МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

КУРСОВАЯ РАБОТА
по дисциплине «Курсовое проектирование»

Проектирование системы автоматизации тестирования ERP-системы в компании «Мир Упаковки»
Б-ИН-22-184938.2470-а.05.000 КР

Студент
гр. БИН-22-01 _____ В.В. Босенко

Руководитель,
профессор,
доктор тех. наук _____В.М. Гриняк

Владивосток 2026

Содержание

МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
на курсовое проектирование

Студенту Босенко Виктории Владимировне, гр. БИН-22-1

Целью выполнения Курсовой работы является закрепление знаний, углубление знаний, а также приобретение практических навыков в области проектирования автоматизированного тестирования ПО для предприятия.

Для достижения этой цели, должны быть выполнены следующие задачи:

1 Анализ предметной области в компании ООО «Мир Упаковки». Необходимо исследовать бизнес-процессы предприятия, описать объект автоматизации и сформировать представление о нем.

2 Выбор инструментов. На основе анализа предметной области подбираются соответствующие инструменты для проектирования среды автоматизации тестирования.

3.Проектирование среды тестирования. Сформировать архитектуру среды через схемы взаимодействия элементов.

4. Описание взаимодействия пользователя со средой. С помощью диаграмм описать действия пользователя в процессе выполнения поставленных задач через спроектированный механизм автоматизации.

4 Выбор стратегии разработки тестов. Структурировать работу во время написания тестов, придерживаясь существующих методологий.

5 Разработка тестов для каждого выбранного вида. Описать примеры для каждого вида выбранного тестов.

Срок сдачи отчета на кафедру: 13.01.2026

Руководитель,
профессор,
доктор тех. наук В.М. Гриняк

Задание получил: В.В.Босенко

Аннотация

Курсовая работа посвящена теме «Проектирование системы автоматизации тестирования ERP-системы в компании «Мир Упаковки»».

Основная цель проекта — разработка системы автоматизированного тестирования.

Для выполнения поставленной цели в ходе работы был проведен анализ текущего состояния бизнес-процесса закупки товаров, существующий инструментов для автоматизации тестирования, методологий для качественного покрытия тестами конфигурации и метрик для оценки эффективности внедрения автотестов с финансовой точки зрения.

На основе проведенного анализа была построена архитектура среды автоматизированного тестирования, включающая в себя такие инструменты, как: фреймрок направленный на написание автоматизированных тестов для платформы 1С:Предприятия, инструмент для формирования понятного пользователю отчета о полученных дефектах, а также программу позволяющею отправлять уведомления с результатами прохождения тестовых сценариев. Для качественного покрытия тестами функционала процесса закупок были выбраны несколько видов тестов, которые проверят работу системы на разных уровнях от базовых функций платформы до бизнес-логики.

Результатом стала концептуальная модель среды автоматизированного тестирования, примеры для каждого вида выбранных тестов и набор метрик с формулами для расчета ценности проделанной работы.

Введение

Современная сфера продаж товаров и услуг имеет большую конкуренцию, где важно подстраиваться под стремительно изменяющиеся потребности покупателей. Компании вынуждены управлять огромным количеством данных, сложными цепочками поставок, продажами и динамичным ценообразованием. Ручное управление таким потоком информации и процессов является невозможным, поэтому многие ритейл компании выбирают использовать в качестве управления ресурсами предприятия ERP-системы. Преимущество таких систем перед другими ПО заключается в гибкой настройке, позволяющей адаптировать их под любые бизнес-процессы конкретной компании. С их помощью организация может не просто внедрить готовый продукт, но и разработать собственную систему управления со своей внутренней логикой работы, а также включить в нее сторонние программы.

Однако это ключевое достоинство может стать и недостатком. Любое изменение в конфигурации может привести к потенциальной ошибке, которая повлечет за собой масштабные финансовые потери, искажающие управленческие отчетности и, как следствие, приводит к принятию некорректных стратегических решений. Ручное тестирование, требующее значительных временных и человеческих ресурсов, становится неэффективным для объемных проверок после каждой доработки. Такой подход повышает вероятность того, что ошибка будет пропущена из-за человеческого фактора в динамически изменяющейся среде.

Таким образом, автоматизация тестирования ERP-системы становится важным элементом в развитии компании, позволяющим минимизировать риски и финансовые потери, сократить время на проверки и ускорить процесс внедрения новых разработок.

1 Описание предметной области

1.1 Общая характеристика предприятия ООО «Мир Упаковки»

Общество с ограниченной ответственностью «Мир Упаковки» - сервисная компания B2B, которая стремительно развивалась на рынке с 1999 года, занимающаяся оптово-розничной торговлей расходных и упаковочных материалов на территории России и СНГ. За более чем двадцатилетний период предприятие из маленького павильона выросла до лидера рынка на Дальнем Востоке.

Быстрый рост компании за счёт открытия новых филиалов, увеличения клиентской базы и товарного ассортимента, а также в усложнение внутренних бизнес процессов, потребовал перехода от разрозненных систем учета к комплексному решению. Ключевым шагом в этом направлении стало внедрение интегрированной ERP-системы на платформе «1С:Предприятие». Данная система была выбрана в качестве единой информационной среды, которая позволяла обеспечить согласованную работу всех подразделений: от отдела по логистике до технической поддержки. Ее внедрение было необходимо, чтобы централизовать управление и автоматизировать важные производственные операции.

1.2 Бизнес-процесс тестирования

На текущей день компания продолжает активно модернизировать свои бизнес-процессы, что влечет за собой внедрение в конфигурацию «1С:Предприятие» новых функциональных блоков и доработку существующих. Система отвечает за особенно важные для бизнеса операции, например, за расчет себестоимости, ценообразование с учетом логистических издержек, управление цепочками поставок, складской учет в режиме реального времени и формирование отчетностей. В связи с этим любая ошибка, допущенная на этапе разработки или внедрения нового функционала, может привести к масштабным сбоям: искажению складских остатков, некорректному расчету цен и, как следствие, к прямым финансовым и репутационным потерям.

Во избежание ошибок и сбоев в работе ERP-системе многие IT организации включают тестирование в процесс разработки. В компании «Мир Упаковки» также выделен этап тестирования, предшествующий выходу обновлений в продакшен (рисунок 1.1).

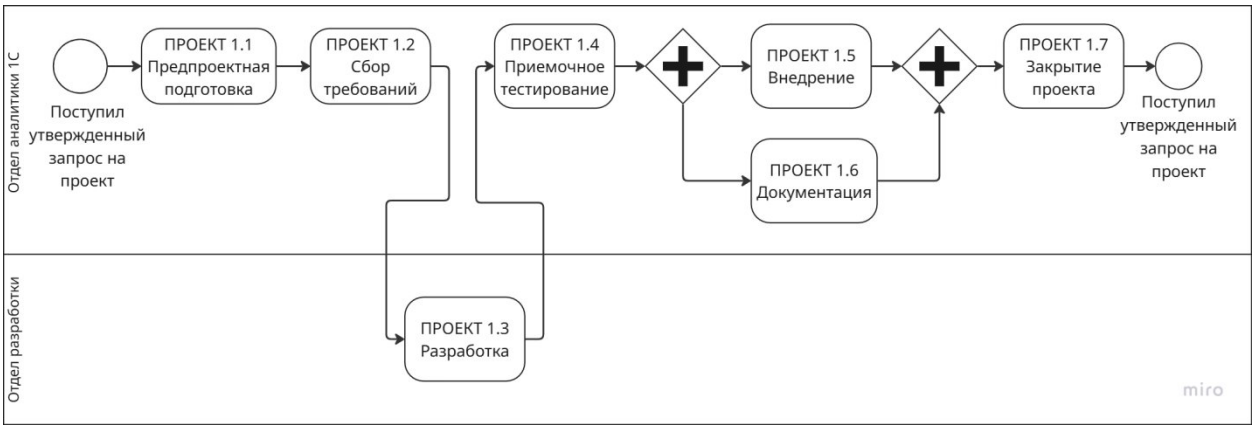

Рисунок 1.1 – Бизнес-процесс отдела разработки ПО

Однако данный процесс осуществляется преимущественно вручную сотрудниками IT-отдела, что вызывает ряд проблем. Во-первых, такой подход требует значительных временных затрат и человеческих ресурсов. Кроме того, потери возрастают, поскольку новые разработки часто нуждаются во множестве доработок, что влечёт необходимость проведения регрессионного тестирования для проверки, чтобы убедиться, что внесенные изменения не нарушили существующую функциональность.

Во-вторых, ручное тестирование подвержено человеческим ошибкам, так как при проверке большого количества повторяющихся сценариев, высока вероятность пропустить дефекты. В результате этого обновления выпускаются в эксплуатацию с недочётами, которые приводит к новым циклам доработок и тестированию, создавая замкнутый круг.

Для решения этих проблем необходимо внедрение автоматизированного тестирования. Автотесты позволяют выполнять проверку функциональности быстрее и проводить регрессионное тестирование после каждого изменения без увеличения трудозатрат. Это не только сокращает время на выпуск обновлений, но и значительно повышает их качество за счёт исключения человеческого фактора и возможности проверки большего количества сценариев. Автоматизация тестов становится ключевым инструментом для поддержания стабильности системы в условиях непрерывной разработки новых функций и процессов в «1С:Предприятие».

