МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 «Прикладная информатика»
(код и наименование направления подготовки, специальности)

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка проекта автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер»

Студент А.Д. Чиркин
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н., Н.В. Хрипунов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Тема бакалаврской работы – «Разработка проекта автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер»».

Как показывает практика, высокая эффективность процесса интеграционного тестирования достигается с помощью его автоматизации, которая позволяет снизить негативное влияние человеческого фактора на результаты тестирования. С этой целью в компании ООО «НетКрэкер» принято решение разработать проект автоматизации интеграционного тестирования информационных систем.

Объектом исследования бакалаврской работы является автоматизация интеграционного тестирования информационных систем в ООО «НетКрэкер».

Предметом исследования бакалаврской работы является проект автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер».

Цель выпускной квалификационной работы – разработка проекта автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер».

Практическая значимость бакалаврской работы заключается в разработке проектного решения для автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер», обеспечивающей повышение его эффективности.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы.

Бакалаврская работа состоит из 46 страниц текста, 16 рисунков, 7 таблиц и 25 источников.

Оглавление

Введение

Одной из задач тестирования программного обеспечения информационной системы является проверка соответствия ее проектируемых единиц требованиям, предъявляемым к функциональности и надежности.

Для решения данной задачи используется интеграционное тестирование.

Интеграционное тестирование - это способ тестирования программного обеспечения путем группировки программных компонентов.

Интеграционное тестирование проводится для оценки соответствия информационной системы или компонента заданным функциональным требованиям.

Как показывает практика, высокая эффективность процесса интеграционного тестирования достигается с помощью его автоматизации, которая позволяет снизить негативное влияние человеческого фактора на результаты тестирования.

С этой целью в компании ООО «НетКрэкер» принято решение разработать проект автоматизации интеграционного тестирования информационных систем.

Реализация данного проекта представляет актуальность и научно-практический интерес.

Объектом исследования бакалаврской работы является автоматизация интеграционного тестирования информационных систем в ООО «НетКрэкер».

Предметом исследования бакалаврской работы является проект автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер».

Цель выпускной квалификационной работы – разработка проекта автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер».

Для достижения данной цели необходимо выполнить следующие задачи:

- произвести анализ предметной области и выполнить постановку задачи на разработку проекта автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер»;
- спроектировать автоматизированную систему нтеграционного тестирования информационных систем в ООО «НетКрэкер»;
- реализовать проектное решение автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер».

Методы исследования – методы тестирования и проектирования информационных систем.

Практическая значимость бакалаврской работы заключается в разработке проектного решения для автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер», обеспечивающей повышение его эффективности.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы.

Во введении описаны актуальность, объект, предмет, цель и задачи исследования.

Первая глава посвящена анализу предметной области и постановке задачи на разработку проекта автоматизации интеграционного тестирования.

Вторая глава посвящена проектированию автоматизированной системы интеграционного тестирования для ООО «НетКрэкер».

В третьей глава рассматривается реализация проектных решений.

В заключении описываются результаты выполнения выпускной квалификационной работы.

Бакалаврская работа состоит из 46 страниц текста, 16 рисунков, 7 таблиц и 25 источников.

Глава 1 Анализ предметной области и постановка задачи на разработку проекта автоматизации интеграционного тестирования

1.1 Анализ предметной области автоматизации

ООО «НетКрэкер» (до 2016 года — NetCracker Technology или NetCracker) — дочерняя компания корпорации NEC, специализирующаяся на создании, внедрении и сопровождении систем эксплуатационной поддержки (OSS), систем поддержки бизнеса (BSS), а также SDN/NFV-решений для операторов связи, крупных предприятий и государственных учреждений.

Организационная структура компании ООО «НетКрэкер» в г. Тольятти представлена на рисунке 1.

Рисунок 1 – Организационная структура ООО «НетКрэкер»

«Общая характеристика предприятия ООО «НетКрэкер» показала, что на современном этапе развития данная организация является одним из лидеров по разработкам программного обеспечения.

Показатели результативной деятельности предприятия являются удовлетворительными и свидетельствуют о большем потенциале организации» [10].

Профессионализм сотрудников и тщательно налаженные процессы разработки позволяют ООО «НетКрэкер» поставлять своим заказчикам самые эффективные ИТ-решения, сочетающие лучшие черты заказных и тиражируемых продуктов.

Одной из профессиональных задач, которые решаются в компании ООО «НетКрэкер», является интеграционное тестирование программного обеспечения информационных систем, которое выполняется тестировщиками в отделе тестирования.

Для повышения эффективности интеграционного тестирования в компании принято решение внедрить современные технологии его автоматизации.

В этой связи представляет интерес проектирование ИТ-решения, обеспечивающего автоматизацию интеграционного тестирования информационных систем в ООО «НетКрэкер».

Цель проекта – разработка ИТ-решения для автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер».

1.2 Анализ процесса интеграционного тестирования

Поток работ интеграционного тестирования информационной системы (ИС) изображен на рисунке 2.

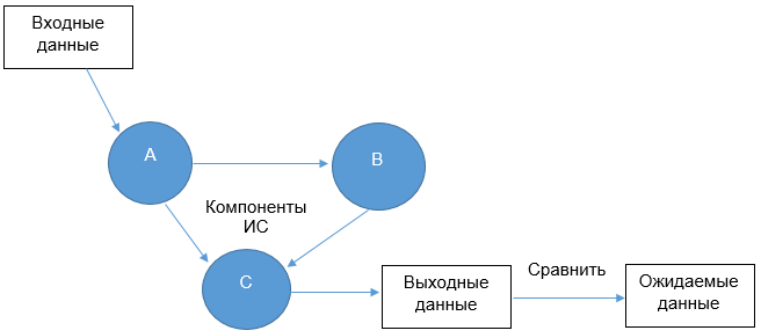

Рисунок 2 - Поток работ интеграционного тестирования ИС

Здесь тестовые входные данные предоставляются системному компоненту A, который затем взаимодействует с компонентами B и C.

Результат выходных данных затем проверяется с ожидаемыми выходными данными.

Существующий бизнес-процесс интеграционного тестирования состоит из следующих операций [25]:

– разработка функций и написание тестового примера.

Инженеры начинают с написания кода для функций и их тестовых примеров. После завершения разработки эти функции будут перенесены в удаленные ветки, которые вызовут запуск интеграционных тестов;

– создание качественной среды интеграционного тестирования.

Необходима выделенная, а не общая среда, поскольку она обеспечивает хорошее управление ресурсами, контроль и ведение журнала для диагностики и отладки. Следовательно, каждый запуск интеграционного теста должен начинаться с очищенной среды;

– планирование и проведение тестов.

Для каждого запроса на проверку новой бизнес-функции должна быть запланированная интеграционная тестовая сборка и запуск для проверки этой функции;

– формирование отчета о результатах интеграционного тестирования.

Отчет о результатах тестирования предупреждает членов команды о статусе разработки функции и позволяет им быстро отреагировать в случае возникновения проблемы. Это означает, что после завершения выполнения всех тестовых примеров должна существовать процедура отчетности, которая будет отправлять отчет о результатах интеграционного тестирования.

Тестирование выполняется тестировщиком в ручном режиме.

Для анализа существующего бизнес-процесса интеграционного тестирования используем методологию реинжиниринга бизнес-процессов

[11].

Целью реинжиниринга является повышение эффективности интеграционного тестирования информационных систем в ООО «НетКрэкер».

На первом этапе необходимо построить модель исследуемого бизнес-процесса «КАК ЕСТЬ».

Для моделирования бизнес-процессов используем методологию ARIS и программное средство ARIS Express [14].

Главным преимуществом данного подхода является возможность разработки, моделирования и улучшения бизнес-процессов в одном относительно простом инструменте.

На рисунке 3 представлена диаграмма бизнес-процесса интеграционного тестирования ИС «КАК ЕСТЬ» в ООО «НетКрэкер», построенная в нотации eEPC ARIS [6].

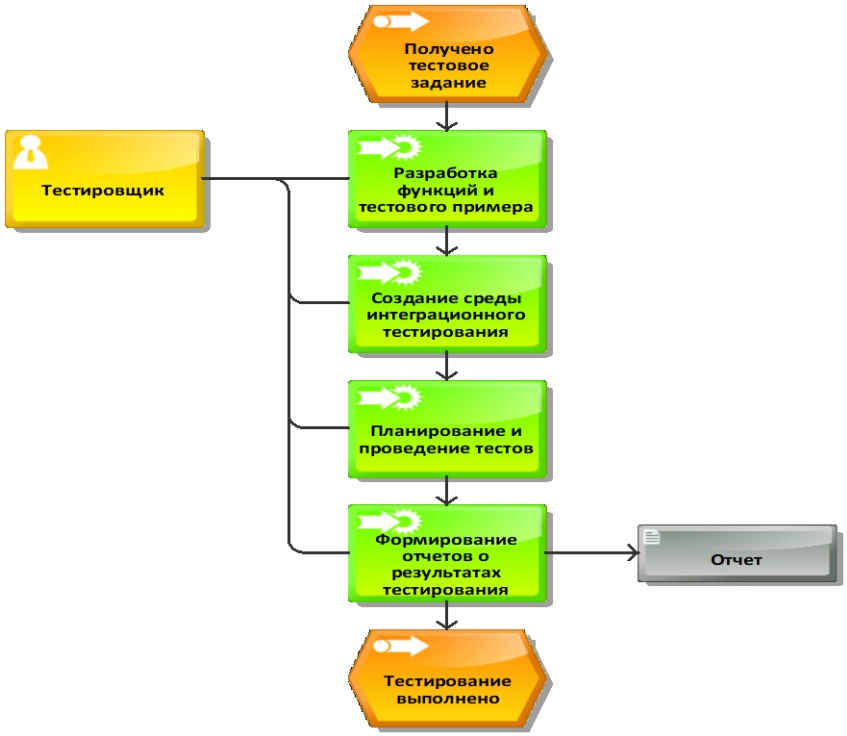

Рисунок 3 - Диаграмма бизнес-процесса интеграционного тестирования «КАК ЕСТЬ»

Диаграмма построена с точки зрения тестировщика.

Анализ существующего бизнес-процесса произведен при участии бизнес-аналитика.

В результате анализа выявлен основной недостаток существующего бизнес-процесса – низкая эффективность ручного тестирования, обусловленная его трудоемкостью, которая связана с необходимостью самостоятельной разработки среды интеграционного тестирования, а также негативным влиянием человеческого фактора [2].

