ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

РАЗРАБОТКА АВТОМАТИЗИРОВАННЫХ ТЕСТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРОЕКТА «САУМИ»

Выпускная квалификационная работа
обучающегося по направлению подготовки 38.03.05 «Бизнес-информатика»
заочной формы обучения, группы 07001356
Аленгоз Аделины Руслановны

Научный руководитель:
к.э.н., ст.пр. Ильинская Е.В.

БЕЛГОРОД 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире разрабатываются миллионы новых программных продуктов. А как известно программирование имеет человеческий фактор, и процесс разработки не возможен без ошибок. Программные продукты в ходе разработки подвергаются неоднократным изменениям и, как правило, не одним, а несколькими разработчиками. Современные компании заинтересованы в том, чтобы выпускать максимально качественный продукт в установленный срок для извлечения хорошей прибыли с наименьшими затратами. Для того, чтобы свести затраты к минимуму и выпустить качественный продукт необходимо вести контроль качества, то есть проводить тестирование на всех этапах разработки. Естественно, нельзя сказать, что затраты на тестирование не существенные. Многие компании тратят огромные суммы на создание и подержание работы департамента обеспечения качества, но данные вложения считаются вполне оправданными. Поскольку, не имея информации о качестве, нельзя выпустить достойный продукт. Можно потратить куда больше ресурсов и времени на решение неполадок постфактум и будет не удивительно, что такая компания на рынке ПО будем менее конкурентоспособна наравне с другими.

Актуальность темы заключается в том, что современные программные продукты – это сложный объект и справиться с ним «ручным методом» очень дорого и трудно. Для того что бы снизить затраты на тестирование и увеличить объем проверок применяются средства автоматизации.

Под автоматизацию обычно попадают сценарии проверки основного функционала, самых критических мест, мест сложных для проверки вручную или рутинные сценарии «ручных» тестировщиков.

Актуальность темы работы связана со значительным распространением автоматизированного тестирования и заключается в необходимости разработки рекомендаций по автоматизации работы тестировщиков.

Объект исследования: компания ООО «Бюджетные и финансовые технологии» – ведущий российский разработчик программных и консалтинговых решений для госсектора и бизнеса

Предмет исследования: проект «САУМИ» - автоматизированная система управления государственной и муниципальной собственностью.

Цель – увеличить объем проверок и снизить трудозатраты на выполнение постоянно повторяемых идентичных тестов посредством автоматизации тестирования на проекте САУМИ для предприятия ООО «БФТ».

Для достижения указанной цели поставлены следующие задачи:

– определить теоретические аспекты тестирования;

– провести анализ деятельности предприятия и основных бизнес-процессов;

– определить целесообразность автоматизации тестирования на проекте «САУМИ»;

– спроектировать информационную модель предметной области;

– разработать автоматизированный тест программного обеспечения;

– провести анализ и расчет экономической целесообразности автоматизации тестирования.

Методы исследования:

– эксперимент;

– теоретический анализ;

– беседа;

– классификация.

Структура работы обусловлена предметом, целью и задачами исследования. Работа состоит из введения, трех глав, заключения, списка литературы и двух приложений. Содержит 25 рисунков и 4 таблицы.

Введение раскрывает актуальность, определяет степень научной разработки темы, объект, предмет, цель, задачи и методы исследования, раскрывает теоретическую и практическую значимость работы.

В первом разделе рассматриваются понятие тестирования и качества, классификация, виды и методы тестирования, жизненный цикл разработки программного обеспечения, процесс тестирования. Во втором разделе раскрываются особенности деятельности и организационной структуры предприятия, произведен анализ бизнес-процессов, определена целесообразность автоматизации тестирования. Третий раздел посвящен проектированию информационной модели предметной области, разработке автоматизированных тестов и расчетам их экономической целесообразности.

В заключении подводятся итоги исследования, формируются окончательные выводы по рассматриваемой теме.

1 Теоретические аспекты процесса тестирования

1.1 Определение понятия и классификация видов тестирования

Тестирование – это проверка, определяющая степень соответствия реального поведения программы ожидаемому результату, в ходе выполнения специальных тестов. Цель тестирования заключается в своевременном обнаружении и предотвращении дефектов, для повышения уверенности в качестве продукта и предоставления информации для принятия решений. Дефектом в тестировании является изъян в программе, который вызывает несоответствие реальных результатов с ожидаемыми.

Тестирование и качество взаимосвязаны [1]. Существуют такие понятия как обеспечение качества (Quality Assurance) и контроль качества (Quality Control).

В обеспечение качества входит:

– усовершенствование процессов;
– контроль качества;
– управление изменениями.

Контроль качества – часть активности по обеспечению качества.

В контроль качества входит:

– тестирование;
– рецензирование кода;
– статический анализ кода;
– внешняя оценка и аудит.

Получается, что тестирование – одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis).

Схема тестирования представлена на рисунке 1.1.

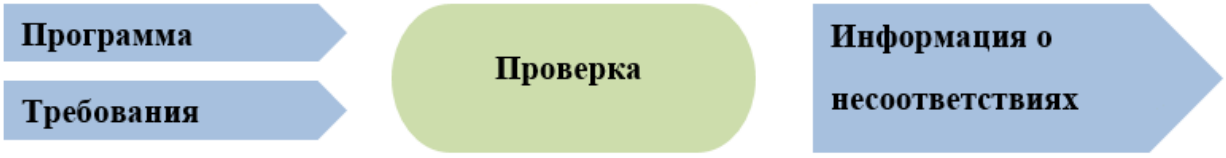

Рисунок 1.1 – Общая схема тестирования

На входе тестировщик получает доступ к программе, которую необходимо протестировать и требования, исходя из которых, проводится тестирование. Наблюдая за работой программы, при определенных условиях и действиях на выходе тестировщик получает информацию о том, где работа программы соответствует требованиям, а где нет. Полученная информация используется для исправления несоответствий или корректировке требований непосредственно в ходе разработке самого продукта.

Специалист по тестированию занимается тем, что производит действия, имитирующие поведение реальных пользователей для проверки поведения программы, а затем проводит сравнение реального поведения программы с ожидаемым.

Следовательно, тестирование как такого не может повлиять на качество продукта, но может своевременно определить проблемную область и дать необходимую информацию для ее решения. Задача тестировщика заключается не только в обнаружении дефектов, но и в том по какой причине они возникают. При таком подходе к процессу тестирования разработчики могут быстро и эффективно устранять ошибки.

Для тестирования программного продукта, применяются различные методы и виды тестирования [2]. Методы тестирования представлены в приложении А. Существует довольно подробная классификация, где виды тестирования сгруппированы по следующим признакам: по объекту, по субъекту, времени, позитивности, изолированности, степени автоматизированности, степени подготовки к тестированию.

По объекту тестирования:

– Функциональное тестирование (Functional Testing).

Функциональное тестирование является одним из самых распространенных видом тестирования, поскольку данный вид тестирования позволяет определить насколько, соответствует программное обеспечение, требованиям заказчика с точки зрения функционала, то есть проверка на то, может ли приложение решать требуемые задачи.

Функциональное тестирование по объекту включает в себя:

– Тестирование безопасности (Security Testing). Тестирование безопасности – это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным;

– Тестирование взаимодействия. (Interoperability Testing)

Тестирование взаимодействия – это тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя тестирование совместимости, и интеграционное тестирование.

– Нефункциональное тестирование (Non-functional Testing). Это тестирование всех свойств программы, не относящихся к функциональности системы. Такими свойствами могут быть предъявленные характеристики с точки зрения таких параметров как: надежность, производительность, удобство, масштабируемость, портируемость и другие качества.

Нефункциональное тестирование по объекту включает в себя:

– Тестирование удобства пользования (UI Testing). В ходе юзабилити-тестирования выявляется насколько приложение удобно в использовании и насколько соответствует заданным требованиям и прототипу. Задача тестирования заключается в выявлении структурных и визуальных недостатков в интерфейсе.

– Инсталляционное тестирование (Installation Testing). Данный вид тестирования проводится для получения информации о корректности установки приложения в искусственно созданной среде, с целью выявления ее готовности

к эксплуатации. Причина проведения подобного тестирования заключается в необходимости проверки инсталяции приложения в зависимости от количества свободного места на винчестере, или разработанных системных требований. Подобное тестирование может исключить проблемы с установкой приложения на различных устройствах и дает возможность проработать системные требования.

– Конфигурационное тестирование (Configuration Testing). Специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.).

– Стрессовое тестирование (Stress Testing). Проверка работоспособности приложения и системы в целом, способность к регенерации после прекращения стрессового воздействия.

– Нагрузочное тестирование (Performance and Load Testing). Проверка работоспособности системы при определенном количестве пользователей.

– Тестирование стабильности и надежности (Stability / Reliability Testing). Проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки;

– Объемное тестирование (Volume Testing). Получение оценки производительности при увеличении объемов данных в базе данных приложения;

По субъекту тестирования:

– Альфа-тестирование (Alpha Testing). Данный вид тестирования обычно проводится на раннем этапе разработки как правило самими разработчиками, заключается в имитации реального использования продукта для систематической проверки всех функций программы.

– Бета-тестирование (Beta Testing). Интенсивное тестирование почти готового продукта на протяжении долгого времени с целью выявления и устранения всех ошибок до выпуска. Имитируется реальное использование

продукта, оценивается стабильность работы. По мимо тестировщиков могут привлекаться реальные пользователи.

По времени проведения тестирования (Тестирование, связанное с изменениями)**:**

– Дымовое тестирование (Smoke Testing). Дымовое тестирование рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции.

В ходе поверхностного тестирования проверяется работоспособность основных функций, наиболее важных модулей, приложения на предмет возможности выполнения требуемых задач и наличия быстро находимых критических и блокирующих дефектов. Если такие дефекты в системе отсутствуют, то дымовое тестирование считается пройденным. В противном случае тестирование считается проваленным, и приложение отправляется на доработку.

Рекомендуется делать автоматизацию дымовых тестов, для ускорения последующего процесса тестирования и обнаружения дефектов на ранних стадиях разработки программного обеспечения.

– Регрессионное тестирование (Regression Testing). Данный вид тестирования направлен на проверку изменений в новой версии сборки для того, чтобы удостовериться, что предыдущие ошибки были исправлены, а новые изменения не принесли за собой новые ошибки в функционале.

Как правило, для регрессионного тестирования используются тест кейсы, написанные на ранних стадиях разработки и тестирования. Это дает гарантию того, что изменения в новой версии приложения не повредили уже существующую функциональность. По критерию позитивности сценария:

– Позитивное тестирование (Positive Testing). Обычные тестовые сценарии, при выполнении которых не должно возникать проблем в использовании приложения. Если проблем нет, в таком случае позитивное тестирование считается пройденным.

– Негативное тестирование (Negative Testing). Сценарии, соответствующие нештатному поведению программы для проверки поведения в экстренных ситуациях. Это позволяет удостовериться в том, что программа выдает правильные сообщения об ошибках, не повреждает пользовательские данные и ведет себя корректно в целом при ситуациях, в которых не предусмотрено штатное поведение продукта.

По степени изолированности тестируемых компонентов:

– Компонентное тестирование (Component Testing). Процесс в программировании, который позволяет проверить единицы исходного кода, наборы из одного или более модулей программы.

– Интеграционное тестирование (Integration Testing). Тестирование, которое проводится после компонентного тестирования, получая информацию о протестированных модулях, группирует их в крупные множества и выполняет тесты над ними. После чего, можно приступить к системному тестированию.

– Системное тестирование (System Testing). Тестирование программного обеспечения, выполняемое на интегрированной среде, для проверки соответствия системы ожидаемым требованиям.

По степени автоматизированности тестирования:

– Ручное тестирование (Manual Testing). Тестирование, которое проводится без использования программных средств, для проверки приложения путем имитации действий пользователя.

– Автоматизированное тестирование (Automated Testing). Автоматизированное тестирование использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

Под автоматизированным тестом (авто-тест) понимается скрипт, имитирующий взаимодействия пользователя с приложением, который выполняется с помощью определенных программных средств.

К преимуществам автоматизированного тестирования можно отнести:

– сокращение времени на выполнение тестов в сравнении с

ручным тестированием;

– минимизация влияния человеческого фактора;

– возможность исполнения авто-тестов внерабочее время тестировщиков;

– раннее обнаружение дефектов;

– вероятность обнаружения большего количества дефектов;

– увеличение скорости появления нового функционала;

– увеличение частоты обновления приложения.

К недостаткам можно отнести:

– требуется более квалифицированный персонал;

– повторяемость – все написанные тестовые сценарии всегда будут выполняться однообразно. При ручной работе тестировщик может обратить внимание на какие-либо детали, выполнив дополнительные действия для нахождения дефекта. Авто-тест этого сделать не сможет;

– затраты на сопровождение – чем чаще меняется функционал, тем чаще необходимо дорабатывать сценарии;

– смешанное / полуавтоматизированное тестирование (Semi-Automated Testing).

Тестирование, которое предполагает, как и использование средств для автоматизации для конкретных тестовых случаев (рутинные сценарии), так и использование ручного способа тестирования там, где автоматизация не оправдана.

По степени подготовки к тестированию:

– Тестирование по документации (Documented Testing). Тестирование, которое производится по заранее написанным тест-кейсам (тестовым сценариям).

– Интуитивное тестирование (Ad Hoc Testing). Тестирование, которое проводится без использования какой-либо документации на интуитивном уровне в попытке найти дефекты. Такое тестирование может проводиться в случае, если документация по функционалу отсутствует, но есть общее

понимание как в принципе это должно работать. Иногда данный вид тестирования полезен, поскольку, тестируя по определенным пользовательским кейсам можно пропустить некоторые ошибки, а интуитивный подход дает больше воли для того, чтобы поломать систему. С другой стороны, дефекты, найденные таким образом, которым пользователь может никогда и не столкнуться считаются незначительными и не всегда тратятся ресурсы на их исправления.

Таким образом, тестировщик, используя различные виды тестирования, может определить проблемную область в работе программного обеспечения, выявить несоответствие фактической работы программы с ожидаемым результатом, найти причину и представить подробный отчет разработчикам для устранения неисправностей.

1.2 Жизненный цикл разработки программного обеспечения и место тестирования в нем

Под жизненным циклом разработки программного обеспечения понимается временной отрезок, начиная с принятия решения о необходимости создания продукта до полного его изъятия из эксплуатации. Представления жизненного цикла могут отличаться в зависимости от компании и производимого продукта, но суть остается одной. Для начала, необходимо разобраться на какие этапы разбивается данный цикл [3].

Существует 6 основных этапов цикла разработки программного обеспечения:

– концепция;
– требования;
– проектирование;

– реализация;

– тестирование;

– выпуск.

Исходя из представленных этапов, можно выделить основные этапы тестирования:

– анализ и планирование тестирования;

– подготовка к тестированию;

– проведение тестирования и отчетность.

Этапы тестирования на жизненном цикле ПО представлены на рисунке 1.2.

