

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов «__»
_____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Экимов Дмитрий Витальевич гр. 19ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>
2. Тема работы Программные средства автоматизации контроля оформления текстовых документов
- Тема утверждена приказом ПГУ № _____ от "___" _____ 2023 г.
3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
- 1) *Назначение: программные средства автоматизации контроля оформления текстовых документов на примере нормоконтроля выпускных квалификационных работ.*
- 2) *Среда выполнения: операционная система Windows 10 и выше*
Языковые средства разработки: C#, UML
Инструментальные средства разработки: Microsoft Visual Studio 2019, Draw.io, Microsoft Project, Microsoft Office 2013 и выше
Технологии разработки: ООП, Windows Forms, VSTO
- 3) *Основные функции программного средства:*
- а) *проверка параметров страниц;*
- б) *проверка параметров абзацев основного текста;*
- в) *проверка параметров абзацев для заголовков;*
- г) *проверка параметров абзацев для рисунков и подписуточных надписей;*
- д) *проверка параметров абзацев названий таблиц;*
- е) *проверка параметров нумераций страниц;*
- ж) *проверка тире и неразрывных пробелов;*
- и) *проверка кавычек в тексте;*
- й) *проверка параметров нумераций страниц;*

5. Перечень подлежащих разработке вопросов:

1) анализ предметной области и постановка задачи;
2) проектирование;
3) реализация;
4) качества программного обеспечения;
5) план разработки бюджета;
6) оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	14.04.2023 – 27.04.2023	
2	Проектирование	25%	27.04.2023 – 10.05.2023	
3	Реализация	35%	10.05.2023 – 30.06.2023	
4	Контроль качества программного обеспечения	10%	31.05.2023 – 02.06.2023	
5	План разработки бюджета	5%	02.06.2023 – 03.06.2023	
6	Оформление пояснительной записки	15%	03.06.2023 – 14.06.2023	

Дата выдачи задания «22» декабря 2022 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 80 листов, 31 рисунок, 11 таблиц, 11 использованных источников и 2 приложения.

ПРОГРАММНОЕ СРЕДСТВО, АВТОМАТИЗАЦИЯ, ТЕКСТОВЫЙ ДОКУМЕНТ, НОРМОКОНТРОЛЬ, VSTO, UML, C#.

Объектом разработки являются средства автоматизации контроля оформления текстовых документов.

Цель работы – создание программных средств автоматизации контроля оформления текстовых документов для осуществления нормоконтроля выпускных квалификационных работ.

Разработка программы происходила на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2019. Дополнительно использован набор инструментов и расширений Visual Studio Tools for Office для разработки приложений Microsoft Office с использованием Visual Studio.

Программное средство спроектировано для работы на устройствах под управлением операционной системы Windows 10 и выше, а также использования в программных продуктах Microsoft Office Word 2013 и выше.

					ПГУ 09.03.04– 8ВР192.20ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программные средства автоматизации контроля оформления текстовых документов Пояснительная записка	Лит.	Лист	Листов
Разраб.		Экимов Д. В.					4	80
Пров.		Балашова И.Ю.				Группа 19ВП2		
Н. Контр.		Такташкин Д.В.						
Утв.		Козлов А.Ю.						

Содержание

Введение.....	6
1 Анализ средств автоматизации контроля оформления текстовых документов.....	8
1.1 Анализ предметной области	8
1.2 Анализ программного обеспечения аналогичного назначения	10
1.3 Анализ функциональных требований.....	13
1.4 Анализ нефункциональных требований.....	17
1.5 Планирование разработки и оценка бюджета средства автоматизации контроля оформления текстовых документов	18
2 Проектирование средства автоматизации контроля оформления текстовых документов	20
2.1 Выбор технологий и инструментальных средств разработки.....	20
2.2 Проектирование программного средства	22
2.3 Проектирование интерфейса пользователя.....	23
3 Реализация средства автоматизации контроля оформления текстовых документов.....	26
4 Тестирование программного продукта.....	35
4.1 Тестирование приложения методом черного ящика.....	35
4.2 Функциональное тестирование.....	37
4.3 Метрики кода.....	38
4.3.1 Метрики размера программы.....	40
4.3.2 Метрики Холстеда	41
Заключение	43
Список использованных источников	44
Приложение А. Исходный код программы	46
Приложение Б. Результаты функционального тестирования.....	74

Введение

Оформление текстовых документов, соответствующее установленным стандартам, является важным критерием во многих сферах деятельности. Примерами являются стандарты подготовки технической документации, соответствия формам документов в различных организациях, положения по оформлению учебных работ, диссертаций, отчетов и т. п.