Для улучшения бизнес-процесса предложено его автоматизировать с помощью специализированного ПО.

«Автоматизированное тестирование - это метод тестирования ПО, который выполняется с использованием специальных программных средств для выполнения набора тестовых примеров.

Напомним, что ручное тестирование выполняется человеком, сидящим перед компьютером, и тщательно выполняющим шаги теста.

ПО для автоматизации тестирования также может вводить тестовые данные в тестируемую систему, сравнивать ожидаемые и фактические результаты и создавать подробные отчеты о тестах.

Цель автоматизации тестирования - уменьшить количество тестовых примеров, которые нужно запускать вручную, а не полностью исключить ручное тестирование» [21].

Автоматизированное тестирование тесно связано с именем Майка Кона, который представил процесс автоматизации системы и проверки продуктов в виде пирамиды.

Разработанная М. Коном пирамида состоит из трех элементов, представленных на рисунке 5 [23].

Рисунок 4 – Пирамида автоматизированного тестирования

Пирамида автоматизированного тестирования состоит из следующих основных слоев:

– автоматизированное модульное тестирование;
– автоматизированное интеграционное тестирование;
– автоматизированное сквозное тестирование.

Таким образом, автоматизированное интеграционное тестирование находится на в середине пирамиды тестирования.

Представленная пирамида типична для тестирования программного обеспечения.

Но ее модификаций очень много. Все зависит от типа проверки, будь то мобильное тестирование или тестирование приложений.

Составные элементы пирамиды могут быть изменены, как и их расположение.

Диаграмма автоматизированного бизнес-процесса интеграционного тестирования («КАК ДОЛЖНО БЫТЬ») в ООО «НетКрэкер» представлена на рисунке 5.

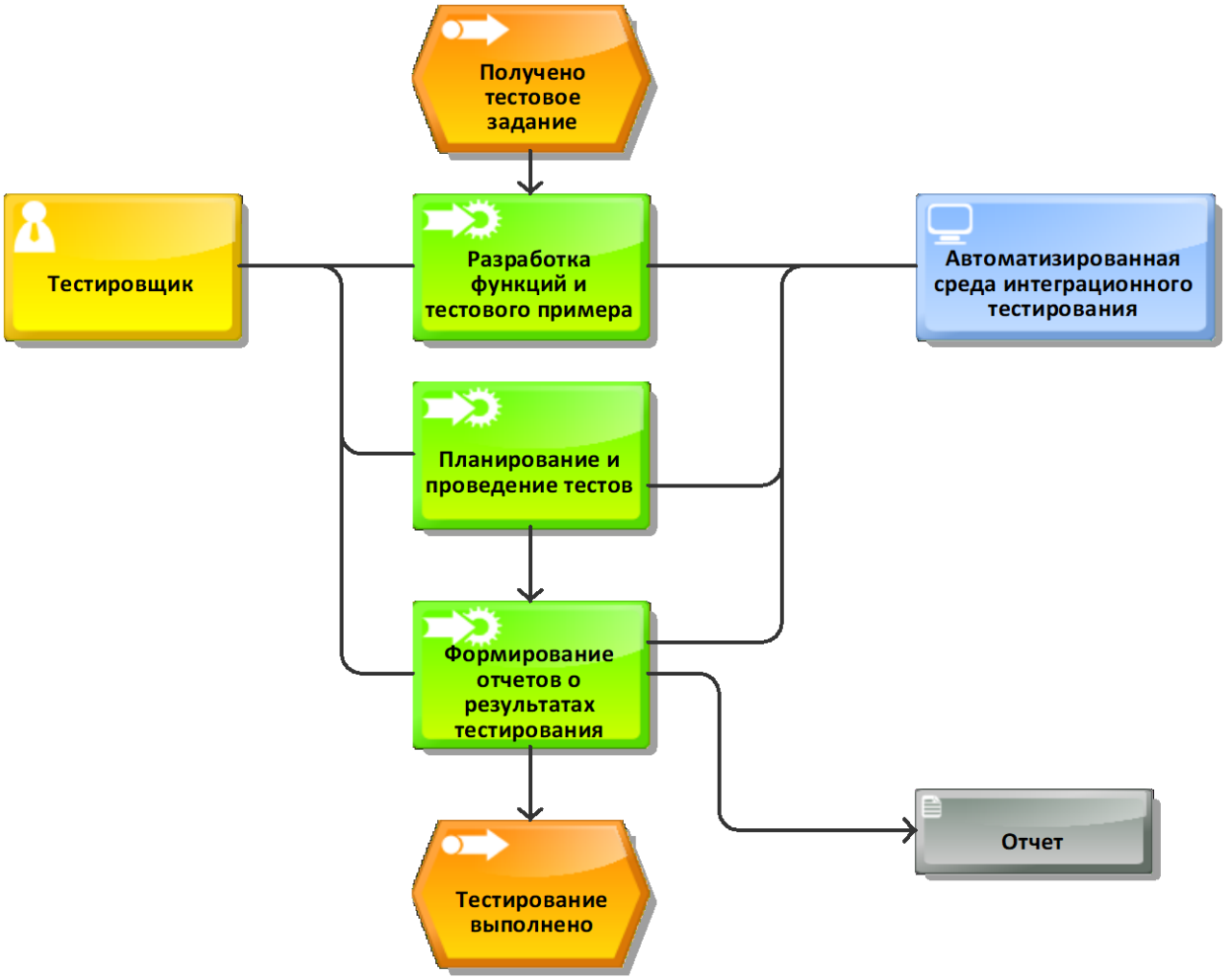

Рисунок 5 - Диаграмма бизнес-процесса интеграционного тестирования «КАК ДОЛЖНО БЫТЬ»

Таким образом, улучшение существующего бизнес-процесса достигается за счет внедрения в него автоматизированной среды интеграционного тестирования (АСИТ).

АСИТ - это комбинация оборудования, программного обеспечения, данных и конфигурации, необходимых для выполнения тестовых примеров.

В среде данного типа тестировщик интегрирует отдельные программные модули, а затем проверяет поведение интегрированной системы. Набор интеграционных тестов используется для проверки того, что система ведет себя так, как указано в документе с требованиями [22].

В АСИТ можно интегрировать один или несколько модулей своего приложения и проверить функциональную корректность.

При современном подходе DevOps к разработке программного обеспечения, где непрерывное тестирование является нормой, АСИТ, вероятно, будет использоваться ежедневно или несколько раз в день.

Следовательно, наличие эффективной АСИТ имеет первостепенное значение для эффективного процесса доставки программного обеспечения.

1.3 Разработка технического задания на проектирование

Для разработки технического задания (ТЗ) на проектирование автоматизированной среды интеграционного тестирования ИС используем ГОСТ 34.602-89 [4].

Согласно данному ГОСТу «ТЗ является основным документом, определяющим требования и порядок создания (развития или модернизации) автоматизированной системы (АС), в соответствии с которым проводится разработка АС и ее приемка при вводе в действие».

ТЗ состоит из следующих разделов:

Пункт 1. Общие сведения.

Полное наименование системы: Автоматизированная среда интеграционного тестирования ИС.

Порядок оформления и предъявления заказчику результатов работ: результатом разработки является АСИТ, доступная для проведения приемочных испытаний.

Пункт 2. Назначение и цели создания системы.

АСИТ предназначена для автоматизации интеграционного тестирования ИС.

Пункт 3. Цель создания системы.

Цель создания АСИТ – информационная поддержка процесса интеграционного тестирования ИС в ООО «НетКрэкер».

Пункт 4. Характеристика объекта автоматизации.

Объектом автоматизации является интеграционного тестирования ИС в

ООО «НетКрэкер».

Пункт 5. Требования к системе.

Для разработки требований к АСИТ используем методологию FURPS+ [7].

FURPS+ - это метод проверки приоритетных требований после понимания потребностей и потребностей клиента.

Аббревиатура FURPS означает «функциональность, удобство использования, надежность, производительность и возможность поддержки».

Методология FURPS+ в классификации требований делает упор на понимание различных типов функциональных и нефункциональных требований [24].

Функциональное требование описывает, что должна делать программная система, в то время как нефункциональные требования накладывают ограничения на то, как система ИС это делать.

В таблице 1 представлены основные требования к ПО автоматизированной среды интеграционного тестирования с учетом особенностей методологий FURPS+.

Таблица 1– Требования к АСИТ

№	Требование	Статус	Полезность	Риск	Стабильность
	1	2	3	4	5
Functionality — Функциональные требования					
1.	Обеспечение автоматизации интеграционного тестирования	Одобренное	Критическая	Средний	Низкая
Usability— Требования к удобству использования					
2.	Дружественный интуитивный интерфейс	Одобренное	Критическая	Средний	Низкая
Reliability— Требования к надежности					
3.	Допустимая частота/периодичность сбоев: 1 раз в 300 часов	Одобренное	Важная	Средний	Средняя
4.	Среднее время сбоев: 1 раб. день	Одобренное	Важная	Средний	Средняя

Продолжение таблицы 1

№	1	2	3	4	5
5.	Возможность восстановления системы после сбоев: 1 раб. день	Одобренное	Важная	Средний	Средняя
6.	Режим работы: 7/24/365	Одобренное	Важная	Средний	Средняя
Performance — Требования к производительности					
7.	Допустимое количество одновременно работающих пользователей: 1	Предложенное	Важная	Средний	Средняя
8.	Время реакции на возникновение аварийной ситуации: 10 с	Предложенное	Важная	Средний	Средняя
Supportability — Требования к поддержке					
9.	Время устранения критических проблем: в течение рабочего дня	Предложенное	Важная	Средний	Средняя
Проектные ограничения					
10.	Применение современных ИТ	Предложенное	Критическая	Средний	Низкая
11.	Низкая стоимость владения	Предложенное	Критическая	Средний	Низкая

Разработанный перечень требований является основой для реализации проектного решения автоматизации интеграционного тестирования ИС в ООО «НетКрэкер».

Пункт 5.1. Требования к графическому дизайну приложения:

- основное меню АСИТ должно быть доступно сразу после активизации приложения;
- на рабочем пространстве программы не должно быть большого объема текстовой информации;
- в дизайне программы не должны присутствовать агрессивные цветовые сочетания и графические решения;
- в целом оформление приложения не должно ущемлять его информативность;
- приложение должно быть удобно пользователям в плане навигации

и интересно для многократного посещения.

Пункт 5.2. Требования к структуре АСИТ.

