Проблемы искусственного интеллекта. 2026. N 3 (42). С. 107-121
Problems of Artificial Intelligence. 2026;3(42):107-121.
Системный анализ, управление и обработка информации, статистика
Научная статья

УДК 004.415.53+346.544.44 **DOI** 10.24412/2413-7383-2026-3-42-107-121

А.А. Вартанян[1], Е.В. Курнасов[2], А.В. Атаманчук[2], М.Р. Забиров[3]
[1]ООО «ВЕКАС»
111020, г. Москва, ул. Синичкина 2-я, д. 9а, стр. 10
[2]МИРЭА – Российский технологический университет,
119454, г. Москва, Проспект Вернадского, д. 78
[3]ООО «МПС Софт»
105066, г. Москва, Нижняя Красносельская, 40/12к2

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА СОПРОВОЖДЕНИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОБОРУДОВАНИЯ ЦИФРОВОЙ МАРКИРОВКИ

A.A. Vartanyan[1], E.V. Kurnasov[2], A.V. Atamanchuk[2], M.R. Zabirov[3]
[1]VEKAS LLC
111020, Moscow, st. Sinichkina 2-ya, 9a, building 10
[2]MIREA – Russian Technological University,
119454, Moscow, Vernadsky avenue., building. 78
[3]MPS Soft LLC
105066, Moscow, Nizhnyaya Krasnoselskaya, 40/12k2

INTELLIGENT SOFTWARE TESTING SUPPORT SYSTEM FOR DIGITAL MARKING EQUIPMENT

В статье представлена интеллектуальная система сопровождения тестирования программного обеспечения оборудования цифровой маркировки. Взаимодействие с системой осуществляется через веб-приложение. Система автоматически формирует черновики тест-кейсов из технических заданий с использованием больших языковых моделей (LLM), поддерживает ревью требований, контекстное усиление промптов (RAG) на основе векторного хранилища, парсинг документов форматов DOCX и PDF, валидацию результатов, интеграцию с Confluence и GitLab. Экспериментальная оценка на контрольном наборе из 18 документов показала сокращение времени подготовки черновика тест-кейсов, пригодного к экспертной доработке, примерно на 75 % по сравнению с ручным baseline. Система приводит структуру тест-кейсов к единому формату; надёжность конвейера обеспечивается механизмом повторных попыток и контролируемой деградацией при недоступности RAG.
Ключевые слова: автоматизация тестирования, генерация тест-кейсов, сопровождение тестирования, большие языковые модели, RAG, оборудование цифровой маркировки, веб-приложение, Confluence.

The article presents an intelligent testing support system for digital marking equipment software, accessed via a web application. It generates draft test cases from technical specifications using large language models (LLMs), supports requirements review, retrieval-augmented prompting (RAG) with a vector store, parsing of DOCX and PDF documents, result validation, and integration with Confluence and GitLab. On a control set of 18 documents, the average time to obtain a review-ready draft was reduced by approximately 75 % compared with a manual baseline. The system standardizes test case structure; pipeline reliability is supported by retries and controlled degradation when RAG is unavailable.
Key words: test automation, test case generation, testing support, large language models, RAG, digital labeling equipment, web application, Confluence.

Введение

В современной разработке программного обеспечения для оборудования цифровой маркировки процесс создания тест-кейсов является важным этапом обеспечения качества [1], [2]. Оборудование цифровой маркировки представляет собой специализированные устройства и программные комплексы, предназначенные для нанесения и проверки уникальных кодов маркировки на товары в соответствии с требованиями законодательства. Такое оборудование должно обеспечивать надежную работу с кодами Data Matrix, QR-кодами и другими форматами маркировки, а также интеграцию с государственными информационными системами, такими как система маркировки «Честный ЗНАК» [3-5].

Тест-кейс – это документированный набор условий и действий, выполняемых для проверки конкретной функциональности программного обеспечения [2], [6]. Каждый тест-кейс содержит описание предварительных условий, последовательность шагов выполнения и ожидаемый результат. Однако ручное создание тест-кейсов на основе технических заданий требует много времени и усилий от QA-инженеров. При ручном создании тест-кейсов специалист должен вручную анализировать техническое задание, формировать тестовые сценарии и описывать ожидаемые результаты [2], [7]. Это трудоемкий процесс, который требует глубокого понимания требований и опыта в области тестирования.

Ситуация осложняется тем, что темпы разработки программного обеспечения опережают скорость создания тестовой документации. Современные методологии разработки, такие как Agile и DevOps, предполагают быстрые итерации и частые релизы, что требует соответствующей скорости подготовки тестовой документации. Ручной метод может становиться узким местом при высоких темпах разработки, что может приводить к отставанию тестирования от разработки. Кроме того, ручной метод подвержен субъективным ошибкам и может привести к неполному покрытию требований тестами [2], [6]. Также, разработчики активно используют ИИ-ассистентов для генерации кода, тогда как практика применения аналогичных инструментов для генерации тест-кейсов и тестовой документации остается недостаточно развитой [8], [9].

Актуальность данной работы обусловлена необходимостью автоматизации процесса генерации тест-кейсов с использованием современных технологий искусственного интеллекта [8-11]. Современные большие языковые модели (LLM), разрабатываемые такими компаниями, как OpenAI и Anthropic, демонстрируют способность к анализу текстовых документов, интерпретации требований и генерации структурированных артефактов, включая тест-кейсы [8]. Использование LLM представляется особенно актуальным при разработке и тестировании программного обеспечения оборудования цифровой маркировки, поскольку данная предметная область характеризуется высокой регламентированностью и частыми изменениями нормативных требований. Тестирование таких систем включает проверку многочисленных состояний жизненного цикла кодов маркировки и интеграцию с внешними государственными информационными системами. Объём и вариативность требований создают значительную нагрузку на процесс подготовки тестовой документации, что делает применение LLM для анализа требований и генерации тест-кейсов обоснованным направлением автоматизации.

В данной работе предлагается интеллектуальная система сопровождения тестирования, включающая автоматизированную генерацию тест-кейсов для программного обеспечения оборудования цифровой маркировки. Система использует большие языковые модели для парсинга технических заданий (DOCX, PDF), генерации и ревью тест-кейсов, контекстного усиления промптов (RAG) и интеграции с Confluence и GitLab.

Анализ существующих методов и их ограничений

Традиционный подход к созданию тест-кейсов основан на ручном методе [2], [6]. QA-инженер вручную анализирует текстовые требования, выделяет сценарии и описывает шаги выполнения тестов. Основными проблемами ручного метода являются субъективность интерпретации требований, низкая скорость создания тест-кейсов и их высокая стоимость. Разные специалисты могут по-разному понимать одни и те же требования, что приводит к созданию различных наборов тест-кейсов для идентичных требований.

Современные подходы автоматизации проектирования тестов, такие как model-based и template-based, преимущественно опираются на формальные модели или заранее структурированные входные данные [2], [6]. Однако в реальных проектах требования часто представлены в виде технических заданий, содержащих локальные термины, исключения и неявные зависимости. Это ограничивает применимость традиционных методов генерации тест-кейсов. В отличие от перечисленных подходов, большие языковые модели способны анализировать неструктурированный текст требований и извлекать из него логические условия, бизнес-правила и граничные сценарии [8]. Однако использование LLM в «сыром» виде не обеспечивает управляемости, воспроизводимости и контроля качества генерируемых тест-кейсов [9].

Описание предлагаемого решения

Целевое назначение интеллектуальной системы – обработка неструктурированных или слабо формализованных требований, представленных в текстовом виде, их семантического анализа с использованием больших языковых моделей и формирования структурированного набора тест-кейсов, пригодных для дальнейшей экспертной валидации и использования в процессах тестирования.