1.3 Описание процесса закупки товаров

В качестве объекта для автоматизации была выбрана закупка товаров (рисунок 1.2). Развитие бизнес логики внутри этапа, а также корректировка смежных процессов, влечет за собой постоянные доработки, которые требуют регрессивного тестирования. Нарушение работы выбранного этапа, влияет на финансовые результаты и искажает складской учет.

Сложность расчетов и большое количество взаимосвязанных объектов делают этот модуль подходящим для автоматизации, поскольку ручная проверка всех сценариев будет

трудоёмка и подвержена ошибкам, в то время как автоматизированные тесты позволят проводить полную регрессионную проверку быстро и стабильно.

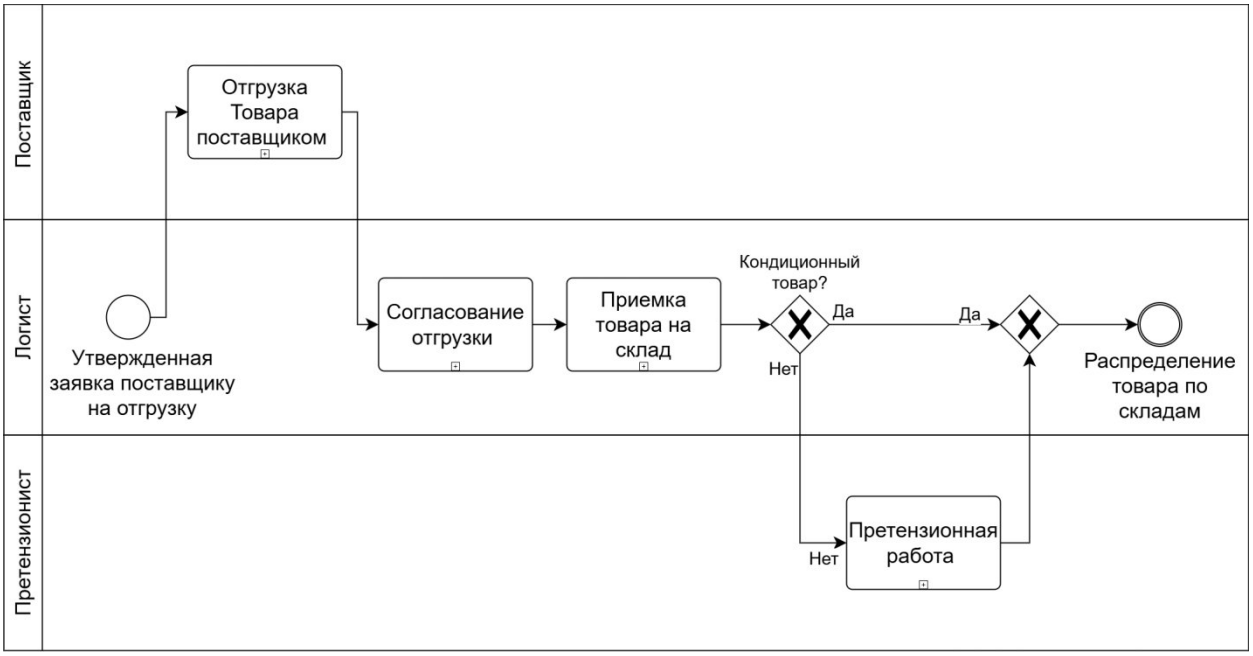

Рисунок 1.2 – Бизнес-процесс закупки товаров

Запуск этапа закупки товаров начинается с утверждения заявок поставщику на отгрузку. Логисты связываются с каждым поставщиком для согласования деталей отгрузки. В приоритете находятся местные поставщики. Для некоторых поставщиков уже существуют утвержденные графики отгрузок с фиксированными днями, что упрощает процесс согласования. По результатам звонков делаются соответствующие записи, чтобы сохранить полученные данные для информирования других сотрудников, которые будут работать с заявкой в будущем.

Если поставщик не готов выполнить отгрузку в запланированные сроки, логист фиксирует это в заявке и передает задачу менеджеру УТЗ для дальнейшей работы с поставщиком. После корректировки даты отгрузки, менеджером УТЗ, вносит изменения в соответствующие поля заявки, и весь процесс начинается заново с обновленными данными. Согласованные заявки распределяются по разным дням отгрузки в зависимости от приоритетов и загруженности. Все изменения в графиках, включая переносы дней отгрузки по инициативе поставщика, тщательно документируются, чтобы тщательно контролировать процесс.

Отгружая товары в транспортную компанию, поставщик выставляет компании УТЗ счет-фактуру и УПД. Это документы, которые подтверждают факт поставки и служат основанием для дальнейших операций. Перед погрузкой контейнера логисты согласуют ставку на перевозку контейнера с транспортной компанией, чтобы избежать дополнительных

расходов. На этом этапе реальную сумму и объем поставки точно не знают, поскольку информация в УПД носит предварительный и неточный характер. Уточнение происходит после проведения заказа, который создается на основании заявки поставщику, который подкрепляется УПД. К заказу также добавляется документ факта оплаты, что подтвердить транзакцию.

Далее из сформированных заказов логистом создается документ рейс ТК, в котором отражается перемещение товара, сумма расходов и прочая информация о рейсе. На основе этих данных формируются разные отчетности, необходимые для аналитики и контроля закупок. Рейс ТК может содержать разное количество заказов от разных поставщиков, что позволяет оптимизировать логистику. Чтобы сэкономить на перевозе товаров, было решено выделить этап консолидации - процесс ожидания наполнения контейнера. Это позволяет собирать грузы от нескольких поставщиков в один контейнер, снижая общие расходы. Однако некоторые поставщики отгружают товар целыми контейнерами, и в таком случае этап консолидации пропускается, чтобы не задерживать поставку.

После загрузки контейнера транспортная компания присылает реестр загруженных товаров. Этот реестр представляет собой пул заявок поставщику и служит подтверждением загрузки. Затем после погрузки контейнера ТК присылает счет за перевозку логисту. Логист в документе рейса ТК записывает все расходы за транспортировку.

Затем информация о заказах передается диспетчеру. Диспетчер следит за своевременность выхода контейнера и прохождением контрольных точек. Контролирует он этот процесс с помощью документа «Рейс ТК, где проставляет соответствующие даты на каждом этапе пути. Это обеспечивает прозрачность каждого этапа и позволяет быстро реагировать на задержки.

По прибытии к пункту назначения транспортная компания связывается с диспетчером и сообщает о прибытии груза. Диспетчер уточняет время для выгрузки. Поскольку компания использует услугу дверь в дверь, транспортная компания сама доставляет груз до склада. Диспетчер связывается со складом, чтобы уточнить, когда склад готов принять товары. После согласования транспортной компании передается время, затем товар привозят на выгрузку в оговоренный срок. Выгрузка обычно планируется за сутки, чтобы подготовить склад и персонал.

Также диспетчер создает документы Поступление товаров и услуг на все заказы, которые готовы к приемке. Эти документы подаются в работу, чтобы склад мог начать подготовку. Диспетчер печатает реестр поступлений. Получая эту информацию, склад знает, какие товары принимать и в каком объеме.

В назначенный день, склад выполняет приемку - проверяет, полностью ли совпадает поступление с заявками или есть расхождения. Результат приемки фиксируется в документе Поступление. Если выявлены расхождения, они анализируются для коррекции будущих поставок. Такой подход обеспечивает точность учета товаров.

Весь процесс требует координации между логистикой, поставщиками и складом, чтобы минимизировать задержки и затраты. Регулярное документирование этапов помогает отследить все шаги и движение товара.

2 Выбор инструментов тестирования

2.1 Среда написания автотестов

Разработка автоматизированных тестов требует применения специализированных программ для создания, выполнения и управления тестовыми сценариями. Выбор инструментов является неотъемлемой частью разработки качественного и практичного продукта. Не менее необходимыми будут инструменты для предоставления результатов прохождения тестов. Следовательно, в создании среды автоматизированного тестирования будут участь три ключевых компонента: фреймворки для написания автотестов и для формирования отчетов, а так же канал для быстрого информирования команды об ошибке.

ERP-система компании «Мир Упаковки» реализована на платформе «1С:Предприятие» из-за чего круг доступных решений сужается. Наиболее распространёнными средствами автоматизации тестирования в 1С считаются Vanessa-Automation, 1С:Сценарное тестирование и Vanessa-ADD. Каждый из этих инструментов обладает уникальным набором характеристик, преимуществ и ограничений, что делает целесообразным их детальное сравнение.

Для наглядного сопоставления функциональных и технических особенностей упомянутых инструментов, ниже приведена таблица 1 сравнений.