Структура ИСУ должна состоять из следующих разделов:

- рабочая среда;
- меню пользователя;
- количество и наименование пунктов меню могут корректироваться по согласованию с Заказчиком в ходе проектирования АСИТ.

Пункт 5.3. Требования к управлению АСИТ.

Для управления АСИТ должны быть введены следующие роли:

- администратор АСИТ, обеспечивающий общее управление приложением;
- пользователь АСИТ;
- для работы с АСИТ от пользователей не должна требоваться предварительная регистрация;
- пользователи АСИТ (тестировщики) должны обладать специальными техническими навыками интеграционного тестирования ИС.

Пункт 5.4. Требования к информационному обеспечению.

Результаты тестирования должны храниться в специальной папке на диске компьютера тестировщика.

Пункт 5.5. Требования к лингвистическому обеспечению:

- пользовательский интерфейс АСИТ должен быть выполнен на английском или русском языках;
- пользовательский интерфейс АСИТ должен обеспечивать наглядное, интуитивно понятное представление информации, быстрый и логичный переход к разделам и страницам;
- навигационные элементы должны обеспечивать однозначное понимание пользователем их смысла: ссылки на страницы должны быть снабжены заголовками, условные обозначения

соответствовать общепринятым.

Пункт 5.6. Требования к программному и техническому обеспечению:

а) АСИТ должна быть разработана на свободно распространяемом ПО или платформе;

б) рекомендуемое программное и техническое обеспечение АСИТ:

- операционная система Linux, Windows 7/8/10;
- техническое обеспечение компьютера тестировщика должно обеспечивать поддержку вышеперечисленного программного обеспечения.

Пункт 5.7. Требования к документированию.

Вся документация должна быть подготовлена и передана как в печатном, так и в электронном виде (в форматах DOC, DOCX или PDF) Заказчику.

Выводы к главе 1

Первая глава посвящена анализу предметной области и постановке задачи на разработку проекта автоматизации интеграционного тестирования.

Результаты проделанной работы позволили сделать нижеследующие выводы.

В результате анализа выявлен основной недостаток существующего бизнес-процесса интеграционного тестирования в ООО «НетКрэкер» – низкая эффективность ручного тестирования, обусловленная его трудоемкостью, которая связана с необходимостью самостоятельной разработки среды интеграционного тестирования, а также негативным влиянием человеческого фактора. Улучшение существующего бизнес-процесса достигается за счет внедрения в него АСИТ.

Для разработки требований к проекту использована методология FURPS+. Разработанный перечень требований является основой для реализации проектного решения автоматизации интеграционного тестирования ИС в ООО «НетКрэкер».

Глава 2 Проектирование автоматизированной системы интеграционного тестирования для ООО «НетКрэкер»

2.1 Обзор и анализ средств интеграционного тестирования

Большинство существующих технологий интеграционного тестирования основано на использовании фреймворков [1].

Фреймворк - это программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта [12].

Фреймворк автоматического тестирования - это набор условий, концепций и практик, направленный на переиспользование, уменьшение затрат на поддержку и повышение надежности использования тестов.

В ООО «НетКрэкер» интеграционное тестирование использует для решения следующих задач:

- тестирование веб-приложений;
- тестирование баз данных;
- тестирование фронт-энда.

Для автоматизации интеграционного тестирования данных задач выбраны следующие фреймворки, представленные в таблице 2.

Таблица 2 Рекомендуемые фреймворки по областям интеграционного тестирования

Область тестирования	Фреймворк
Тестирование веб-приложений	Selenium
Тестирование баз данных	h2+DbUnit
Тестирование фронт-энда	Cucumber

Выбор фреймворков основан на личном опыте проектанта и с учетом успешного опыта их применения специалистами-тестировщиками [19].

Рассмотрим характеристики указанных фреймворков на предмет использования в проектируемой АСИТ для ООО «НетКрэкер».

2.1.1 Фреймворк Selenium

Selenium - это масштабный проект с открытым исходным кодом, предлагающий несколько решений для автоматизации тестирования. Он используется в основном для веб-приложений, и может быть использован для интеграционного тестирования.

Фактически, Selenium стал отраслевым стандартом благодаря своей гибкости.

Вот некоторые особенности и особенности, которые делают его самым популярным инструментом:

- поддержка нескольких языков - C#, Java, JavaScript, PHP, Python, Ruby, Perl;
- работа в разных операционных системах - Windows, Mac, Linux;
- работа со всеми популярными браузерами, включая Chrome, Firefox, Safari и Headless;
- использование стандарта W3C упрощает создание сценариев и тестирование;
- возможность запускать параллельные тесты с разными гибридными тестовыми данными;
- поддержка различных расширений и интеграции с другими программными решениями.

При конфигурировании Selenium Server нужно указать всего 4 параметра: адрес и порт Selenium сервера, используемый браузер и адрес, по которому расположено тестируемое веб-приложение.

Код интеграционного демо-теста, созданный с помощью Selenium, представлен ниже.

«public class SeleniumExampleTests

{

[Fact]

public void SeleniumExample1()

{

ISelenium selenium = new DefaultSelenium("localhost",

5561,

"*firefox",

"localhost:8099");

selenium.Start();

selenium.Open("/Account/LogOn");

selenium.WaitForPageToLoad("10000");

selenium.Type("Name","username");

selenium.Type("Password","12345");

selenium.Click("//input[@value='Войти']");

selenium.WaitForPageToLoad("10000");

Assert.True(selenium.IsTextPresent("Вход успешен!"));»[20]

selenium.Stop();

}

}

Среди недостатков следует выделить сложность написания сценариев и построения фреймворков. Также Selenium не идеален для тестирования API веб-сервисов.

Следует учесть, что Selenium - это только часть стадии интеграционного тестирования, поскольку он не объединяет продукт в группу и не проверяет целостность данных. Он просто управляет продуктом как пользователь.

2.1.2 Фреймворк h2+DbUnit

h2 – это полезная база данных (БД), которую можно использовать при написании интеграционных тестов для локальной среды разработки [18].

Она также используется для языка программирования Java.

Основные преимущества:

– это самая быстрая база данных из доступных, и она имеет расширенные функции безопасности;
– h2 имеет три режима подключения - встроенный, серверный и смешанный. Фактически, файл .jar занимает всего 2 МБ;
– фреймворк поставляется с надежной резервной копией - веб-сайтом с инструкциями, документацией, поддержкой и т. д.

h2 используется совместно с фреймворком DbUnit.

DbUnit — расширение для JUnit, которое может быть использовано для инициализации БД в известное состояние перед выполнением каждого интеграционного теста и заполнения БД нужными данными [12].

Пример кода интеграционного теста, созданного с помощью с помощью Dbunit и h2, представлен ниже.

```
import static org.assertj.core.api.Assertions.assertThat;
 @RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = {PersisitenceTestConfig.class},
      loader = AnnotationConfigContextLoader.class)
@TestExecutionListeners
({DependencyInjectionTestExecutionListener.class,
      DirtiesContextTestExecutionListener.class,
      TransactionalTestExecutionListener.class,
      DbUnitTestExecutionListener.class})
@DbUnitConfiguration(databaseConnection = "h2Connection")
@DatabaseSetup("classpath:claim_search_dataset.xml")
public class ClaimServiceTestIT {
   @Autowired
  TopicRepo topicRepo;
  @Autowired
  TopicService TopicService;
```

```
    @Test
    public void testTopicSearch() {
        Topic topic1 = topicRepo.findOne(1l);
        Topic topic2 = topicRepo.findOne(2l);
        Topic topic20 = topicRepo.findOne20l);
        ...
        List<Topic> topiscs = topicService.findBySomeCriteria(...);
        assertThat(claims)
                .hasSize(3)
                .doesNotContainNull()
                .contains(topic1)
                .contains(topic2)
                .contains(topic20);
    }
}
```

Следует отметить, что у DbUnit есть свои недостатки, но это очень полезный инструмент, позволяющий разделить тестовые данные и тестовый код.

2.1.3 Фреймворк Cucumber

Cucumber - это инструмент тестирования, который поддерживает метод BDD [16].

Он предлагает способ писать тесты, понятные каждому, независимо от их технических знаний.

В BDD пользователи (бизнес-аналитики, владельцы продуктов) сначала пишут сценарии или приемочные тесты, которые описывают поведение системы с точки зрения клиента, для проверки и утверждения владельцами продуктов, прежде чем разработчики напишут свои коды.

Фреймворк Cucumber использует язык программирования Ruby.

Фрагмент кода теста, созданного с помощью фреймворка Cucumber, представлен ниже.

```
package name.alexkosarev.sandbox.web.controllers.contacts;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;
import org.junit.runner.RunWith;
@RunWith(Cucumber.class)
@CucumberOptions(
        strict = true,
        glue = "name.alexkosarev.sandbox.web.controllers.contacts",
        features = "classpath:features/contacts/FindAll.feature")
public class FindAllTest {
}
```

Преимущества Cucumber:

– читаемость способствует командной работе;
– возможность повторного использования снижает количество необходимых шагов для тестирования;
– возможность создания универсального, но конкретного кода.

По мнению многих тестировщиков фронт-энда, Cucumber является лучшим инструментом для интеграционного тестирования.

2.2 Логическое моделирование автоматизированной среды интеграционного тестирования

Логическое проектирование определяет функции и особенности информационной системы и отношения между ее компонентами.

Логическое проектирование включает выходные данные, которые должны быть произведены информационной системой, входные данные, необходимые для системы, и процесс, который должен быть произведен системой, независимо от того, как задачи будут выполняться физически.

Логическое проектирование определяет то, что должно происходить в информационной системе, а не то, как это должно быть выполнено.

Логическое моделирование – это обязательный этап логического проектирования ИС.

Для разработки логической модели АСИТ используем язык UML и CASE-средство Rational Rose, которое поддерживает методологию RUP [9].

Для определения функциональных требований применяется диаграмма вариантов использования UML.

Диаграмма вариантов использования наглядно представляет взаимодействие между основными сервисами (бизнес-прецедентами), которые предоставляет исследуемый бизнес-процесс, и теми, кому эти сервисы предоставлены (бизнес-субъекты или акторы).

Как инструмент визуального моделирования и бизнес-анализа Rational Rose позволяет системному аналитику отслеживать бизнес-цели и сопоставлять их с системными требованиями, что существенно повышает эффективность процесса формирования последних.

На этапе управления требованиями RUP необходимо, чтобы все прецеденты и участники были определены, и было разработано большинство описаний прецедентов.

Акторами процесса интеграционного тестирования являются Тестировщик и фреймворки: Selenium, h2+DbUnit, Cucumber. Варианты использования (прецеденты) представлены в таблицах 3-5.