Студенты по окончании обучения должны освоить навыки подготовки отчетности, необходимые не только в учебном заведении, но и при дальнейшей работе, ведении совместных проектов с коллегами, формировании документации для организации. Поэтому в процессе обучения, учащиеся изучают инструменты форматирования и демонстрируют знания на практике. Однако из-за большого объема пунктов в разделе правил оформления студенты часто допускают ошибки по невнимательности. Обучающиеся считают это маловажным фактом, хотя эта точка зрения совершенно неверна [1].

Наиболее распространена проверка отчетности вручную, что также приводит к риску возникновения ошибок. Студенты последних курсов обучения выполняют выпускную квалификационную работу, которая подлежит наиболее тщательной проверке, и эта задача распределяется между студентом и консультантом по нормоконтролю. Но при большом количестве работ на одного консультанта имеется риск пропустить мелкие ошибки, что приводит к неправильному оформлению и несоответствию стандартам ГОСТ. Актуальным решением на текущий момент для решения проблемы проверки соответствия текста требованиям форматирования является автоматизация. Имея средство автоматизации контроля оформления текстовых документов, учебные заведения помогут студентам иметь более ясное представление о заданном эталоне оформления в соответствии с установленными положениями и стандартами. Тем самым вероятность допуска выпускной квалификационной работы (ВКР) с ошибками к защите может быть значительно снижена, а точность проверок консультанта максимально эффективной.

Объект исследования – средства автоматизации проверки документов на соответствие стандартам.

Целью данной выпускной квалификационной работы является разработка программных средств автоматизации контроля оформления текстовых документов, применяемых нормоконтролером высшего учебного заведения в процессе проверки оформления выпускных квалификационных работ студентов. Разработка включает в себя анализ требований, проектирование, реализацию и тестирование программного обеспечения.

1 Анализ средств автоматизации контроля оформления текстовых документов

1.1 Анализ предметной области

На сегодняшний день с развитием технологий количество электронных документов значительно возросло. Во многих организациях существуют свои шаблоны и требования к оформлению, вследствие чего существует необходимость проверки соответствия этим требованиям.

В Российской Федерации существует значительное количество ГОСТов, связанных с различными аспектами текстовых файлов, включая их оформление и обработку. Они устанавливают правила и рекомендации относительно структуры, формата, типографики, таблиц, списков и других аспектов оформления документов. Разные организации могут иметь свои специфические требования, которые нужно учесть при разработке средства автоматизации [2].

При проверке документов на соответствие ГОСТам присутствуют следующие шаги и этапы:

- идентификация применимых ГОСТов;
- получение актуальных версий ГОСТов;
- анализ требований ГОСТов;
- сопоставление требований и фактического содержания документов;
- выявление расхождений и недостатков;
- формирование отчета о проверке;
- корректировка и обновление документов.

Процесс проверки документов является долгой и монотонной работой, что совместно с человеческим фактором приведет к ошибкам. Поэтому важно автоматизировать данный процесс с целью сэкономить время и усилия, которые могут быть затрачены на ручную проверку и исправление текстовых документов. Подобные средства автоматизируют процесс контроля и исправления, что способствует повышению производительности и ускоряет процесс создания и редактирования документов.

Средства по контролю оформления текстовых документов помогают в обеспечении соответствия текстовых документов установленным стандартам оформления, таким как правила оформления научных статей, стандарты цитирования и т. д. Это помогает сохранить единообразие и профессиональный вид документов, а также облегчает их восприятие и понимание.

Средство автоматического контроля оформления текстовых документов помогает обеспечить качество, точность и согласованность документов, повышает профессионализм текстов и улучшает эффективность работы с документами. Также помогает в правильном оформлении цитат, сносок и библиографических ссылок, предлагает стандартные форматы и указывает на несоответствия в оформлении, что облегчает составление более точных и академически корректных ссылок [3].

Разрабатываемое программное средство предназначено для проверки ВКР, курсовых работ, отчетов по практикам. В Пензенском государственном университете реализован стандарт университета, в котором прописаны требования регламентирующие данные работы [4]. Исходя из этого стандарта, были определены следующие основные функции нормоконтролера:

- проверка документа на соответствие ГОСТ 7.32–2001 «Отчет о научно-исследовательской работе. Структура и правила оформления»;
- проверка документа на соответствие ГОСТ 2.105–95 «Общие требования к текстовым документам»;
- проверка документа на соответствие ГОСТ Р 7.0.5–2008 «Библиографическая ссылка. Общие требования и правила составления»;
- проверка документа на соответствие ГОСТ 7.1–2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления»;
- проверка документа на соответствие ГОСТ 7.80–2000. «Библиографическая запись. Заголовок. Общие требования и правила составления».

Данные функции в полной мере должны быть реализованы в работе.

1.2 Анализ программного обеспечения аналогичного назначения

Для проведения сравнительного анализа выбраны приложения, являющиеся наиболее приближенными аналогами разрабатываемого средства. В таблице 1 приведено описание подобных программ, разработанных в данной области другими высшими учебными заведениями.

