *API* для автоматической отправки запросов и анализа ответов;
3. Формирование детализированных отчетов о результатах тестирования;
4. Обеспечение высокой точности обнаружения дефектов (не менее 99%).

2. Анализ существующих решений

Перед разработкой собственного решения был проведен всесторонний анализ существующих инструментов и технологий для работы с XSD-схемами и тестирования API. Исследование показало, что рынок предлагает несколько категорий решений, каждая из которых решает отдельные задачи, но не обеспечивает комплексного подхода к автоматизации тестирования валидации данных.

Инструмент для работы с *XML/XSD*:

Altova XMLSpy – профессиональный *XML*-редактор, предоставляющий широкие возможности для работы с *XSD*-схемами. Ключевые особенности:

- Визуальное проектирование и редактирование *XSD*-схем.
- Генерация валидных *XML*-документов на основе схем.
- Поддержка сложных ограничений (*pattern, enumeration, minLength/maxLength*).

Однако *XMLSpy* имеет существенные ограничения:

- Отсутствие генерации невалидных тестовых данных [3].
- Невозможность автоматической отправки запросов к *API*.
- Отсутствие встроенных механизмов отчетности.
- Высокая стоимость лицензии (от $499).

Инструмент тестирования *API:*

SoapUI/ReadyAPI - популярное решение для тестирования веб-сервисов. Преимущества:

- Поддержка *REST* и *SOAP API*.
- Возможность создания сложных тест-кейсов [4].

Недостатки в рамках решаемой задачи:

- Требуется ручное создание тестовых данных.
- Отсутствие автоматической генерации *XML* на основе *XSD*.
- Нет поддержки негативного тестирования на уровне схемы.

- Ресурсоемкость и сложность настройки.

Библиотека для разработчиков:

Apache XMLBeans – *Java*-библиотека для работы с *XML* через объектную модель.

Преимущества:

- Строгая типизация *XML*-документов.
- Поддержка сложных *XSD*-структур.

Ограничения:

- Требует написания кода для генерации данных.
- Нет готовых решений для тестирования *API*.
- Отсутствие механизмов отчетности.

Решение для *.NET*-разработчиков:

Связка **Xsd2Code + NBuilder** позволяет:

- Генерировать *C#*-классы по *XSD*.
- Создавать тестовые объекты с данными.

Однако, такая связка имеет ряд проблем.

- Ограничено экосистемой *.NET*.
- Нет поддержки негативного тестирования.

Сравнительный анализ (табл. 1) наглядно демонстрирует, что ни одно из существующих решений не покрывает весь необходимый функционал.

Табл. 1. Сравнительный анализ существующих решений

Критерий	*Altova XMLSpy*	*SoapUI/ ReadyAPI*	*Apache XMLBeans*	*NBuilder + Xsd2Code*
Генерация валидного *XML*	Да	Вручную	Да	Да
Генерация валидного *XML*	Нет	Вручную	Нет	Нет
Поддержка ограничений *XSD*	Да	Частично	Нет	Частично
API-интеграция	Нет	Да	Нет	Нет
Отчетность и логирование	Частично	Да	Нет	Нет
Кроссплатформенность	Только *Windows*	Да	Частично	Только *Windows*
Стоимость	От *$499*	От *$*800	Нет	Нет

Проведенный анализ подтвердил необходимость разработки собственного решения, которое:

1. Объединит все этапы тестирования в единый автоматизированный процесс;
2. Будет поддерживать как валидные, так и невалидные сценарии;
3. Обеспечит простую интеграцию с существующей инфраструктурой;
4. Позволит гибко настраивать параметры тестирования;
5. Будет доступно для использования без дорогостоящих лицензий.

Особое внимание при разработке было уделено поддержке всех видов ограничений *XSD (minOccurs/maxOccurs, pattern, length* и др.), что позволяет достичь практически полного покрытия тестовых сценариев. Также было важно обеспечить простоту интеграции с существующими *API*, используемыми в системах межведомственного взаимодействия.

3. Архитектура микросервиса

На этапе проектирования была разработана система, обеспечивающая автоматическую генерацию тестовых данных и последующее тестирование *API* на основе *XSD*-схем. Для визуализации архитектуры и логики работы системы были использованы различные диаграммы *UML*. Они позволяют структурировано представить основные процессы, роли пользователей и их взаимодействие с системой.