Таблица 1 - Сравнительный анализ инструментов для написания автотестов

Критерий	Vanessa Automation	1С:Сценарное тестирование	Vanessa-ADD
Язык написания тестов	Gherkin	1С	1С
Сложность освоения	Низкий порог входа за счет языка сценариев на русском	Средний порог входа, требует знаний 1С и программирования	Высокий порог входа, требует глубоких знаний 1С
Встроенная визуализация результатов	Нет	Да	Нет
Зрелость и сообщество	Высокая зрелость, активное сообщество, обширная документация	Высокая зрелость как стандартного инструмента от «1С», но ориентирован на ручное тестирование	Низкая зрелость, нишевое использование, отсутствие широкой поддержки
Ориентирован на вид тестов	Тестирование интерфейса пользователя, Тестирование функциональности системы	Функциональное, интеграционное, запись и воспроизведение пользовательских действий в 1С	Модульное (юнит) Сценарное, Регрессивное, Нагрузочное

Продолжение таблицы 1

Критерий	Vanessa Automation	1С:Сценарное тестирование	Vanessa-ADD
Интеграция с CI/CD	Поддерживается Jenkins, GitLab CI, TeamCity; легко интегрируется в пайплайны для непрерывного тестирования.	Встроенная в 1С платформу, поддержка базовых CI-инструментов через 1С-сервера; менее гибкая	Поддерживается Jenkins, GitLab CI, TeamCity; легко интегрируется в пайплайны для непрерывного тестирования.
Open source	Да	Платное	Да

В процессе анализа трех фреймворков было решено сделать выбор в пользу Vanessa-Automation. Несмотря на определенные недостатки, такие как необходимость дополнительного времени, на настройку инфраструктуры, она была выбрана как наиболее подходящий вариант. Главным плюсом данного инструментом стала не только её совместимость с 1С, но также со сторонними программами, например, для детализированных отчетов или для подключения уведомлений, что делает её очень гибким инструментом, а также открывает новые возможности для автоматизации и улучшения процессов. Кроме того, Vanessa Automation является бесплатным open-source решением с активным сообществом пользователей и разработчиков, где доступно множество готовых решений, шаблонов, плагинов и документации, что значительно упрощает внедрение и поддержку. Инструмент ориентирован на такие виды тестов, как тестирование интерфейса пользователя (UI-тесты), тестирование функциональности системы (функциональные и интеграционные проверки бизнес-логики) и тестирование формирования отчетов, обеспечивая комплексное покрытие критически важных аспектов работы приложений в «1С:Предприятие».

«1С:Сценарное тестирование» не было выбран по нескольким ключевым причинам, несмотря на его встроенность в платформу «1С» и возможность формировать отчеты без сторонних программ . Во-первых, этот инструмент является платным, что повышает расходы компании, в случае его введения в работу. Во-вторых, он менее гибкий в плане интеграции с внешними системами CI/CD, ограничиваясь базовыми возможностями через серверы «1С», без полноценной поддержки популярных инструментов типа Jenkins или GitLab, что усложняет автоматизацию некоторых этапов тестирования. Кроме того, с помощью «1С:Сценарное тестирование» нельзя написать все виды тестов: он преимущественно ориентирован на запись и воспроизведение пользовательских действий в клиентской части «1С», что делает его неэффективным для модульного тестирования или проверки

низкоуровневой бизнес-логики, в результате чего тесты могут покрыть не весь функционал системы, оставляя пробелы в выявлении дефектов на более глубоких уровнях.

Vanessa ADD представляет собой достойную альтернативу, не уступающую Vanessa Automation в определенных аспектах, и ее можно эффективно использовать там, где тесты Vanessa Automation не покрывают код. Однако написание автотестов для «1С:Предприятие» все же отдали предпочтение Vanessa Automation из-за множества готовых решений, а также широкого спектра видов тестирования, включающего интерфейсы, функциональность и отчеты, что делает ее более универсальной и подходящей для комплексной автоматизации в современных проектах.

2.2 Инструмент для анализа результатов тестирования

Для выбранного инструмента тестирования, такого как Vanessa Automation, требуется отдельный инструмент формирования отчетности результатов тестирования. Основная причина заключается в отсутствии встроенной визуализации результатов в Vanessa Automation. Инструмент генерирует выходные данные в виде JSON или XML, однако эти данные сложно интерпретировать без преобразования в пользовательские отчеты. Без специализированного решения команды тратят время на ручной анализ логов, что снижает эффективность в процессах непрерывной интеграции и доставки, затрудняет выявление трендов дефектов и распределение задач по исправлению. Подходящий инструмент должен обеспечивать детальные отчеты, включая историю прохождения тестов, графики, диаграммы и поддержку ссылок на шаги сценариев Gherkin.

Существует несколько вариантов для формирования отчетности: от простых HTML-генераторов в составе фреймворков до полных систем вроде Allure или Cucumber. Они различаются по возможностям интеграции и глубине анализа данных. Выбор среди них позволяет учесть специфику работы с Vanessa Automation, где важна совместимость с BDD-подходом и CI/CD-инструментами. Сравнение основных альтернатив представлено в следующей таблице 2.

Таблица 2 - Сравнительный анализ инструментов формирования отчетов

Критерий	ExtentReports	Cucumber Reports	Allure Framework
Легкость адаптации с Vanessa Automation	Низкая. Формат, который выводит Vanessa вызывает сложности для расшифровки в отчет. Требуются стронинние инструменты	Высокая. Vanessa Automation поддерживает формирование отчётов в формате Cucumber	Высокая. Vanessa Automation поддерживает формирование отчётов в формате Allure

Продолжение таблицы 2

Критерий	ExtentReports	Cucumber Reports	Allure Framework
Интеграция с CI/CD-инструментами	Интеграция с CI/CD-системами (GitHub Actions, CircleCI)	Интеграция с CI/CD-системами (Jenkins, CircleCI, GitLab CI)	Интеграция с CI/CD-системами (Jenkins, TeamCity, GitLab CI)
Что отображает в отчетах	Общая статистика, список тестов, Детализация теста, отчёт об ошибках, временные метрики и т.д	Отчет по общую статистику, список всех сценариев с детализацией, отчёт по шагам, история выполнения, Отчёт о времени выполнения и т.д	Общая статистика, группировка дефектов по категориям, графики динамики, временная шкала выполнения тестов
Формат отчетов	HTML, XML, JSON	HTML, JSON, JUnit, Pretty	HTML, JSON, CSV
Кастомизация категорий ошибок	Да	Да	Да
Отображает момент ошибки	Отображает момент ошибки. Подсвечивает красным шаг, на котором тест упал	Отображает момент ошибки в тесте через встроенные плагины и пользовательские реализации. Отображает проваленный шаг красным	Отображает последовательность шагов, приведших к ошибке и прикреплённые артефакты (скриншоты, логи)
Open source	Да	Да	Да

В ходе сравнения инструментов для формирования отчетов были выделены Cucumber Reports и Allure Framework, поскольку они обладают схожими характеристиками: оба легко адаптируются с Vanessa Automation благодаря прямой поддержке соответствующих форматов, обеспечивают интеграцию с ведущими CI/CD-системами, такими как Jenkins и GitLab CI, и предоставляют кастомизацию категорий ошибок. При этом оба инструмента являются бесплатными, а также визуально выделяют моменты ошибок в тесте, что упрощает анализ. Однако предпочтение было отдано Allure Framework из-за наличия большой базы открытых обучающих материалов, что ускорит решение возможно возникших проблем в процессе интеграции инструмента в среду автоматизированного тестирования.

ExtentReports был исключен как возможный вариант выбора из-за сложностей в расшифровке файла, которые формируют Vanessa Automation после прохождения тестов.

Для решения потребуются сторонние инструменты, что увеличивает время на обработку данных. Это делает его менее эффективным по сравнению с Allure или Cucumber Reports.

Таким образом, Allure становится подходящим элементом для визуализации результатов прохождения тестов, поскольку встроенная поддержка формата Allure в Vanessa Automation обеспечивает автоматическую генерацию детализированных отчётов и их бесшовную интеграцию в CI/CD-пайплайны. Кроме того, поддержка множественных форматов, включая CSV, упрощает экспорт для аналитики, а open source-статус Allure гарантируют долгосрочную актуализацию без зависимостей от поставщиков.

2.3 Инструмент для уведомления о результатах прохождения тестов

Для полной автоматизации процесса тестирования с использованием Vanessa Automation и Allure необходимо ввести систему оперативного уведомления о результатах прохождения тестов. Это позволит команде разработки получать актуальную информацию о некорректной работе ERP-системы. Без такого инструмента команда может упустить дефекты, особенно в условиях непрерывных доработок, где новая функция может повлиять на работу текущей конфигурации.

Выбор такого инструмента пал на чат-бот в социальной сети. Это решение было обосновано несколькими факторами, начиная с его простоты внедрения и минимальных требований для создания. Чат-ботов легко настроить, поскольку они опираются на публичные API мессенджеров, не требуя дополнительных сервисов. Кроме того, сообщения из приложений для общения редко остаются незамеченными, в отличие, например, от почты, где письма часто попадают в спам. Более того, многие мессенджеры поддерживают групповые чаты, что позволит уведомлять не только ответственных разработчиков, но и весь отдел разработки.