Таблица 1 – Разработки в сфере нормоконтроля оформления документов

№	Автор / Университет	Стек технологий	Анализируемые форматы	Охватываемый спектр задач
1	Государственный университет им. Витуса Беринга Камчатский	Delphi, SQL	docx, doc	Автоматизация формирования титульных листов. Проверка форматирования содержания. Формирование приложения к экзаменационной ведомости.
2	Белорусский государственный технологический университет	ASP.Net, Open XML SDK	docx	Проверка форматирования содержания.
3	НИУ «Высшая школа экономики»	Open XML SDK, C#, ASP.Net	docx	Проверка форматирования содержания.
4	Уральский государственный педагогический университет	C#	docx	Проверка форматирования содержания. Обучение оформлению документации.
5	Вятский государственный университет	HTML, VBA	doc, docx, rtf	Проверка форматирования содержания.
6	Университет ИТМО	C#, .Net, Python	docx	Проверка форматирования содержания. Обучение оформлению документации.

Из приведенных выше данных можно сделать вывод, что в разработке авторы выбирали одни из наиболее распространенных форматов текстовых документов – doc и docx. Также стоит отметить схожий функционал программных решений. Однако стек технологий программных продуктов различается. Для проведения сравнительного анализа, в качестве критериев,

выбраны наиболее важные возможности, которые напрямую влияют на функциональность сервиса. Критерии были разбиты по критериям и приведены в таблице 2.

Таблица 2 – Критерии оценки средства автоматизации контроля оформления текстовых документов

№	Наименование
	Функциональные возможности
1	Проверка параметров абзацев
2	Проверка параметров страниц
3	Проверка параметров интервалов
4	Проверка параметров отступов
5	Проверка параметров выравнивания
6	Проверка параметров абзаца рисунков и подрисуночных надписей
7	Проверка параметров абзаца названий таблиц
8	Проверка параметров нумерации страниц
9	Проверка параметров списков
	Необязательные функциональные возможности
10	Формирование приложения к экзаменационной ведомости
11	Обучение оформлению документации
12	Поддержка нескольких форматов текстовых документов
	Технические возможности
13	Запуск и отмена нормоконтроля
14	Удаление логов о проверке
	Эргономика
15	Интуитивно-понятный интерфейс
16	Удобство восприятия результата
17	Описание ошибки

Для оценки описанных выше критериев используется трехбалльная шкала соответствия, от 0 – функция не реализована, 1 – реализована частично, 2 – полностью реализована. В таблице 3 представлены характеристики оценки.

Таблица 3 – Шкала баллов

Балл	Характеристика оценки
0	функция не реализована
1	реализована частично
2	полностью реализована

Для анализа были взяты сервисы из таблицы 1. Заполним таблицу 4, оценив реализованный функционал описанных выше систем. Затем произведем расчет аддитивной суммы интегральной оценки для каждого сравниваемого продукта по следующей формуле:

$$O_{\text{пп}} = \sum_{i=1}^n (Z_i),$$

где $O_{\text{пп}}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – значение выполнения критерия (0 – функция не реализована, 1 – реализована частично, 2 – полностью реализована).

Таблица 4 – Критериальная оценка программных продуктов

№	Критерий	Программный продукт №					
		1	2	3	4	5	6
1	Проверка параметров абзацев	1	2	2	2	2	2
2	Проверка параметров страниц	1	1	1	1	2	2
3	Проверка параметров интервалов	1	2	2	2	2	2
4	Проверка параметров отступов	2	2	2	1	2	2
5	Проверка параметров выравнивания	1	2	2	2	2	2
6	Проверка параметров абзаца рисунков и подрисуточных надписей	1	1	1	2	2	2
7	Проверка параметров абзаца названий таблиц	1	2	2	2	1	2
8	Проверка параметров нумерации страниц	1	2	1	1	2	2
9	Проверка параметров списков	2	1	1	1	2	1
10	Формирование приложения к экзаменационной ведомости	1	0	0	2	0	2
11	Обучение оформлению документации	1	0	0	2	0	2
12	Поддержка нескольких форматов текстовых документов	1	0	0	0	2	0
13	Запуск и отмена нормоконтроля	1	1	2	1	2	2
14	Удаление результатов проверки	1	1	1	2	2	2
15	Интуитивно-понятный интерфейс	1	1	2	1	2	1
16	Удобство восприятия результата	2	2	2	2	2	2
17	Описание ошибки	1	1	2	1	2	2
	Интегральная оценка	20	21	23	25	29	30

Подводя итог сравнения, можно сделать вывод, что среди аналогичных программных продуктов большая часть критериев были полностью или частично реализованы. Лидером является программное средство от Университета ИТМО. Используя результаты сравнительного анализа, были определены наиболее важные функции и критерии, которые должны быть в приложении.