Система функционирует как надстройка над большими языковыми моделями, обеспечивая управление процессом генерации, структурирование входных данных, нормализацию выходного формата и контроль воспроизводимости результатов. Это позволяет снизить трудозатраты на подготовку тестовой документации при сохранении необходимости экспертной проверки и утверждения сгенерированных тест-кейсов.

Система сопровождения тестирования реализует последовательный процесс обработки требований, начиная с загрузки технического задания и заканчивая формированием черновиков тест-кейсов, пригодных для использования в процессах тестирования и документооборота. Модель системы объединяет описание технологического процесса, функциональную декомпозицию (нотация IDEF0) (в соответствии с рисунком 1) и модель потоков данных (нотация DFD) (в соответствии с рисунком 2), что позволяет рассмотреть систему одновременно с точки зрения выполняемых функций и обрабатываемой информации.

На рисунках 1–2 отражён базовый сценарий генерации тест-кейсов (A1–A5). Ревью технического задания, контекстное усиление промптов (RAG) и сценарии взаимодействия с GitLab реализованы в том же веб-приложении как дополнительные функции контура сопровождения тестирования.

Исходной точкой процесса является взаимодействие с внешней сущностью – QA-инженером, который загружает техническое задание в форматах DOCX или PDF через веб-интерфейс. Документ регистрируется в системе как проект и проходит первичную валидацию формата и ограничений по размеру. На данном этапе формируются и сохраняются метаданные проекта в хранилище Projects.

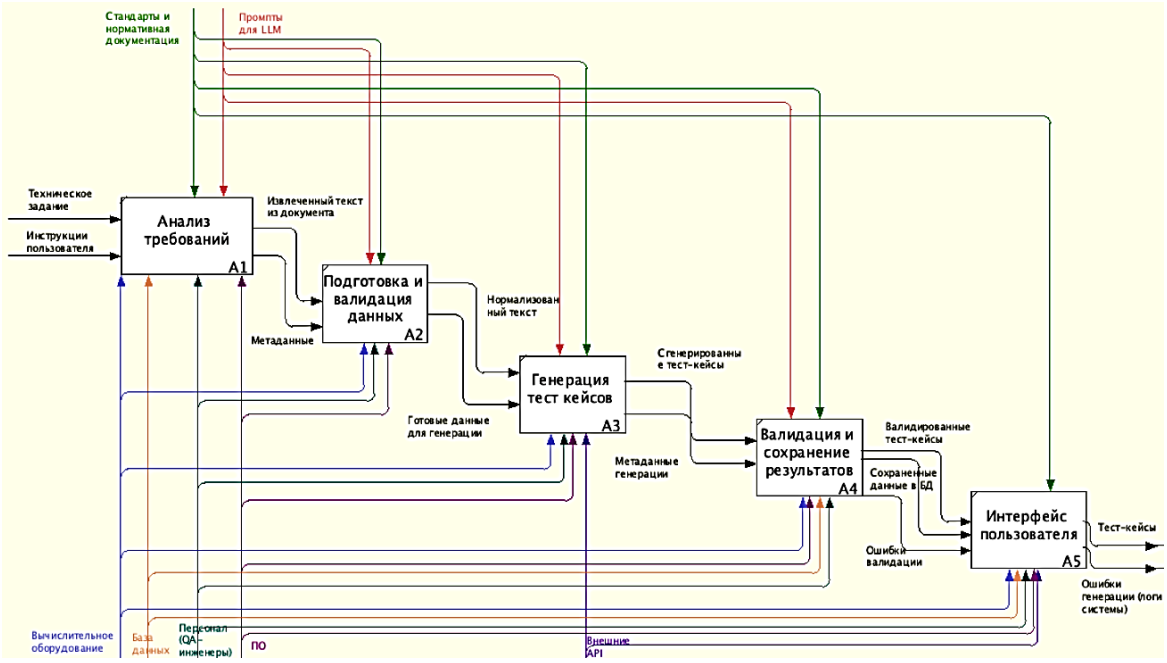


Рисунок 1 – Контекстная диаграмма интеллектуальной системы сопровождения тестирования

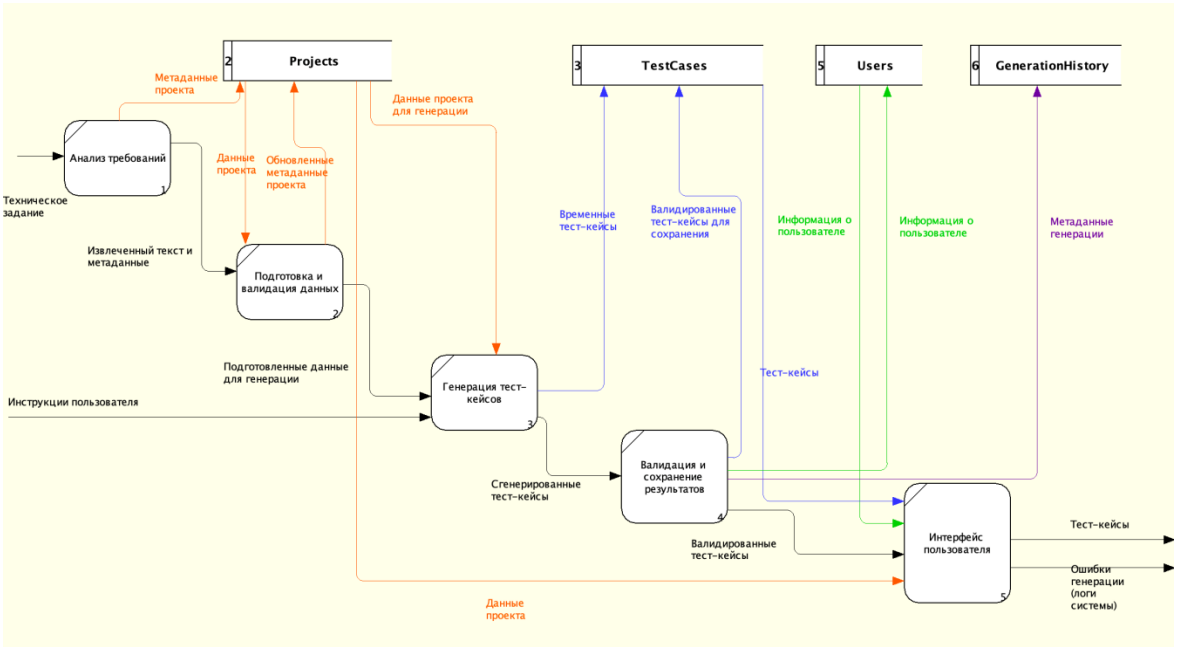


Рисунок 2 – Диаграмма потоков данных интеллектуальной системы сопровождения тестирования

С функциональной точки зрения данный этап соответствует функции A1 «Приём и регистрация технического задания» в модели IDEF0. Результатом выполнения функции является извлечённый текст документа и набор метаданных, передаваемых далее по процессу обработки.

Следующий этап (A2 «Подготовка данных») включает нормализацию структуры, проверку пригодности требований для генерации тест-кейсов. При необходимости выполняется разбиение документа на смысловые блоки для обеспечения корректной обработки большой языковой моделью. Подготовленные данные сохраняются в рамках проекта и передаются в функцию генерации.

Ключевым элементом системы является функция А3 «Генерация тест-кейсов», реализующая взаимодействие с внешней сущностью LLM API. Подготовленный текст требований и пользовательские инструкции используются для формирования запроса к большой языковой модели. В результате система получает структурированные тест-кейсы, включающие позитивные, негативные и граничные сценарии, отражающие специфику программного обеспечения оборудования цифровой маркировки.