Таблица 3 – Описание прецедента: Интеграционное тестирование веб-приложений

Прецедент: Интеграционное тестирование веб-приложений
ID: 1
Краткое описание: Интеграционное тестирование веб-приложения
Главный актор: Тестировщик
Второстепенный актор: фреймворк Selenium
Предусловие: нет

Продолжение таблицы 3

Прецедент: Интеграционное тестирование веб-приложений
Основной поток: Тестировщик выполняет интеграционное тестирование веб-приложения в АСИТ
Постусловие: нет
Альтернативные потоки: нет

Таблица 4 – Описание прецедента: Интеграционное тестирование БД

Прецедент: Интеграционное тестирование БД
ID: 2
Краткое описание: Интеграционное тестирование БД
Главный актор: Тестировщик
Второстепенный актор: фреймворк h2+DbUnit
Предусловие: нет
Основной поток: Тестировщик выполняет интеграционное тестирование БД в АСИТ
Постусловие: нет
Альтернативные потоки: нет

Таблица 5 – Описание прецедента: Интеграционное тестирование фронт-энда

Прецедент: Интеграционное тестирование фронт-энда
ID: 3
Краткое описание: Интеграционное тестирование фронт-энда
Главный актор: Тестировщик
Второстепенный актор: фреймворк Cucumber
Предусловие: нет
Основной поток: Тестировщик выполняет интеграционное тестирование фронт-энда в АСИТ
Постусловие: нет
Альтернативные потоки: нет

На рисунке 6 представлена диаграмма вариантов использования АСИТ.

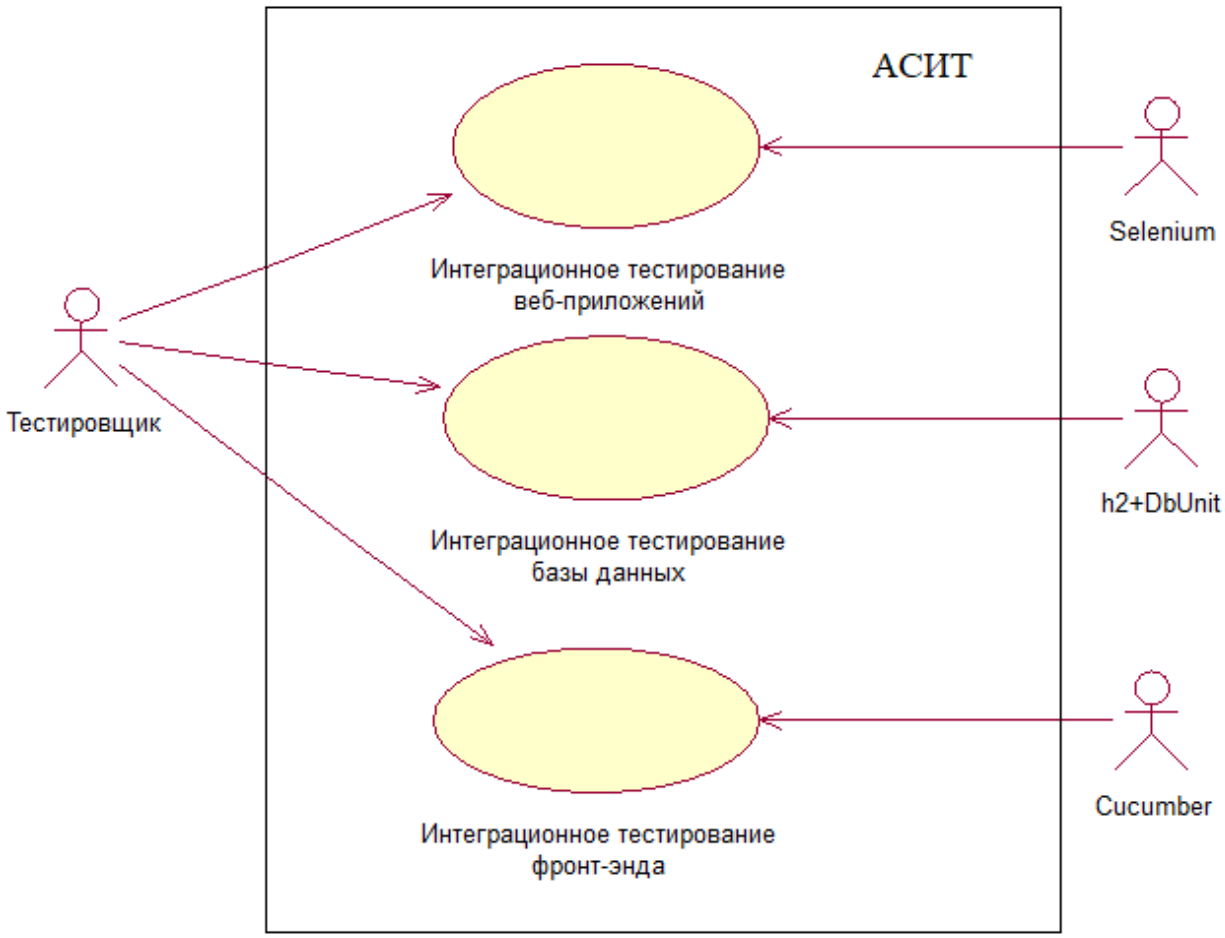

Рисунок 6 - Диаграмма вариантов использования АСИТ

Разработанная диаграмма вариантов использования отражает функциональный аспект АСИТ.

Для отражение статического аспекта АСИТ используем диаграмму классов UML.

Диаграмма классов в основном используется для визуализации структуры ИС.

Помимо этого, она также показывает основные компоненты, их отношения друг с другом и их соответствующие атрибуты.

Диаграмма классов предназначена в первую очередь для разработчиков, чтобы помочь им создать программу. Этот тип диаграммы помогает разработчикам понять компоненты, поток и атрибуты для каждого случая.

На рисунке 7 изображена диаграмма классов АСИТ.

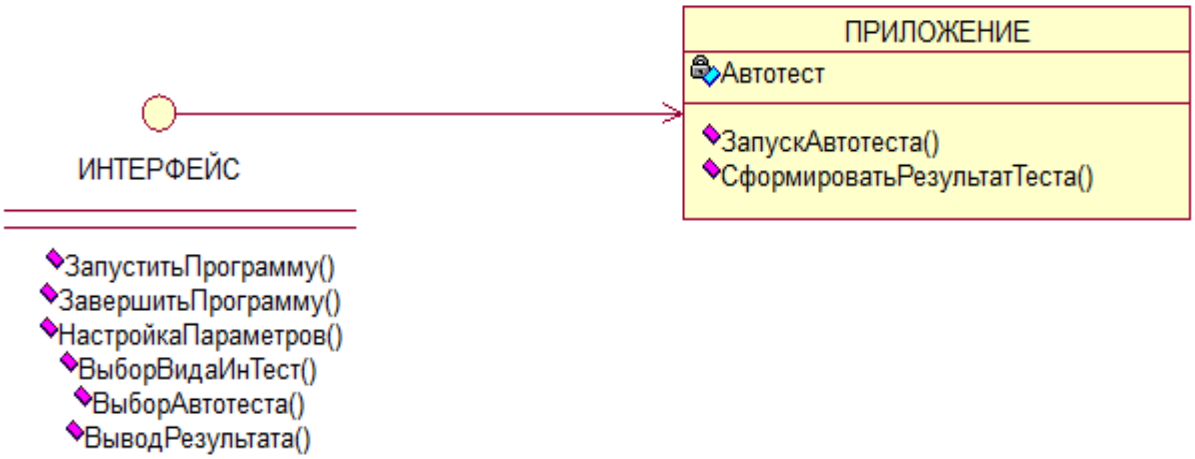

Рисунок 7 - Диаграмма классов АСИТ

Спецификация классов АСИТ представлена в таблице 6.

Таблица 6 - Спецификация классов АСИТ

Класс	Описание
Интерфейс	Класс объектов, моделирующих на логическом уровне интерфейс программы
Приложение	Класс объектов, моделирующих на логическом уровне приложение программы

Для представления сценария выполнения программных функций используем диаграмму последовательности UML.

Диаграммы последовательности подчеркивают хронологический ход обмена информацией.

Диаграммы последовательностей легче понять разработчикам и аналитикам.

Диаграмма последовательности иллюстрирует различные сценарии бизнес-варианта использования.

На рисунке 8 изображена диаграмма последовательности сценария интеграционного тестирования в АСИТ.