В таблице 5 приведено описание основного порядка выполнения ролями «Пользователь» и «Программное средство» варианта использования «Запустить проверку на нормоконтроль».

Таблица 5 – Спецификация прецедента «Запустить проверку на нормоконтроль»

Прецедент: запустить проверку на нормоконтроль
ID: 1
Краткое описание: Обработка текстового документа на соответствие нормоконтролю
Главные актеры: Пользователь Программное средство
Основной поток: 1. Пользователь запускает проверку 1.1. Пользователь открывает текстовый документ 1.2. Пользователь переходит во вкладку «Нормоконтроль» 1.3. Пользователь нажимает кнопку «Проверить» 2. Программное средство начинает обработку текстового документа в соответствии с установленными параметрами

В таблице 6 приведено описание основного порядка выполнения ролями «Пользователь» и «Программное средство» варианта использования «Запустить проверку на нормоконтроль».

Таблица 6 – Спецификация прецедента «Прервать проверку на нормоконтроль»

Прецедент: прервать проверку на нормоконтроль
ID: 2
Краткое описание: Прерывание обработка текстового документа на соответствие нормоконтролю
Главные актеры: Пользователь Программное средство
Основной поток: 1. Программное средство обрабатывает текстовый документ в соответствии с установленными параметрами 2. Пользователь в открывшемся диалоговом окне нажимает кнопку «Прервать» 3. Программное средство прерывает обработку текстового документа

В таблице 7 приведено описание основного порядка выполнения ролями «Пользователь» и «Программное средство» варианта использования «Запустить проверку на нормоконтроль».

Таблица 7 – Спецификация прецедента «Очистить результаты нормоконтроля»

Прецедент: очистить результаты нормоконтроля
ID: 3
Краткое описание: Очистка текстового документа от заметок, содержащих информацию о найденных ошибках
Главные актеры: Пользователь Программное средство
Основной поток: 1. Пользователь переходит во вкладку «Нормоконтроль» 2. Пользователь нажимает кнопку «Очистить» 3. Программное средство удаляет созданные заметки из текстового документа

Для раскрытия прецедента «Запустить проверку на нормоконтроль» основываясь на рисунке 1 была составлена диаграмма деятельности (рисунок 2), чтобы структурировать процесс и понять логику его выполнения.

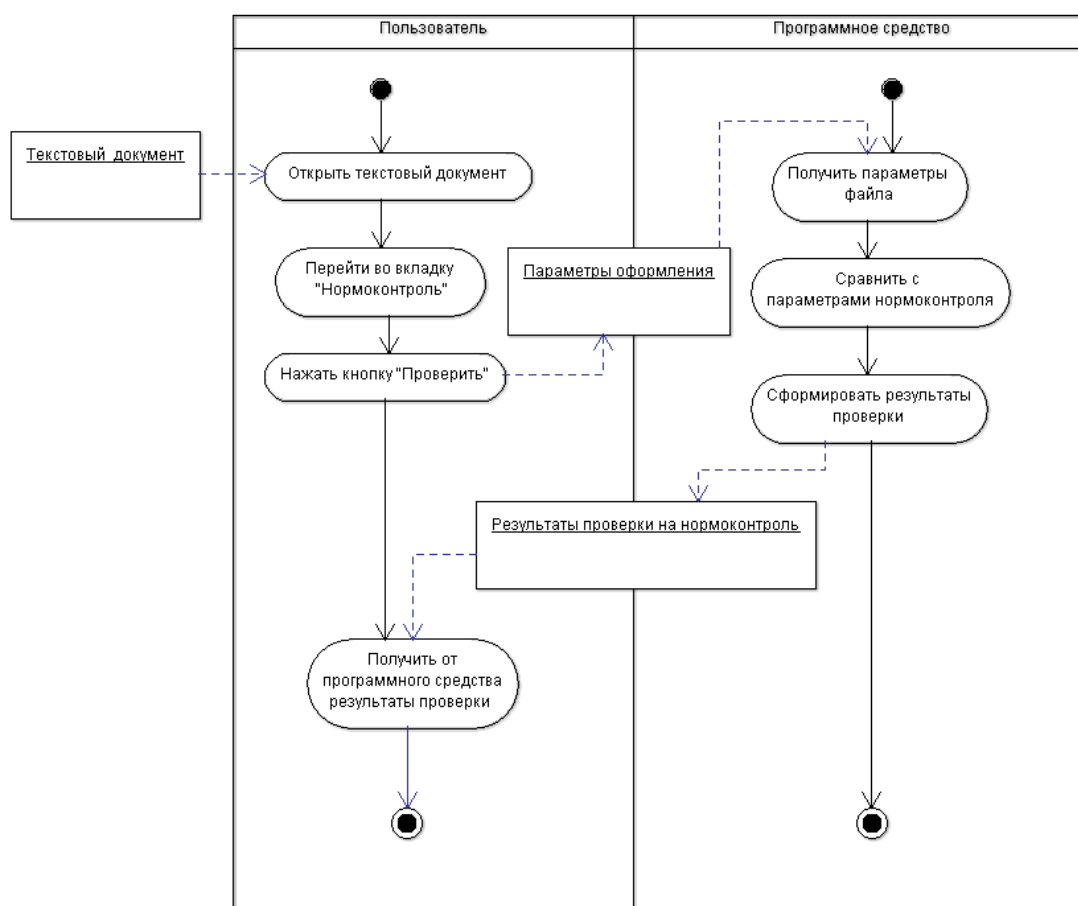


Рисунок 2 – Диаграмма деятельности для прецедента «Запустить проверку на нормоконтроль»