С точки зрения DFD на данном этапе происходит обмен данными между внутренним процессом генерации и внешней сущностью LLM API. Сгенерированные тест-кейсы передаются на этап валидации (А4).

Функция А4 «Валидация и сохранение результатов» обеспечивает проверку корректности структуры тест-кейсов и их соответствия установленной схеме. Только тест-кейсы, успешно прошедшие проверку, сохраняются в хранилище TestCases с привязкой к соответствующему проекту. Одновременно обновляется информация в хранилище Users и фиксируются параметры генерации в GenerationHistory, что обеспечивает прослеживаемость и аудит использования LLM.

Заключительный этап (А5 «Представление и экспорт результатов») реализует взаимодействие с пользователем и внешними системами. Сохранённые тест-кейсы отображаются в веб-интерфейсе с возможностью фильтрации и анализа. По запросу пользователя выполняется экспорт во внешнюю систему документооборота через Confluence API либо выгрузка в формат CSV. На уровне DFD данный этап отражает поток данных из внутренних хранилищ во внешнюю сущность Confluence API.

Сформированные тест-кейсы отображаются в веб-интерфейсе в едином формате: для каждого сценария указываются тип (позитивный, негативный, граничный), шаги выполнения и ожидаемый результат; предусмотрены фильтрация по типу и просмотр записей перед экспертной доработкой и экспортом (в соответствии с рисунком 3).

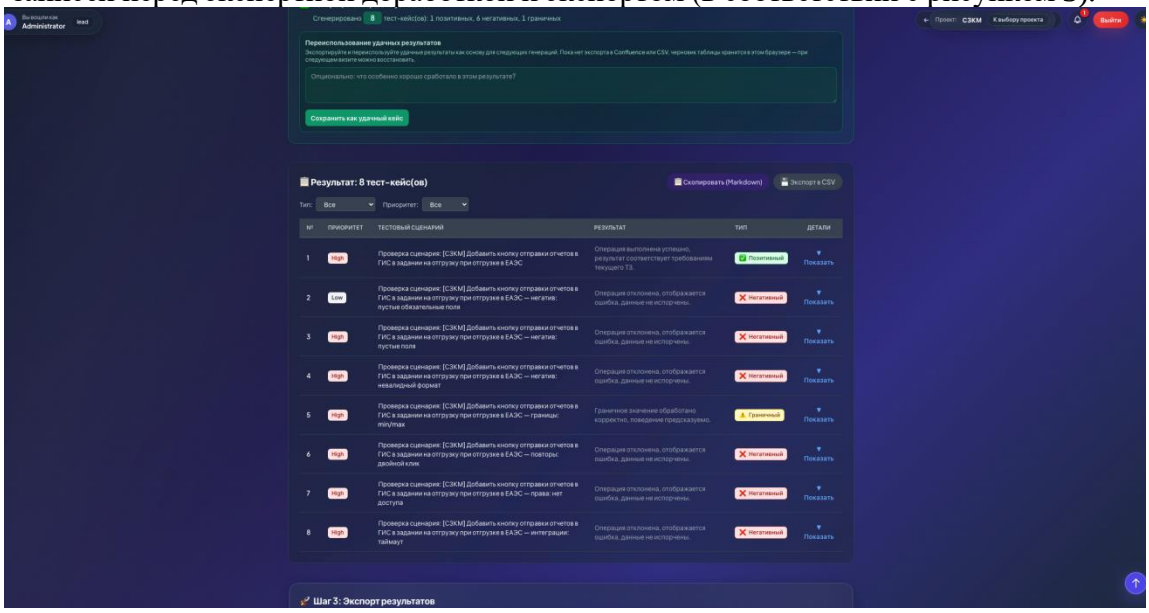


Рисунок 3 – Интерфейс подсистемы генерации тест-кейсов: отображение и фильтрация сформированных сценариев

Подсистема ревью технического задания выполняет анализ полноты и непротиворечивости требований и формирует рекомендации по уточнению пользовательского промпта для последующей генерации тест-кейсов; результаты анализа выводятся в отдельной вкладке веб-приложения (в соответствии с рисунком 4).

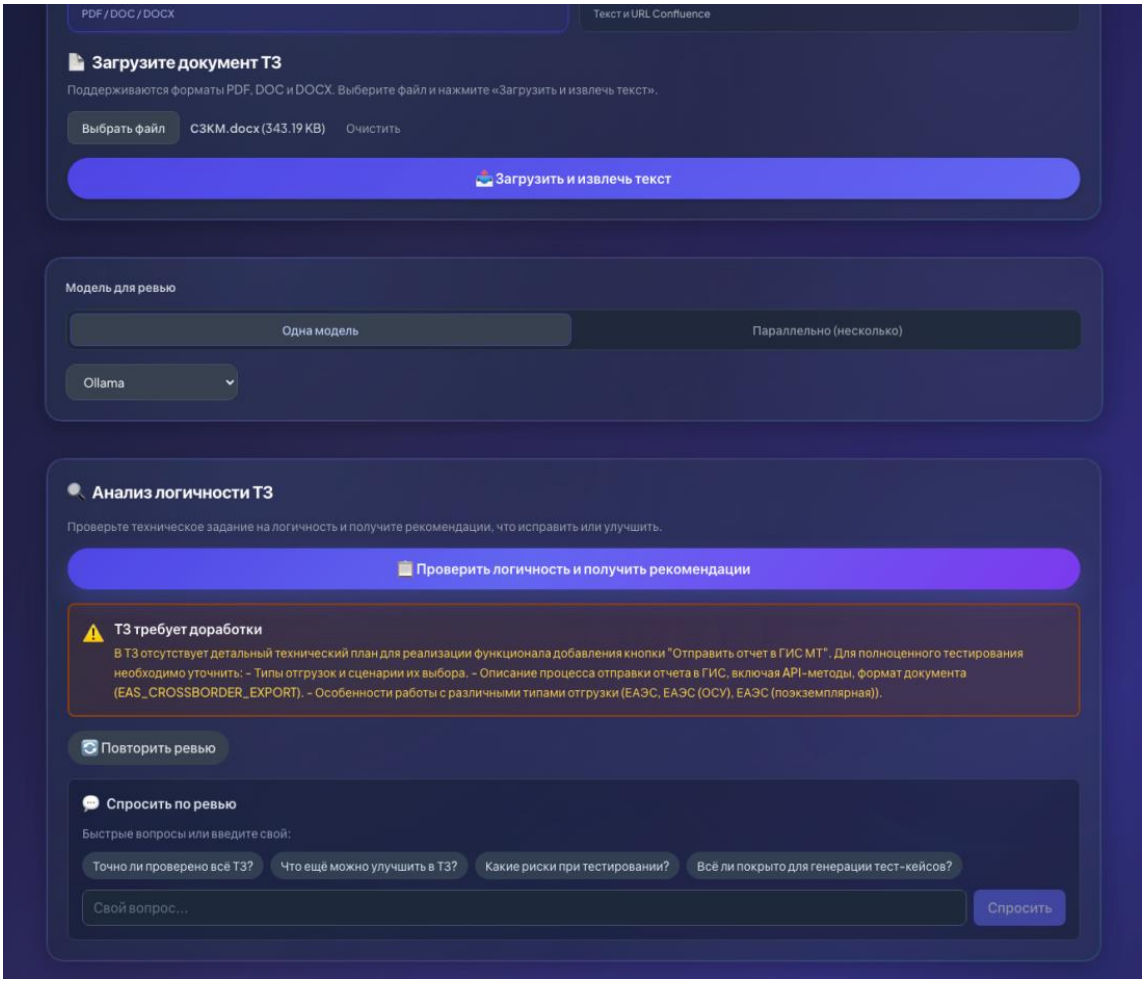


Рисунок 4 – Интерфейс ревью технического задания: результаты анализа и рекомендации

Дополнительно в контуре сопровождения реализованы контекстный поиск по векторному хранилищу (RAG) при генерации и сценарии взаимодействия с GitLab для анализа автотестов. Указанные средства не заменяют экспертную приёмку, но сокращают подготовительный этап.