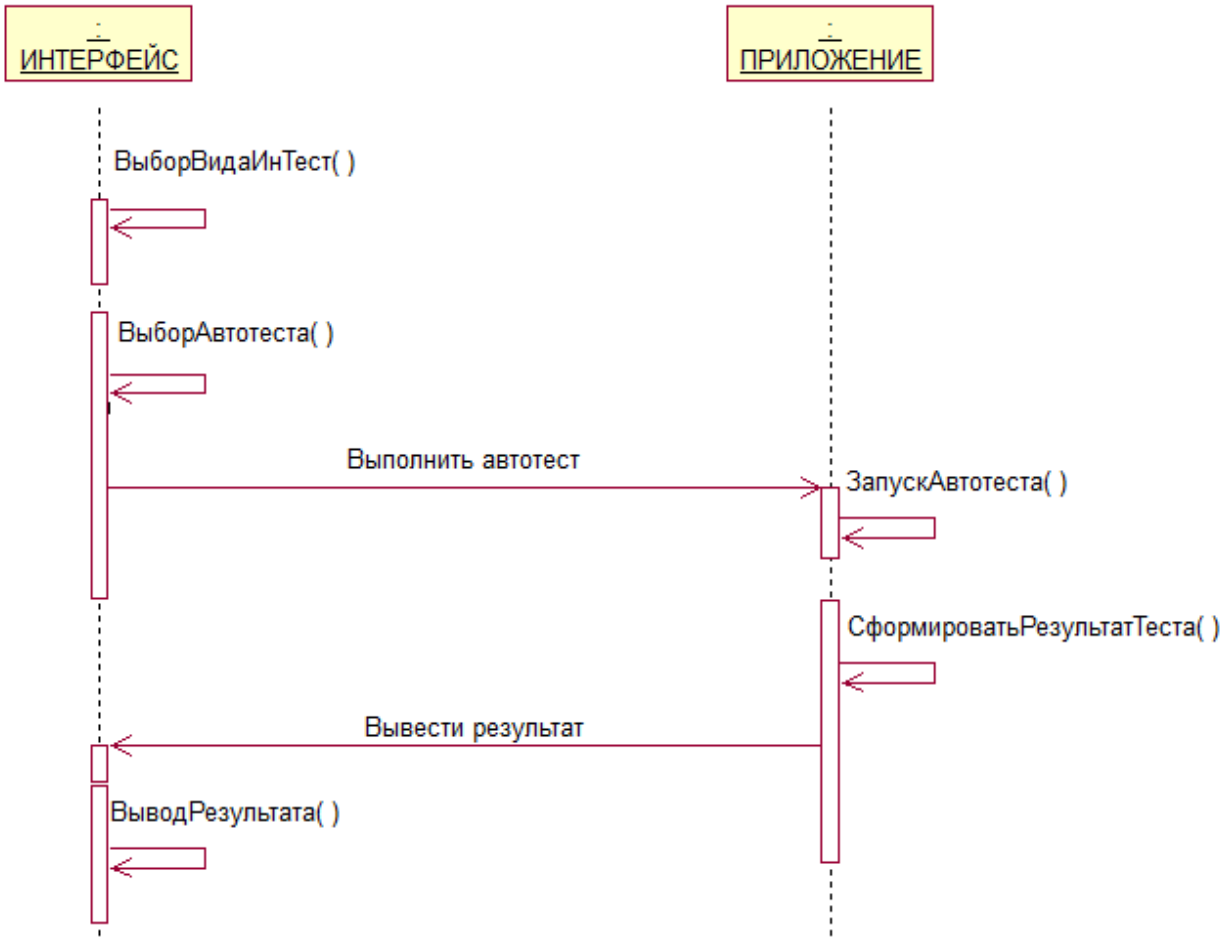

Рисунок 8 – Диаграмма последовательности сценария интеграционного тестирования в АСИТ

Сценарий интеграционного тестирования организован следующим образом:

В случайный момент времени объект Интерфейс по команде Тестировщика активизирует функцию выбора вида интеграционного

тестирования и автоматического выбора фреймворка.

Далее объект Интерфейс по команде Тестировщика активизирует функцию выбора автотеста.

Объект Приложение выполняет процедуру запуска автотеста и обращается к объекту Интерфейс с запросом на вывод результата тестирования.

Объект Интерфейс выводит результат тестирования на экран и/или на печать.

Процесс интеграционного тестирования завершается.

Диаграмма последовательности отображает динамический аспект АСИТ.

Выводы к главе 2

Вторая глава посвящена проектированию АСИТ.

Результаты проделанной работы позволили сделать нижеследующие выводы.

Для автоматизации интеграционного тестирования данных задач выбраны следующие фреймворки: Selenium, h2+DbUnit и Cucumber.

Выбор фреймворков основан на личном опыте проектанта и с учетом успешного опыта их применения другими специалистами-тестировщиками.

Для построения логической модели АСИТ разработаны базовые диаграммы языка UML, отражающие различные аспекты системы: диаграмму вариантов использования, диаграмму классов и диаграмму последовательности.

Для разработки логической модели АСИТ использовано CASE-средство Rational Rose, которое поддерживает методологию RUP.

Глава 3 Реализация проектных решений по автоматизации интеграционного тестирования ООО «НетКрэкер»

3.1 Выбор среды разработки автоматизированной среды интеграционного тестирования

Для реализации АСИТ используем технологию Integrated Development Environment (IDE).

«IDE – интегрированная среда разработки представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

Интегрированная среда разработки позволяет программистам объединять различные задачи написания компьютерной программы.

Интегрированные среды разработки повышают продуктивность программиста за счет объединения общих действий по написанию программного обеспечения в одном приложении: редактирование исходного кода, создание исполняемых файлов и отладка» [5].

В состав типовой интегрированной среды разработки входят:

- текстовый редактор;
- транслятор – компилятор или интерпретатор;
- средства автоматизации сборки;
- отладчик.

Рассмотрим функциональные и архитектурные особенности популярных интегрированных сред разработки.

3.1.1 Интегрированная среда разработки Visual Studio

Интегрированная среда разработки или IDE Visual Studio– это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений.

«Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает компиляторы, средства

автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки (рисунок 9).

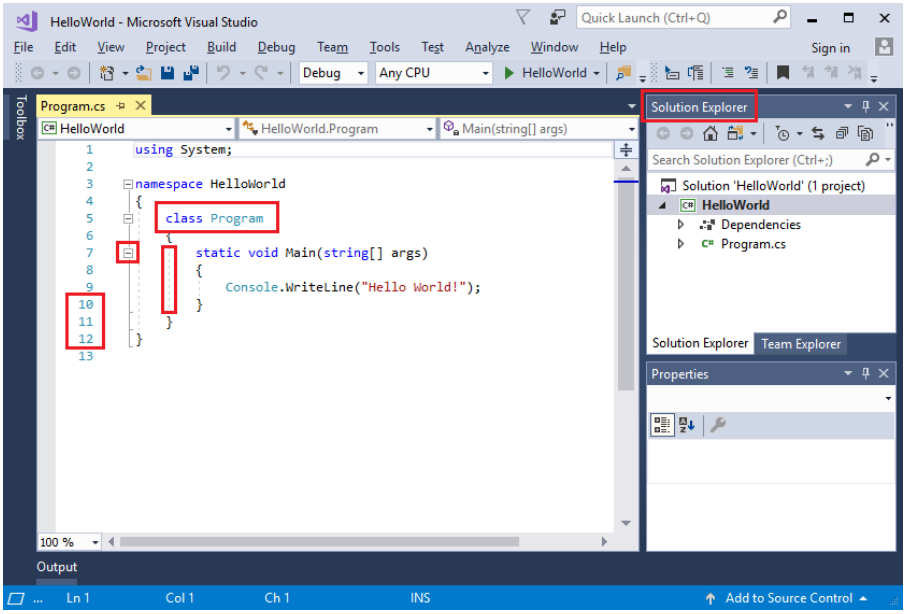

Рисунок 9 – Экран обозревателя IDE Visual Studio

Основными инструментами разработчика является следующие:

- обозреватель решений, который позволяет просматривать файлы кода, перемещаться по ним и управлять ими. Обозреватель решений позволяет упорядочить код путем объединения файлов в решения и проекты;
- редактор, отображающий содержимое файла. Здесь можно редактировать код или разрабатывать пользовательский интерфейс, например, окно с кнопками или текстовые поля;
- командный обозревать, который позволяет отслеживать рабочие элементы и использовать код совместно с другими пользователями с помощью технологий управления версиями, таких как Git и система управления версиями Team Foundation» [8].

Среда Visual Studio доступна для операционных систем Windows и Mac.

Возможности Visual Studio оптимизированы для разработки кроссплатформенных и мобильных приложений.

Существует три выпуска Visual Studio: Community, Professional и Enterprise.

3.1.2 Интегрированная среда разработки NetBeans

NetBeans – это бесплатная интегрированная среда разработки с открытым исходным кодом для разработчиков программного обеспечения (рисунок 10).

«Среда NetBeans предоставляет все средства, необходимые для создания профессиональных десктоп-приложений, корпоративных, мобильных и веб-приложений на платформе Java, а также C++, PHP и других языках» [apache.org] .

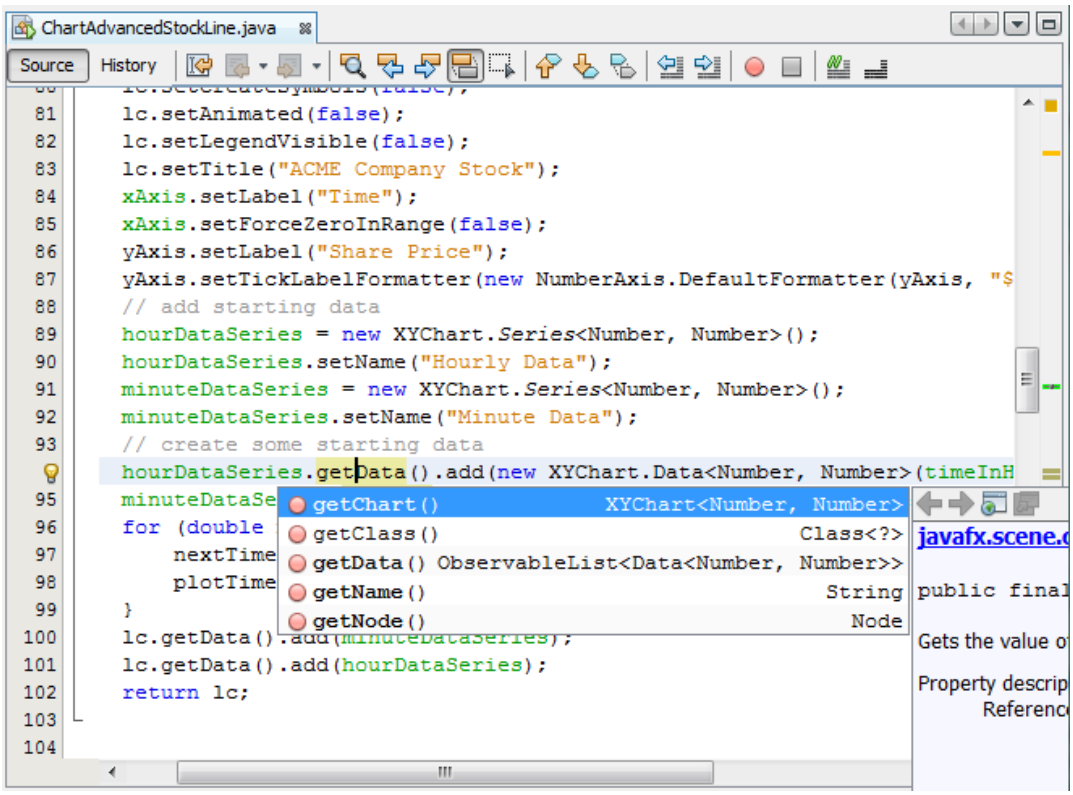

Рисунок 10 – Экран IDE NetBeans

Основные характеристики среды NetBeans следующие:

- рабочая область среды является полностью настраиваемой – существует возможность пользовательской настройки действий, выполняемых с помощью панели, назначения «горячих» клавиш и т. д.;
- «среда имеет в своем составе расширенный многоязыковой редактор для различных языков программирования, прежде всего для Java. Существует возможность расширения функций редактора с целью поддержки любого другого языка;
- Редактор среды делает отступы строк, проверяет соответствие скобок и слов, подсвечивает синтаксис исходного кода.
- Производятся проверка ошибок во время ввода, отображение вариантов для автозавершения кода и фрагментов документации по требуемому языку программирования.
- Редактор может генерировать и вставлять в исходный код стандартные фрагменты кода на Java или других языках.
- Браузер классов позволяет просматривать иерархию и структуру любого класса Java и другие.