Для описания процессов моделирования средства автоматизации контроля оформления текстовых документов была использована методология функционального моделирования IDEF0. Разработан функциональный блок (рисунок 3), который будет представлять средство как единое целое.

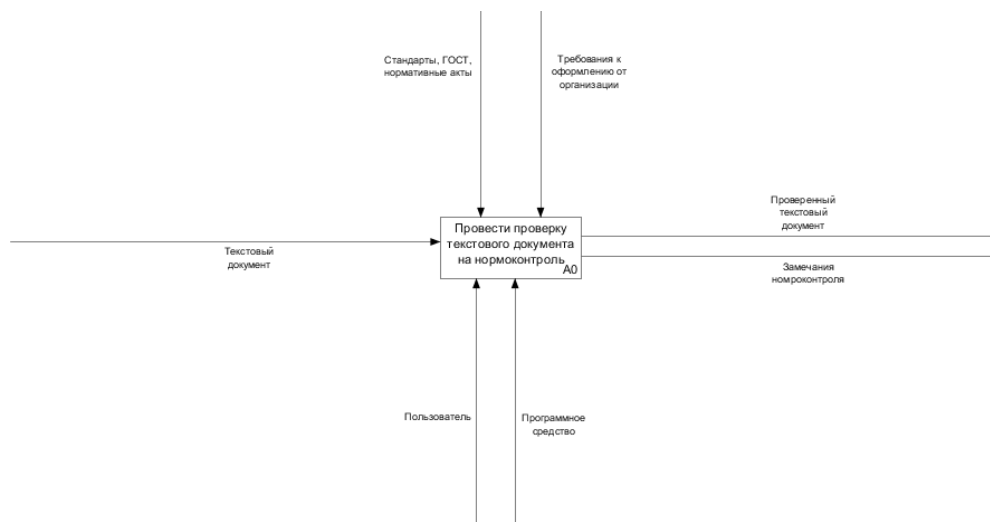


Рисунок 3 – Функциональный блок «Провести проверку текстового документа на нормоконтроль»

Проведена декомпозиция функционального блока (рисунок 4), которая детализирует работу средства автоматизации контроля оформления текстовых документов, показывая основные действия и механизмы.

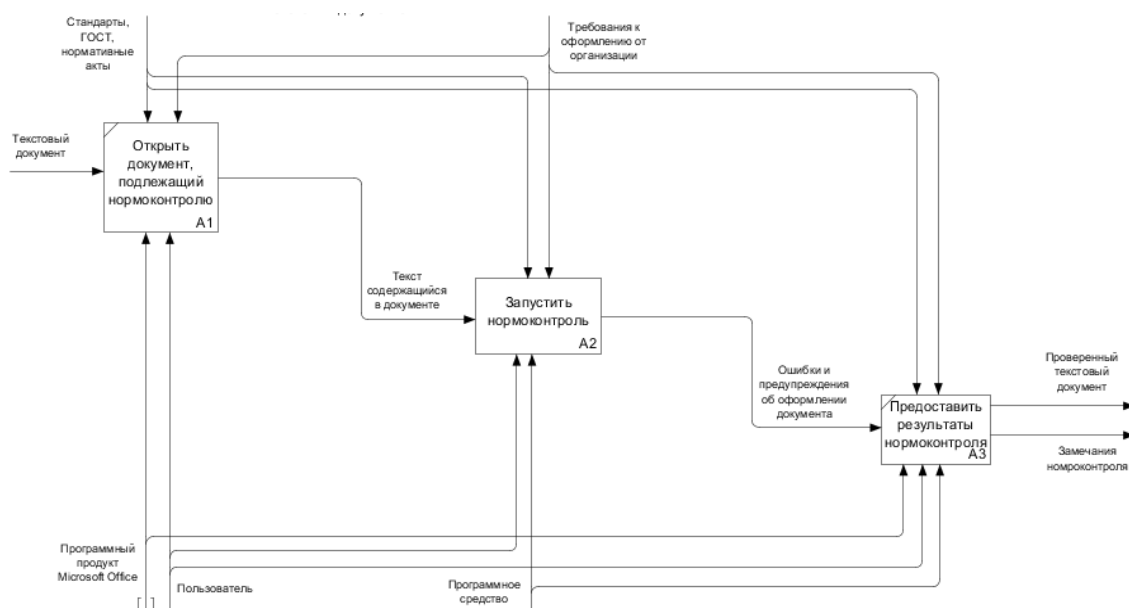


Рисунок 4 – Декомпозиция функционального блока «Провести проверку текстового документа на нормоконтроль»

На рисунке 5 отображена декомпозиция основного функционального блока средства автоматизирования.