Сценарий сопровождения автотестов реализован во вкладке «Автотесты»: пользователь задаёт параметры подключения к репозиторию GitLab, запускает анализ и получает рекомендации по покрытию и доработке тестовых сценариев (в соответствии с рисунком 5).

Таким образом, интеллектуальная система сопровождения тестирования представляет собой интегрированную функционально-информационную структуру, в которой технологический процесс генерации тест-кейсов реализован как последовательность взаимосвязанных функций, обменивающихся данными через внутренние хранилища и взаимодействующих с внешними сервисами.

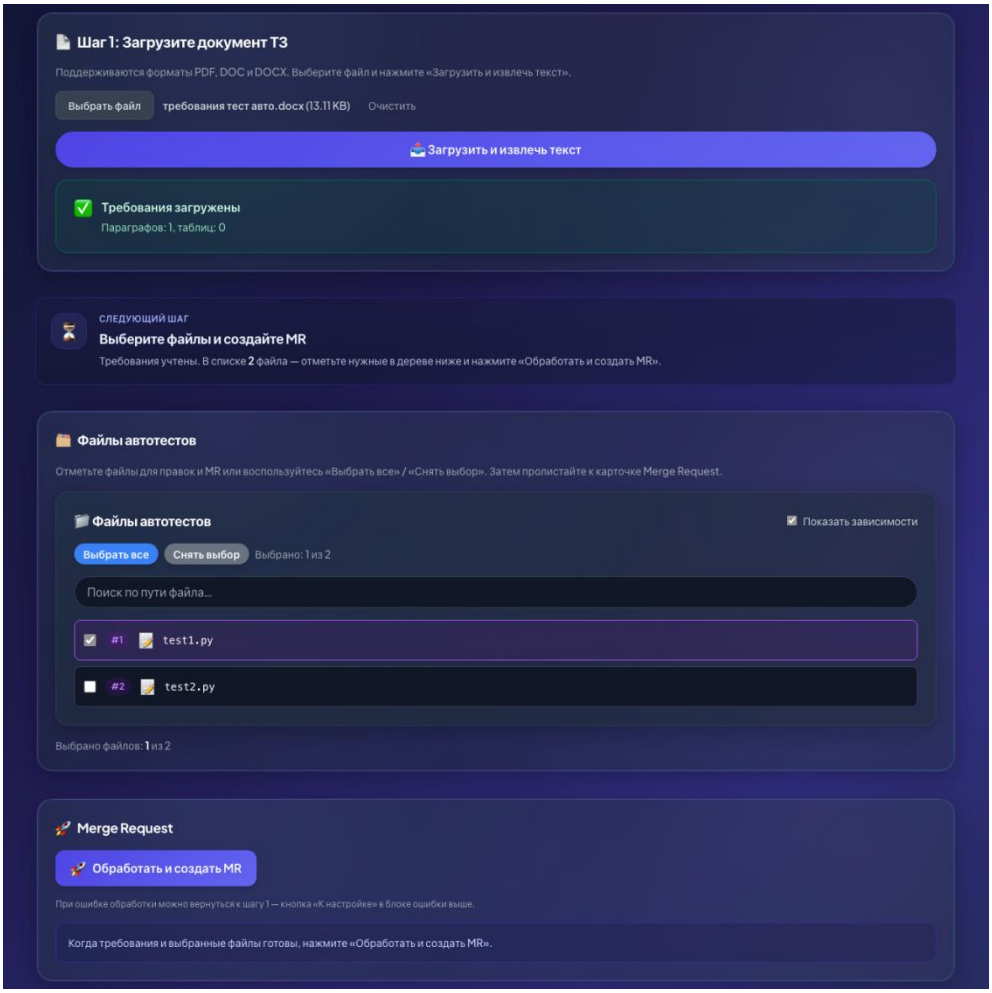


Рисунок 5 – Интерфейс подсистемы анализа автотестов: запуск анализа репозитория GitLab

Архитектура предлагаемого решения

Архитектура интеллектуальной системы сопровождения тестирования построена по принципу трехуровневой модели, которая обеспечивает разделение ответственности между компонентами и упрощает разработку, тестирование и поддержку системы [12].

Первый уровень – уровень представления (Presentation Layer), реализованный как React приложение с компонентами для загрузки документов, ввода инструкций, отображения тест-кейсов и экспорта. React – это популярная JavaScript-библиотека для создания пользовательских интерфейсов, которая позволяет создавать интерактивные веб-приложения с компонентной архитектурой.

Второй уровень – уровень бизнес-логики (Business Logic Layer), реализованный как FastAPI-приложение с сервисами парсинга документов, генерации и ревью тест-кейсов, RAG, работы с БД и интеграций. FastAPI – современный веб-фреймворк для Python, который обеспечивает высокую производительность за счет асинхронной обработки запросов.

Третий уровень – уровень данных (Data Layer), включающий реляционную БД SQLite, векторное хранилище ChromaDB (эмбеддинги и RAG) и внешние API (провайдеры LLM, Confluence и GitLab). SQLite – это встраиваемая реляционная база данных, которая хранит данные в файле и не требует отдельного сервера, что упрощает развертывание системы (в соответствии с рисунком 6).