Чат-бот будет отправлять структурированные уведомления после запуска тестов в Vanessa Automation. В каждом сообщении будет указано общее количество выполненных тестов, число успешных и число неудачных. Это позволяет быстро оценить общий статус сценариев без необходимости углубляться в детали. Кроме того, в сообщения будет включена прямая ссылка на отчет в Allure Framework. Такая связь с Allure обеспечивает переход от общей информации к детальному анализу, где буду отображены графики, шаги выполнения и детали ошибок. Бот также может быть настроен на отправку уведомлений только при неудачах или по расписанию, чтобы избежать перегрузки ненужной информации.

Отдельные платформы вроде ВКонтакте или Max имеют меньшую популярность в корпоративной среде, поэтому был выбран Telegram, где разработчики обсуждают как

рабочие вопросы, так и повседневную рутину, поэтому чат с результатами тестирования плавно интегрируется в рабочих процессов.

Таким образом, бот в Telegram станет инструментом для оповещения о результатах тестирования, обеспечивая возможность быстро реагировать и устранять дефекты в процессе разработки конфигурации.

3 Проектирование среды тестирования

3.1 Роль пользователя

Пользователями создаваемой системы будут выступать тестировщики и разработчики. Главная задача тестировщика заключается в написании автотестов и поддержании их в актуальном статусе. Для этого должен использоваться инструмент создания сценариев. После подготовки нужных тестов, тестировщик должен запустить механизм, который будет управлять прогоном тестов, формировать отчеты и уведомлять о завершении процесса тестирования.

Разработчик будет являться получателем оповещения, содержащее информацию о дефектах. После устранения ошибок, разработчик самостоятельно может запустить подготовленные ранее тестеровщиком тесты без его участия. Соответственно, автотесты могут запускаться до полного устранения неполадок работы бизнес-процесса

3.2 Пользовательский интерфейс

Действия пользователя в рамках среде тестирования должны быть следующим (рисунок 3.1):

Рисунок 3.1 – Процесс автотестирования

Тестировщик получает задание на тестирование изменений, которые были внесены в бизнес логику закупок товаров. Для проведения полноценного тестирования необходимо будет добавить в реестр новые сценарии, опираясь на техническое задание, разработанное аналитиками.

Получив задание, тестировщик открывает через конфигуратор 1С Vanessa Automation, где создает тесты, которые должны проверять только доработку. В случае необходимости, тестировщик должен актуализировать тесты, находящиеся в ранее созданном реестре. Каждый написанный тест группируется в зависимости от его вида.

По завершению работы над сценариями, тестеровщик готовит файлы с расширение «.feature», они выгружаются вручную в каталог, в котором находятся остальные тесты, проверяющие полный функционал закупок товаров.

Последним этапом для тестировщика является запуск базы тестов. Для этого он активирует BAT-файл, где прописан алгоритм действий механизма. Система начинает прогон всех сценариев. Результатом её работы станет сообщение в чат-боте, если доработка повела себя некорректно и вызвала дефект. Разработчик получает уведомление с ссылкой на отчет Allure, переходя по которой находит подробную характеристику ошибок и моменты их появления. Устранив все дефект, разработчик может повторно запустить базу тестов через БАТ-файл, как делал это тестировщик.

3.3 Архитектура системы

Выбранная комбинация инструментов – Vanessa-Automation, Allure Framework и чат-бот будут формировать полноценную среду для автоматизированного тестирования в рамках «1С:Предприятие». Они покрывают все этапы от создания теста до информирования о результате их провождения.

В рамках курсовой было решено использовать БАТ файл, чтобы смоделировать работу автоматизированного проведения среды. Последовательный запуск через консоль позволит наглядно продемонстрировать функцию каждого элемента системы и их взаимодействие. Общая архитектура решения представлена на схеме (рисунок 3.2):

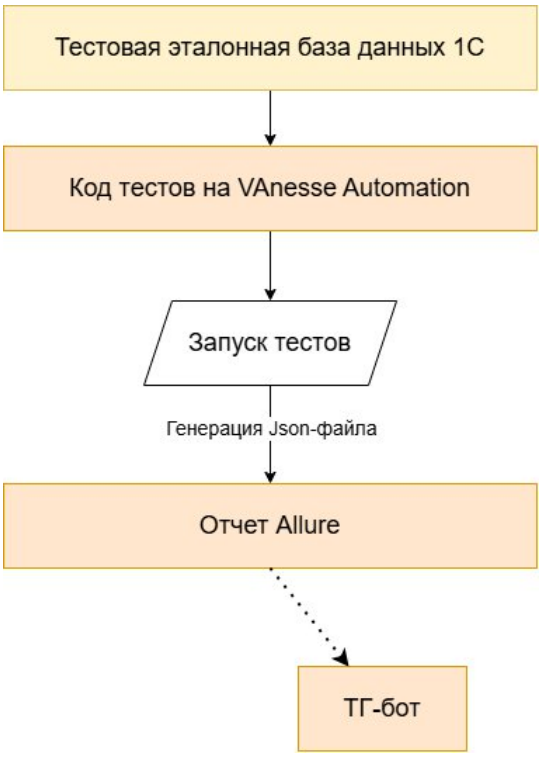

Рисунок 3.2 – Архитектура среды тестирования

Первым этапом станет подготовка тестового окружения. На выделенной виртуальной машине или локальном компьютере будет развернута информационная база «1С» предназначенная для тестирования с актуальной конфигурацией и загруженным набором эталонных данных. После завершения настройки базы в нее загружается внешняя обработка Vanessa-Automation, которая находится в открытом доступе и ее можно скачать без дополнительных затрат.

Чтобы воспользоваться функциональностью инструмента, нужно запустить два экземпляра 1С предприятия, первый с ключом менеджера тестирования (/TESTMANAGER), второй с ключом клиента тестирования (/TESTCLIENT) и установить соединение между менеджером и клиентом тестирования. Менеджер тестирования - клиентское приложение, запущенное для исполнения сценария тестирования. Менеджерами тестирования могут выступать только толстый или тонкий клиенты. Клиент тестирования - клиентское приложение, запущенное для имитации интерактивных действий, описываемых сценарием тестов. Клиент тестирования получает команды от менеджера тестирования, воспроизводит их и возвращает результаты выполнения этих команд (рисунок 3.3). Клиентом тестирования может быть любое клиентское приложение, кроме конфигуратора.

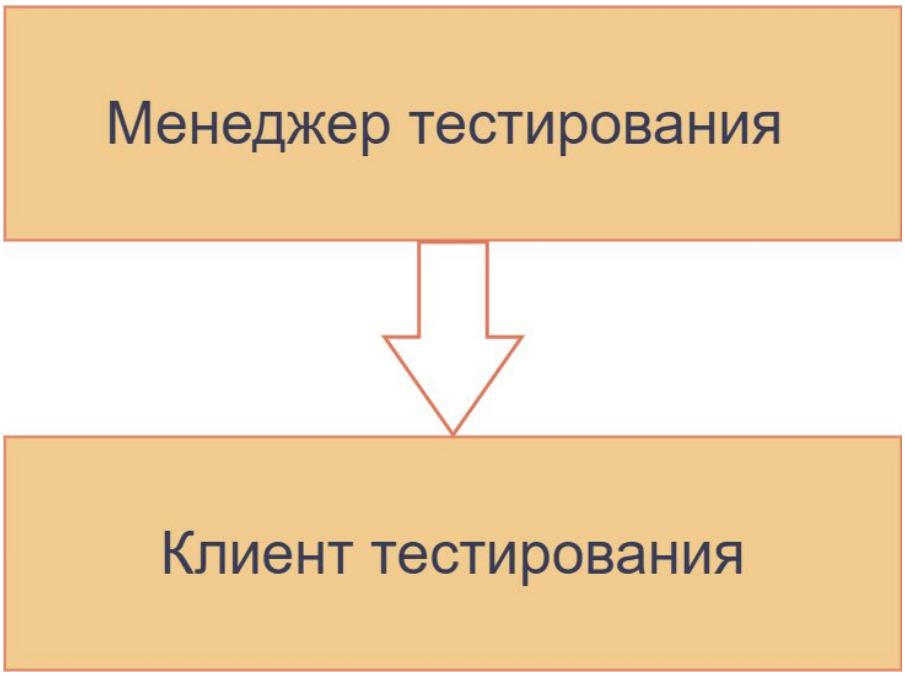

Рисунок 3.3 – Схема взаимодействия менеджер-клиент тестирования

Практическая реализация данного взаимодействия выглядит следующим образом: сначала открывается «1С» в режиме предприятия, которому передается роль менеджера. Затем инициируется запуск второго экземпляра, который получает все необходимые параметры, включая роль клиента. Для установления стабильного соединения и обмена командами между менеджером и клиентом используется протокол TCP (Transmission Control

Protocol). Важно отметить, что архитектура поддерживает работу по схеме «один ко многим»: единственный менеджер может централизованно координировать и контролировать несколько клиентов, работающих одновременно (например, для тестирования под разными ролями пользователей), в то время как каждый клиент подключается только к одному менеджеру.

Написанные сценарии в Vanessa будут храниться в фиче-файлах с соответствующим расширением .feature.