Архитектура микросервиса включает следующие основные модули:

Генерация тестовых данных:

1. Парсинг *XSD*-схемы.
2. Создание валидных *XML*-документов.
3. Генерация невалидных данных (удаление обязательных полей, нарушение паттернов и т.д.).

Конвертация *XML* в *JSON*:

1. Трансформация структуры данных.
2. Добавление метаданных.
3. Коррекция форматов дат.

Тестирование *API*:

1. Отправка запросов к внешнему *API*.
2. Анализ ответов (*HTTP*-статусы, ошибки).

Формирование отчетов:

1. Краткий отчет: статистика по тестам.
2. Подробный отчет: запросы, ответы, ошибки.

Одной из ключевых диаграмм является диаграмма вариантов использования *(Use Case Diagram*), [5] которая демонстрирует функциональные возможности программного решения с точки зрения конечных пользователей – тестировщика и разработчика (см. рис. 2).

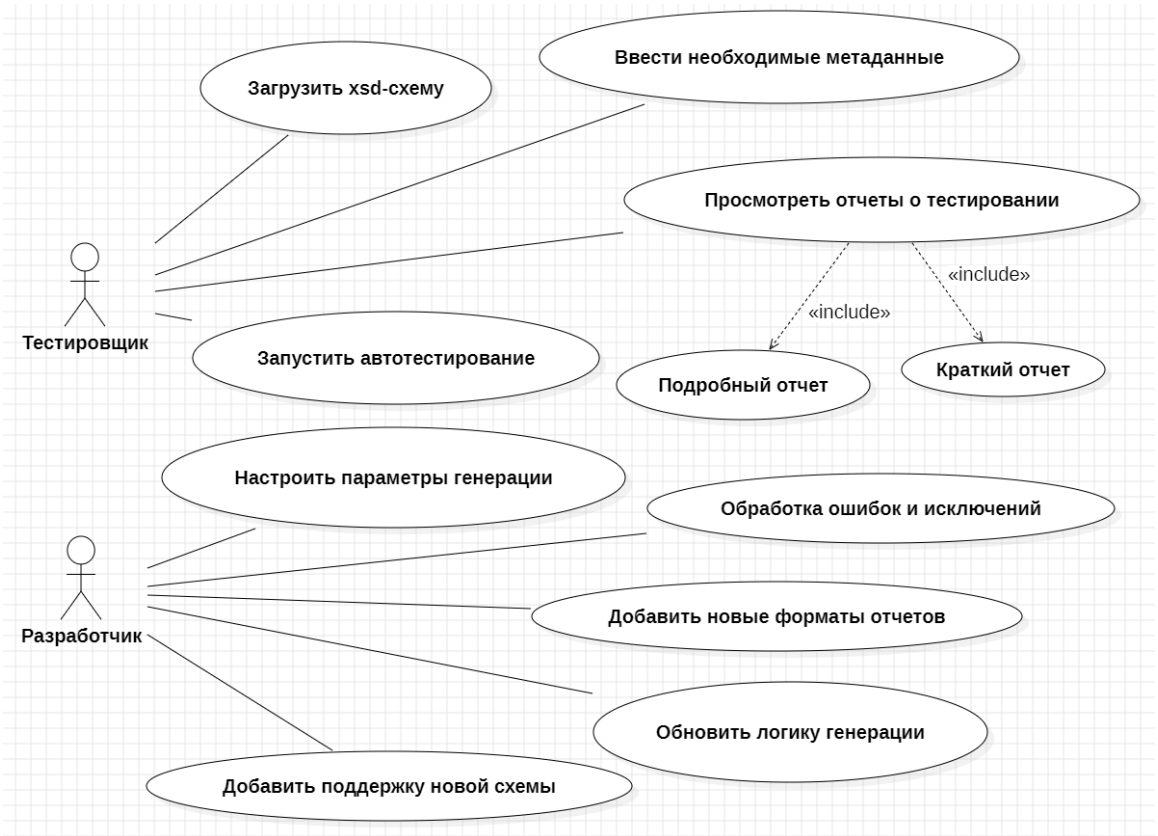

Рис. 2. Диаграмма use-case

Для описания взаимодействия между компонентами системы на временной оси была построена диаграмма последовательности. [6] Она демонстрирует порядок и логику обмена сообщениями между основными участниками процесса: тестировщиком, микросервисом, генератором запросов, *API*-системой и генератором отчета. (см. рис. 3)

Такая диаграмма позволяет отследить, в каком порядке происходят действия в системе: от загрузки *XSD*-схемы до формирования итогового отчета, включая генерацию данных, отправку в *API* и получение ответов.