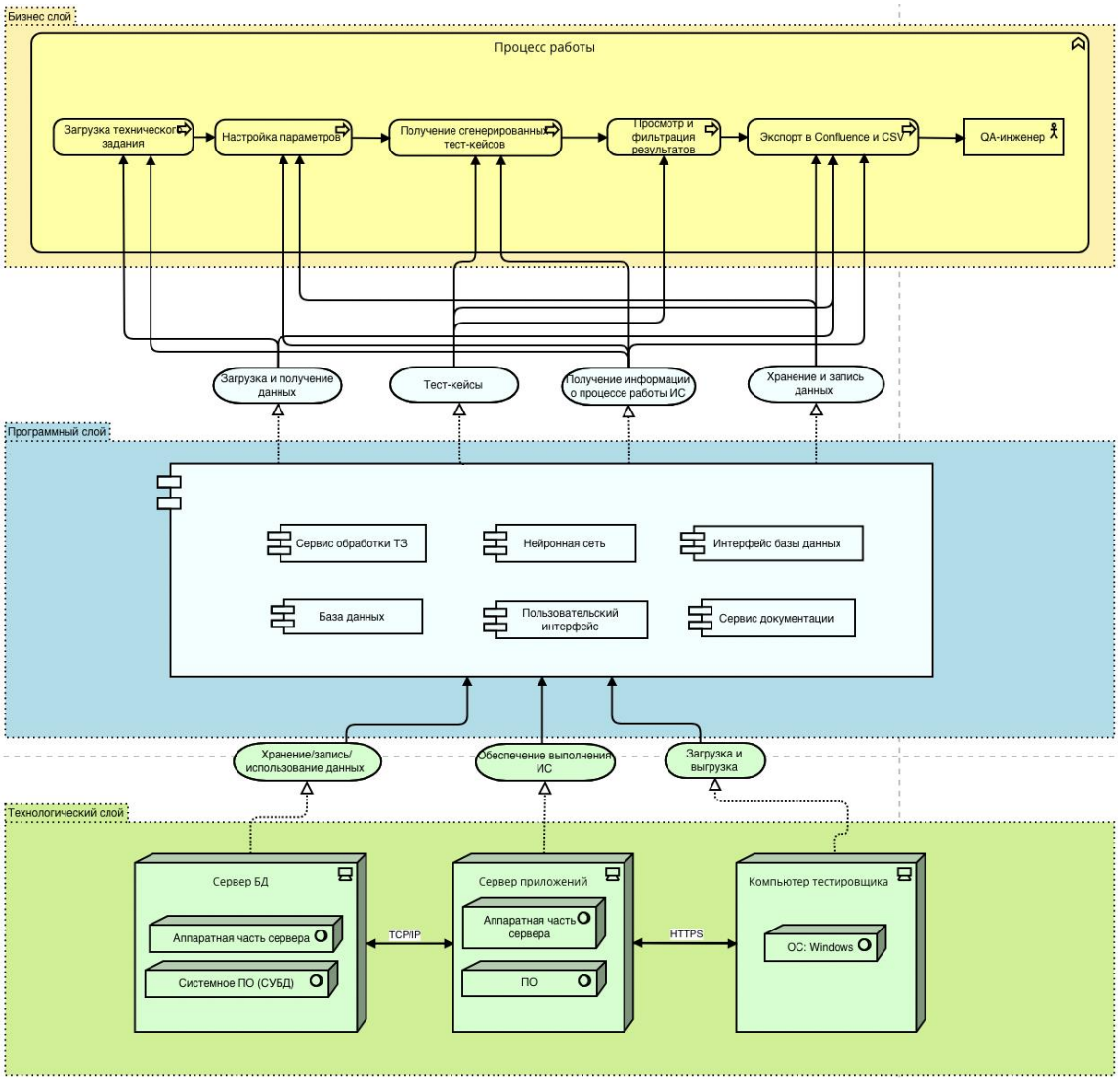


Рисунок 6 – Архитектура интеллектуальной системы

Организация хранения данных в системе

База данных системы построена на основе реляционной модели и включает шесть основных таблиц (в соответствии с рисунком 7), обеспечивающих хранение и управление данными о пользователях, проектах тестирования, тест-кейсах, истории генераций, экспортах в Confluence и результатах прохождения тест-кейсов [12].

Таблица «users» предназначена для управления пользователями системы (QA-инженерами). Хранит учётные данные (логин, email, хешированный пароль), роль пользователя (admin, lead, tester и др.) и агрегированную статистику (созданные тест-кейсы, выполненные прогоны, зафиксированные дефекты).

Таблица «projects» служит для организации тест-кейсов по проектам тестирования, соответствующим функциональным модулям ПО оборудования цифровой маркировки. Каждый проект содержит метаданные загруженного технического задания (имя файла, извлеченный текст), дополнительные инструкции пользователя для кастомизации генерации, а также статистику по тест-кейсам (общее количество, пройденные, проваленные) и статус проекта.

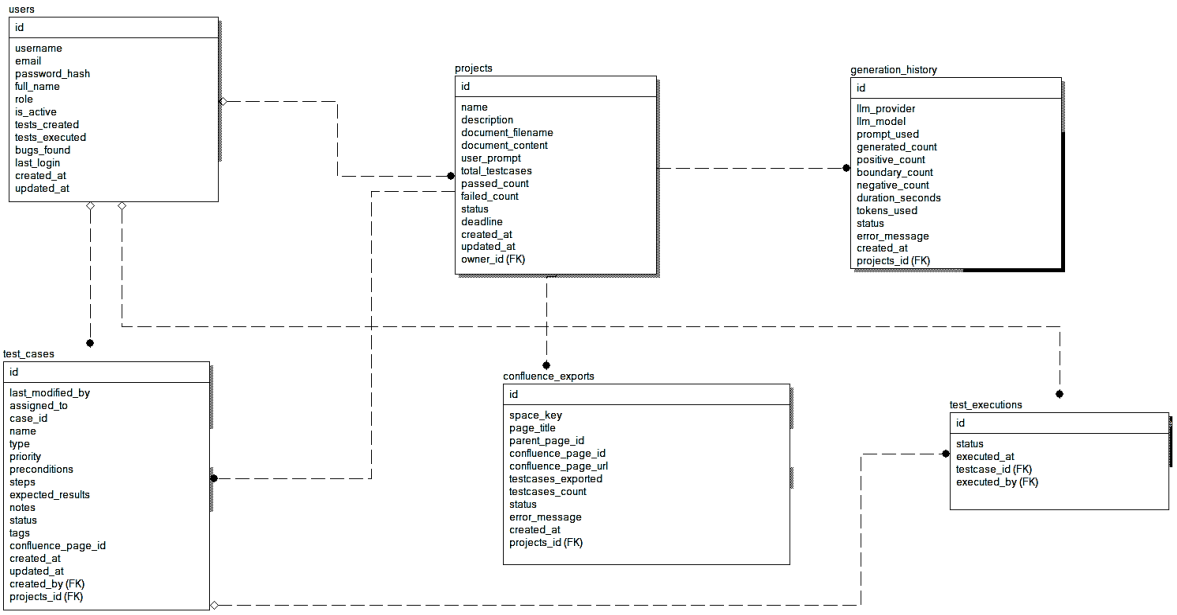


Рисунок 7 – Структурная схема базы данных

Таблица «test_cases» является центральной таблицей системы и хранит все сгенерированные тест-кейсы. Каждый тест-кейс содержит идентификатор (case_id), название, тип (позитивный, негативный, граничный), приоритет, предварительные условия, шаги выполнения (в формате JSON), ожидаемый результат и статус выполнения.

Таблица «generation_history» фиксирует метаданные о каждом процессе генерации: LLM-провайдер и модель, использованный промпт, количество сгенерированных тест-кейсов по типам, длительность, объём токенов (при наличии), оценочную стоимость запроса и статус выполнения.

Таблица «confluence_exports» хранит историю экспортов тест-кейсов в систему документооборота Confluence. Для каждого экспорта сохраняются данные о созданной странице (ключ пространства, название, URL), количество экспортированных тест-кейсов и статус операции.

Таблица «test_executions» предназначена для фиксации результатов ручного или регламентированного прохождения тест-кейсов (статус, фактический результат, среда, пояснения при несоответствии), а также связи с тест-кейсом и исполнителем.

Для решения задач семантического связывания исходного текста технического задания с доменным контекстом и накопленным опытом предыдущих запусков, в интеллектуальной системе предусмотрено векторное хранилище. В таблице 1 приведены коллекции векторного хранилища, используемые системой.

Таблица 1 – Коллекции векторного хранилища

Коллекция	**Содержимое**	**Режим использования**
semantic_model	термины, определения, правила	постоянный контекст генерации
learning_feedback	успешные кейсы и обратная связь	контекстное улучшение качества
documents	чанки текущего документа	временная индексация/поиск/очистка

Управление векторным контуром хранения данных осуществляется через разработанный административный интерфейс веб-приложения (для ролей admin и lead), который позволяет добавлять текстовые чанки в коллекцию semantic_model, просматривать сохранённые элементы и выполнять фильтрацию по источнику данных. Такой подход упрощает актуализацию доменного контекста без прямого обращения к служебным API и повышает удобство сопровождения системы в процессе эксплуатации (в соответствии с рисунком 8).