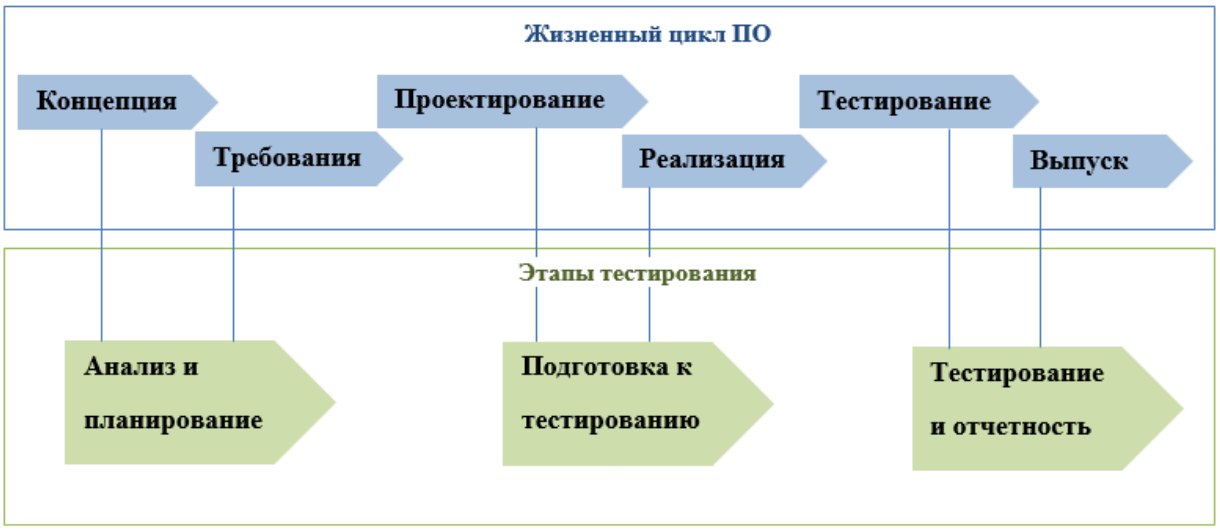

Рисунок 1.2 – Жизненный цикл разработки ПО и этапы тестирования

На этапах создания концепции и требований, производится анализ требований, разработка чек-листа, разработка стратегии тестирования. Определяются задачи тестирования и приоритеты. Производится предварительная оценка трудозатрат.

На этапах проектирования и реализации производится подготовка среды (стенда) для тестирования, ревью требований и чек-листа. Формируются тестовые сценарии. Определяются случаи для автоматизации. Производится корректировка приоритетов и трудозатрат.

На этапах тестирования и выпуска производится непосредственное выполнение тестовых сценариев, выявление и документирование дефектов, верификация билдов, проведение различных видов тестирования. Подготавливается отчет о ходе тестирования и отчет о готовности версии.

Таким образом жизненный цикл разработки ПО определяет основные этапы, по которым проводятся работы всех участников проекта. А этапы тестирования непосредственно определяют работы группы тестирования от зарождения проекта до его выпуска.

1.3 Описание процесса тестирования

Тестирование представляет собой процесс проверки программного продукта, на соответствие требований заявленным заказчиком. Осуществляется данный процесс путем специально созданных искусственных ситуаций, максимально приближенных к реальному поведению пользователя.

Из требований заказчика разрабатываются тесты, и формируются в тест-кейсы (группы тестов) и выполняются на ПО. После выполнения таких тестов мы получаем необходимую информацию о работе приложения, о ее соответствиях и не соответствиях.

Процесс тестирования схематично показан на рисунке 1.3.

Рисунок 1.3 – Процесс тестирования

Тест-кейс (ТК) – тестовый артефакт, суть которого заключается в выполнении некоторого количества действий и/или условий, необходимых для проверки определенной функциональности разрабатываемой программной системы [6].

Для выполнения таких кейсов нет необходимости знакомиться с требованиями, поскольку в них пошагово описаны действия необходимые при проверки конкретной функциональности и описан ожидаемый результат.

Тест-кейсы должны составляться таким образом, чтобы любой человек даже не имеющий отношения к тестированию смог пошагово проверить функционал и выявить несоответствия с ожидаемым результатом.

Готовый тест-кейс представлен на рисунке 1.4.

	Шаг теста	Данные теста	Ожидаемый результат
1	Перейти в пункт меню СПРАВОЧНИКИ->СИСТЕМА->РОЛИ ПОЛЬЗОВАТЕЛЕЙ		Открылась списковая форма <Роли пользователей системы>
2	Нажать кнопку "Новый" на панели инструментов		Открылась форма <Новая роль пользователя>
3	Заполнить поля	[Наименование]={Роль} [Описание]={Описание} [Тип]={Функциональная} [Роль для ЭП]=чекбокс активен [Заблокирована]=чекбокс не активен	Данные отражаются в полях
4	Нажать кнопку "ОК"		Форма <Новая роль пользователя> закрылась. Созданная запись отражается на списковой форме

Рисунок 1.4 – Пример тест-кейса

Такие тест-кейсы собираются в цикл по тематике, чтобы группы тестов можно было выполнять последовательно и отслеживать какие кейсы пройдены, а в каких были обнаружены дефекты.

Таких циклов может быть несколько, их собирают с группу, каждый цикл создается на какой-то конкретный функционал.

Таким образом, как тест-кейсы были созданы и собраны в циклы можно приступить к их выполнению.

Выполнение тест-кейсов может быть оправдано, когда есть необходимость проверить основной функционал системы, не используя требования. Бывают такие ситуации, что тестированием занимается не только группа тестировщиков, но и так же привлекаются лица, никак не связанные с данным родом занятия. В таких случаях и выполняются данные ТК.

Руководитель направления тестирования сообщает о необходимости выполнения тест-кейсов. Назначает сотрудника ответственного за их выполнение. Действия сотрудника:

– открыть цикл, т.к кейсы связаны последовательно в нем;

– открыть ТК, перевести в статус «В работе»;

– пошагово выполнить ТК в программном продукте;

– проанализировать полученные результаты и сопоставить с ожидаемыми результатами описанных в ТК;

– при обнаружении дефекта перевести в статус «Fail» («Не пройден»), и вынести отдельный дефект;

– при успешном выполнении ТК, перевести в статус «Pass» («Пройден»);

– перейти к следующему;

– по окончанию всех ТК цикла, цикл считается выполненным. Результаты в процентном соотношении показываются рядом с циклом.

На создание и выполнение тест-кейсов уходит большое количество времени, так же имеются определенные риски, связанные с тем, что подобные тесты охватывают лишь малую часть и могут быть попросту не эффективны в обнаружении дефектов если проверять только, используя шаги ТК. Для принятия решения о проведении таких тестов необходимо учитывать их целесообразность, а также брать во внимание тот факт, что проект имеет ограничения по времени и бюджету. Целесообразно автоматизировать подобные тест кейсы и при необходимости запускать их на отдельной машине, что сэкономит трудозатраты, поскольку в то время пока выполняются авто-тесты по основному функционалу, параллельно ручной тестировщик может тестировать программу на более глубоком уровне.

2 Анализ предметной области и целесообразность автоматизации тестирования в компании ООО «БФТ» на проекте «САУМИ»

2.1 Анализ деятельности предприятия ООО «Бюджетные и финансовые технологии»

Компания ООО «Бюджетные и финансовые технологии» основанная в 1997 году, является ведущим российским разработчиком проектных решений на базе собственных методологических и программных продуктов для государственного сектора и бизнеса [7].

Компания ООО «БФТ» входит в состав:

– Экспертной группы при Координационной комиссии по созданию и развитию государственной интегрированной информационной системы управления общественными финансами «Электронный бюджет».

– Рабочей группы Министерства финансов Российской Федерации по вопросам совершенствования государственного (муниципального) контроля.

– Консультативного совета по вопросам развития инфраструктуры электронного правительства при Министерстве связи и массовых коммуникаций Российской Федерации.

– Межведомственной комиссии по информатизации в сфере закупок при Министерстве экономического развития Российской Федерации.

Компания ООО «БФТ» является лидером по числу реализованных централизованных систем в госсекторе, что подтверждают следующие цифры:

– 98 побед во Всероссийских конкурсах и рейтингах Министерства финансов;

– реализовано более 3800 проектов;

– 22 субъекта в которых реализованы централизованные системы в госсекторе;

– 20 субъектов, в которых располагаются центры технической поддержки и сопровождения;

– сотрудничество с 9500 муниципальными образованиями.

Продуктовая линейка Компании охватывает ключевые сферы государственного и муниципального управления:

– централизованные ERP-системы для государственного и муниципального управления;

– BI-системы для оценки эффективности государственного и муниципального управления;

– автоматизация управления имущественными и земельными отношениями;

– методическое сопровождение деятельности органов власти (Консалтинг);

– проектное управление;

– портальные и мобильные решения;

– автоматизация деятельности МФЦ;

– колл-центры и контакт-центры;

– ECM-система для управления документооборотом;

– защита информации;

– ИТ-решения для коммерческих организаций;

– администрирование государственных и муниципальных платежей.

Подтверждением эффективности работы ООО «Бюджетные и финансовые технологии» являются награды за участие в различных профессиональных конкурсах ИТ-проектов и ИТ-решений представленные на рисунке 2.1.

Рисунок 2.1 – Награды ООО «БФТ»

Программные продукты ООО «БФТ» разрабатываются с учетом бюджетной реформы, всех требований законодательства РФ и интегрированы друг с другом.

Компания ООО «Бюджетные и финансовые технологии» стремиться к постоянному совершенствованию выпускаемой продукции, территориальному расширению и повышению количества счастливых клиентов. Для этого компания придерживается своей миссии и цели.

Миссия - создание инструментов для повышения эффективности государственного управления, бизнеса и взаимодействия государства и гражданина.

Цель - предоставление лучших по качеству и эффективности продуктов максимальному количеству пользователей в каждой из отраслей экономики, с которой работает Компания.

Компания представляет собой большую и развитую структуру, состоящую из множества департаментов, производственных центров (ПЦ), управлений, отделов.

На рисунке 2.2 представлена небольшая часть организационной структуры, которая связана с разработкой и выпуском программного продукта.

Рисунок 2.2 – Упрощенная организационная структура

Организационная структура является линейно-функциональной. Данная структура базируется как на линейных, так и на функциональных полномочиях. Линейные полномочия передаются вертикально, непосредственно начиная с руководителя предприятия – иерархия. Функциональность заключается в присутствии на одном уровне группы лиц, имеющих одинаковый уровень подчинения вышестоящим звеньям в иерархичной цепи, в то же время каждая группа выполняет свои конкретные задачи и обязанности. Функциональные элементы взаимодействуют друг с другом, но в работу друг друга не вмешиваются.

По большей части структура строится по привязке к производственным центрам. Каждый производственный центр занимается рядом программных продуктов. Например, в производственном центре ЖКХ ведутся работы с АЦК-Капитальный ремонт, АЦК-РИС ЖКХ, АЦК-РИС ГЖН, АЦК-СиАР. В производственном центре находятся отделы разработки, аналитики, внедрения, сопровождения.

Департамент обеспечения качества (ДОК) разбито по направлениям со своими руководителями:

– руководитель направления ЖКХ;
– руководитель направления АЦК-Планирование;

– руководитель направления АИС УБП

– руководитель направления АЦК-Финансы.

Для управления проектами ООО «Бюджетные и финансовые технологии» используют продукты от Atlassian – это австралийская компания разработчик программного обеспечения для управления разработкой программного обеспечения. Одними из самых широко используемыми продуктами являются - Jira и Confluence. Именно эти продукты и использует ООО «БФТ».

Jira непосредственно используется для управления проектами компании, данная система помогает организовать и отслеживать работу по проектам всей команды.

Данный продукт имеет массу преимуществ для всех подразделений команды:

Со стороны руководителей:

возможность планирования спринтов и распределения задач между разработчиками/тестировщиками;

– возможность расставления приоритетов задач, что позволяет сосредоточиться в первую очередь на особо значимых проблемах;

– возможность распределения времени на конкретную задачу и сотрудника, отслеживание трудозатрат;

– составление наглядной отчетности с помощью диаграмм и графиков. Например, можно составить диаграмму по количеству созданных задач тестировщиками в количественном или процентном соотношении. Либо отслеживать выполнение значимых задач.

Со стороны аналитика/ разработчика / тестировщика:

– составление фильтров прямо на рабочем столе по задачам различным проектов, что помогает видеть текущие задачи, назначенные на конкретного сотрудника и не терять их из виду;

– удобный пользовательский интерфейс;

– возможность вести работу над задачами и создавать их;

– быстрый поиск необходимого проекта или конкретной задачи;

– удобное отслеживание личных трудозатрат. Можно видеть сколько было потрачено на конкретную задачу и какого числа.

Со стороны всех пользователей (интеграция):

– интеграция с Git репозиториями;

– интеграция с Jenkins. Работая над задачей можно сэкономить время, на том, что нет необходимости идти в Jenkins и искать в какой версии были сделаны исправления по задаче. Благодаря интеграции прямо в задаче можно увидеть в какой сборке и на какие проекты попали изменения по задаче.

– интеграция с Confluence. В один клик можно просмотреть постановку по конкретной задаче.

Confluence используется по большей части для работы над контентом всей команды по определенному проекту. Так же там содержится различная документация по проектам, инструкции, организационная информация. Могут создаваться чек-листы приоритетных задач к показу, благодаря интеграции с Jira, можно наблюдать, не покидая Confluence статус работ по задачам, тематику и людей, работающих над задачей.

Стандартный процесс работы в Jira представлен на рисунке 2.3.

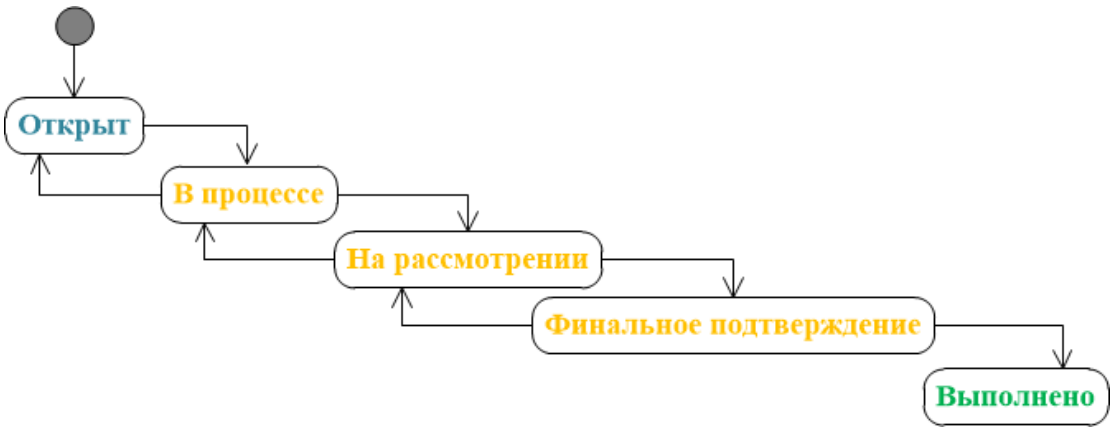

Рисунок 2.3 – Процесс работы в Jira

Не зависимо от выполняемой работы существуют определенные статусы процесса, которые присваиваются при решении любой задачи на проекте на каждом этапе работы.