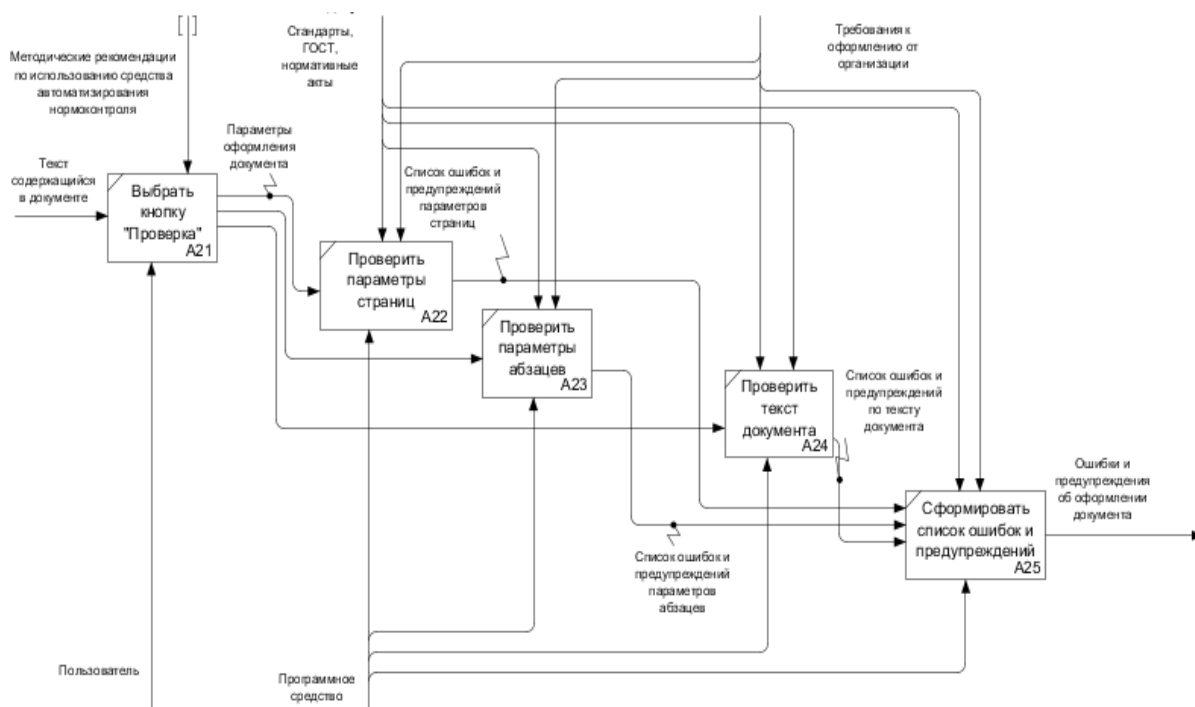


Рисунок 5 – Декомпозиция функционального блока «Запустить нормоконтроль»

Данная функциональная диаграмма используется при проектировании и разработки приложения.

1.4 Анализ нефункциональных требований

В ходе анализа технологий аналогичных средств и их функционала было определено что для решения задачи будет использован язык C# с использованием Windows Forms .NET. Использование C# и Windows Forms .NET в разработке обеспечивает простоту, скорость, гибкость и надежность, что позволяет создавать качественные и функциональные приложения для операционной системы Windows.

Программное средство автоматизации контроля оформления текстовых документов должно поддерживаться версиями Microsoft Office с 2013 по 2021. При разработке необходимо будет использовать редактор исходного кода Visual Studio 2019.

Необходимо использовать набор средств разработки Visual Studio Tools for Office, упрощающий создание пользовательских областей форм и настройку ленты. Можно создавать области задач уровней документа и приложения и без проблем создавать и выполнять отладку решений рабочего процесса [5].

1.5 Планирование разработки и оценка бюджета средства автоматизации контроля оформления текстовых документов

Планирование разработки и оценка бюджета была выполнена при помощи Microsoft Project [6]. Дата начала проекта – 03 апреля 2023 года. В результате планирования было выделено 11 задач. Задачи проекта, а также их длительность и данные о предшествующих задачах представлены на рисунке 6. Также для визуализации плана проекта применяется диаграмма Ганта. Диаграмма Ганта – это визуальное представление графика работ, построенное согласно плану проекта. Она является мощным инструментом для планирования, управления и отслеживания проектов. Это помогает улучшить организацию работы, оптимизировать ресурсы, сократить задержки в разработке. График работ состоит из ряда отрезков, размещенных вдоль временной оси.