Среда NetBeans позволяет разрабатывать Java десктоп-приложения с профессиональными графическими интерфейсами пользователя, а также создавать веб- и корпоративные приложения в соответствии с современными стандартами» [13].

3.1.3 Интегрированная среда разработки Eclipse

Интегрированная среда разработки Eclipse является бесплатной программной платформой с открытым исходным кодом, контролируется организацией Eclipse Foundation [17].

Среда написана на языке программирования Java.

Основной целью её создания является повышение продуктивности процесса разработки программного обеспечения.

Фрагмент среды представлен на рисунке 11.

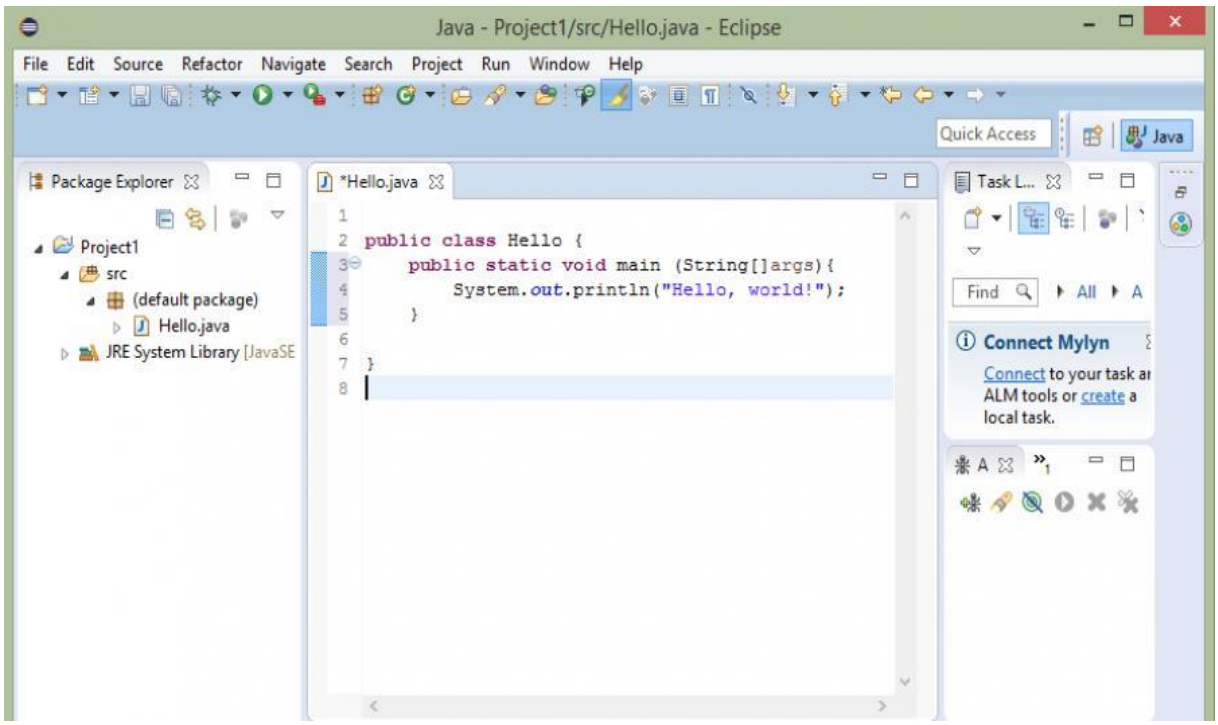

Рисунок 11 – Экран IDE Eclipse

«Основным компонентом является исполняемая среда – Eclipse Runtime, в которой выполняются коды расширений и модулей.

Она обеспечивает всю базовую функциональность – управление расширениями и обновлениями, взаимодействие с операционной системой, обеспечение работы системы помощи.

Следующим ключевым компонентом является собственно интегрированная среда разработки – она отвечает за управление элементами программы, управление проектами, отладку и сборку проектов, поиск по файлам и командную программу.

Eclipse претендует на статус наиболее популярной Java IDE и является единственным конкурентом такой мощной платформы, как NetBeans» [17].

Но в отличие от последней, использующей для создания элементов пользовательского интерфейса платформонезависимую библиотеку Swing, Eclipse использует платформозависимую библиотеку SWT.

«Интегрированные среды разработки, созданные на платформе Eclipse,

широко применяются для создания программного обеспечения на различных языках программирования.

Это обусловлено универсальностью Eclipse, работающей по принципу «Плагины для Eclipse разрабатываются в самой Eclipse» [17].

Для выбора интегрированной среды разработки используем таблицу 7, составленную на основе анализа блогов по данной тематике.

Критерии оценивания:

0 – полное несоответствие требованиям;

1 – значительное несоответствие требованиям;

2 – незначительное несоответствие требованиям;

3 – полное соответствие требованиям.

Таблица 7 – Сравнительный анализ интегрированных сред разработки

Характеристика/балл	Visual Studio	NetBeans	Eclipse
стоимость	2	1	3
юзабилити	3	2	2
простота интеграции с фреймворками	1	1	3
предпочтение разработчика	1	3	2
Итого	7	7	10

Таким образом, наилучшими характеристиками обладает IDE Eclipse.

Выбираем IDE Eclipse в качестве среды для разработки АСИТ.

3.2 Архитектура и функционирование автоматизированной среды интеграционного тестирования

Для представления программной архитектуры АСИТ используем диаграмму компонентов UML.

Диаграмма компонентов UML показывает компоненты, предоставленные и требуемые интерфейсы, порты и отношения между ними.

Диаграммы компонентов UML используются для создания архитектурного ландшафта конкретной системы.

Диаграмма компонентов позволяет моделировать программные компоненты высокого уровня и интерфейсы с этими компонентами.

Диаграмма компонентов имеет более высокий уровень абстракции, чем диаграмма классов UML.

На рисунке 12 изображена диаграмма компонентов АСИТ.

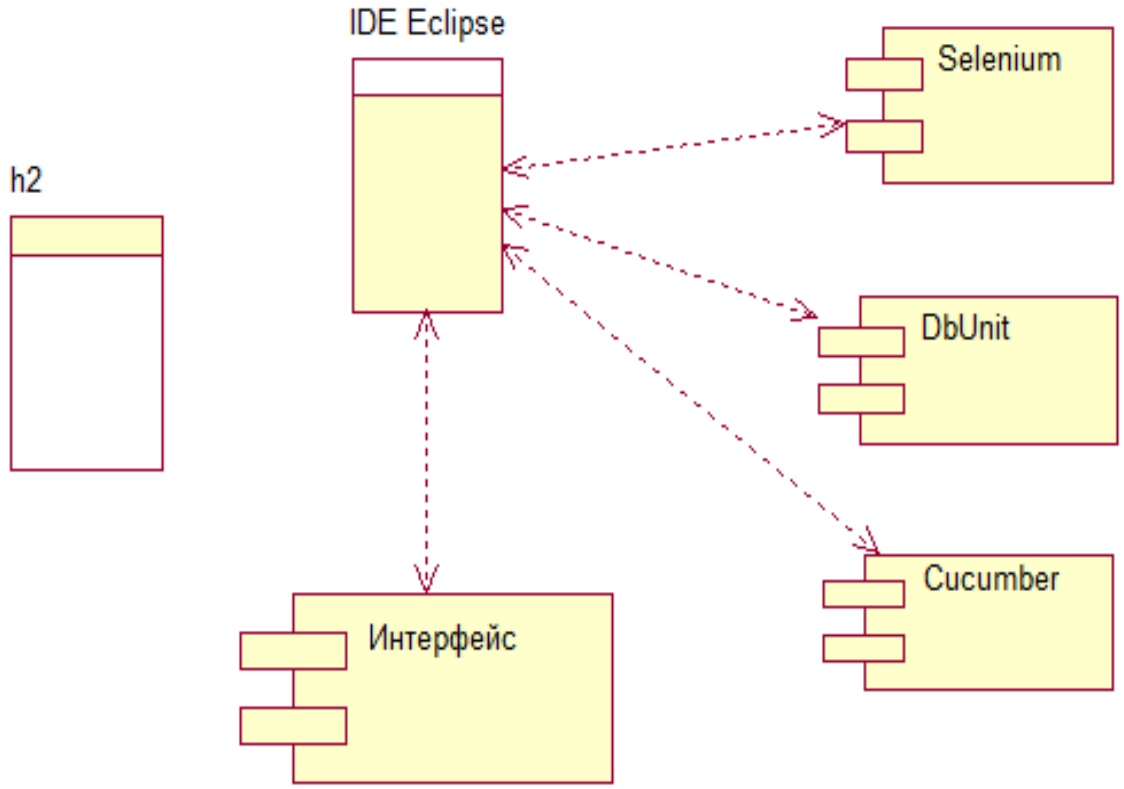

Рисунок 12 – Диаграмма компонентов АСИТ

Рассмотрим пример разработки автотестов с использованием Selenium WebDriver в среде Eclipse IDE.

Перед тем, как начать кодирование любого приложения, необходимо выполнить некоторые настройки и конфигурации в Eclipse IDE, которые будут использованы для разработки.

Это обеспечит правильную конфигурацию Selenium WebDriver и его доступности для разработки тестовых примеров веб-автоматизации [15].

Методика разработки тестовых примеров состоит из следующих шагов:

Шаг 1. Загрузить и запустить Eclipse. После того, как Eclipse будет запущен, необходимо создать рабочее пространство. Рабочее пространство Eclipse - это каталог в системе, в котором будут храниться все плагины Eclipse, конфигурации или другая информация, относящаяся к проекту.

Шаг 2. Создать «Проект» в Eclipse как показано на рисунке 13.