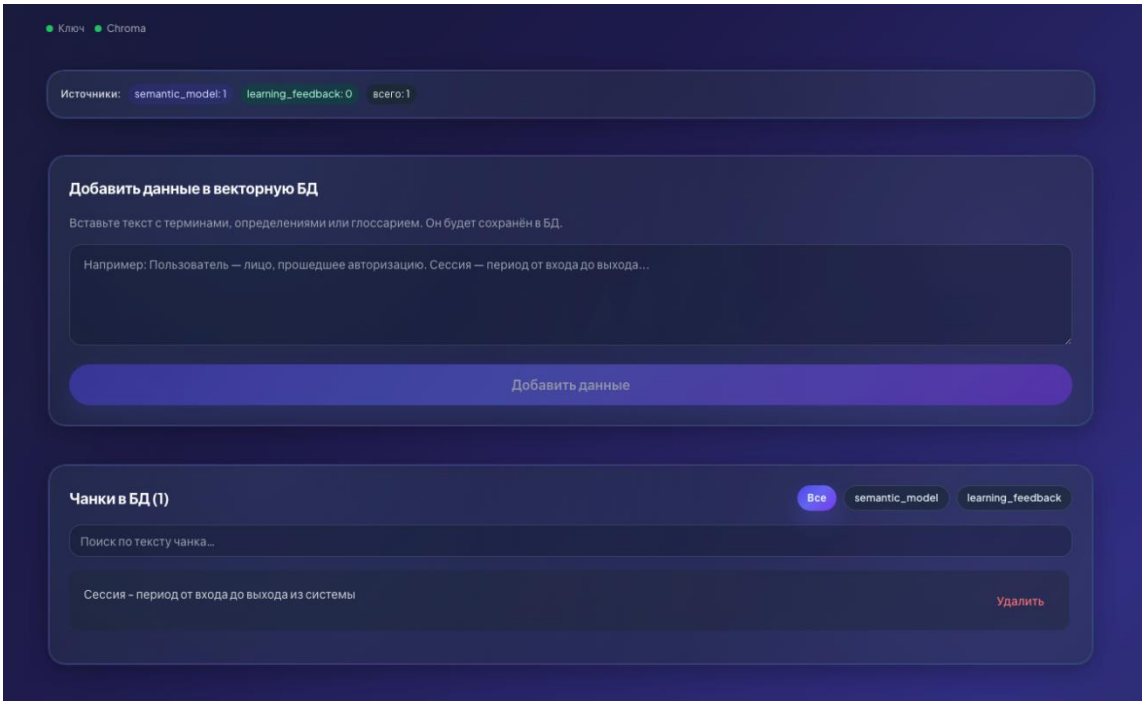


Рисунок 8 – Интерфейс администрирования векторного хранилища (добавление и просмотр чанков)

Алгоритм работы следующий: после получения запроса на генерацию тест-кейсов выполняется проверка доступности векторного контура, затем извлекаются релевантные фрагменты из постоянных коллекций semantic_model и learning_feedback, после чего при объёме технического задания свыше 100 000 символов запускается временная индексация в коллекции documents и отбор наиболее значимых чанков для включения в промпт. По завершении обработки временные данные удаляются. Такой механизм предотвращает накопление нерелевантного контента и сохраняет управляемость качества retrieval-выборки. Кроме этого, в этом решении обеспечивается контролируемая деградация функциональности RAG: при недоступности Chroma или отсутствии валидного ключа OPENAI_API_KEY система не блокирует основной сценарий, а продолжает генерацию без retrieval-компонента. Это снижает вероятность полного отказа пользовательского процесса и повышает надёжность решения в нестабильной интеграционной среде.

Технологический стек и реализация

Серверная часть представляет собой FastAPI-приложение, которое обрабатывает запросы от клиента, выполняет генерацию тест-кейсов и ревью технического задания через API языковых моделей, интеграции с Confluence и GitLab и сохраняет результаты в реляционную базу данных SQLite (через SQLAlchemy). Для организации обработки запросов используются асинхронные функции (async/await), что позволяет эффективно обрабатывать большие документы и выполнять параллельную генерацию тест-кейсов.

Для работы с искусственным интеллектом используются библиотеки для взаимодействия с LLM-провайдерами: openai (модели GPT-4o-mini и др.), anthropic (модели Claude 3.5 Sonnet) и aiohttp (для Ollama); httpx применяется для HTTP-запросов к внешним API (в том числе GitLab); в экспериментальной оценке использовалась модель GPT-4o-mini с параметром температуры 0.3. Для обработки документов используются библиотеки python-docx и pypdf, которые позволяют извлекать текст из файлов форматов

.docx и .pdf (для DOCX — с сохранением структуры документа). Для экспорта тест-кейсов в систему документооборота используется библиотека atlassian-python-api для работы с Confluence REST API [13].

Клиентская часть реализована с помощью React и TypeScript (в соответствии с рисунком 9). TypeScript обеспечивает типобезопасность и улучшает качество кода. Сборка выполняется средствами Vite, взаимодействие с backend — через Axios. Для стилизации используется Tailwind CSS, что позволяет быстро создавать современный и адаптивный дизайн [14]. Для обеспечения безопасности доступа к системе используется аутентификация пользователей через JWT-токены [15]: токен передаётся в заголовке Authorization и хранится в sessionStorage браузера. Библиотека python-jose используется для создания и проверки JWT-токенов, а библиотека passlib с bcrypt — для безопасного хранения хешированных паролей пользователей в базе данных.

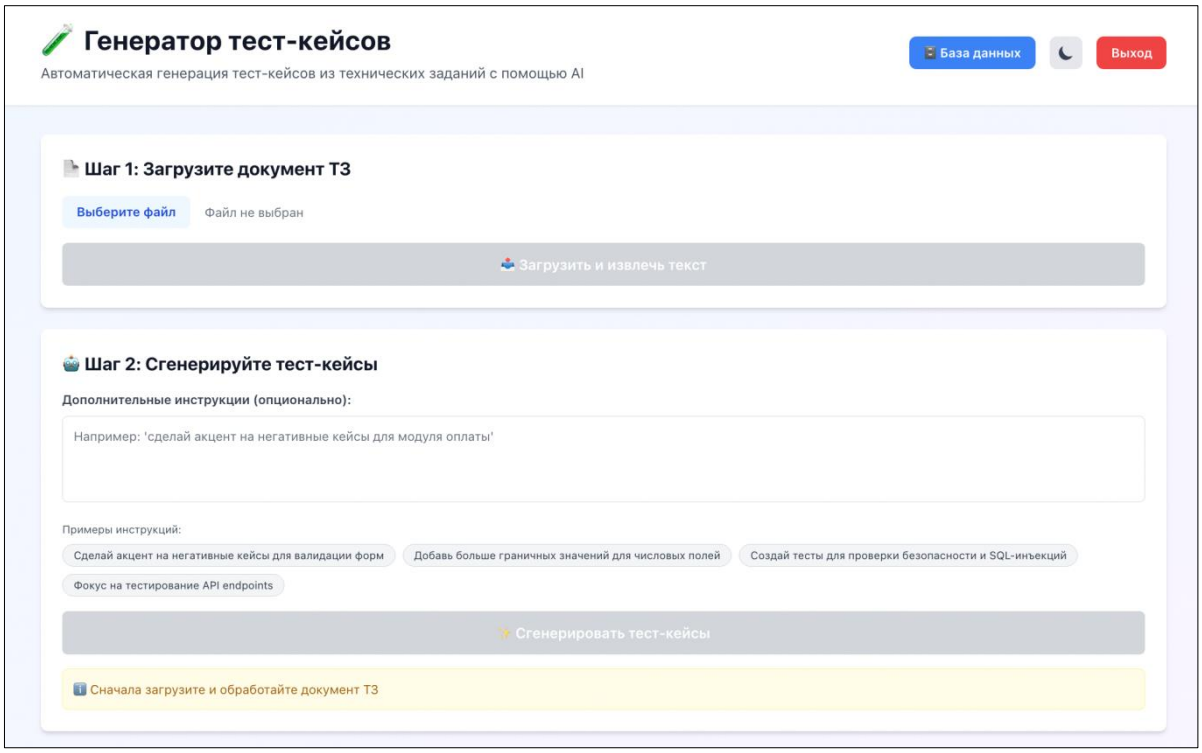


Рисунок 9 – Главная страница веб-интерфейса системы

Экспериментальная оценка эффективности

Оценивалась практическая эффективность внедрения автоматизированной генерации черновиков тест-кейсов в сравнении с ручным методом [1], [16]. Под черновиком, пригодным к экспертному ревью, понимался набор тест-кейсов, удовлетворяющий чек-листу (наличие шагов, ожидаемых результатов, основных типов сценариев).