Следующим этапом необходимо будет преобразовать сырые данные в читабельную аналитическую информацию. Для генерации детализированного отчета необходимо будет установить Allure из github. После полного прогона тестовых сценариев, фреймворк Vanessa-Automation формирует набор структурированных данных в формате JSON. Каждый такой файл содержит информацию о соответствующем тесте: его итоговый статус (успех/провал/пропуск), точное время выполнения, детализированную последовательность шагов, а также приложенные артефакты: скриншоты и логи. Эти файлы Allure будет получать на вход, обрабатывать их и создавать по ним отчет.

Завершающим этапом будет интеграция бота Telegram в механизм для организации автоматической отправки уведомлений непосредственно в рабочие чаты команды. Процесс настройки начинается с создания чат-бота через официальный инструмент BotFather, который предоставляет уникальный API-токен для авторизации всех исходящих запросов. Далее создается групповой чат или канал, куда будут поступать сообщения. Эта группа имеет свой уникальный идентификатор, передающийся боту, чтобы сообщения попали в нужный чат.

Подобныя архитектура дает возможность отследить каждый этап процесса тестирования: от запуска клиентов до получения готового отчета в мессенджере. Это позволит быстрее выявить ошибку или удостоверится в корректной работе каждого элемента, что упростит переход на полную автоматизацию процесса.

3.4 Модель данных

В рамках полученной архитектуры данные будут двигаться по следующей цепочке (рисунок 3.4):

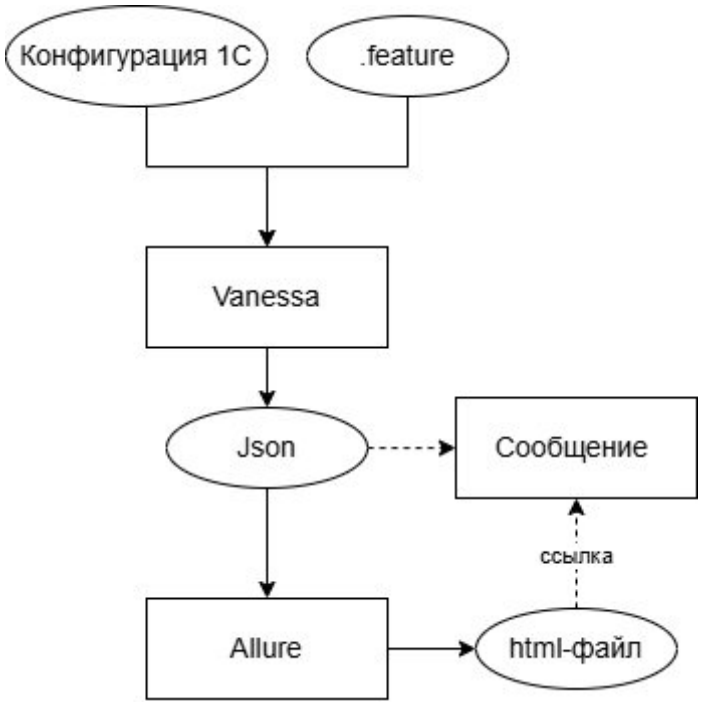

Рисунок 3.4 – Движение данных в рамках среды тестирования

На вход Vanessa Automation должна получить актуальную конфигурацию 1С, в которой содержится код бизнес-процесса закупок товара, и фича-файлы с готовыми тестами. Используя введенные данные, ванесса производит прогон базы сценариев, а результат в json формат файлов помещает в назначенную папку. На основе этих файлов Allure формирует html отчеты, которые также помещаются в папку с остальными отчетами. БАТ-файл, управляющий запуском фреймворков и передвижением данных, собирает краткую сводку о результатах выполнения тестов и создает текстовое сообщение, к которому прикрепляет ссылку на соответтвующиц html файл. Результатом движения файлов станет уведомление в Телеграмм.

3.5 Алгоритм

Весь алгоритм действий работы среды будет прописан в БАТ-файле, соответственно управление данными будет происходить через него.

В качестве первого шага алгоритм должен выполнить выгрузку конфигурации в тестовую базу данных, исключив тем самым риск попадания в рабочую систему изменений, способных повлиять на текущий процесс работы. После подготовки базы, БАТ-файл должен активировать последовательный запуск тестов через Vanessa Automation: сначала юнит-тесты, затем дымовые и сценарные. Завершив прогон тестов, базу необходимо будет снова обновить, чтобы очистить от изменений, полученных в ходе работы. Затем должна произойти выгрузка json-файлов в папку с результатами прохождения автотестов. Следующий этап - подготовка среды Allure. Инструмент принимает на вход путь к каталогу с

результатами. Информация, содержащаяся в файлах созданных Ванессой, расшифровывается в отчет. Кроме того, если в указанном каталоге присутствует подпапка history с данными от предыдущих запусков, Allure автоматически проводит сравнение и строит динамические графики. Результатом работы является новая папка, содержащая отчеты в формате HTML.

В БАТ-файле будет содержаться шаблон текстового сообщения, для которого потребуется получить информацию с итогами тестирования. Чтобы это осуществить необходимо будет прописать функции, которые будут делать это автоматически.

На первом этапе осуществляется парсинг и агрегация результатов тестов. Программа будет анализировать JSON-файл, сформированный инструментом Vanessa-Automation, и извлекать из него ключевые метрики: общее количество выполненных тестов, количество успешно пройденных, количество упавших с ошибками, а также количество не прошедших тестов.

На следующем этапе будет происходить формирование текстового сообщения. На основе извлеченных данных составляется читаемое и информативное сообщение по следующему шаблону (рисунок 3.5).

Рисунок 3.5 – Шаблон текстового сообщения

Кроме того, программа также должна получить ссылку, которое будет добавлено в тело сообщения, на соответствующий отчет.

После успешного сбора информации для текстового шаблона БАТ-файл инициирует отправку уведомления. Для этого будет выполняться запрос к API Telegram (https://api.telegram.org/bot<ТОКЕН>/sendMessage) с передачей параметров в виде идентификатора чата и сформированного ранее текстовое сообщение. В результате чего, уведомление доставляется в созданный чат команде разработчиков.

На этом цикл работы БАТ-файла заканчивается и начинается заново при следующем запуске для повторения прописанного алгоритма.

3.6 Метрики качества

Для оценки практической и финансовой ценности внедрения автоматизированного тестирования следует выделить несколько ключевых метрик. Основной задачей стало определение целесообразности затраченных ресурсов на разработку автотестов в сравнении с ручным подходом.

Первой метрикой стал процент успешного прохождения тестов. Данный показатель вычисляется как отношение количества успешно пройденных тестов к общему числу выполненных тестовых сценариев за один прогон. Формула (3.1) для расчета выглядит следующим образом:

$$\text{Успешное прохождение тестов} = \left(\frac{N_{\text{пройдено}}}{N_{\text{всего}}} \right) \times 100\% , \qquad (3.1)$$

где $N_{\text{пройдено}}$ - количество пройденных тестов;

$N_{\text{всего}}$ - общее количество выполненных тестов;

Высокий процент успеха свидетельствует о стабильности функционала и качестве продукта. Динамика изменения этого показателя от прогона к прогону позволяет отслеживать общую тенденцию - улучшается или ухудшается состояние системы с течением времени. Резкое падение процента успеха часто сигнализирует о появлении критических регрессионных ошибок после внесения изменений в код. В тоже время неизменно высокий процент успешности может свидетельствовать о неполном покрытии кода.

Второй необходимой метрикой является время, затрачиваемое на полный прогон набора автотестов. Этот показатель необходимо учитывать для оценки скорости получения обратной связи от тестовой системы. Сравнение времени автоматического и ручного прогона одного и того же набора тест-кейсов наглядно будет демонстрировать преимущество автоматизации. Для ручного тестирования типичного набора тестов может потребоваться несколько дней, в то время как автоматизированный прогон часто укладывается в несколько часов или минут. Это позволяет запускать тесты значительно чаще, например, после каждого коммита в репозиторий, что сокращает время на обнаружение дефекта.

Третий параметром для анализа будет дополнением ко второй метрике. Он будет содержать в себе стоимость трудозатрат. Эта метрика поможет сравнить финансовые затраты на ручное и автоматизированное тестирование. Для расчета стоимости прогона используется формула (3.2): количество человеко-часов, необходимых для полного проведения всех тестов, умножается на почасовую ставку тестировщика.

$$\text{Стоимость прогона} = t_{\text{прогон}} \times S_{\text{тест}}, \quad (3.2)$$

где $t_{\text{прогон}}$ - время прогона, час;

$S_{\text{тестер}}$ - ставка тестировщика, руб./час;

Эта метрика отобразит, сколько компания сможет сэкономить на работе тестироващика после внедрения автоматизированных тестов.