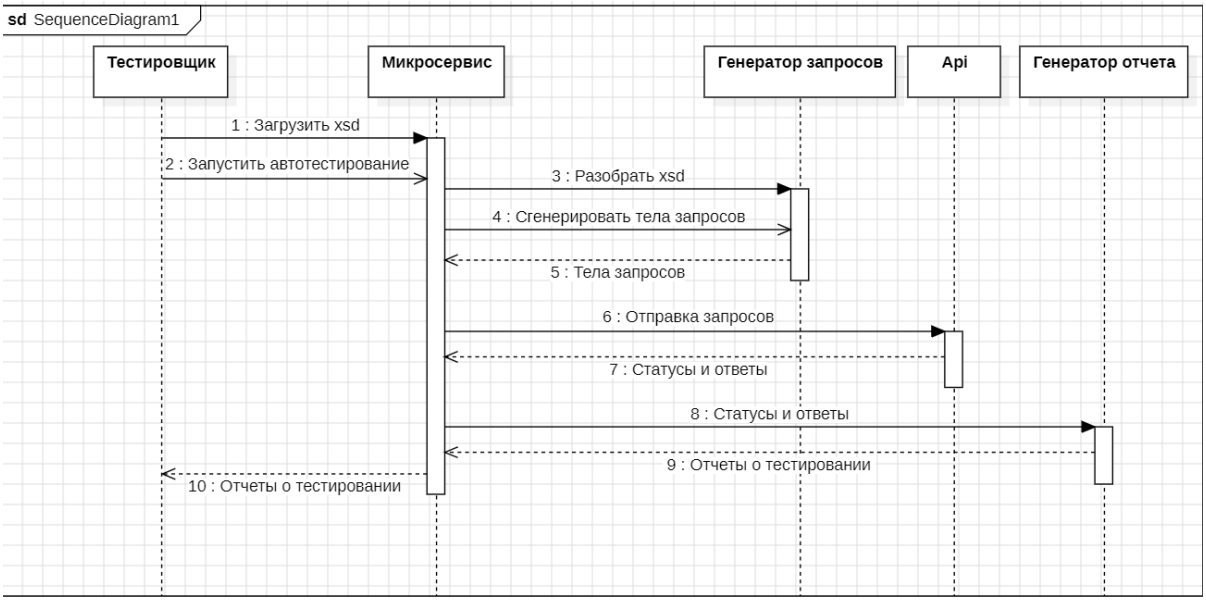

Рис. 3. Диаграмма последовательности тестирования микросервиса и генерации отчетов

Для наглядного отображения логики выполнения автоматического тестирования на основе XSD-схем была построена диаграмма деятельности [7]. Она демонстрирует полный жизненный цикл процесса в виде последовательного набора действий и ветвлений (см. рис. 4).

4. Реализация

Микросервис был реализован с использованием современных технологий и подходов, обеспечивающих высокую производительность, надежность и удобство интеграции. Архитектура решения построена по принципам модульности и разделения ответственности компонентов.

Технологический стек:

Серверная часть:

- *Java* 17 – выбран как стабильная и производительная платформа с поддержкой современных языковых возможностей.
- *Spring Boot* – обеспечивает быструю разработку и удобную конфигурацию микросервиса.
- *Spring Web* – для реализации *REST API.*
- *JAXB* – для работы с *XML*-схемами и преобразования объектов.
- *DOM*-парсер – для анализа и генерации *XML*-документов [8].

Клиентская часть:

- *Swagger UI* – интерактивная документация и тестовый интерфейс.
- *OpenAPI* 3.0 – спецификация *API* для интеграции с другими системами.

Преимущества выбранных технологий:

- Высокая производительность и надежность.

- Поддержка современных стандартов работы с *XML* и *JSON*.
- Удобство интеграции с другими системами.