– открыт;
– в процессе;
– на рассмотрении;
– финальное подтверждение;
– выполнено.

Данные статусы присваиваются последовательно, но при необходимости можно вернуть статус назад. Как показано стрелочками на рисунке 2.3. Возвращаться можно неоднократно до тех пор, пока финальное подтверждение не будет пройдено. Таким образом процесс работы над задачей считается завершенным и переходит в статус «Выполнено».

Основные бизнес-процессы протекают при решении текущих задач.

– Задачи формируются в Jira. Данный продукт помогает отслеживать ошибки.

– Создание запроса в Jira.

При создании запроса первым делом выбирается проект, в котором необходимо завести задачу. Представлено на рисунке 2.4.

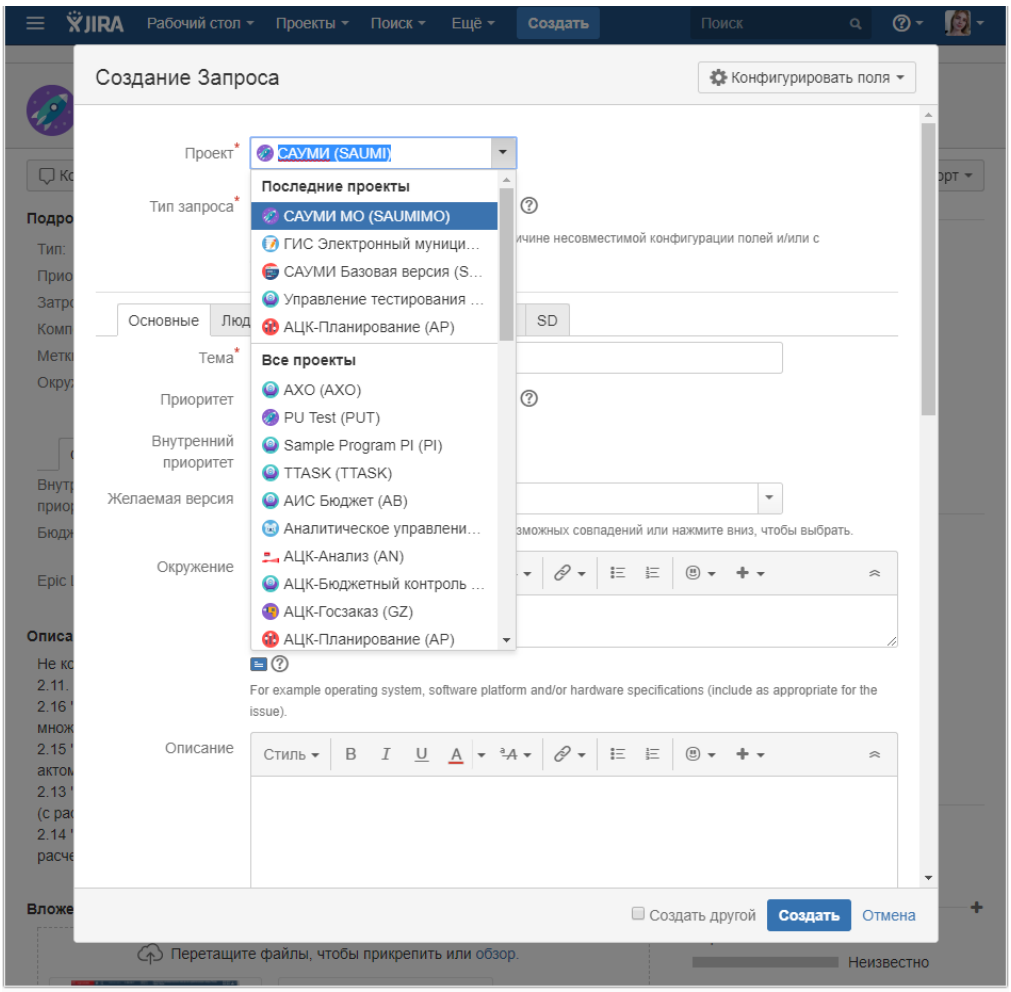

Рисунок 2.4 – Создание запроса

Выбрать тип запроса, как на рисунке 2.5.

Типы запроса делятся на:

– документирование;
– поручение;
– доработка;
– бизнес-задача;
– test;
– тестирование;
– дефект;
– аналитика.

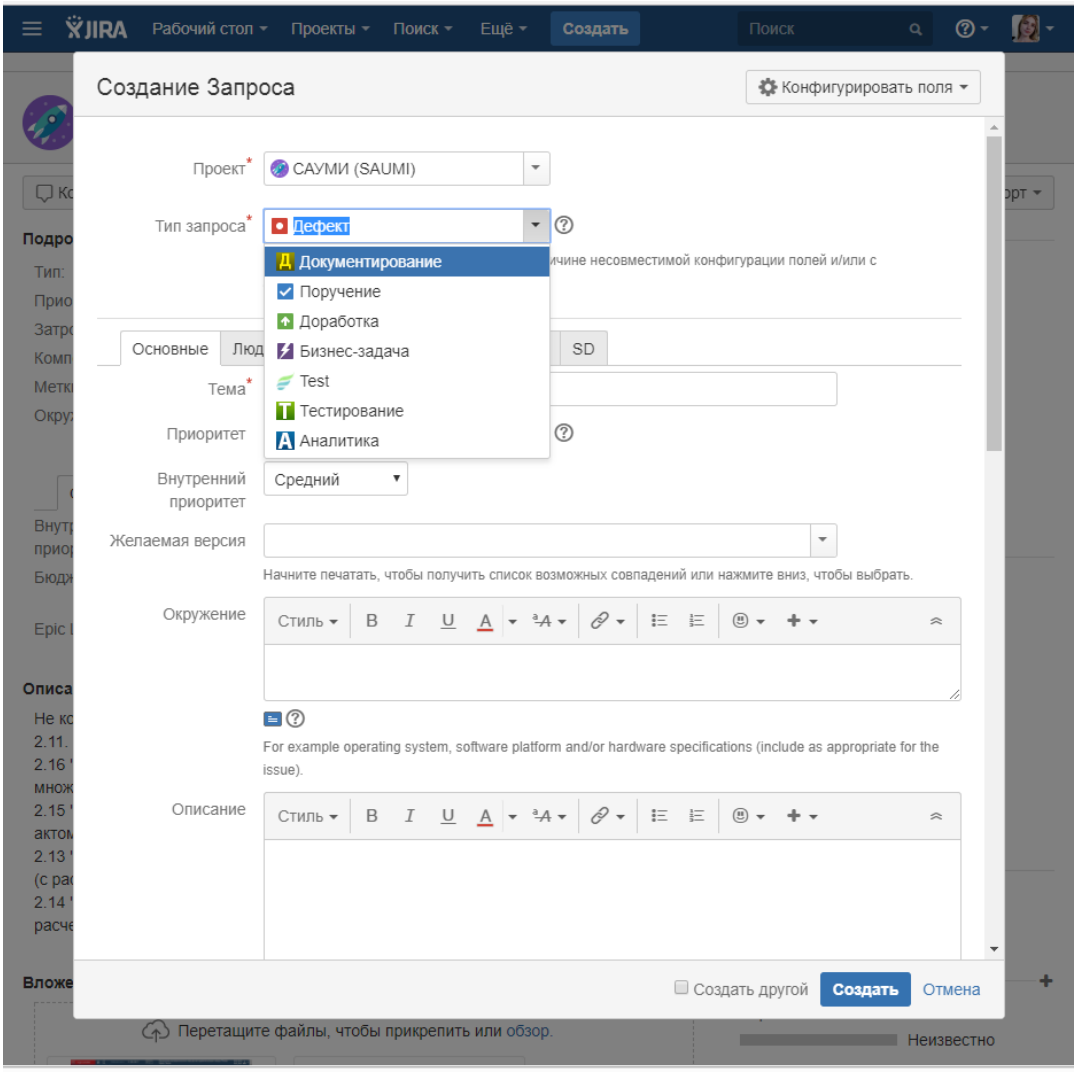

Рисунок 2.5 – Типы запросов

Далее уже в зависимости от выбранного типа запроса заполняется тело задачи с обязательными атрибутами, такими как:

– тема;
– приоритет;
– окружение;
– описание;
– вложение.

В зависимости от выбранного типа, решается, кто будет работать над задачей.

Основные типы, которые часто используются это:

– поручение;
– доработка;

– дефект.

В задаче типа «Поручение» существуют подзадачи аналитики, разработки и тестирования.

В задаче типа «Доработка» существуют подзадачи аналитики, разработки, тестирования, документирования.

В задаче типа «Дефект» существуют подзадачи разработки и тестирования.

Не смотря на разделение обязанностей по типу задачи, тем не менее, разработка ПО – командная работа, поэтому в ходе решения задачи могут принимать участие все указанные лица.

По типам задач можно выделить следующие бизнес-процессы:

На рисунке 2.6. представлен бизнес-процесс решения задачи под названием «Дефект».

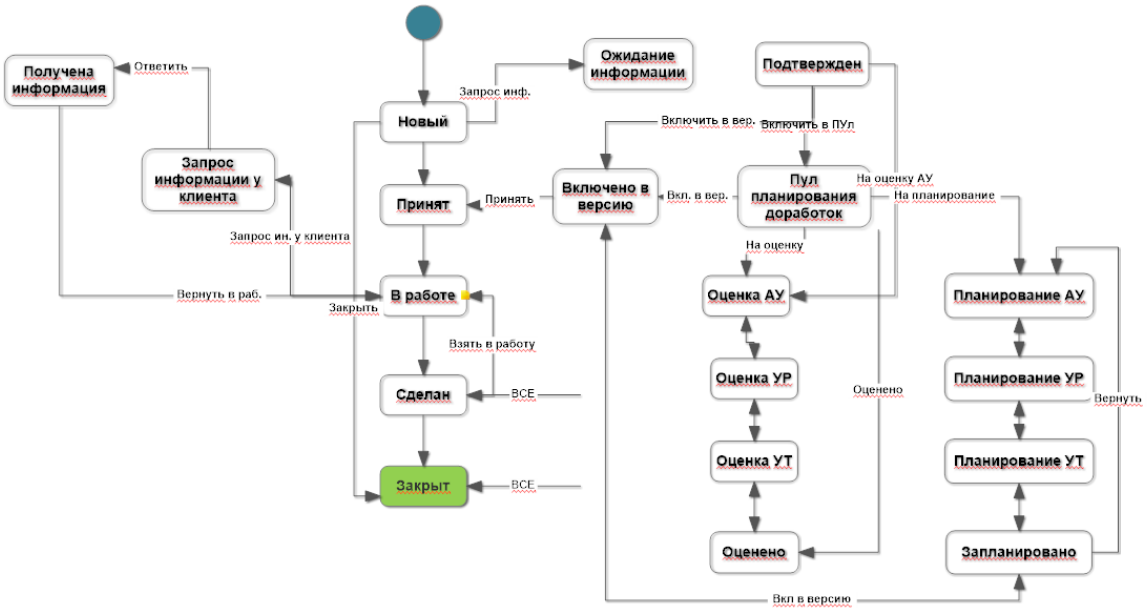

Рисунок 2.6 – Бизнес-процесс «Дефект»

На рисунке 2.7. представлен бизнес-процесс решения задачи под названием «Поручение».

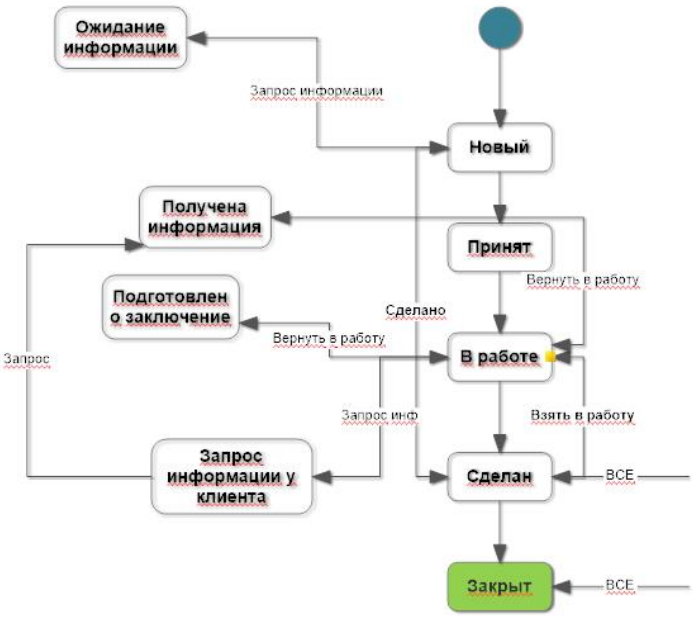

Рисунок 2.7 – БП «Поручение»

Проанализировав деятельность, организационную структуру и основные бизнес-процессы предприятия можно увидеть каким образом взаимодействуют сотрудники компании для достижения общих целей в ходе разработки программного обеспечения.

2.2 Целесообразность автоматизации тестирования программного обеспечения на проекте «САУМИ»

Целью компании ООО «Бюджетные и финансовые технологии» является предоставление качественного программного продукта. Для достижения высокого качества необходимо проводить широкомасштабное тестирование.

Поэтому перед департаментом обеспечения качества стоит задача в нахождении и документировании некорректного поведения программы. С развитием тестирования на помощь тестировщикам приходят инструменты автоматизации.

На данный момент существует огромное множество средств автоматизации тестирования, но прежде, чем автоматизировать процессы тестирования для начала необходимо определить целесообразность внедрения такого процесса [9].

Основной задачей внедрения автоматизации в процессы тестирования является повышение качества тестирования путем снижения трудозатрат на ручное тестирование и увеличение объема проверок с помощью автоматизированного. Под повышением качества подразумевается, что с помощью автоматизации можно охватить больший объем тестов либо выполнение таких тестов, которые невозможно или нецелесообразно проводить вручную. Под экономией времени понимается то, что определенные автоматизированные тесты будут выполняться гораздо быстрее, чем ручным способом.

В компании ООО «БФТ» на проекте «САУМИ» самые высокие трудозатраты на проведение дымового тестирования. Дымовое тестирование подразумевает короткий цикл тестов, выполняемый для подтверждения того, что после новой сборки кода, приложение стартует и выполняет свои основные функции. Для проверки того, что программа стартует и выполняет свои основные функции, создаются тест кейсы, по которым обычно вручную проводится тестирование. Данный функционал является первостепенным, при нахождении ошибки в нем не имеет смысла проводить дальнейшее тестирование. Данные ошибки относятся к типу «Блокирующий» или «Критический».

Дымовое тестирование необходимо проводить как можно чаще, а также необходимо выделять на него много времени. На проекте сейчас есть четыре тестировщика, в среднем на выполнение дымового тестирования каждый

тестировщик тратит до 12 часов, трудозатраты по бюджету на проведение одного такого тестирования четырьмя тестировщиками составляют 64 часа. Поэтому целесообразно автоматизировать данный вид тестирования, что позволит быстрее обнаруживать блокирующие и критические дефекты, и соответственно быстрее их исправлять.