Четвертым анализируемым параметром будет время разработки одного автоматизированного теста. Данная метрика включает в себя все этапы: проектирование сценария: от написания кода до отладки. Учет этих трудозатрат нужен для расчета окупаемости автоматизации. Первоначальные затраты на создание автотеста, как правило, значительно выше, чем на написание одного ручного тест-кейса. Окупаемость достигается только после многократного использования этого теста в повторяющихся прогонах. Формула (3.3) для оценки трудоемкости разработки выглядит следующим образом :

$$\text{Общее время на разработку} = \sum_{i=1}^{n} \left(t_{\text{разраб 1 теста}} \right)_i, \quad (3.3)$$

где n - общее количество автотестов в проекте;

$t_{\text{разраб 1 теста}}$ - время разработки одного теста, час;

Необходимо будет замерить время на разработку одного автоматического теста, а затем просуммировать все время, полученное на разработку каждого такого сценария.

Пятым и наиболее показательная метрика - стоимость дефекта. Этот показатель оценивает не только прямые затраты на его исправление, но и потенциальные бизнес-потери. Прямые затраты включают время работы разработчика на исправление ошибки и время тестировщика на проведение повторного тестирования после исправлений. Однако потери со стороны бизнеса оценить может быть сложнее, так как они могут быть как финансовые, так и репутационными. Для расчета будет использоваться формула (3.4):

$$\text{Стоимость дефекта} = t_{\text{исправ}} \times S_{\text{разраб}} + t_{\text{тест}} \times S_{\text{тестер}}, \quad (3.4)$$

где $t_{\text{исправ}}$- время на исправление, час;

$S_{\text{разраб}}$ - ставка разработчика, руб./час;

$t_{\text{тест}}$ - время на ретест, час;

$S_{\text{тестер}}$ - ставка тестировщика, руб./час;

Главная цель автоматизации - находить дефекты на самых ранних стадиях разработки или доработки. Дефект, обнаруженный в продакшене, может обойтись в несколько раз дороже, чем тот же дефект, найденный на этапе разработки или в тестовом окружении. Поэтому автоматизированные тесты, интегрированные в непрерывный процесс работы,

позволят выявлять большинство ошибок мгновенно после их появления, что сократит общую стоимость дефектов в проекте.

В итоге, комплексный анализ пяти метрик поможет увидеть полноценную картину экономической эффективности автоматизации. Такой подход нужен для перехода от субъективных ощущений к объективным числовым данным, на основе которых можно принимать взвешенные решения о дальнейшем развитии автоматизированного тестирования.

4 Требования к тестам

4.1 Метод тестирования

Перед началом проектирования автотестов необходимо разработать стратегию тестирования, которая позволит структурировать работу, а также покрывать тестами функции, играющую важную роль в бизнес логике закупок товаров. Это позволит равномерно распределить усилия в процессе работы.

В разработке программного обеспечения существует несколько методологий тестирования. Два наиболее известных - это TDD (Test-Driven Development) и BDD (Behavior-Driven Development).

TDD - это разработка через тестирование, при которой сначала пишутся тесты, а затем код, удовлетворяющий этим тестам. Её суть заключается в строгом следовании короткому циклу, состоящему из трёх шагов: сначала пишется тест на основании тз, который проверяет ещё не реализованную функциональность, затем пишется минимальный объем кода, чтобы этот тест прошёл успешно, и наконец проводится рефакторинг нового кода, чтобы улучшить его структуру, не нарушая работоспособности. (рисунок 4.1).

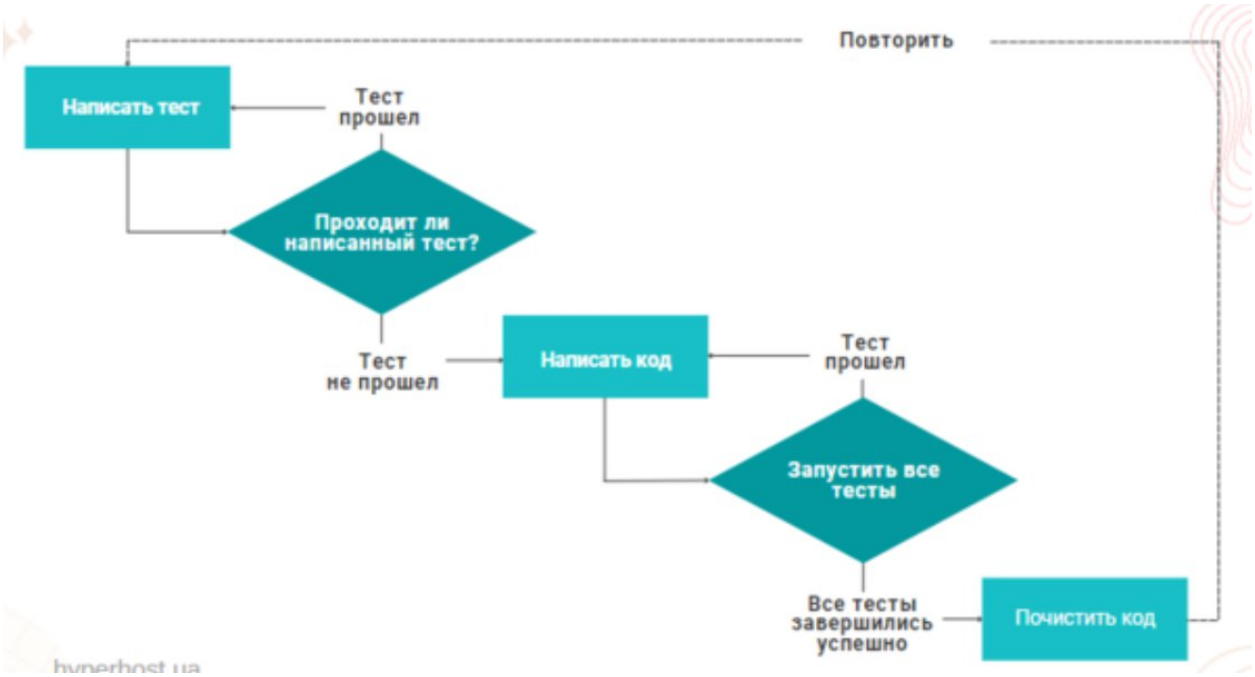

Рисунок 4.1 – Схема работы методологии TDD

Основная цель TDD - создание чистого, рабочего кода. Код, написанный по таким правилам, изначально оказывается покрытым тестами, что значительно упрощает его дальнейшее сопровождение и модификацию. Однако при работе с уже существующими бизнес-процессами, где основная логика уже реализована, эта методология не подходит.

BDD в отличие от TDD, которая фокусируется на технической реализации, направлена на поведение системы с точки зрения пользователя. Тесты в BDD описываются на понятном бизнесу языке, часто с использованием структуры Gherkin (Given-When-Then). Эти сценарии описывают, как система должна реагировать на конкретные действия пользователя в определённом контексте. Поскольку в рамках данной работы автоматизируется уже готовый бизнес-процесс закупки, выбор в пользу BDD становится логичным. Данная методология позволит не погружаться в техническую часть вопроса, а смотреть на механизм с точки зрения пользователя, что сэкономит время на изучение кода. Такой подход также позволит найти возможные упущения в сценариях, которые могут усложнять работу самого процесса закупок.

Дальше необходимо выбрать виды тестов и их соотношение, чтобы обеспечить оптимальное покрытие системы, так как исчерпывающее тестирование невозможно, поэтому важно распределить тесты на жизненно важные функции. Для решения этой задачи используется концепция тестовой пирамиды.

Пирамида тестирования (англ. Testing Pyramid) - концептуальная модель, которая помогает организовать различные виды тестов в процессе разработки программного обеспечения. Цель данной концепции - минимизировать затраты на тестирование при полном покрытии кода и функциональности системы. Чтобы достигнуть её, пирамиду разделяют на несколько уровней, по которым распределяются разные виды тестов. В классической модели есть три таких уровня (рисунок 4.2).

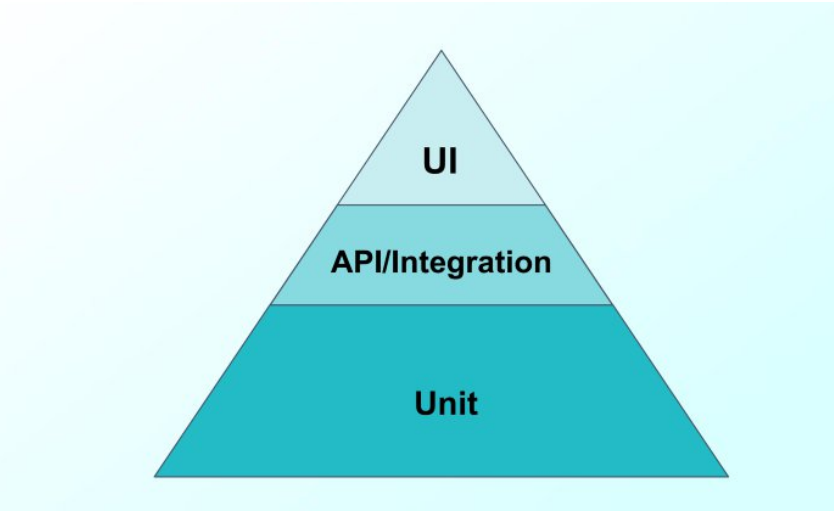

Рисунок 4.2 – Канон тестирования

Основание - это самая большая ячейка, которая состоит из Unit-тестов. Они фокусируются на проверке отдельных компонентов кода. Такие тесты должны быть быстрыми и простыми. Средний уровень - интеграционные тесты. Их должно быть меньше, чем Unit-тестов и они отвечают за проверку взаимодействия между разными модулями

системы. На верхнем уровне пирамиды расположены UI тесты известные как end-to-end. Эти тесты имитируют действия реального пользователя в системе и поэтому они самые медленные и сложные в поддержке, поэтому, как правило, в базе тестов их меньше всего. Таким образом, чем выше уровень, тем меньше тестов, содержатся на нем.