	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало проекта	0 дней	Пн 03.04.23	Пн 03.04.23		
2	Анализ требований	9 дней	Пн 03.04.23	Чт 13.04.23		
3	Изучение предметной области	5 дней	Пн 03.04.23	Пт 07.04.23	1	Компьютер[1]
4	Анализ аналогичных приложений	4 дней	Пн 10.04.23	Чт 13.04.23	3	Компьютер[1]
5	Анализ требований завершен	0 дней	Чт 13.04.23	Чт 13.04.23	4	
6	Выбор средств	5 дней	Пн 10.04.23	Пт 14.04.23		
7	Выбор программных средств	3 дней	Пн 10.04.23	Ср 12.04.23	3	Программист ;Компьютер[1]
8	Выбор аппаратных средств	2 дней	Чт 13.04.23	Пт 14.04.23	7	Программист ;Компьютер[1]
9	Завершение выбора средств	0 дней	Пт 14.04.23	Пт 14.04.23	8	
10	Проектирование приложения	10 дней	Пн 17.04.23	Пт 28.04.23		
11	Проектирование пользовательского интерфейса	5 дней	Пн 17.04.23	Пт 21.04.23	9	Программист ;Компьютер[1]
12	Проектирование реализуемых функций	5 дней	Пн 24.04.23	Пт 28.04.23	11	Программист ;Компьютер[1]
13	Завершение проектирование приложения	0 дней	Пт 28.04.23	Пт 28.04.23	12	
14	Реализация приложения	20,05 дней	Пн 01.05.23	Пн 29.05.23		
15	Разработка пользовательского интерфейса	9 дней	Пн 01.05.23	Чт 11.05.23	13	Программист ;Компьютер[1];[1]
16	Разработка реализуемых функций	11,05 дней	Пт 12.05.23	Пн 29.05.23	15	Программист [56%];Компьютер[1]
17	Завершение реализации приложения	0 дней	Пн 29.05.23	Пн 29.05.23	16	
18	Тестирование	4 дней	Пн 29.05.23	Пт 02.06.23		
19	Модульное тестирование	2 дней	Пн 29.05.23	Ср 31.05.23	17	Программист ;Тестировщик;Компьютер[1]
20	Функциональное тестирование	2 дней	Пн 29.05.23	Ср 31.05.23	17	Тестировщик[75%];[1]
21	Нагрузочное тестирование	2 дней	Ср 31.05.23	Пт 02.06.23	19,20	Тестировщик[67%];[1]
22	Тестирование завершено	0 дней	Пт 02.06.23	Пт 02.06.23	21	
23	Конец проекта	0 дней	Пт 02.06.23	Пт 02.06.23	22	

Рисунок 6 – Список задач проекта

Список ресурсов, используемых при выполнении задач представлен на рисунке 7.

	Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. едини	Стандартная ставка	Ставка сверхурочн	Затраты на использ.	Начисление	Базовый календарь
1	Программист	Трудовой		П		200%	р.50 000,00/мес	р.300,00/ч	р.0,00	Пропорциональное	Стандартный
2	Тестировщик	Трудовой		Т		200%	р.35 000,00/мес	р.200,00/ч	р.0,00	Пропорциональное	Стандартный
3	Компьютер	Материальный		К			р.0,00		р.0,00	Пропорциональное	

Рисунок 7 – Список ресурсов

Диаграмма Ганта, которая получена в результате проектирования, представлена на рисунке 8.