С выполнением дымового тестирования готовые авто-тесты справятся за 4 часов, и те же четыре тестировщика могут заниматься другими приоритетными задачами либо параллельно тестировать функционал системы на более глубоком уровне.

Написание авто-тестов занимает много времени, но это экономия трудозатрат в перспективе на будущее. Поскольку данные тесты легко поддерживать из-за неизменности основного функционала на проекте, и их можно будет выполнять бесконечное количество раз. Помимо избавления тестировщика от рутинной повторяющейся работы и экономии за счет этого времени, авто-тесты можно запускать внерабочее время. В результате получается, что трудозатраты на данный вид тестирования уменьшились, а объем проверок увеличился.

3 Разработка автоматизированных тестов программного обеспечения и расчет экономической целесообразности

3.1 Проектирование информационной модели предметной области

Перед тем, как приступить к разработке есть необходимость в проектировании информационной модели предметной области. Для проектирования необходим инструмент, который позволит подробно отобразить функциональную структуру и взаимодействие основных процессов поддерживающий нотации IDEF0. Один из таких инструментов – Microsoft Visio.

Методология функционального моделирования и графическая нотация IDEF0 (Integration Definition for Function Modeling), используется для описания и формализации бизнес-процессов.

На рисунке 3.1 представлена контекстная диаграмма модели информационной системы контроля качества выпускаемого программного продукта «САУМИ», выполненная в нотации IDEF0.

Рисунок 3.1 – Информационная система контроля качества выпускаемого программного продукта «САУМИ»

Информационная система контроля качества выпускаемого программного продукта «САУМИ» имеет на входе следующие потоки данных:

– писание ПО;
– требования заказчика;
– сроки;
– заявка на работы.

На выходе система формирует следующие потоки данных:

– отчет по дефектам;
– отчет по качеству;
– отчет руководству.

Управляющие потоки:

– регламент;
– ребования к качеству.

Механизмами информационной системы контроля качества выпускаемого программного продукта АЦК-Капитальный ремонт выступают:

– тестировщики;
– ПО;
– технические средства;
– специальные средства тестирования.

После представления информационной системы в общем виде и её взаимосвязь с окружающим миром, то есть после построения контекстной диаграммы, производится разбитие её на крупные фрагменты. Этот процесс называется функциональной декомпозицией. Далее крупные фрагменты разбиваются на более мелкие и до тех пор, пока не будет получена требуемая степень подробности.

На рисунке 3.2 представлена декомпозиция информационной системы контроля качества выпускаемого программного продукта «САУМИ», выполненная в IDEF0.

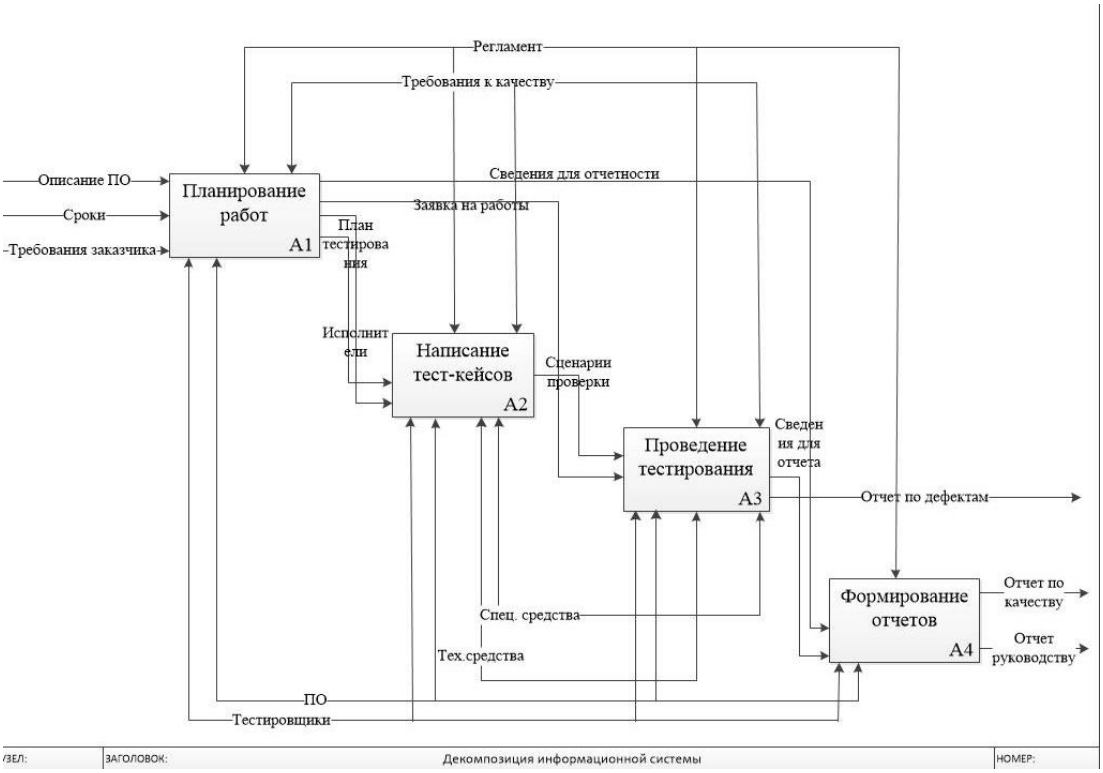

Рисунок 3.2 – Декомпозиция информационной системы

В результате процедуры декомпозиции определены четыре основных процесса:

– планирование работ;
– написание тест-кейсов;
– проведение тестирования;
– формирование отчетов.

В процесс «Планирование работ» на входе поступают потоки данных «Описание ПО», «Сроки» и «Требования заказчика». Управляется этот процесс «Регламент», «Требования к качеству», а механизмами выступают потоки данных «Тестировщики» и «ПО». На выходе процесса «Планирование работ» формируются потоки данных «Сведения для отчета», «Заявка на работы», «Исполнители» и «План тестирования», которые поступают в процесс

«Написание тест-кейсов», «Проведение тестирования» и «Формирование отчетов».

В процессе «Написание тест-кейсов» на входе поступают потоки данных «План тестирования», «Исполнители» и «Заявка на работы». Управляется этот процесс «Регламент», «Требования к качеству», а механизмами выступают потоки данных «Тестировщики», «ПО», «Технические средства», «Специальные средства тестирования». На выходе процесса «Написание тест-кейсов» формируются потоки данных «Сценарии проверок», который поступает в процесс «Проведение тестирования».

В процесс «Проведение тестирования» на входе поступают потоки данных «Заявка на работы» и «Сценарии проверок». Управляется этот процесс «Регламент», «Требования к качеству», а механизмами выступают потоки данных «Тестировщики», «ПО», «Технические средства», «Специальные средства тестирования». На выходе процесса «Проведение тестирования» формируется поток данных «Сведения для отчета», который поступает в процесс «Формирование отчетов» и выходной поток из системы «Отчет по дефектам».

В процессе «Формирование отчетов» на входе поступают потоки данных «Сведения для отчета» из рассмотренных ранее процессов. Управляет этот процесс «Регламент», а механизмами выступают потоки данных «Тестировщики», «ПО». На выходе процесса формируются потоки данных «Отчет по качеству», «Отчет руководству» и эти потоки являются выходными потоками из системы.

Так же необходимо произвести декомпозицию основных процессов «Написание тест-кейсов» и «Проведение тестирования» информационной системы.

На рисунке 3.3 представлена декомпозиция основного процесса «Написание тест-кейсов» информационной системы контроля качества выпускаемого программного продукта «САУМИ», выполненная в IDEF0.

В результате процедуры декомпозиции определены три основных процесса:

– выявление областей покрытия тестами;
– написание тест-кейсов по требованиям;
– разработка авто-тестов.

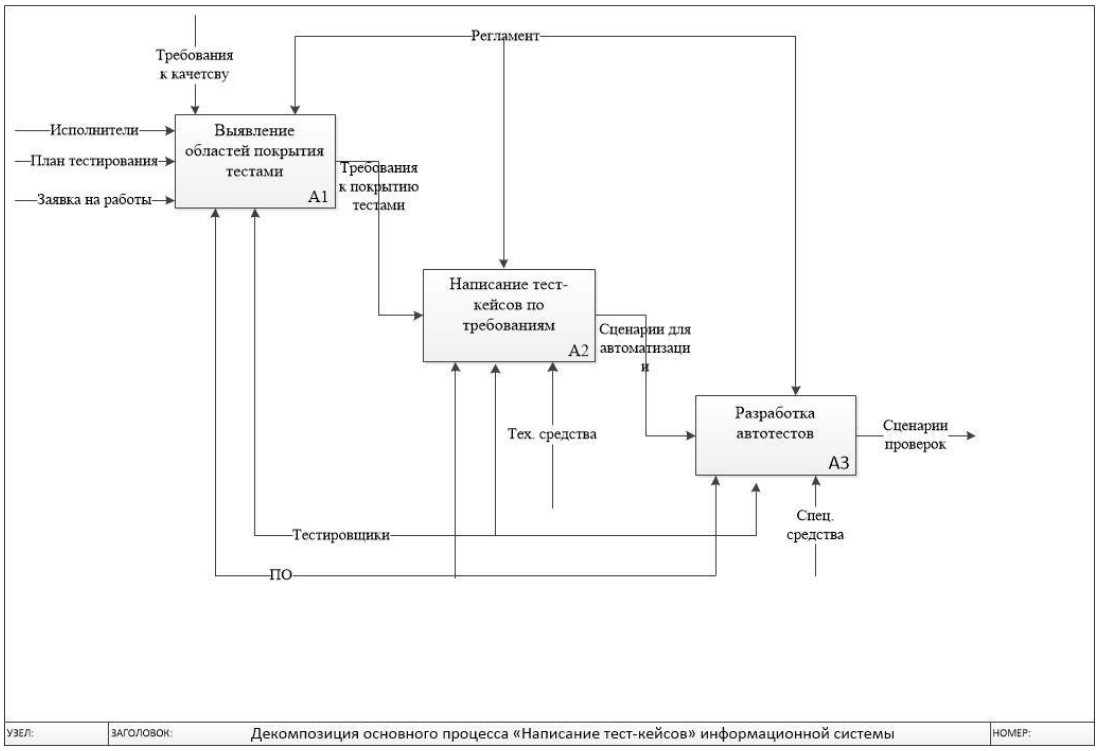

Рисунок 3.3 – Декомпозиция основного процесса «Написание тест-кейсов» информационной системы

«В процесс «Выявление областей покрытия тестами» на входе поступают потоки данных «Исполнители», «План тестирования» и «Заявка на работы». Управляется этот процесс «Регламент», «Требования к качеству», а механизмами выступают потоки данных «Тестировщики» и «ПО». На выходе процесса «Выявление областей покрытия тестами» формируется поток данных «Требования к покрытию тестами», который поступает в процесс «Написание тест-кейсов по требованиям».

В процесс «Написание тест-кейсов по требованиям» на входе поступает поток данных «Требования к покрытию тестами». Управляет этот процесс «Регламент», а механизмами выступают потоки данных «Тестировщики»,

Технические средства» и «ПО». На выходе процесса «Написание тест-кейсов по требованиям» формируется поток данных «Сценарии для автоматизации», который поступает в процесс «Разработка авто-тестов».

В процесс «Разработка авто-тестов» на входе поступает поток данных «Сценарии для автоматизации». Управляет этот процесс «Регламент», а механизмами выступают потоки данных «Тестировщики», «Специальные средства тестирования» и «ПО». На выходе процесса «Разработка авто-тестов» формируется поток данных «Сценарии проверок», который поступает в основной процесс «Проведение тестирования».

На рисунке 3.4 представлена декомпозиция основного процесса «Проведение тестирования» информационной системы контроля качества выпускаемого программного продукта «САУМИ», выполненная в IDEF0.

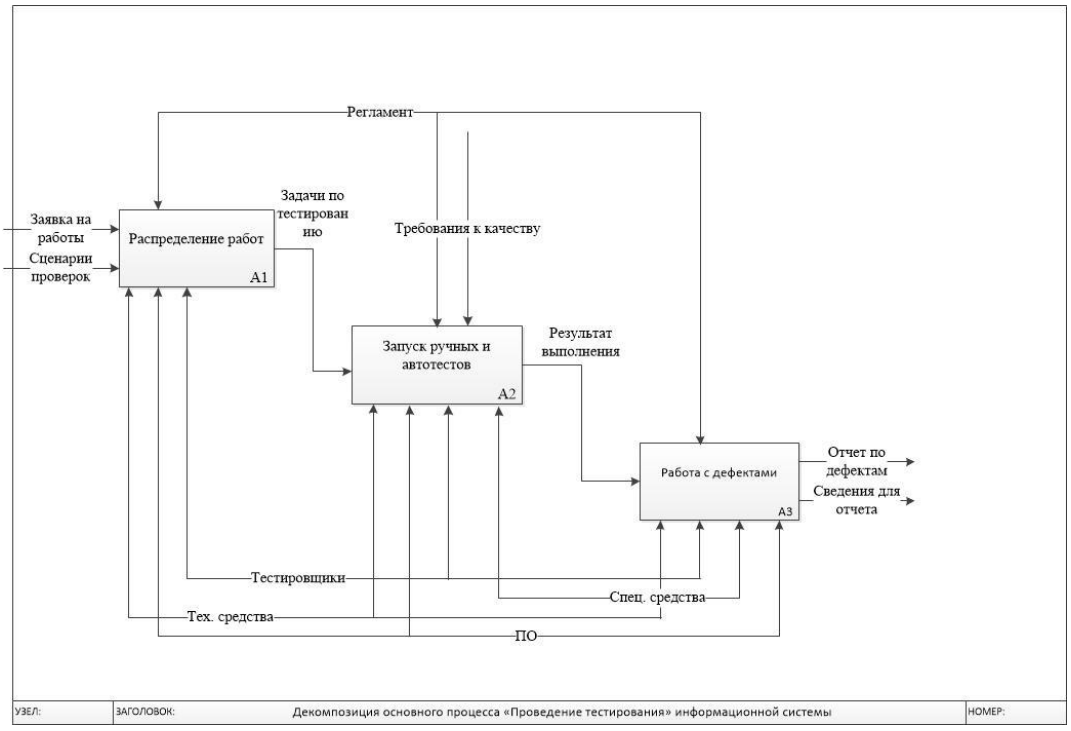

Рисунок 3.4 – Декомпозиция основного процесса «Проведение тестирования» информационной системы

В результате процедуры декомпозиции определены три основных процесса:

– распределение работ;
– запуск ручных и авто-тестов;

– работа с дефектами.

В процесс «Распределение работ» на входе поступают потоки данных «Сценарии проверок» и «Заявка на работы». Управляется этот процесс «Регламент», «Требования к качеству», а механизмами выступают потоки данных «Тестировщики», «Технические средства» и «ПО». На выходе процесса «Распределение работ» формируется поток данных «Задачи по тестированию», который поступает в процесс «Запуск ручных и авто-тестов».