Использовался контрольный набор из 18 документов трёх классов объёма: малые (8–12 стр.), средние (35–55 стр.), крупные (120–190 стр.). Для каждого документа выполнялись 5 повторов генерации при фиксированных параметрах модели (GPT-4o-mini, температура 0.3). Ручной baseline выполнялся на тех же текстах без использования системы.

Результаты по трудозатратам. Среднее полное время ручной подготовки набора тест-кейсов с нуля составило 3 ч 10 мин. При использовании системы среднее время, затраченное экспертом на загрузку документа, запуск генерации и приведение

черновика к финальному виду, составило 47 мин. Чистое машинное время генерации для малых и средних документов составляет десятки секунд – минуты. Сокращение трудозатрат эксперта относительно ручного baseline достигло в среднем 75 %.

Качество сгенерированных черновиков оценивалось экспертом на подвыборке из 6 документов. Доля тест-кейсов, принятых без изменений, составила 38 %; потребовавших незначительных правок – 47 %; потребовавших значительной переработки – 15 %. Основной причиной правок являлись неоднозначные требования в исходных ТЗ.

Надёжность конвейера. Полные отказы генерации не зафиксированы; механизм повторных попыток обеспечил успешное завершение во всех тестовых запусках.

На контрольном наборе автоматизированный конвейер демонстрирует сокращение трудозатрат эксперта на 75 %, при этом 85 % сгенерированных тест-кейсов требуют лишь незначительных правок либо не требуют их вовсе.

Заключение

В работе предложена модель интеллектуальной системы сопровождения тестирования программного обеспечения оборудования цифровой маркировки с ядром автоматизированной генерации тест-кейсов. Система представляет собой веб-приложение, состоящее из серверной части на FastAPI и клиентской части на React. Система использует большие языковые модели для генерации тест-кейсов и ревью технических заданий в форматах DOCX и PDF.

Проведённая экспериментальная оценка подтвердила работоспособность конвейера генерации на контрольном наборе документов. Результаты показывают, что система позволяет сократить время подготовки черновика тест-кейсов примерно на 75 % по сравнению с ручным baseline и обеспечить единообразие формата; дальнейшая оценка полноты покрытия требований выполняется экспертом при приёмке сгенерированных сценариев. Система пригодна к опытно-промышленному внедрению; развёртывание в промышленной (production) среде целесообразно после настройки безопасности, масштабирования и пилотной эксплуатации.

Ограничение исследования. В работе не оценивался раздельный вклад компонента RAG в улучшение качества тест-кейсов, а также отсутствует сравнение с альтернативными методами генерации (template-based, model-based). Эксперимент выполнялся на документах в области цифровой маркировки; обобщение на другие домены требует дополнительной проверки.

Дальнейшее развитие системы предполагает несколько направлений. Первое направление связано с интеграцией системы с популярными Issue-трекерами, такими как Jira, что позволит автоматически генерировать тест-кейсы непосредственно из задач и требований. Второе направление связано с развитием доменного корпуса и механизмов RAG (включая накопление learning feedback) для повышения релевантности контекста без обязательного дообучения базовых LLM. Третье направление связано с расширением типов генерируемых тест-кейсов, включая более сложные негативные и граничные сценарии.

Разработанная система демонстрирует потенциал применения технологий искусственного интеллекта для автоматизации процессов тестирования в разработке программного обеспечения. Дальнейшее развитие системы в указанных направлениях позволит расширить область ее применения и повысить эффективность процессов тестирования.

Список источников

1. ГОСТ Р ИСО/МЭК 12207-2010. Информационные технологии. Процессы жизненного цикла программных средств. Введ. 2012-01-01. М.: Стандартинформ, 2012. 120 с.
2. Савин, В. В. Тестирование программного обеспечения: учебное пособие / В. В. Савин. М.: Издательский дом «Форум» : ИНФРА-М, 2018. 368 с.
3. Мелихов М.Д. Система маркировки «Честный знак» и ее значение при осуществлении внешнеэкономической деятельности в РФ // Наукосфера. 2025. № 7-2. С. 119-126. DOI: 10.5281/zenodo.16410748.
4. Цифровизация предприятий пищевой промышленности с использованием «Честного знака»: вызовы, особенности, перспективы / Абрамов В.И., Абрамов О.В., Гордеев В.В., Столяров А.Д. // Продовольственная политика и безопасность. 2025. Т. 12. № 1. С. 281-301. DOI: 10.18334/ppib.12.1.122489.
5. Проблемы и перспективы развития системы маркировки «Честный знак» в сфере оборота продукции / Казанкина О.А., Кублин И.М., Немцев А.Д., Воротилин А.В.// Вестник Волжского университета им. В.Н. Татищева. 2025. № 6 (115). С. 84-96. DOI: 10.51965/2076-7919_2025_6_84.
6. Романчик, В. С. Методы и средства тестирования программного обеспечения: учебное пособие / В. С. Романчик. М.: Издательство МГУ, 2017. 320 с.
7. Новиков, Ф. А. Методологии проектирования информационных систем: учебное пособие / Ф. А. Новиков, Д. Б. Яковлев. М.: Издательство МГУ, 2018. 384 с.
8. Марков, А. А. Применение больших языковых моделей для автоматизации тестирования программного обеспечения / А. А. Марков, И. В. Петров // Программные продукты и системы. 2023. № 4. С. 612-625.
9. Козлов, В. П. Искусственный интеллект и машинное обучение в разработке программного обеспечения / В. П. Козлов, Е. Н. Волкова. М.: Издательство МГУ, 2022. 288 с.
10. Пикалёв Я.С. Обзор архитектур систем интеллектуальной обработки естественно-языковых текстов / Я.С. Пикалев // Проблемы искусственного интеллекта. 2020. № 4 (19). С. 45-68.
11. Ермоленко Т.В. Классификация ошибок в тексте на основе глубокого обучения / Т.В. Ермоленко // Проблемы искусственного интеллекта. 2019. № 3 (14). С. 47-57.
12. Лебедев, И. В. Архитектура программных систем: учебное пособие / И. В. Лебедев, А. С. Федоров. М.: Издательство МГТУ им. Н. Э. Баумана, 2021. 352 с.
13. Atlassian Python API [Электронный ресурс]. Режим доступа: https://atlassian-python-api.readthedocs.io/ (дата обращения: 25.03.2026).
14. Tailwind CSS Documentation [Электронный ресурс]. Режим доступа: https://tailwindcss.com/docs (дата обращения: 25.03.2026).
15. Jones M. JSON Web Token (JWT) / M. Jones, J. Bradley, N. Sakimura // IETF RFC 7519. 2015. Режим доступа: https://datatracker.ietf.org/doc/html/rfc7519 (дата обращения: 25.03.2026).
16. П ГОСТ Р ИСО/МЭК 25051-2014. Требования к программным продуктам и оценке готового программного продукта. Введ. 2015-09-01. М.: Стандартинформ, 2015. 20 с.