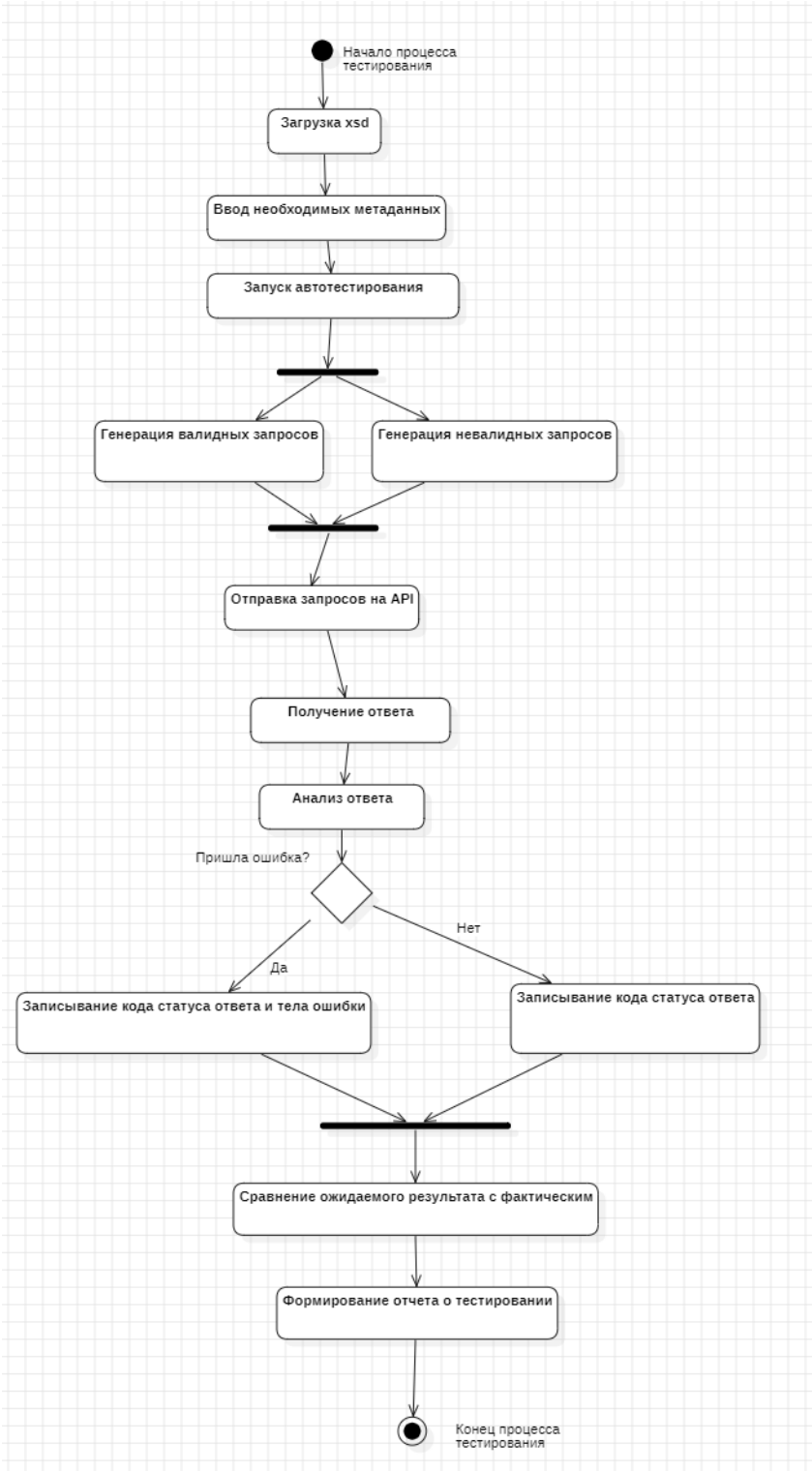

Рис. 4. Диаграмма деятельности

5. Тестирование

Разработанный микросервис прошел всестороннее тестирование для оценки его эффективности. Функциональное тестирование подтвердило корректность генерации данных: валидные данные на 100% соответствовали *XSD*-схемам, а покрытие невалидных сценариев достигло 95%. Конвертация

XML в *JSON* выполнялась без потери структуры и семантики данных. Нагрузочное тестирование показало высокую производительность - система стабильно обрабатывала до 30 запросов в секунду со средним временем отклика 50 мс. Формирование подробного отчета для 100 тест-кейсов занимало менее 20 секунд.