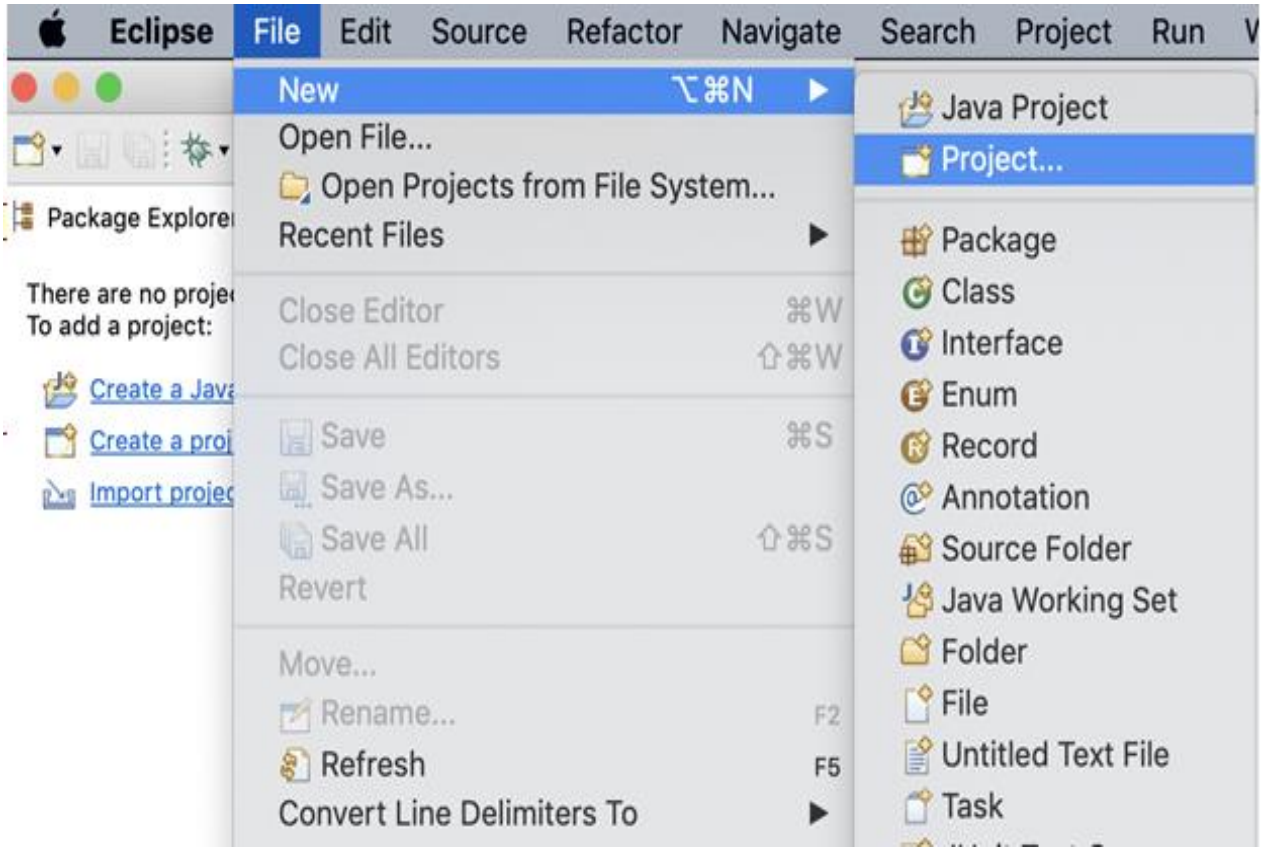

Рисунок 13 – Создание проекта в Eclipse

Шаг 3. Создать новый Java-пакет под проект как показано на рисунке 14.

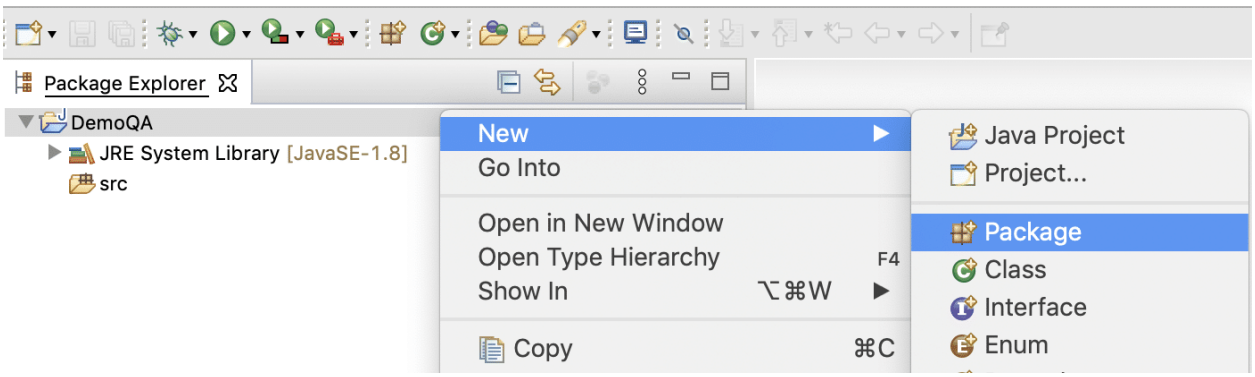

Рисунок 14 – Создание Java-пакета

Шаг 4. Создать новый класс в пакете. Он отобразится в окне «Project Explorer» Eclipse, как показано на рисунке 15.

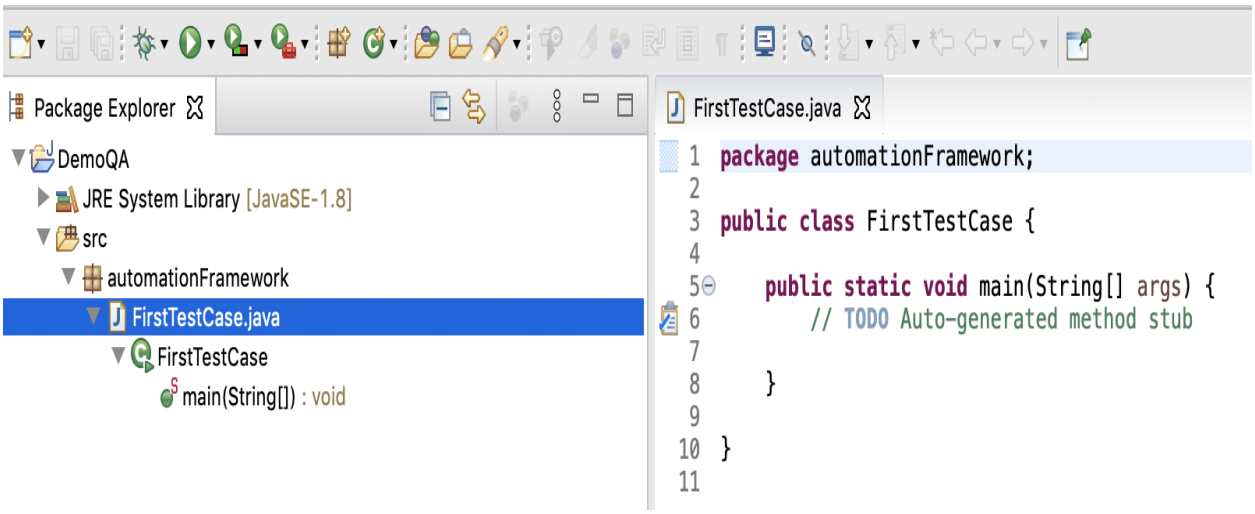

Рисунок 15 – Создание Java-класса

Шаг 5. Включить различные библиотеки, предоставляемых Selenium WebDriver, во вновь созданный проект.

Selenium WebDriver предоставляет набор файлов jar, которые необходимо включить в путь сборки Eclipse (рисунок 16).

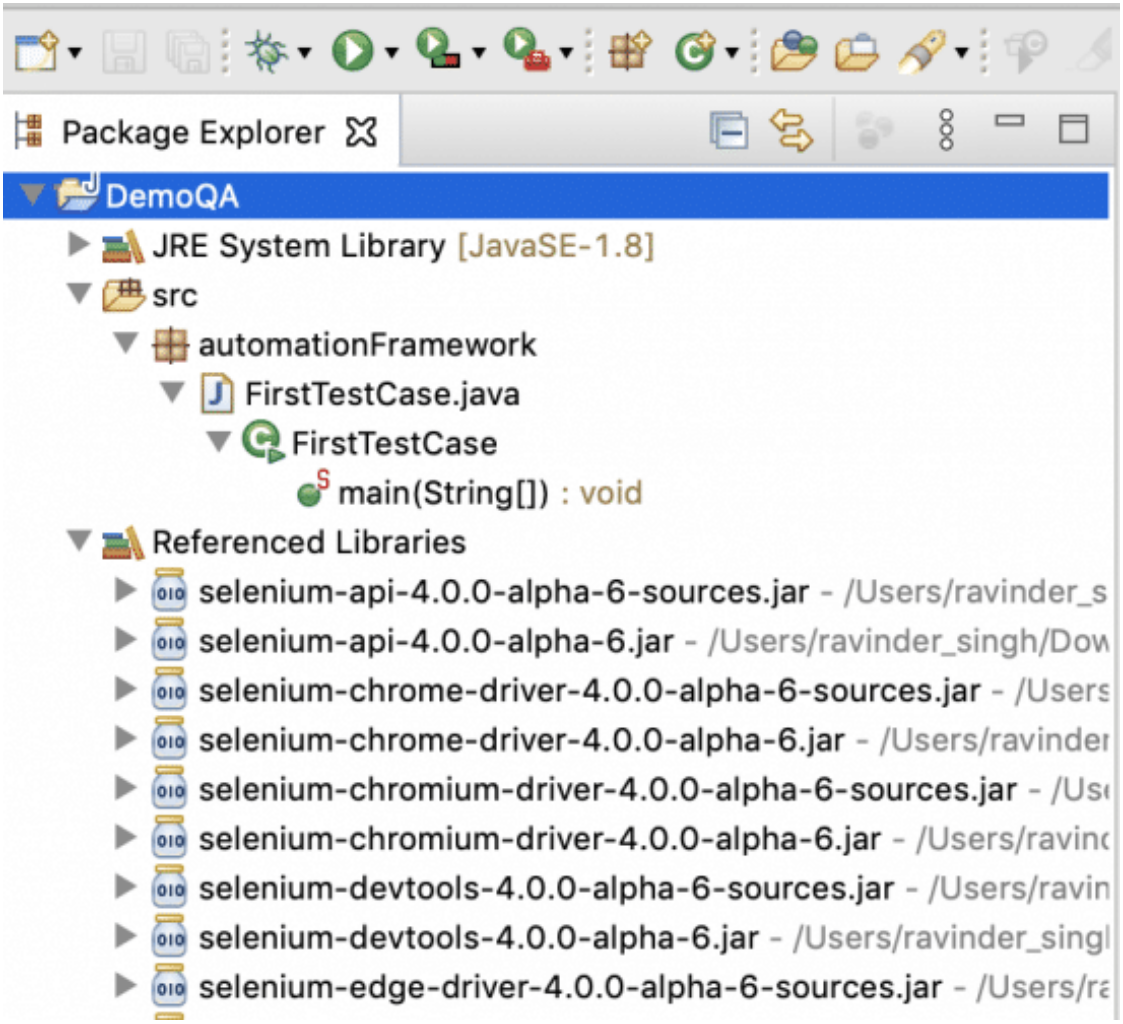

Рисунок 16 – Добавление jar-файлов Selenium WebDriver в пакет

После того, как библиотеки Selenium WebDriver станут доступными в пути сборки, будет получен доступ ко всем функциям Selenium WebDriver в текущем проекте.