В процесс «Запуск ручных и авто-тестов» на входе поступает поток данных «Задачи на тестирование». Управляет этот процесс «Регламент» и «Требования к качеству», а механизмами выступают потоки данных «Тестировщики», «Технические средства», «Специальные средства тестирования» и «ПО». На выходе процесса «Запуск ручных и авто-тестов» формируется поток данных «Результат выполнения», который поступает в процесс «Работа с дефектами».

В процесс «Работа с дефектами» на входе поступает поток данных «Результат выполнения». Управляет этот процесс «Регламент» и «Требования к качеству», а механизмами выступают потоки данных «Тестировщики», «Технические средства», «Специальные средства тестирования» и «ПО». На выходе процесса «Работа с дефектами» формируются потоки данных «Сведения для отчета, который поступает в основной процесс «Формирование отчетов» и выходной поток из системы «Отчет по дефектам».

В результате проектирования информационной системы предметной области были определены основные процессы контроля качества выпускаемого продукта – планирование работ, написание тест-кейсов, проведение тестирования, формирование отчетов. Более подробно описаны процессы написания тест кейсов и проведения тестирования.

На этапе написания тест кейсов были выявлены следующие процессы:

– выявление областей покрытия тестами;
– написание тест-кейсов по требованиям;
– разработка авто-тестов.

На этапе проведения тестирования были выявлены следующие процессы:

– распределение работ;
– запуск ручных и авто-тестов;
– работа с дефектами.

Таким образом, с помощью средств моделирования бизнес-процессов, можно произвести разбиение модели информационной системы на фрагменты. Определение процессов на этапах написания тест-кейсов и проведения тестирования было необходимо для того, чтобы определить работы да данных этапах, после чего можно перейти к разработке проекта автоматизированных тестов.

3.2 Разработка автоматизированных тестов программного обеспечения

Для того чтобы автоматизировать процесс тестирования необходимо решить следующие задачи:

– изучить приложение;
– определить область эффективной автоматизации;
– выбрать программные средства для разработки тестовых скриптов, выполнения тестов и отчетности;
– разработать тестовые скрипты;
– обеспечить хранение тестовых скриптов;
– провести анализ экономической эффективности автоматизации.

Для достижения необходимого результата нужно учесть несколько факторов:

– участие тестера в работе скрипта должно быть минимальным;
– следует выбирать сценарии, которые легко автоматизировать;

– скрипты должны подходить для дымового тестирования;

– скрипт должен быть легко поддерживаем и расширяем;

– должна быть возможность запускать скрипты на разных удаленных машинах и в любое время суток;

– инструмент автоматизации тестирования должен обладать графическим интерфейсом;

– инструмент автоматизации должен быть бесплатным;

– результаты выполнения должны представляться в виде отчетов.

Процесс разработки автоматизированного тестирования должен начинаться с подробного изучения тестируемого приложения и соответствующей документации. Для разработки решений по автоматизации необходимо иметь полное представление о функциональных возможностях системы и узких местах. Для этого необходимо провести анализ требований, предъявляемых к работе программного продукта и определить функциональность, которая необходима заказчику.

После подробного изучения функциональных возможностей программного продукта, необходимо выбрать область эффективной автоматизации. Для проекта «САУМИ» было решено, что целесообразно автоматизировать процесс дымового тестирования. Для разработки тестовых скриптов будут использоваться написанные заранее тест-кейсы, покрывающие основной функционал системы.

Тогда как область автоматизации определена, необходимо определиться с выбором инструментария для автоматизации тестирования. Выбор программных средств для автоматизации чаще всего зависит от требований к тестовым сценариям и объектов тестирования. Так как инструменты тестирования могут содержать определенный набор технологий, используемых при разработке программного продукта. Часто принятие решение о выборе инструмента сводится к простому способу проб и ошибок. И можно встретить случаи использования нескольких инструментов для автоматизации различных функций приложения.

В конкретном случае для автоматизации тестирования на проекте «САУМИ» были выбраны следующие программные средства:

– IntelliJ IDEA;
– Maven;
– GitHub;
– Jenkins;
– Allure Framework;
– TestNG Framework;
– Katalon Recorder (Selenium IDE for Chrome);
– Monte Screen Recoder.

Для создания проекта тестовых скриптов используется IntelliJ IDEA. Данный инструмент является бесплатным, имеет удобный пользовательский интерфейс для сбора, написания и редактирования тестовых сценариев. В IntelliJ IDEA можно сразу открывать мавен-проект, что довольно удобно, поскольку нет необходимости переключаться между рабочими пространствами. Для сборки проекта используется Maven [11]. Для автоматической сборки небольшого проекта есть возможность сборки с помощью командной строки. Помимо удобства и быстроты сборки, можно выделить такие преимущества как осуществление сборки в любой ОС, хорошая интеграция со средами разработки, в том числе IntelliJ IDEA. Не нужно ничего настраивать, проект всегда готов к разработке.

В качестве хранилища копий файлов проекта, используется крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки – GitHub.

После того как сделали сборку проекта, для ее запуска авто-тестов на стенде используется Jenkins – это проект для непрерывной интеграции с открытым исходным кодом, написанный на Java.

Результаты прохождения тестирования попадают в Allure – фреймворк, который служит для получения отчетов о прохождении авто-тестов.

Отчеты отображают информацию в понятном виде и для тестировщиков, и для менеджеров проекта. К ним можно прикладывать

скриншоты, логи и любые другие файлы. Allure позволяет разбивать сложные тесты на шаги, включать информацию о параметрах тестов и тестового окружения.

Если необходима видеозапись выполнения автотестов, можно воспользоваться библиотекой Monte Screen Recoder, установив ее в локальный репозиторий.

Все авто-тесты должны быть написаны на Java с использованием тестового фрейворка TestNG и в качестве формирования сценариев можно использовать инструмент Selenium [13].

Selenium IDE используется для облегчения работы по разработке авто-тестов [12]. Он позволяет сформировать заготовки авто-тестов, а в некоторых случаях готовый к выполнению. Он представляет собой простое в использовании дополнение к браузеру Chrome и путем имитации действий пользователя происходит составление сценариев для тестов [9]. Для полноценной разработки автоматизированных сценариев необходимо созданные заготовки доработать в среде разработки IntelliJ IDEA, тем самым создать проект. Под доработкой можно понимать их модернизацию, добавление проверок, реализацию для нескольких браузеров.

Процесс запуска выполнения автоматизированного тестирования должен включать следующие виды работ:

– назначается ответственный сотрудник за выполнение авто-тестов;

– проводится пошаговая подготовка для запуска авто-тестов для восстановления бэкапа базы данных;

– в Jenkins производится запуск всего проекта авто-тестов;

– анализ результатов выполнения авто-тестов и занесение найденных дефектов в Jira;

Процесс всегда должен начинаться с составления плана тестирования (Тест-плана) на основании требований, предъявляемых к программному продукту, сроков и иных работ, которые необходимо реализовать. Данные документ должен составлять руководитель направления тестирования, либо

самый ответственный тестировщик. Должны быть определены стратегии и описаны риски. В случае внедрения автоматизации, должны быть продуманы затраты на разработку и обработку тестовых скриптов.

Тест-план отражает все основные работы, описывает то, каким образом оно будет проводиться, в какой последовательности и с какими сроками. Также определяются работы каждого сотрудника.

В процессе работы тест-план может корректироваться, так как всегда есть возможность появления незапланированных работ, как со стороны заказчика, так и со стороны руководства. Но документ всегда должен быть актуальным, чтобы все участники процесса понимали объект работ.

План тестирования согласуется и утверждается с рабочей группой, которая состоит из руководителя тестирования, руководителя разработки и руководителя аналитики.

После составления тест-плана можно перейти к выявлению областей покрытия тестами.

На проекте «САУМИ» уже разработаны тест-кейсы, которые покрывают весь основной функционал системы.

Руководитель направления выделяет сценарии, подходящие для автоматизации дымового тестирования.

Под автоматизацию попадают сценарии на основной функционал программы, который остается неизменным. И особенно те сценарии, которые относятся к критичным или блокирующим компонентам системы.

Так как автоматизация должна быть выгодна компании, отдавать под автоматизацию сложные тест-кейсы не имеет смысла, так как их разработка будет сложной, сопровождение скриптов будет затратным.

После того как были выбраны тест-кейсы для автоматизации можно перейти к разработке.

На рисунке 3.5 продемонстрирован тест-кейс, направленный для разработки скрипта автоматизации. Данный тест-кейс описывает работу авторизации пользователя в системе.

Описание

Запуск браузера, ввод логина/пароля, вход в систему с использованием функции <Запонить меня>

Сведения о тесте

	Шаг теста	Данные теста	Ожидаемый результат
1	1. Запустить один из следующих web-браузеров: Google Chrome 26+, Mozilla Firefox 19+,Microsoft Internet Explorer 9+. 2. В окне браузера в адресной строке задать адрес приложения вида «http://хост:порт/saumi», где часть "хост:порт" задается администратором системы.		Web-браузер запущен и работает корректно. Открыта главная страница приложения с полями [Пользователь:], [Пароль:] и кнопкой "Войти"
2	Заполнить обязательные поля	[Пользователь]*= {"логин"} [Пароль]*= {"пароль"} [Запомнить меня]= {активирован}	В поле [Пользователь]* отобразилось значение {"логин"} В поле [Пароль]* отобразилось количество точек равное количеству знаков в {"пароль"} Напротив поля [Запомнить меня] появилась галочка
3	Нажать кнопку "Войти"		Произошла авторизация пользователя в системе, открыт <Рубрикатор>

Рисунок 3.5 – Пример тест-кейса направленного на разработку скрипта

К моменту передачи тест-кейсов на разработку автоматизированных скриптов, должна быть подготовлена рабочая машина тестировщика для разработки авто-тестов. Для таких объемов автоматизации будет достаточно рабочего компьютера тестировщика.

В качестве основного веб-браузера будет использоваться Chrome версии 66.0.3359.181.

Затем необходимо установить инструмент для разработки тестовых сценариев Katalon Recorder (Selenium IDE for Chrome). Он представляет собой простое дополнение к браузеру Chrome. Его можно бесплатно загрузить с официального сайта.

После установки потребуется перезапустить браузер. Selenium IDE будет доступен в меню «Инструменты».

Selenium IDE используется в качестве первого этапа написания автоматизированных сценариев. В этом инструменте будут создаваться простые автоматизированные тесты, а также шаблоны к более сложным.

Selenium IDE представляет собой рекордер для записи шагов прохождения сценариев. Для старта необходимо запустить плагин и начать

запись действий. Теперь каждое действие, производимое в браузере, будет записываться. Создание таких авто-тестов с записью простых линейных последовательностей с жестко определенными данными, несложными проверками, простой структурой, так же может считаться автоматизацией не требующий знаний программирования. Но ни о каких циклах не может быть и речи.

На рисунке 3.6 представлен пример тестового сценария «Проверка успешной авторизации пользователя».

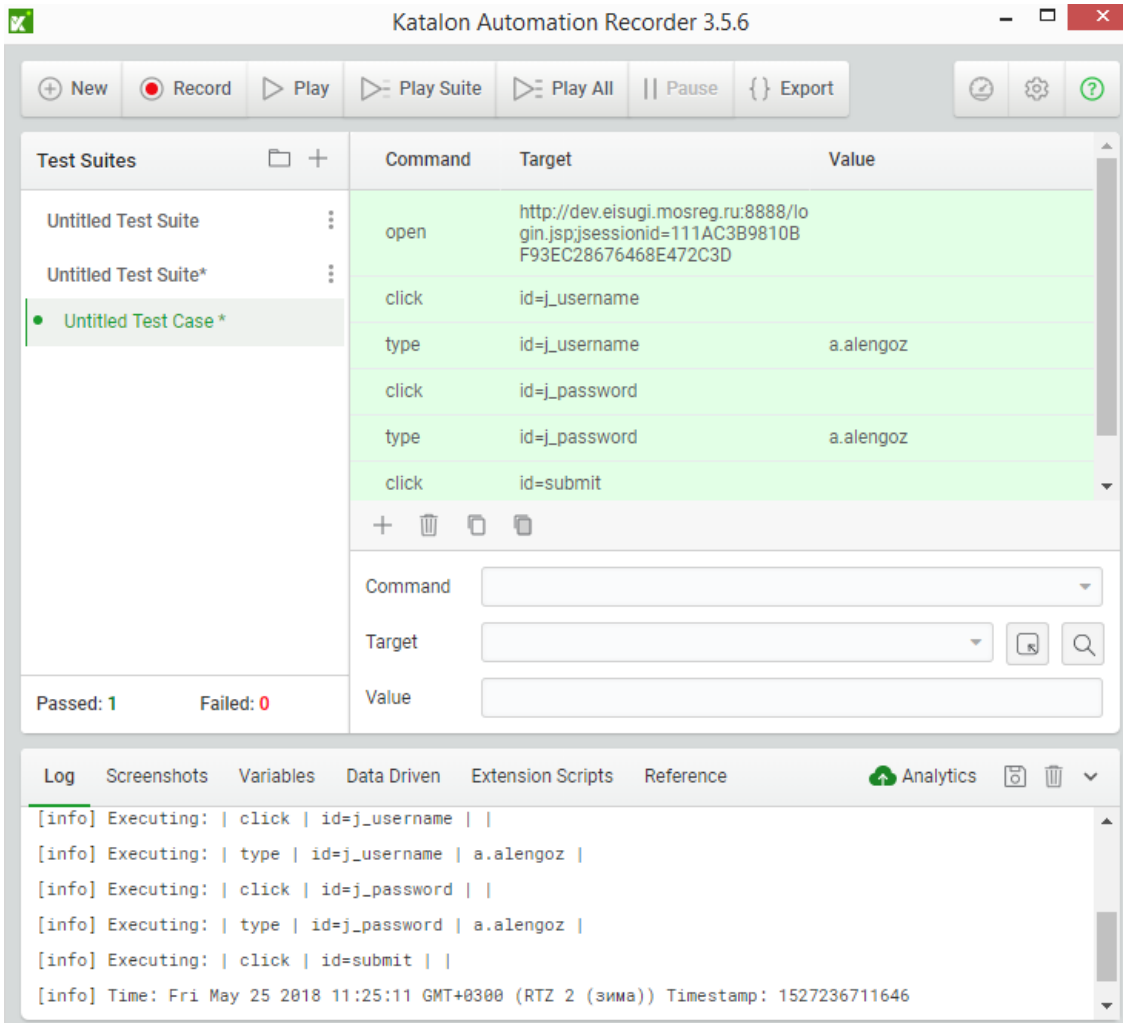

Рисунок 3.6 – Проверка успешной авторизации пользователя

После того как тест был записан его действия можно воспроизвести, нажав кнопку «Выполнить». И все записанные действия воспроизведутся автоматически, по прохождению результат можно посмотреть внизу окна на вкладке «Log». На рисунке 3.6 видно, что все действия выполнились успешно.

Для полноценной разработки сценариев автоматизированного тестирования необходимо будет продолжить работу в среде разработки – IntelliJ IDEA и перейти на полноценный язык программирования. В Selenium IDE происходит по большей части заготовка, которую необходимо дорабатывать.