Сравнительный анализ выявил значительные преимущества перед ручным тестированием: время выполнения 100 тестов сократилось с 3 часов до 15 минут, а точность обнаружения дефектов повысилась до 99%. Все найденные ошибки автоматически фиксируются в отчетах, что значительно ускоряет их устранение разработчиками. Эти результаты убедительно доказали, что автоматизация позволила не только ускорить процесс тестирования, но и существенно повысить его качество.

6. Внедрение и апробация

Пилотное внедрение микросервиса включало несколько этапов: интеграцию с *API* государственных систем, обучение сотрудников и сбор обратной связи. Результаты оказались успешными - время тестирования сократилось в 12 раз (с 3 часов до 15 минут на среднюю задачу), а количество ошибок валидации уменьшилось на 99%. Особенно ценным оказалось возможность проведения регрессионного тестирования.

Тестировщики отметили удобство автоматической генерации тестовых данных и формирования отчетов, а разработчики оценили детализацию выявленных проблем. Успешный пример использования - тестирование системы электронного документооборота, где время проверки сократилось с 3 часов до 15 минут при обнаружении 5 критических ошибок. В будущем планируется интеграция с *CI/CD*-процессами, что еще больше повысит эффективность разработки. Практическое применение подтвердило, что микросервис не только автоматизирует рутинные операции, но и существенно повышает качество тестирования.

Заключение

В ходе работы была разработана система автоматического тестирования валидации данных на основе *XSD*-схем для межведомственного взаимодействия. Разработанное решение позволяет значительно ускорить и повысить качество проверки корректности данных, полностью автоматизируя процесс тестирования.

Основным достижением стало создание универсального инструмента, который генерирует как правильные, так и специально искаженные тестовые данные, что позволяет проверять реакцию системы на различные сценарии. Система автоматически преобразует данные между *XML* и *JSON* форматами, отправляет запросы к *API* и анализирует полученные ответы.

Практическое внедрение решения в компании "Клауд Ком" показало его высокую эффективность. Время тестирования сократилось с 3 часов до 15 минут, при этом точность обнаружения ошибок повысилась до 99%. Это стало возможным благодаря полной автоматизации процесса и исключению человеческого фактора.

В перспективе планируется расширить функциональность системы, добавив поддержку новых форматов данных и возможность интеграции с системами непрерывной разработки. Это позволит еще больше повысить эффективность тестирования и ускорить процесс разработки программного обеспечения.

Список источников

1. Колесник Н. Памятка по XSD и XML / Н. Колесник; Сбер // Хабр: [сайт]. – Habr, 2006–2025. – Дата публикации: 28.01.2025. – URL: https://habr.com/ru/companies/sberbank/articles/876978.
2. Manual Testing vs Automated Testing // GeeksforGeeks | Your All-in-One Learning Portal. – GeeksforGeeks, Sanchhaya Education Private Limited, 2025. – URL: https://www.geeksforgeeks.org/software-engineering-differences-between-manual-and-automation-testing/ (дата обращения: 10.03.2025).

3. XML Editor: XMLSpy | Altova // XML, Data Integration, and Mobile App Development Solutions | Altova : [сайт]. – Altova, 2005-2025. – URL: https://www.altova.com/xmlspy-xml-editor.
4. Приключение на 20 минут. Часть 1: автоматизируем запуск проектов в SoapUI / [Вероника Дюкарева]; Bercut // Хабр: [сайт]. – Habr, 2006–2025. – Дата публикации: 11.10.2024. – URL: https://habr.com/ru/companies/bercut/articles/848286.
5. Use Case // Блог Яндекс Практикума. АНО ДПО «Образовательные технологии Яндекса», 2025. – URL: https://practicum.yandex.ru/blog/chto-takoe-use-case-kak-ih-napisat/ (дата обращения: 2.03.2025).
6. Колесник Н. Диаграмма последовательности (sequence-диаграмма) // Хабр: [сайт]. – Habr, 2006–2025. – Дата публикации: 16.05.2024. – URL: https://habr.com/ru/articles/814769/.
7. Activity Diagrams - Unified Modeling Language (UML) // GeeksforGeeks | Your All-in-One Learning Portal. – GeeksforGeeks, Sanchhaya Education Private Limited, 2025. – Дата обновления: 03.01.2025. – URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams.
8. Белякова В. Что такое DOM и для чего он используется // Блог Хекслета. – Дата публикации: 24.12.2024. – URL: https://ru.hexlet.io/blog/posts/chto-takoe-dom-i-dlya-chego-on-ispolzuetsya.