3.3 Оценка эффективности проектных решений

Для оценки эффективности проектного решения по внедрению ИС используется показатель эффективности управления ИС.

Эффективность управления – это многоплановое понятие, которое включает следующие компоненты:

– целевую эффективность управления;

− экономическую эффективность управления;

− функциональную эффективность управления.

«Целевая эффективность управления характеризует степень достижения цели действий управляемого объекта при фактическом уровне реализации функций управления.

Под экономической эффективностью управления понимается степень полезной отдачи от выделенных средств на разработку, эксплуатацию системы и осуществление управления.

Под функциональной эффективностью управления понимается степень реализации органом управления возлагаемых на него функций.

Величина показателя функциональной эффективности управления $К_{эу}$ может вычисляться с помощью следующей формулы:

$$К_{эу} = \frac{\sum_{i=1}^{n} Р_{yi}}{n}, \quad (1)$$

где n - количество функций управления, реализуемых ИС;

$Р_{yi}$ - вероятность выработки ИС эффективного управляющего воздействия при реализации *i*-й функции управления.

Для ИС с высокой эффективностью управления величина данного показателя должна превышать значение 0.5» [3].

Оценим эффективность АСИТ на основе показателя функциональной эффективности управления.

В рассматриваемом случае АСИТ предназначена для управления процессом интеграционного тестирования и поддерживает следующие функции управления:

− тестирование веб-приложений;

− тестирование БД;

− тестирование фронт-энда.

В каждой сессии используется только одна из вышеперечисленных функций, на выполнение которой может негативно повлиять человеческий фактор.

Пусть вероятность выработки эффективного управляющего воздействия для данной функции равна 0.5.

В этом случае значение показателя функциональной эффективности управления будет равно:

$$К_{эу} = 2.5/3 = 0.83 \quad (2)$$

Таким образом, коэффициент эффективности управления внедренной АСИТ $К_{эу}$> 0.5, что свидетельствует о высокой функциональной эффективности управления среды.

Выводы к главе 3

Третья глава посвящена реализации проектных решений по автоматизации интеграционного тестирования ООО «НетКрэкер».

Результаты проделанной работы позволили сделать следующие выводы.

Как показал анализ, наилучшими характеристиками обладает IDE Eclipse. Поэтому IDE Eclipse выбрана в качестве среды для разработки АСИТ.

Коэффициент эффективности управления АСИТ Кэу> 0.5, что свидетельствует о высокой функциональной эффективности управления внедренной среды.

Заключение

Выпускная квалификационная работа посвящена актуальной проблеме разработки проекта автоматизации интеграционного тестирования информационных систем.

Для достижения поставленной в работе цели в процессе выполнения бакалаврской работы решены следующие задачи:

- произведен анализ предметной области и выполнена постановку задачи на разработку проекта автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер». В результате анализа выявлен основной недостаток существующего бизнес-процесса интеграционного тестирования в ООО «НетКрэкер» – низкая эффективность ручного тестирования, обусловленная его трудоемкостью, которая связана с необходимостью самостоятельной разработки среды интеграционного тестирования, а также негативным влиянием человеческого фактора. Улучшение существующего бизнес-процесса достигается за счет внедрения в него АСИТ. Для разработки требований к проекту использована методология FURPS+. Разработанный перечень требований использован в качестве основы для реализации проектного решения автоматизации интеграционного тестирования ИС в ООО «НетКрэкер»;
- спроектирована АСИТ для ООО «НетКрэкер». Для автоматизации интеграционного тестирования данных задач выбраны следующие фреймворки: Selenium, h2+DbUnit и Cucumber. Выбор фреймворков основан на личном опыте проектанта и с учетом успешного опыта их применения другими специалистами-тестировщиками. Для построения логической модели АСИТ разработаны базовые диаграммы языка UML, отражающие различные аспекты системы: диаграмма вариантов использования, диаграмма классов и

диаграмма последовательности. Для разработки логической модели АСИТ использовано CASE-средство Rational Rose, которое поддерживает методологию RUP;

- выполнена реализация проектного решения автоматизации интеграционного тестирования информационных систем в ООО «НетКрэкер». Для представления программной архитектуры АСИТ разработана диаграмма ее компонентов. Для реализации АСИТ использована технология IDE. Как показал анализ, наилучшими характеристиками обладает IDE Eclipse. Поэтому IDE Eclipse выбрана в качестве среды для разработки АСИТ. Для оценки эффективности проекта использована методика оценки эффективности управления внедренной ИС. Коэффициент эффективности управления АСИТ Кэу> 0.5, что свидетельствует о высокой функциональной эффективности управления внедренной среды.

Результаты бакалаврской работы представляют практический интерес и могут быть рекомендованы для бизнес-аналитиков и тестировщиков, работающими над проектами автоматизации интеграционного тестирования ИТ-компаний.

Список используемой литературы

1. 12 инструментов для интеграционных и unit-тестов в Java [Электронный ресурс]. URL: https://tproger.ru/translations/12-tools-for-unit-tests/ (дата обращения: 08.09.2021).

2. Брекелов В.В., Борисов Е.А., Барыгин И.А. Автоматизация интеграционного тестирования на примере модулей обмена данными по FIX-протоколу // Информатика, телекоммуникации и управление. 2015. №1 (212). С. 88-96..

3. Вдовин В. М., Суркова Л. Е., Шурупов А.А. Предметно-ориентированные экономические информационные системы [Электронный ресурс]: учебное пособие. М. : Дашков и К, 2016. 386 c. URL: https://www.iprbookshop.ru/60492.html (дата обращения: 06.09.2021).

4. ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы [Электронный ресурс]. URL: https://gostexpert.ru/gost/gost-34.602-89 (дата обращения: 15.08.2021).

5. Интегрированные среды разработки программ [Электронный ресурс]. URL: http://bourabai.ru/einf/ide.htm (дата обращения: 06.09.2021).

6. Использование нотации eEPC для графического описания бизнес-процессов [Электронный ресурс]. URL: https://habr.com/ru/post/143273/ (дата обращения: 15.08.2021).

7. Коцюба И.Ю., Чунаев А.В., Шиков А.Н. Методы оценки и измерения характеристик информационных систем [Электронный ресурс] : учебное пособие. Санкт-Петербург : Университет ИТМО, 2016. 264 c. URL: https://www.iprbookshop.ru/67289.html (дата обращения: 15.08.2021).

8. Краткое руководство. Знакомство с интегрированной средой разработки Visual Studio [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/visualstudio/ide/quickstart-ide-orientation?view=vs-2019 (дата обращения: 06.09.2021).

9. Леоненков А. В. Объектно-ориентированный анализ и

проектирование с использованием UML и IBM Rational Rose [Электронный ресурс] : учебное пособие. М. : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. 317 c. URL: https://www.iprbookshop.ru/97554.html (дата обращения: 06.09.2021).

10. Сайт ООО НетКрэкер [Электронный ресурс]. URL: uhttps://www.netcracker.com (дата обращения: 15.08.2021).

11. Сорокин А. А., Орлова А.Ю. Реинжиниринг бизнес-процессов [Электронный ресурс] : учебное пособие. Ставрополь : Северо-Кавказский федеральный университет, 2014. 212 c. URL: https://www.iprbookshop.ru/63003.html (дата обращения: 15.08.2021).

12. Фреймворк как программная платформа [Электронный ресурс]. URL: https://intellect.icu/frejmvork-kak-programmnaya-platforma-klassifikatsiya-i-vidy-frejmvorkov-framework-9515 (дата обращения: 08.09.2021).

13. Apache NetBeans [Электронный ресурс]. URL: https://netbeans.apache.org/ (дата обращения: 06.09.2021).

14. ARIS Express [Электронный ресурс]. URL: https://www.ariscommunity.com/aris-express (дата обращения: 15.08.2021).

15. Configure Selenium WebDriver with Eclipse [Электронный ресурс]. URL: https://www.toolsqa.com/selenium-webdriver/configure-selenium-webdriver-with-eclipse/ (дата обращения: 06.09.2021).

16. Cucumber. Tools & techniques that elevate teams to greatness [Электронный ресурс]. URL: https://cucumber.io/ (дата обращения: 08.09.2021).

17. Eclipse IDE [Электронный ресурс]. URL: https://www.eclipse.org/eclipseide/ (дата обращения: 06.09.2021).

18. H2 Database Engine [Электронный ресурс]. URL: https://www.h2database.com/html/main.html (дата обращения: 08.09.2021).

19. Integration Testing Tools and Practices to Start Using Today [Электронный ресурс]. URL: https://u-tor.com/topic/start-integration-testing-today (дата обращения: 08.09.2021).

20. Selenium. Интеграционное тестирование веб-приложений [Электронный ресурс]. URL: https://korzh.net/2010-12-seleniumin-tegracionnoe-testirovanie-veb-prilozhenij.html (дата обращения: 08.09.2021).

21. Top 15 Automation Testing Interview Questions & Answers [Электронный ресурс]. URL: https://www.guru99.com/automation-testing-interview-questions.html (дата обращения: 15.08.2021).

22. Types of Testing Environments [Электронный ресурс]. URL: https://www.testenvironmentmanagement.com/types-of-testing-environments/ (дата обращения: 08.09.2021).

23. What Is Automation Testing Pyramid? [Электронный ресурс]. URL: https://qatestlab.com/resources/knowledge-center/test-automated-pyramid/ (дата обращения: 15.08.2021).

24. What is the use of Furps+ model in classifying requirements? [Электронный ресурс]. URL: https://findanyanswer.com/what-is-the-use-of-furps-model-in-classifying-requirements (дата обращения: 15.08.2021).

25. Zihao L. Automated Integration Testing [Электронный ресурс]. URL: https://medium.com/@allenliuzihao/automated-integration-testing-a295d21e513a (дата обращения: 15.08.2021).