Дорабатывать заготовку потом гораздо проще, чем создавать все с нуля. К примеру, если есть достаточно большая форма в тестируемом приложении, которую необходимо заполнить, необходимо будет узнавать имена, идентификаторы полей. Для этого необходимо ходить по исходному коду странички, искать нужные теги, смотреть какие атрибуты, что увеличивает время разработки авто-теста. Вот тут Selenium IDE, оказывает большую помощь, сразу находя и записывая необходимую информацию. Так же, можно создавать заготовки, какой-либо части сценария, например, при изменении части функционала приложения.

После полноценной подготовки заготовки авто-теста, необходимо перенести его в среду разработки IntelliJ IDEA. Это осуществляется с помощью кнопки «Export», выбираем формат Java (WebDriver + TestNG) далее можно просто скопировать код и перенести вручную.

Для удобства работы и сопровождения авто-тестов в будущем, необходимо сформировать в созданном проекте дерево папок, которые должны повторять модули/компоненты навигации «САУМИ». Все основные файлы программы должны находится в папках: resources, appmanager, ObjetsData, TestBases, tests.

Перед тем как перенести java код в IntelliJ IDEA, необходимо скачать и распаковать Maven в корень диска C://. Добавить переменные среды в дополнительных параметрах системы, где имя переменной = «M2_HOME», а значение переменной = «C:\apache-maven-3.2.2». В среде разработке создать проект мавен и добавить зависимости.

Создать в папке tests Java класс с наименованием logIn. Перенести в него код. Далее необходимо подтянуть зависимости мавен. После того, как

зависимости были подтянуты необходимо, внести правки. Места, где имеются ошибки, будут выделены красным. После всех правок, можно проверить на корректность авто-тест запустив его в среде разработки. В результате внизу окна будут описаны ошибки, если они есть, и предупреждения. Ошибки имеют красный значок с восклицательным знаком, а предупреждения серый. На рисунке 3.7 уже выполнялся тест, и можно увидеть, что он выполнился без ошибок.

На рисунке 3.7 показан пример перенесенного и доработанного java-кода из Selenium. Полный текст готового кода в приложении Б.

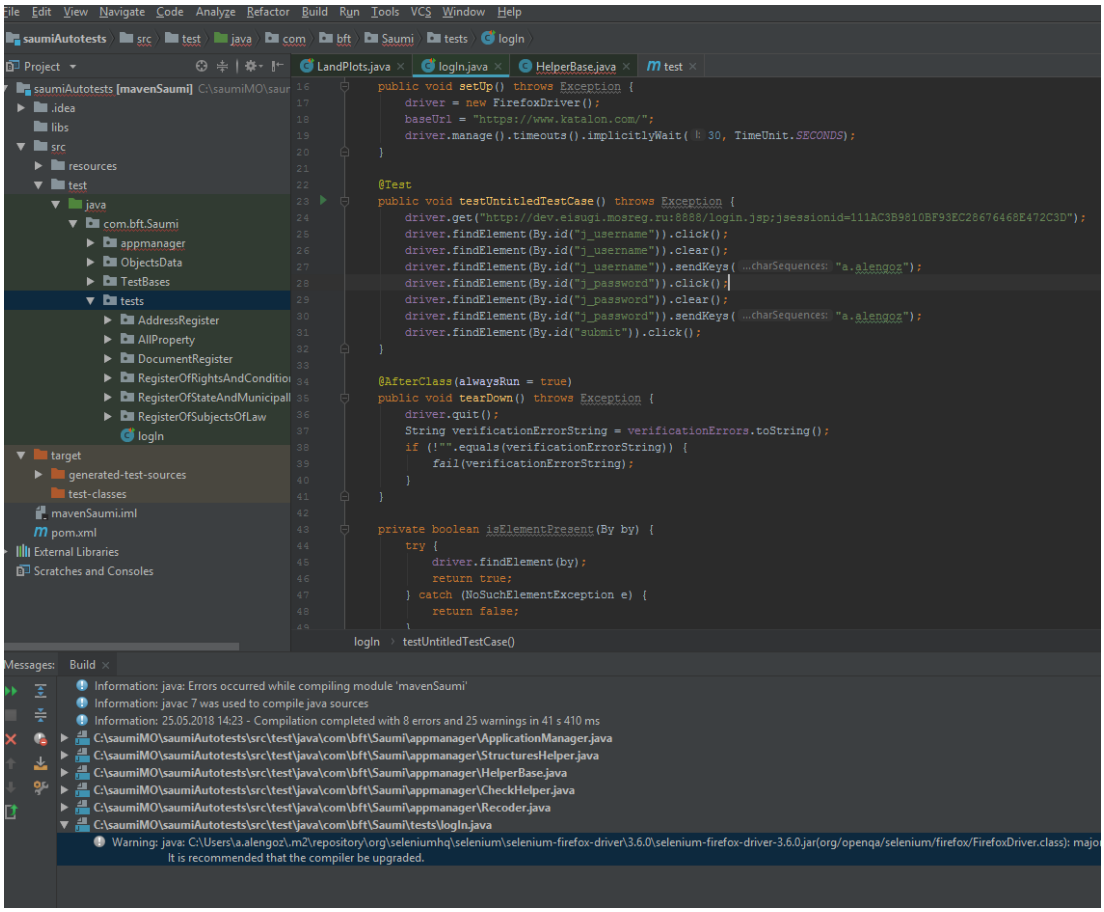

Рисунок 3.7 – Пример разработанного авто-теста в IntelliJ IDEA

На все сценарии, выявленные для автоматизации, должны быть разработаны авто-тесты. При невозможности либо высокой трудоемкости в реализации авто-теста, необходимо сообщить руководителю направления тестирования, для его переоценки.

Все разработанные сценарии в виде java файлов, должны отправляться в общее хранилище, для целостности и сохранности. В компании «Бюджетные и финансовые технологии» принято использовать ftp либо svn.

По завершению разработки автоматизированных сценариев, получаем возможность оперативно и без огромных затрат проводить дымовое тестирование программного продукта «САУМИ».

Перед тем как запустить тестирование, необходимо сделать бэкап базы данных. Делается он для того, что бы на стенде были необходимые тестовые данные, и их не пришлось создавать вручную для каждого теста. Для этого необходимо выполнить следующие шаги:

– остановить службу saumiMO;
– удалить базу saumiBase в pgAdmin 4;
– создать новую пустую базу с тем же именем;
– востановить ее, используя бекап из папки C:\Saumi_Tomcat\backup\;
– Открыть консоль от имени администратора и ввести команды:
– chcp 1251(не обязательно. отображает русские буквы)
– cd C:\Saumi_Tomcat\updater-1.10.2.12\bin
– updater update --config upd.properties --auto --force_version;
– дождаться, когда все успешно обновится и остановить службу saumiMO;
– перейти в папку C:\Saumi_Tomcat\tomcat\webapps\saumimo\META-INF и заменить файл context.xml на файл в этой папке;
– Запустить службу saumiMO.

Для запуска тестирования необходимо в Jenkins выбрать интересующий проект с набором тестов и поставить на выполнение. При необходимости можно выбрать конкретные группы тестов, для получения быстрого результата по части функционала.

В Jenkins есть история сборок, где можно посмотреть информацию о прохождении предыдущих авто-тестов, как представлено на рисунке 3.8.

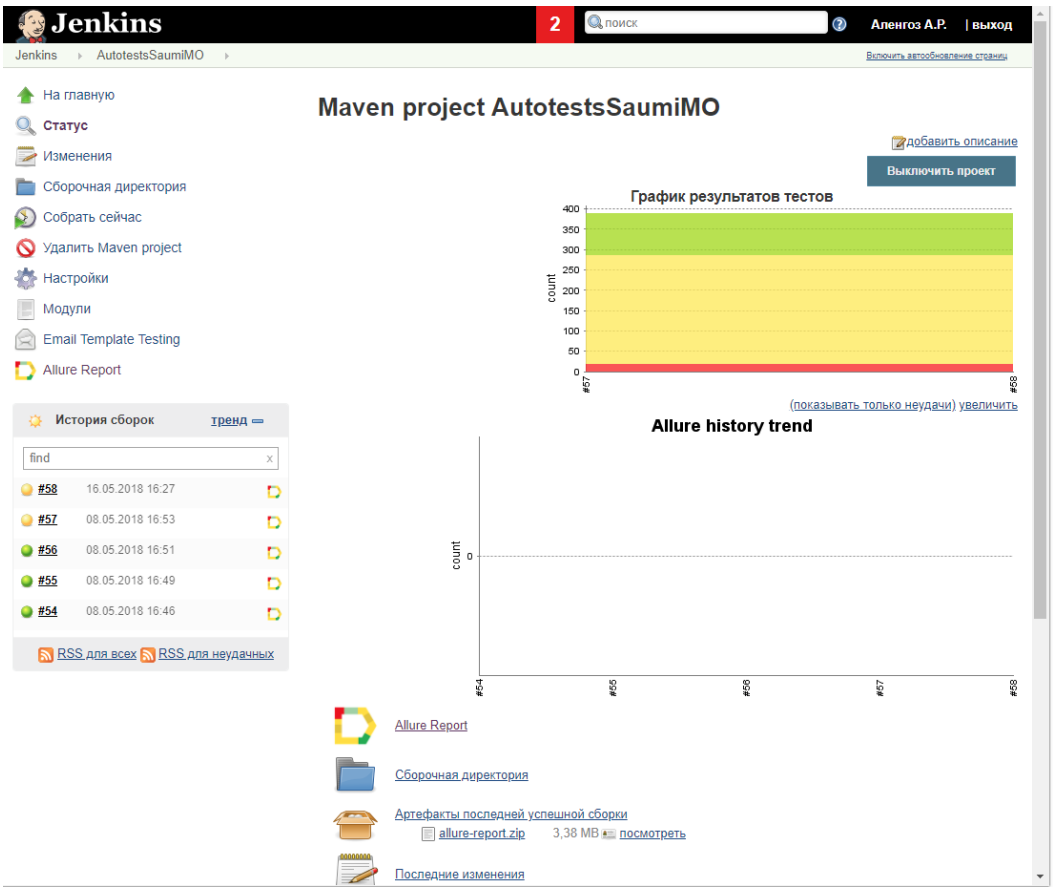

Рисунок 3.8 – История пройденных ранее авто-тестов в Jenkins

Итоги выполнения сценариев можно просмотреть с помощью Allure Framework из Jenkins, там будет информация о количестве запущенных сценариев и количестве сценариев, с какими-либо ошибками, а также времени прохождения. На провалившихся тестах, формируется лог и указывает на какой строчке кода (каком шаге) появилась ошибка.

На рисунке 3.9 представлен общий отчет о прохождении последней группы авто-тестов.

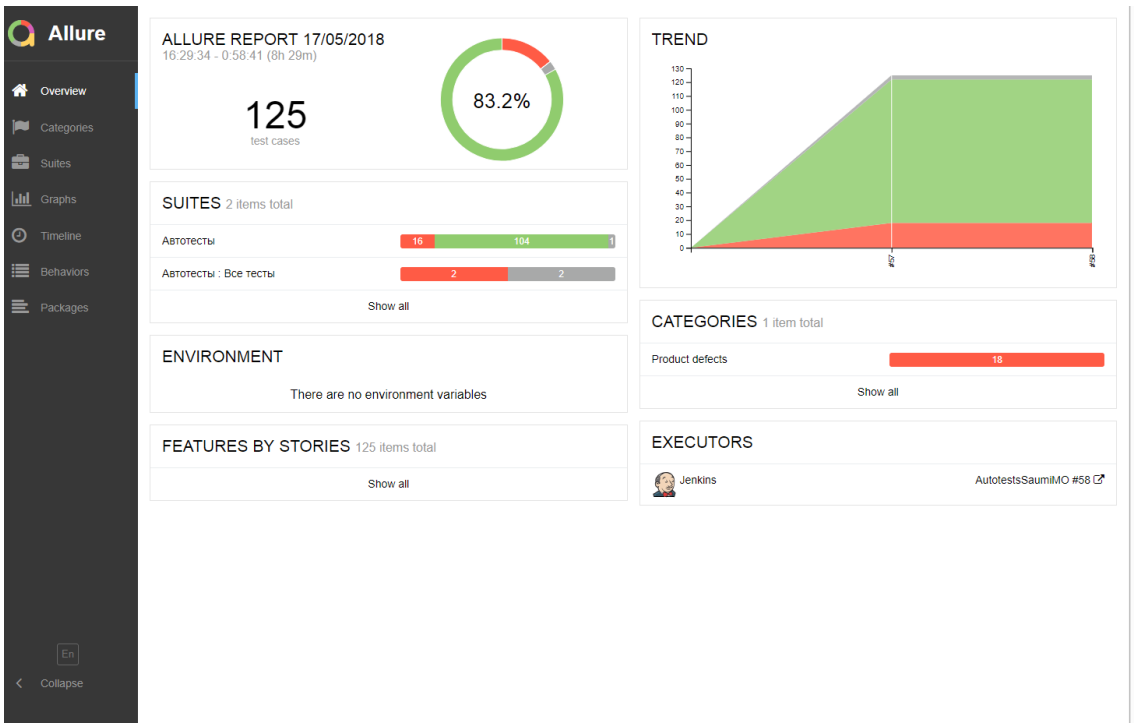

Рисунок 3.9 – Общий отчет о прохождении группы авто-тестов в Allure

На рисунке 3.10 представлена информация по упавшим тестам.