При их написании в рамках этой концепции надо руководствоваться следующими принципами. Во-первых, тесты, которые находятся на верхних уровнях, должны проверять только то, что невозможно проверить на нижних, например, unit-тесты должны покрывать код на модульном уровне, а не интеграционном. Во-вторых, важно исключить дублирование. Тесты не должны проверять логику, которая была покрыта тестами других уровней.

Такая структура позволит найти дефекты и устранить на нижнем уровне до того как запустятся сложные и емкие тесты, которые могут повлиять эти ошибки. Кроме того, такой подход поможет охватить процесс закупок товара на разных уровнях: от технического уровня до бизнес логики.

Однако из-за специфики платформы 1С и используемого инструмента тестирования – Vanessa Automation классическую пирамиду сложно будет применить.

Во-перых, unit-тесты подразумевает под собой проверку изолированных элементов, что в рамках 1С системы не везде можно осуществить. Так как на платформе многие юниты (объекты методанных, элементы интерфейса и т.д) сильно зависят друг от друга, поэтому при попытке изолировать этот код для написания чистых модульных тестов, они могут не пройти или выводить ложный результат. Для них могут потребоваться инструменты направленные исключительно для написания модульных тестов. Во-вторых, фреймвор Vanessa Automation заточена под имитацию пользовательских действий, что делает запуск тестов медленнее, что противоречит задумке о высокой скорости прогона юнит-тестов, поэтому выгоднее делать акцент на пользовательские сценарии. В-третьих, такая пирамида противоречит методологии BDD, где главной задачей является проверкой бизнес-сценариев, которые описывают поведение пользователя, а не просто указания входных и выходных данных. Поэтому стоит использовать перевернутую пирамиду (рисунок 4.3)

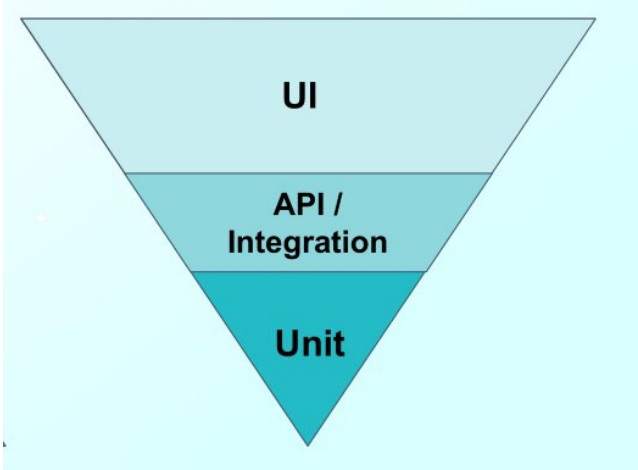

Рисунок 4.3 – Перевернутая пирамида тестирования

В этом подходе основной акцент смещается на стабильные UI тесты, чтобы гарантировать, что ключевые бизнес-процессы будут работать корректно в реальных условиях. Такой подход не отрицает важность разных уровней тестирования, но предлагает более практичное для платформы 1С соотношение, позволяя использовать сильные стороны выбранного инструмента.

4.2 Виды тестов

Для тестирования процесса закупок товаров будут написано 3 вида тестов, среди которых преимущественно будет больше сценарных тестов.

Модульные тесты, как уже было упомянуто выше, представляют собой проверку отдельных частей кода на предмет корректной работы. В случае с платформой 1С, модульные тесты применяются в основном для проверки вычислительных процедур, таких как расчёт дат, сумм, количества и других параметров, которые не зависят от состояния всей информационной базы. Для таких тестов может понадобиться сторонний плагин.

Примером модульного теста в рамках бизнес-процесса закупки товаров может служить проверка расчёта плановой даты отгрузки товара в транспортную компанию в документе «Рейс ТК» (рисунок 4.4).

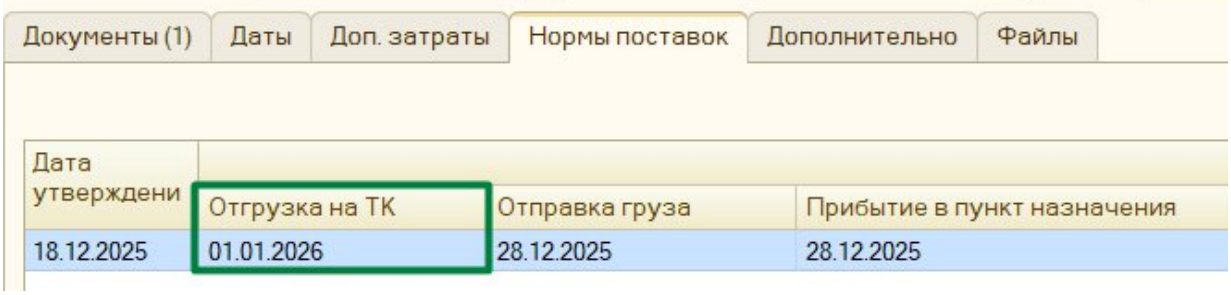

Рисунок 4.4 – Табличная часть документа «Рейс ТК»

Она рассчитывается путем сложения двух переменных. На вход подаются дата утверждения заявки поставщика, например 27.12.25 и норма дней доставки до ТК, например 1 день. Для этого необходимо будет создать документ «Заявка поставщику на отгрузку», где будут находиться вводные данные, затем создать на его основе «Заказ поставщику», и добавить этот заказ в Рейс ТК. В табличной части с плановыми датами должна отобразиться корректно рассчитанная дата, а именно 28.12.25. Тест проверяет, правильно ли система рассчитала плановую дату отгрузки. Если фактический результат совпадает с ожидаемым значит, тест считается пройденным, иначе – нет.

Для модульных тестов на вход могут подаваться различные значения, поэтому необходимо использовать граничные значения, чтобы не перегружать механизм бессмысленными сценариями.

В используемой пирамиде интеграционные тесты будут заменены на дымовые. Дымовые тесты проверяют базовый функционал решений на платформе 1С. Главная цель этих тестов убедиться, что система не «падает» при выполнении стандартный действий. Этот тип тестов позволит убедиться в корректной работе всей системы, а не только в бизнес логике.

Примером дымового теста может быть проведение документа. В процессе передвижения товара диспетчер или логист вводят фактические даты на каждую контрольную точку, после чего документ перепроводится (рисунок 4.5). Суть теста – проверить, что перепроведение документа не меняет движение по регистрам, также проверяется сам факт, что проведение документа не вызвало ошибок.

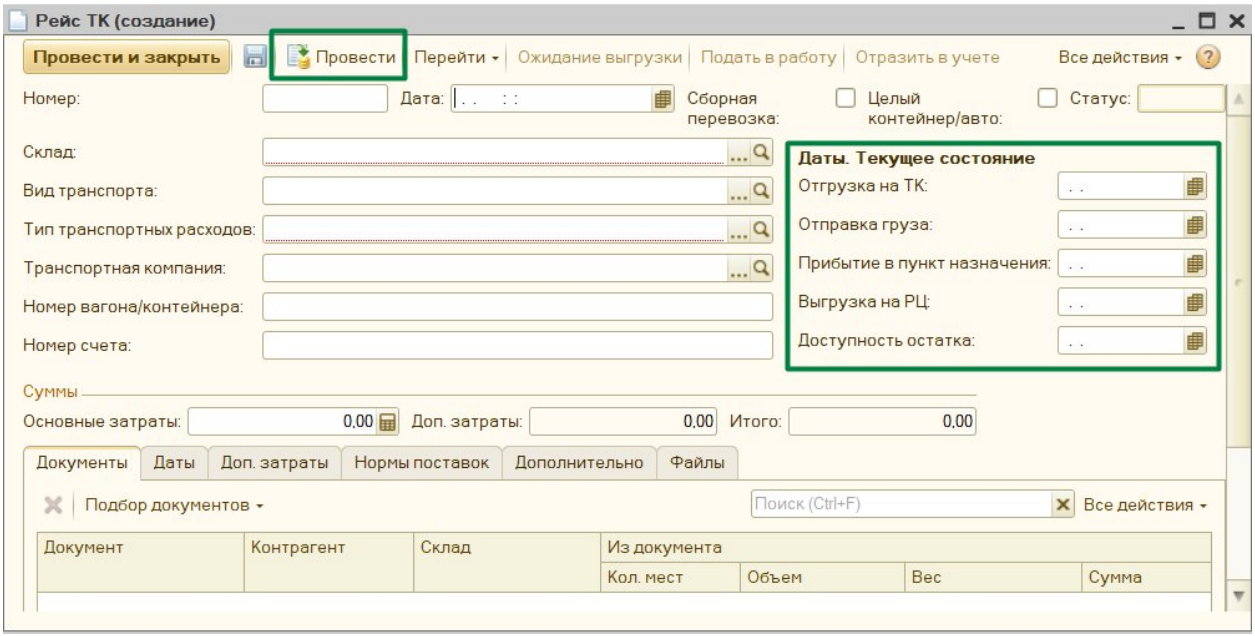

Рисунок 4.5 – Форма документа «Рейс ТК»

Чтобы воспроизвести тест необходимо создать документ рейс ТК, добавить в него необходимые заказы. Затем ввести первую фактическую дату - Отгрузка на ТК, провести документ. После чего добавить следующую дату, а именно Погрузка в ТС, и перепровести документ. Необходимо убедиться, что движение по регистрам за предыдущую дату остались прежними. Если перепроведение прошло, не повлияв на предыдущий результат, тест считается успешным.