References

1. GOST R ISO/IEC 12207-2010. Information technology. Software lifecycle processes. – Introduction. 2012-01-01. – Moscow: Standartinform, 2012. – 120 p.
2. Savin, V. V. Software testing: a textbook / V. V. Savin. – Moscow: Forum Publishing House: INFRA-M, 2018. 368 p.
3. Melikhov M. D. The “Chestny Znak” traceability system and its role in foreign economic activity in the Russian Federation. – 2025. – No. 7-2. – pp. 119-126. DOI: 10.5281/zenodo.16410748.
4. Abramov V. I., Abramov O. V., Gordeev V. V., Stolyarov A. D. Digitalization of food industry enterprises using “Chestny Znak”: challenges, features, and prospects // Food Policy and Security. – 2025. – Vol. 12. – No. 1. – Pp. 281-301. – DOI: 10.18334/ppib.12.1.122489.
5. Kazankina O. A., Kublin I. M., Nemtsev A. D., Vorotilin A. V. Problems and prospects of developing the “Chestny Znak” traceability system in product circulation // Bulletin of Volga University named after V. N. Tatischev. – 2025. – No. 6 (115). – Pp. 84-96. – DOI: 10.51965/2076-7919_2025_6_84.
6. Romanchik, V. S. Methods and means of software testing: a textbook / V. S. Romanchik. – Moscow: Moscow State University Publishing House, 2017. – 320 p.
7. Novikov, F. A. Methodologies for designing information systems: a textbook / F. A. Novikov, D. B. Yakovlev. – Moscow: Moscow State University Publishing House, 2018. – 384 p.
8. Markov, A. A. Application of large language models for software testing automation / A. A. Markov, I. V. Petrov // Software products and systems. – 2023. – No. 4. – Pp. 612-625.
9. Kozlov, V. P. Artificial intelligence and machine learning in software development / V. P. Kozlov, E. N. Volkova. – M.: Publishing House of Moscow State University, 2022. – 288 p.

10. Pikalev Ya. S. A survey of architectures of systems for intelligent natural language text processing // Problems of Artificial Intelligence. – 2020. – No. 4 (19). – Pp. 45-68.
11. Ermolenko T.V. Classification of errors in the text based on deep learning // Problems of Artificial Intelligence. – 2019. – No. 3 (14). – pp. 47-57.
12. Lebedev, I. V. Architecture of software systems: a tutorial / I. V. Lebedev, A. S. Fedorov. – M.: Publishing house of Bauman Moscow State Technical University, 2021. – 352 p.
13. Atlassian Confluence Cloud REST API [Electronic resource]. – URL: https://developer.atlassian.com/cloud/confluence/rest/v2/intro/ (accessed: 25.03.2026).
14. Tailwind CSS Documentation [Electronic resource]. – URL: https://tailwindcss.com/docs (accessed: 25.03.2026).
15. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT) [Electronic resource] // IETF RFC 7519. – 2015. – URL: https://datatracker.ietf.org/doc/html/rfc7519 (accessed: 25.03.2026).
16. GOST R ISO/IEC 25051-2014. Requirements for software products and evaluation of the finished software product. – Introduction. 2015-09-01. – Moscow: Standartinform, 2015. – 20 p.

RESUME

A.A. Vartanyan, E.V. Kurnasov, A.V. Atamanchuk, M.R. Zabirov

Intelligent software testing support system for digital marking equipment

Research on software testing automation methods is one of the actively developing areas in the field of software development and quality assurance. With the development of digital labeling systems, there is an increasing need for efficient and scalable approaches to preparing test documentation.

The use of modern artificial intelligence technologies, in particular large language models, makes it possible to automate the process of analyzing text requirements and generating test cases. The use of such models helps to increase the speed of preparation of test documentation, as well as improve the completeness and structure of test scenarios.

In an environment where software requirements are often presented in an unstructured form, the use of LLM-based approaches allows you to extract logical conditions, business rules, and boundary scenarios directly from text-based technical specifications.

Experimental studies on a control set of 18 documents showed an approximately 75 % reduction in draft test case preparation time compared to the manual baseline, along with suitability of draft results for subsequent expert validation and use in the testing process.

РЕЗЮМЕ

А.А. Вартанян, Е.В. Курнасов, А.В. Атаманчук, М.Р. Забиров

Интеллектуальная система сопровождения тестирования программного обеспечения для оборудования цифровой маркировки

Исследования методов автоматизации тестирования программного обеспечения являются одним из активно развивающихся направлений в области разработки и обеспечения качества ПО. С развитием систем цифровой маркировки возрастает необходимость в эффективных и масштабируемых подходах к подготовке тестовой документации.

Использование современных технологий искусственного интеллекта, в частности больших языковых моделей, позволяет автоматизировать процесс анализа текстовых требований и генерации тест-кейсов. Применение таких моделей способствует повышению скорости подготовки тестовой документации, а также улучшению полноты и структурированности тестовых сценариев.

В условиях, когда требования к программному обеспечению часто представлены в неструктурированном виде, применение подходов на основе LLM позволяет извлекать логические условия, бизнес-правила и граничные сценарии непосредственно из текстовых технических заданий.

Экспериментальные исследования на контрольном наборе из 18 документов показали сокращение времени подготовки черновика тест-кейсов примерно на 75 % по сравнению с ручным методом, а также пригодность черновиков тест-кейсов для последующей экспертной валидации и использования в процессе тестирования.

Вартанян Алина Андраниковна – ведущий инженер тестирования ООО «ВЕКАС», г. Москва, Россия, vartanayn.alina@gmail.com. *Область научных интересов:* тестирование программного обеспечения, искусственный интеллект, большие языковые модели, автоматизация разработки.

Курнасов Евгений Вячеславович – кандидат технических наук, доцент кафедры промышленной информатики Института искусственного интеллекта РТУ МИРЭА, г. Москва, Россия, kurnasv@mirea.ru. *Область научных интересов:* искусственный интеллект, предиктивный анализ данных, системная интеграция в области промышленной автоматизации и управления, разработка интеллектуальных систем управления технологическими объектами и процессами, умные IT-решения для промышленности.

Атаманчук Алексей Викторович – программист, выпускник бакалавриата Института искусственного интеллекта РТУ МИРЭА, г. Москва, Россия, at4man696@yandex.ru. *Область научных интересов:* тестирование программного обеспечения, искусственный интеллект, большие языковые модели, автоматизация разработки.

Забиров Марат Рушанович – программист ООО «МПС Софт», г. Москва, Россия, marat3014@gmail.com. *Область научных интересов:* тестирование программного обеспечения, искусственный интеллект, большие языковые модели, автоматизация разработки.

Vartanyan Alina Andranikovna – lead test engineer, VEKAS LLC, Moscow, Russia, vartanayn.alina@gmail.com. Research interests: software testing, artificial intelligence, large language models, development automation.

Kurnasov Evgeny Vyacheslavovich – candidate of technical sciences, associate professor, department of industrial informatics, Institute of Artificial Intelligence, RTU MIREA, Moscow, Russia, kurnasv@mirea.ru. Research interests: artificial intelligence, predictive data analysis, system integration in the field of industrial automation and control, development of intelligent control systems for technological facilities and processes, smart IT solutions for industry.

Atamanchuk Alexey Viktorovich – software engineer, bachelor's graduate of the Institute of artificial intelligence, RTU MIREA, Moscow, Russia, at4man696@yandex.ru. Research interests: software testing, artificial intelligence, large language models, and development automation.

Zabirov Marat Rushanovich – software engineer, MPS Soft LLC, Moscow, Russia, marat3014@gmail.com. Research interests: software testing, artificial intelligence, large language models, and software development automation.

Статья поступила в редакцию 27.02.2026;
одобрена после рецензирования 01.04.2026; принята к публикации 02.09.2026
The article was submitted 27.02.2026; approved after reviewing 01.04.2026;
accepted for publication 02.09.2026.