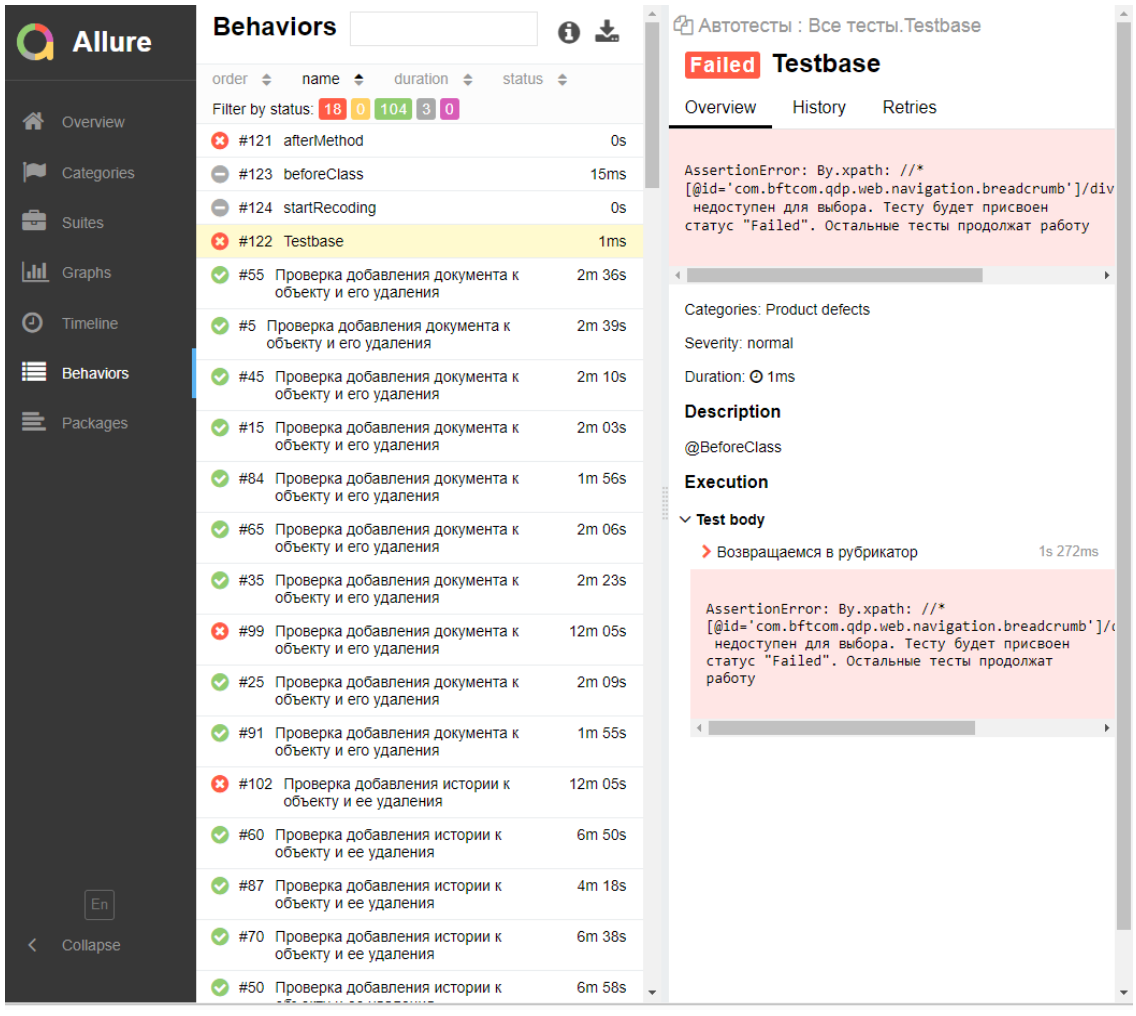

Рисунок 3.10 – Отчет по упавшим авто-тестам в Allure

По всем провалившимся авто-тестам, тестировщик должен провести работы, понять причину возникновения и зарегистрировать ошибку в системе учета дефектов JIRA, приложив всю необходимую информацию (скриншоты, логи).

Руководитель направления тестирования в любой момент может узнать состояние выполняемых работ. Благодаря отчетам из Allure, виден основной объем выполненных/невыполненных работ по видам тестирования.

В случае если найденные ошибки программного продукта оперативного исправляются группой разработчиков, то ранее провальные тест-кейсы перезапускаются.

В итоге разработанная автоматизация дымовых тестов предоставляет возможность получения отчета по тестированию в довольно короткие сроки, не переключая сотрудников с одних работ на другие работы. А обработка результата выполнения авто-тестов не займет много времени.

3.3 Расчет экономической целесообразности автоматизации тестирования

Чтобы определить экономическую эффективность, необходимо выделить, какие показатели будут оценивать внедрение автоматизированной системы. В основном это стоимость человеко-часов работы тестировщика, стоимость разработки авто-тестов в человеко-часах, данные о частоте проведения тестирования.

Для подсчета эффективности внедрения автоматизации тестирования рассмотрим несколько сторон:

– прямой расчет затрат на проведение тестирования;

– временные затраты на тестирование;

– минимизация рисков.

Способ прямого расчета позволяет сравнить стоимость тестирования до и после внедрения автоматизации.

Следующая формула может быть использована для расчета прямых затрат ручного прогона дымового тестирования:

$$Z_r = I \times S_r \times T_r \times P \times K \ , \quad (3.1)$$

где Z_r – затраты на проведение ручного тестирования, руб.;

I – количество спринтов тестирования, ед.;

S_r – стоимость работы ручного тестировщика в час, руб.;

T_r – время на ручное прохождение тестов, час;

P – количество запусков тестирования в спринте, ед.;

K – количество конфигураций в спринте, ед.

Для расчета затрат на автоматизацию необходимо учитывать некоторые факторы:

– стоимость программного обеспечения для внедрения автоматизации;

– стоимость разработки первоначального набора автоматизированных тестов;

– стоимость прогона автоматизированных тестов;

– затраты на сопровождение автоматизации.

Следующая формула может быть использована для расчета затрат прогона дымового тестирования, при внедрении автоматизации:

$$Z_a = R + T_a \times S_a + (P \times T_e + T_s) \times I \times S_a \times K \ , \quad (3.2)$$

где Z_a – затраты на проведение автоматизированного тестирования, руб.;

R – разовые затраты на автоматизацию (стоимость или аренда оборудования, стоимость лицензий), руб.;

T_a – общее время на разработку всех автоматизированных сценариев, час.;

S_a – стоимость работы тестировщика-автоматизатора в час, руб.;

T_e – время прогона автоматизированных сценариев, час.;

T_s – время на сопровождение автоматизированных сценариев, час.

Рассчитать ожидаемую выгоду от внедрения автоматизированного тестирования можно с точки зрения временных затрат на приемочное тестирование. Данный способ основан на сравнении временных затрат, требуемых на проведение тестирования и внедрение автоматизации. Не берется в расчет затраты связанные с денежным выражением.

$$L_r = I \quad \times T_r \times P \tag{3.3}$$

$$L_a = I \quad \times T_a + P \quad \times T_e, \tag{3.4}$$

где L_r – временные затраты на ручное тестирование, час.;

L_a – временные затраты на автоматизированное тестирование, час.

Ещё один способ оценки эффективности внедрения автоматизированного тестирования является оценка с точки зрения минимизации рисков. Данный метод предлагает сравнение средств, затраченных на тестирование с убытками, которые могут возникнуть в результате ошибки функционирования готовой системы на этапе эксплуатации. Стоит отметить, что зачастую сложно точно оценить возможные убытки, потому что данный метод подразумевает точечный анализ возможных рисков на проекте [17].

По выбранной методике расчета показателей эффективности рассчитаем затраты на проведение дымового тестирования на проекте «САУМИ». Рассмотрим временные затраты на проведение дымового тестирования и рассмотрим минимизацию рисков для проекта в общем.

Проведем расчеты показателей эффективности при базовом варианте тестирования. Выясним, на каком спринте после внедрения автоматизации процесса тестирования стоимость автоматизации будет меньше проводимый работ по тестированию вручную. Поочередно, от одного спринта до необходимого количества будем подставлять значение в выбранные формулы расчета.

Для расчета затрат при ручном тестировании используем формулу (3.1), при автоматизированном – формулу (3.2).

Характеристика затрат на выполнение ручного и автоматизированного тестирования при базовом варианте дана в таблице 3.1.

Таблица 3.1 – Характеристика затрат на выполнение ручного и автоматизированного тестирования при базовом варианте

№ п/п	Расчетные параметры затрат	Обозначение	Единица измерения.	Значение	
				ручное тестирование	автоматизированное тестирование
1	2	3	4	5	6
1.	Количество спринтов тестирования	I	ед.	1	1
2.	Стоимость работы ручного тестировщика в час	S_r	руб.	100	–
3.	Время на ручное прохождение тестов	T_r	час.	12	-
4.	Количество запусков тестирования в спринте	P	ед.	2	2
5.	Количество конфигураций в спринте	K	ед.	1	1
6.	Разовые затраты на автоматизацию	R	руб.	-	0
7.	Общее время на разработку всех автоматизированных сценариев	T_a	час.	-	80
8.	Стоимость работы тестировщика-автоматизатора в час	S_a	руб.	-	100

Продолжение таблицы 3.1

1	2	3	4	5	6
9.	Время прогона автоматизированных сценариев	T_e	час.	-	1
10.	Время на сопровождение автоматизированных сценариев	T_s	час.	-	2

Стоит пояснить, что в разрабатываемом проекте автоматизации отсутствуют затраты, связанные с покупкой серверов, лицензий и т.д. Стоимость работ ручного тестировщика и тестировщика-автоматизатора одинаковы, так как штатных изменений не производилось и использовался штатный ресурс сотрудников отдела.

В таблице 3.2 отражены прямые затраты на ручное и автоматизированное тестирование в разрезе спринтов тестирования.

Таблица 3.2 – Прямые затраты на ручное и автоматизированное тестирование в разрезе спринтов тестирования

№ спринта	Затраты на ручное тестирование (руб.)	Затраты на автоматизированное тестирование (руб.)
1	2	3
1	2400	8400
2	4800	8800
3	7200	9200
4	28800	16000
5	3600	17500
6	43200	19000
7	50400	20500

Посмотрим на графике (рисунок 3.11) как будут идти затраты на ручное и автоматизированное тестирование.

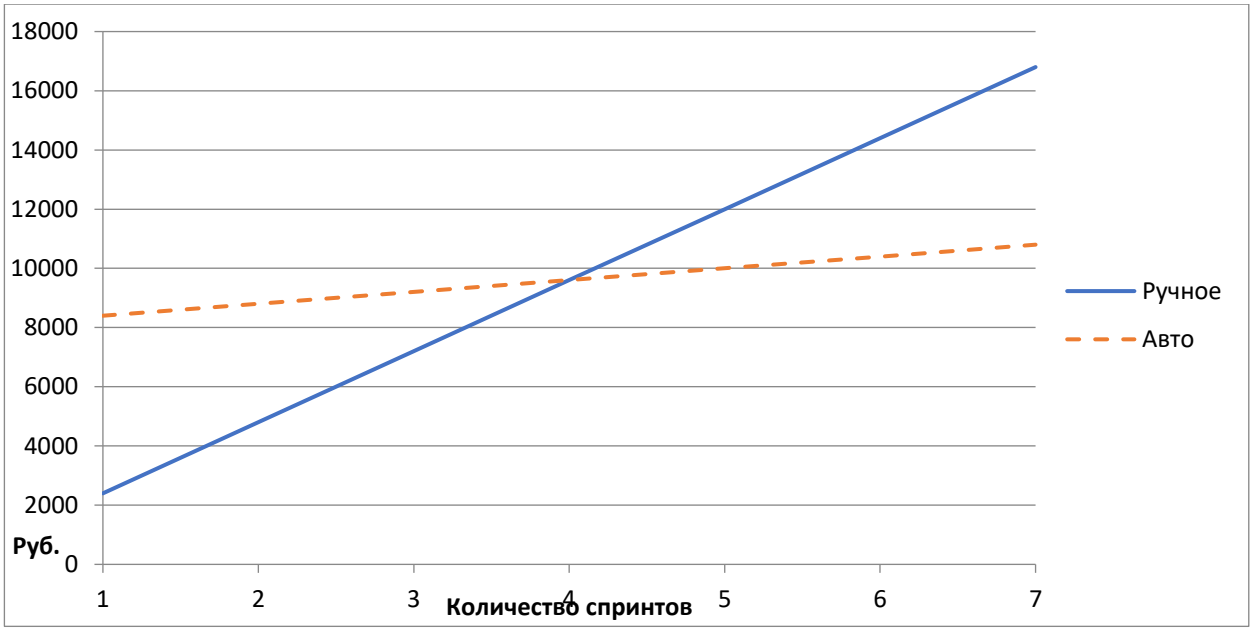

Рисунок 3.11 – График прямых затрат на ручное и автоматизированное тестирование

Рассмотрим расширенный вариант прохождения дымового тестирования, т.е. добавим конфигурации прохождения сценариев. Под конфигурациями будем понимать тестирование в трех веб браузерах, так как проверка работоспособности в различных браузерах так же является важной составляющей в тестировании приложения. Так же увеличатся затраты на формирование авто-тестов. И не стоит забывать про сопровождение разработанных сценариев.

Характеристика затрат на выполнение ручного и автоматизированного тестирования при расширенном варианте дана в таблице 3.3.

Таблица 3.3 – Характеристика затрат на выполнение ручного и автоматизированного тестирования при расширенном варианте

№ п/п	Расчетные параметры затрат	Обозначение	Единица измерения.	Значение	
				ручное тестирование	автоматизированное тестирование
1	2	3	4	5	6
1.	Количество спринтов тестирования	I	ед.	1	1
2.	Стоимость работы ручного тестировщика в час	S_r	руб.	100	–
3.	Время на ручное прохождение тестов	T_r	час.	12	-
4.	Количество запусков тестирования в спринте	P	ед.	2	2
5.	Количество конфигураций в спринте	K	ед.	3	3
6.	Разовые затраты на автоматизацию	R	руб.	-	0
7.	Общее время на разработку всех автоматизированных сценариев	T_a	час.	-	100
8.	Стоимость работы тестировщика-автоматизатора в час	S_a	руб.	-	100
9.	Время прогона автоматизированных сценариев	T_e	час.	-	1
10.	Время на сопровождение автоматизированных сценариев	T_s	час.	-	3

В таблице 3.4 отражены затраты на ручное и автоматизированное тестирование в разрезе спринтов тестирования при расширенном варианте.

Посмотрим на графике (рисунок 3.12) как будут идти затраты на ручное и автоматизированное тестирование, при расширенном варианте тестирования приложения.

Таблица 3.4 – Затраты на ручное и автоматизированное тестирование в разрезе спринтов тестирования при расширенном варианте

№ спринта	Затраты на ручное тестирование (руб.)	Затраты на автоматизированное тестирование (руб.)
1	7200	11500
2	14400	13000
3	21600	14500
4	28800	16000
5	3600	17500
6	43200	19000
7	50400	20500

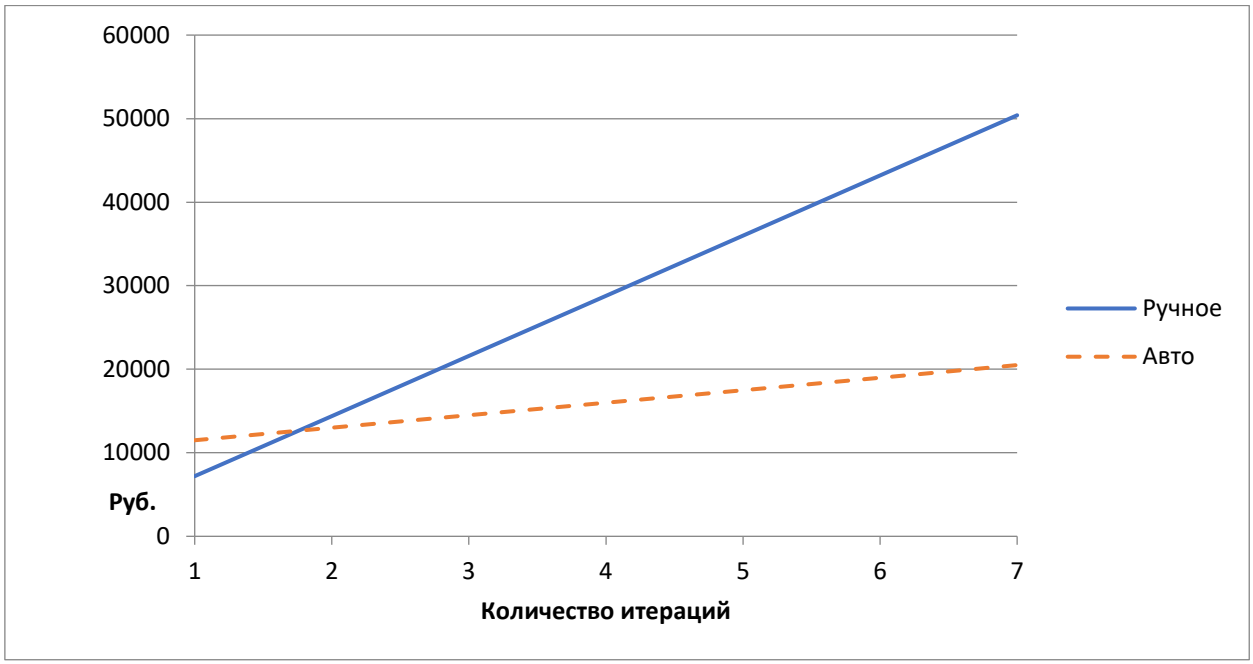

Рисунок 3.12 – График расширенных затрат на ручное и автоматизированное тестирование

Очевидно, что хорошо организованная автоматизация сценариев дымового тестирования для компании несет значительный экономический эффект, который достигается достаточно быстро и со временем только увеличивается.