Особое внимание будет уделено сценарным тестам. Их цель проверить бизнес-процесса в контексте действий пользователя выполняющего определенные задачи от их начала до логичного завершения. Такие тесты проверяют каждый шаг на соответствие с ожидаемым результатом, подтверждая корректность работы всей цепочки действий. Именно сценарные тесты являются основным видом проверки, поскольку компании важна бесперебойная работа бизнес логик.

Для создания сценарных тестов потребуется разбить бизнес-процесс закупка товаров на отдельные блоки в зависимости пользователя, который сопровождает его. Необходимо проверить каждую последовательность действий в зависимости от условий их появления.

Примером такого теста будет процесс оформление документа «Рейса ТК». За создания данного документа отвечает логист, следовательно, все действия должны происходить под ним. Когда логист получает информацию о готовности поставщиков отгрузить товар в транспортную компанию, первым его шагом будет создание заказов поставщиков на основе утвержденной заявки. Ожидаемым результатом станет добавление заказа в регистр с корректными данными полученных из заявки. Следующим действие будет создание документа Рейса ТК и заполение его заказами. По окончанию этого шага тест должен проверить, что добавленные заказы соответствуют состоянию признака Целый контейнер в документе Рейса и в заявке. Если было найдено не соответствие, то платформа должна вывести предупреждение пользователю. Предположим, что в рамках сценария статусы признаков не соответствует, тогда шаг будет считаться успешно выполненным, если программа обнаружит несоответствие и выведет окно предупреждения. На третьем этапе логист вносит плановые даты отгрузки товара на ТК для каждого заказа. При добавлении таких дат, выставляются регламентные даты. Тест должен проверить, что в соответствующей колонке появились числа корректного формата. Если ожидаемые ячейки были заполнены корректно, значит, этот шаг помечается, как успешный. Завершающим действием будет проведение документа Рейса ТК. Если созданный документ попадает в регистр и имеет статус «проведен», тогда функция работает корректно. Сценарий будет считаться успешно завершенным, если ни на одном шаге не было найдено несоответствий с ожидаемым результатом, иначе тест попадет в список дефектов.

Таким образом, модульные тесты позволят проверить корректность выполнения отдельных простых расчетов и логических операций с разными вводными данными. Дымовые тесты обеспечат контроль стабильности системы при выполнении стандартных операций, например, при проведении документов. Сценарные тесты охватят ключевые бизнес-процессы, имитируя действия пользователей и проверяя всю цепочку операций от начала до завершения. Такой комплексный подход позволяет выявлять ошибки на разных уровнях. В результате тестирование будет охватывать не только бизнес-логику, но и базовые функции платформы 1С, что повышает надежность автоматизированного тестирования в рамках процесса закупок.

Заключение

На основании проведенной работы можно сделать следующие выводы. В ходе анализа бизнес-процессов компании для автоматизации было выбрано – тестирование, а в качестве объекта автоматизированного тестирования был выбран процесс закупки товаров в ERP системе. Данный элемент является одним из самых важных бизнес логик компании, поскольку его стабильность напрямую влияет на финансовые результаты.

Также для разработки среды был выбран набор инструментов, включающий фреймворк Vanessa-Automation для написания тестов, Alluredля формирования отчетов, а Telegram бот для оповещения пользователей о результатах прогона. Этот выбор обусловлен их совместимостью с платформой 1С, активным сообществом и открытими решений.

Для выбранного набора была построена схема взаимодействия всех элементов процесса автоматизированного тестирования: от соединения Vanessa с информационной базой 1С до отправки сообщения в Telegram. Также были рассмотрены действия пользователей при работе с спроектированной средой. Для оценки проделанной работы были выделены ключевые метрики, которые позволят рассчитать материальную и нематериальную ценность разработки. Эти показатели позволяют перейти от субъективных оценок к объективным данным для принятия решений на внедрение.

Выбранные методологии и виды тестов позволят структурировать работу и покрыть большинство функций бизнес процесса закупок товара. Модульные тесты, покроют низкоуровневую логику. Дымные тесты позволят убедиться в работоспособности базовых операций системы, а сценарные тесты - проверить бизнес логику в рамках закупок товаров. Такой подход минимизирует дублирование усилий и оптимизирует общее время выполнения всего тестового набора.

Таким образом, проделанная аналитическая работа дает почву для дальнейшей реализации автоматизированных тестов.

Список использованных источников

1 Мир Упаковки: официал. сайт. – URL: https://khv.mirupak.ru/about/ (дата обращения: 11.11.2025).

2 История ООО «Мир Упаковки» // Sutori. – URL: https://goo.su/jy5gqr6 (дата обращения: 11.11.2025).

3 1С-проекты и тестирование: взгляд QA-специалистов // Хаб. – URL: https://habr.com/ru/companies/simbirsoft/articles/752144/ (дата обращения: 13.11.2025).

4 «Фича-файлы с использованием экспортных сценариев Vanessa Automation» // 1C-KoderLine – URL: https://www.koderline.ru/expert/narabotki/article-ficha-fayly-s-ispolzovaniem-eksportnykh-stsenariev-vanessa-automation/ (дата обращения: 15.11.2025).

5 Тестирование: пример создания сценарного UI теста для платформы 1С // ИНФОСТРАРТ База знаний – URL: https://infostart.ru/1c/articles/813062/ (дата обращения: 16.11.2025).

6 TDD vs BDD: в чем различия и что выбрать для разработки // Кекслет – URL: https://ru.hexlet.io/blog/posts/tdd-vs-bdd-v-chem-razlichiya-i-chto-vybrat-dlya-razrabotki (дата обращения: 18.12.2025).

7 Автоматизация тестирования решений на 1С: Ожидания vs Реальность // ИНФОСТРАРТ База знаний – URL: https://infostart.ru/1c/articles/2407502/ (дата обращения: 18.12.2025).

8 Сценарное тестирование в помощь программисту 1С // Хаб – URL: https://habr.com/ru/articles/307808/ (дадта обращения: 19.12.2025).

9 Тестирование: пример из семи шагов создания Unit-теста для платформы 1С // ИНФОСТРАРТ База знаний – URL: https://infostart.ru/1c/tools/663808/ (дата обращения: 20.12.2025).

10 Старт в юнит-тестах // ИНФОСТРАРТ База знаний – URL: https://infostart.ru/1c/articles/2204036/ (дата обращения: 20.12.2025).

11 YAxUnit или модульное тестирование в 1С // ИНФОСТРАРТ База знаний – URL: https://infostart.ru/1c/articles/1976659/ (дата обращения: 20.12.2025).

12 Формирование дымовых тестов для фреймворка тестирования Vanessa-Automation // GitHub – URL: https://github.com/Tavalik/VA-SmokeTests (дата обращения: 20.12.2025).

13 Дымовые тесты // GitHub – URL: https://github.com/vanessa-opensource/add/blob/develop/tests/smoke/readme.md (дата обращения: 20.12.2025).

14 Разработка и сценарное тестирование с Vanessa-ADD. Отчетность Allure. Автоматизация запуска сценариев // ИНФОСТРАРТ База знаний – URL: https://infostart.ru/1c/articles/1010127/#Отчетность_Allure (дата обращения: 22.12.2025).

15 Vanessa Automation: подготовка сценариев для тестирования в 1С // САЙФИЛД – URL: https://saifield.ru/vanessa-automation/ (дата обращения: 24.12.2025).

16 От экспертов «1С-Рарус»: Подходы к сценарному тестированию на примере 1С:Общепит и 1С:Сценарное тестирование // 1С-РАРУС – URL: https://rarus.ru/publications/20201112-ot-ekspertov-1c-scenarnoe-testirovanie-450864/#osnovnye-principy-raboty-resheniya (дата обращения: 24.12.2025).

17 Лучшие метрики тестирования для повышения качества продукта // KursHub – URL: https://kurshub.ru/journal/blog/luchshie-metriki-testirovaniya-dlya-povysheniya-kachestva-produkta/ (дата обращения: 26.12.2025).