Рассмотрим ещё один вариант показателя эффективности - анализ временных затрат на тестирование с внедрением автоматизации. На графике

(рисунок 3.13) отображается временные затраты на проведение дымового тестирования с учетом временных затрат на автоматизацию в разрезе спринтов.

Как только автоматизация окупилась, временные затраты на получение результата дымового тестирования значительно меньше. Раньше прогон тестирования занимал 12 человеко-часов, а после внедрения автоматизации занимает 1 час. А учитывая, что запустить прогон автоматизированных тестов можно не в работающее время – затраты становятся максимально минимальные.

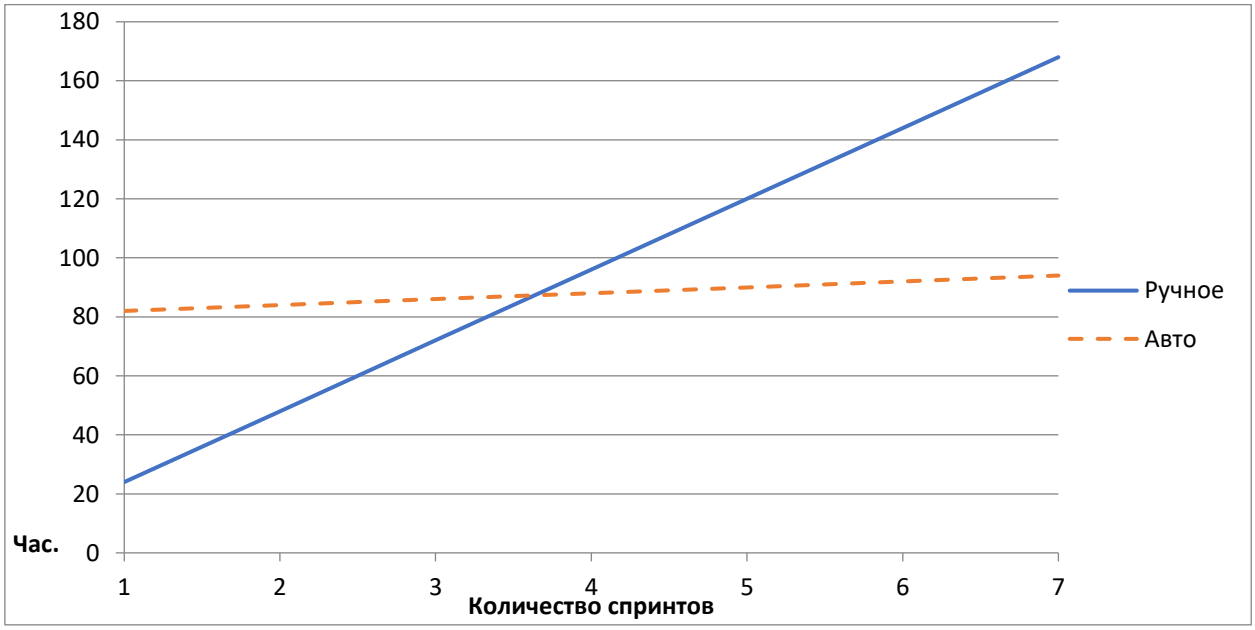

Рисунок 3.13 – График временных затрат на ручное и автоматизированное тестирование

Что касается возможных рисков, так с внедрением автоматизации, появляется возможность запускать прогон автоматизированных тестов каждый вечер. Чем раньше тестирование выявит дефект, тем меньше стоит сам дефект. Например, если разработчик сегодня случайно сломал какой-либо функционал и завтра получил зарегистрированный дефект, ему будет легче вспомнить что и где он правил.

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе была подробно рассмотрена потребность в автоматизации процесса тестирования для компании ООО «Бюджетные и финансовые технологии» на проекте «САУМИ» в части выполнения дымового тестирования.

На этапе определения теоретических аспектов тестирования:

– изучены понятие тестирования, классификация видов тестирования;

– определение места тестирования в жизненном цикле программного обеспечения;

– рассмотрен процесс тестирования.

На этапе проведения анализа деятельности предприятия и основных бизнес-процессов:

– изучена деятельность предприятия ООО «БФТ» и организационная структура;

– описаны основные бизнес-процессы предприятия.

На этапе определения целесообразности автоматизации тестирования на проекте «САУМИ»:

– произведено сравнение трудозатрат и объема работ при ручном и автоматизированном тестировании при проведении дымового тестирования.

– определены преимущества автоматизации.

На этапе проектирования информационной модели предметной области:

– смоделирована контекстная диаграмма информационная система контроля качества выпускаемого программного продукта «САУМИ»;

– произведено разбиение контекстной диаграммы на крупные фрагменты – декомпозиция информационной системы контроля качества выпускаемого программного продукта «САУМИ»;

– в результате процедуры декомпозиции ИС определены четыре основных процесса – планирование работ, написание тест-кейсов, проведение тестирования, формирование отчетов;

– произведена декомпозиция основных, четырех процессов, для определения работ на каждом этапе, чтобы перейти к разработке.

На этапе разработки автоматизированных тестов программного обеспечения:

– определены проблемы, которые необходимо решить;

– описаны программные средства автоматизации, которые используются в работе;

– создан проект автоматизированных тестов с помощью программных средств;

– разработан автоматизированный тестовый сценарий;

– выполнен разработанный тест в среде разработки, результат;

– описано выполнение группы автоматизированных тестов;

– описано представление отчетов по результатам выполнения групп авто-тестов.

На этапе анализа и расчета экономической целесообразности автоматизации тестирования:

– произведен анализ экономической целесообразности;

– произведен расчет экономической целесообразности, расчет представлен в виде графиков.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Обеспечение качества – основные понятия и определения [Электронный ресурс]. – Режим доступа: http://www.protesting.ru/qa/. – (Дата обращения 20.04.2018)

2. Виды тестирования программного обеспечения [Электронный ресурс]. – Режим доступа: http://www.protesting.ru/testing/testtypes.html. – (Дата обращения 19.04.2018)

3. Введение в тестирование. Жизненный цикл разработки ПО. [Электронный ресурс]. – Режим доступа: https://confluence.bftcom.com/pages/viewpage.action?pageId=12493584. – (Дата обращения 20.04.2018)

4. Савин, Р. Тестирование Дот Ком, или Пособие по жесткому обращению с багами в интернет-стартапах / Р. Савин. – М: Дело, 2007. – 312 с.

5. Кенер, С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / С. Кенер, Д. Фолк, У. К. Нгуен. – К.: «ДиаСофт», 2001. – 544 с.

6. Процесс тестирования [Электронный ресурс]. – Режим доступа: http://www.protesting.ru/testing/testprocess.html. – (Дата обращения 20.04.2018)

7. Представительский сайт компании ООО «БФТ» [Электронный ресурс]. – Режим доступа: http://www.bftcom.com/company/. – (Дата обращения 03.05.2018)

8. Колесов, А. Методология и практика тестирования ПО [Электронный ресурс] / А. Колесов – Режим доступа http://www.polytech21.ru/rekomendatsii-po-oformleniyu. – (Дата обращения: 03.05.2018)

9. Гребенюк, В.М. Оценка целесообразности внедрения автоматизированного тестирования [Электронный ресурс] / В.М. Краснов –

Режим доступа: https://naukovedenie.ru/PDF/13tvn113.pdf. – (Дата обращения: 10.05.2018).

10. Плагин Selenium IDE для браузера Chrome [Электронный ресурс]. – Режим доступа: https://chrome.google.com/webstore/detail/katalon-recorder-selenium/ljdobmomdgdljniojadhoplhkpialdid. – (Дата обращения:20.05.2018).

11. Maven [Электронный ресурс]. – Режим доступа: https://www.codenotfound.com/maven-download-install-apache-maven-3-2-windows.html. – (Дата обращения: 20.05.2018).

12. «Что такое selenium?». Автоматизация веб-приложений через браузер [Электронный ресурс]. – Режим доступа: http://selenium2.ru/. – (Дата обращения:20.05.2018).

13. Тестирование с помощью TestNG в Java [Электронный ресурс]. – Режим доступа: devcolibri.com/тестирование-с-помощью-testng-в-java/. – (Дата обращения:20.05.2018).

14. Положение о тестировании в проектах разработки программного обеспечения [Электронный ресурс]. – Режим доступа: http://www.pmprofy.ru/files/581/testing.doc. – (Дата обращения:20.05.2018).

15. Требования к программным продуктам [Электронный ресурс]. – Режим доступа: http://referatdb.ru/geografiya/558/index.html?page=6. – (Дата обращения:20.05.2018).

16. Стратегия автоматизации тестирования [Электронный ресурс]. – Режим доступа: http://agilerussia.ru/practices/test-automation-strategy/. – (Дата обращения:20.05.2018).

17. Hoffman, D. Cost Benefits Analysis of Test Automation [Электронный ресурс] / D. Hoffman – Режим доступа: http://www.softwarequalitymethods.com/papers/star99%20model%20paper.pdf. – (Дата обращения: 25.05.2018).

18. Jonson, D. Test Automation ROI [Электронный ресурс] / D. Jonson – Режим доступа: http://www.dijohnic.com/test_automation_roi.pdf. (Дата обращения: 22.05.2018).

19. Kaner, C. Improving the Maintainability of Automated Test Suites [Электронный ресурс] / C. Kaner – Режим доступа: http://www.kaner.com/pdfs/autosqa.pdf/. – (Дата обращения: 25.05.2018).

20. SQA Days [Электронный ресурс]. – Режим доступа: http://sqadays.com/. – (Дата обращения: 23.05.2018).

21. Тренды в тестировании [Электронный ресурс]. – Режим доступа: http://qapl.net/. – (Дата обращения: 23.05.2018).

22. Надежный тест-дизайн [Электронный ресурс]. – Режим доступа: http://www.myshared.ru/slide/498416/. – (Дата обращения: 24.05.2018).

23. Стратегия автоматизации тестирования [Электронный ресурс]. – Режим доступа: http://agilerussia.ru/practices/test-automation-strategy/. – (Дата обращения: 24.05.2018).

24. Внедрение автоматизации тестирования ПО на уровне проекта [Электронный ресурс]. – Режим доступа: http://software-testing.ru/library/testing/testing-automation/1004-test-automation-setting – (Дата обращения: 25.05.2018).

25. Методология и практика тестирования ПО [Электронный ресурс]. – Режим доступа: http://www.interface.ru/home.asp?artId=6981. – (Дата обращения: 25.05.2018).

ПРИЛОЖЕНИЕ А

Методы тестирования ПО

Методы тестирования программного обеспечения в зависимости от доступа тестировщика к исходному коду представлены на рисунке 1.3.

– Метод черного ящика (Black box);

Метод тестирования, основанный на работе только с внешним интерфейсом системы, не предполагающий знаний устройства системы. Имитация действий пользователя.

– Метод белого ящика (White box);

Метод тестирования основан на знании структуры/устройства и реализации системы. Входные значения выбираются, основываясь на знании кода, который будет их обрабатывать.

– Метод серого ящика (Grey box).

Это комбинация белого и черного ящика, предполагает частичное знание внутреннего устройства программы.

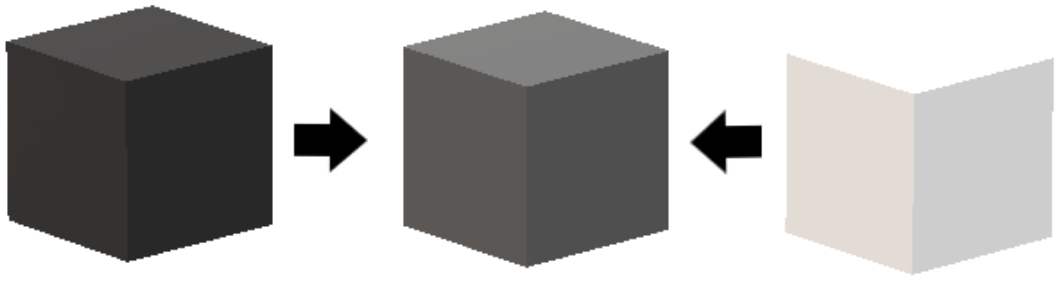

Рисунок А.1 – Схема представления методов тестирования путем черный/белый/серый ящик

ПРИЛОЖЕНИЕ Б

Пример полного готового авто-теста

```
package com.bft.Saumi.tests;

import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.*;
import java.util.concurrent.TimeUnit;
import static org.testng.Assert.*;

public class logIn {
    private WebDriver driver;
    private String baseUrl;
    private boolean acceptNextAlert = true;
    private StringBuffer verificationErrors = new StringBuffer();

    @BeforeClass(alwaysRun = true)
    public void setUp() throws Exception {
        driver = new FirefoxDriver();
        baseUrl = "https://www.katalon.com/";
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    }

    @Test //Шаги теста
    public void testUntitledTestCase() throws Exception {

driver.get("http://dev.eisugi.mosreg.ru:8888/login.jsp;jsessionid=111AC3B9810BF93EC28
676468E472C3D");
        driver.findElement(By.id("j_username")).click();
```

```
    driver.findElement(By.id("j_username")).clear();
    driver.findElement(By.id("j_username")).sendKeys("a.alengoz");
    driver.findElement(By.id("j_password")).click();
    driver.findElement(By.id("j_password")).clear();
    driver.findElement(By.id("j_password")).sendKeys("a.alengoz");
    driver.findElement(By.id("submit")).click();
}

@AfterClass(alwaysRun = true)
public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
        fail(verificationErrorString);
    }
}

private boolean isElementPresent(By by) {
    try {
        driver.findElement(by);
        return true;
    } catch (NoSuchElementException e) {
        return false;
    }
}

private boolean isAlertPresent() {
    try {
        driver.switchTo().alert();
        return true;
    } catch (NoAlertPresentException e) {
        return false;
```

```
        }
    }

    private String closeAlertAndGetItsText() {
        try {
            Alert alert = driver.switchTo().alert();
            String alertText = alert.getText();
            if (acceptNextAlert) {
                alert.accept();
            } else {
                alert.dismiss();
            }
            return alertText;
        } finally {
            acceptNextAlert = true;
        }
    }
}
```