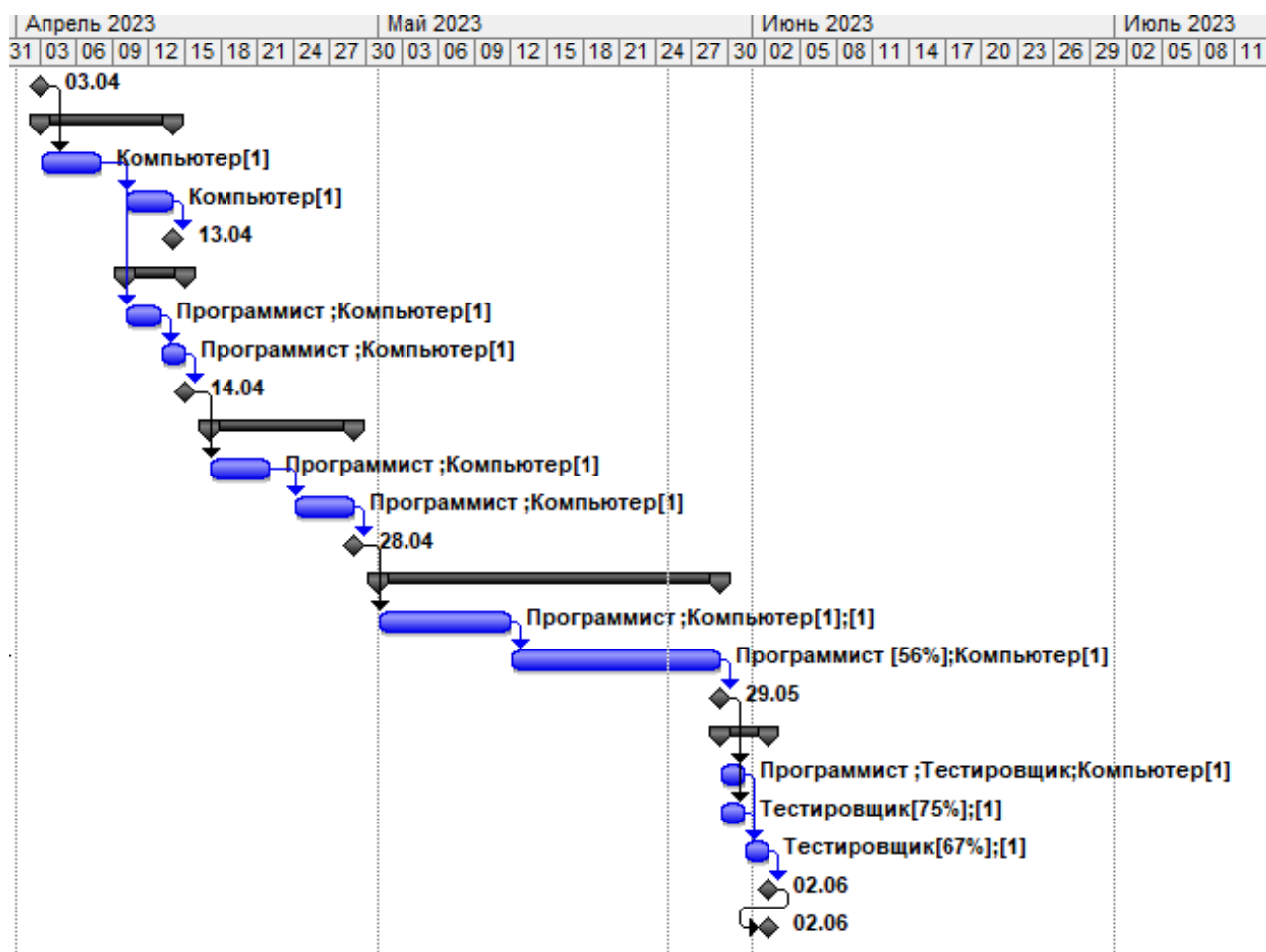


Рисунок 8 – Диаграмма Ганта

В результате оценки бюджета примерная стоимость разработки составила 108193 рубля. Для разработки потребуется 376 человеко-часов. В условиях, указанных при планировании, разработка займет 45 дней.

2 Проектирование средства автоматизации контроля оформления текстовых документов

2.1 Выбор технологий и инструментальных средств разработки

В процессе анализа предметной области было решено использовать Rational Unified Process (RUP) методологию. RUP – это гибкая методология разработки программного обеспечения, разработанная компанией Rational Software. RUP представляет собой итеративную и инкрементальную методологию, ориентированную на объектно-ориентированную разработку. Применение RUP позволяет командам разработчиков эффективно управлять проектами, управлять рисками и обеспечивать качество продукта [7].

Определяет следующие четыре основные фазы разработки:

- а) начальная фаза (inception phase):
 - 1) определение целей проекта и его ограничений;
 - 2) анализ бизнес-требований и определение основных функциональных и нефункциональных требований;
 - 3) определение архитектуры системы;
 - 4) оценка рисков и планирование проекта;
 - 5) создание предварительного плана разработки и оценка затрат;
- б) фаза разработки (elaboration phase):
 - 1) уточнение требований и архитектуры системы;
 - 2) разработка более детального плана разработки;
 - 3) создание прототипов и проведение исследований концепции;
 - 4) идентификация и управление рисками;
 - 5) определение основных компонентов и модулей системы;
 - 6) разработка итерационного плана разработки;
- в) фаза конструирования (construction phase):
 - 1) итеративная реализация системы, включая программирование, тестирование и интеграцию компонентов;

- 2) систематическое выполнение итераций разработки, в каждой из которых происходит детализация требований, проектирование, разработка и тестирование;
- 3) постепенное улучшение системы и добавление новых функций;
- 4) контроль качества и тестирование продукта;
- г) фаза перехода (transition phase):
 - 1) подготовка системы к выпуску и передаче заказчику;
 - 2) тестирование системы на соответствие требованиям и стабильность работы;
 - 3) обучение пользователей и администраторов системы;
 - 4) поддержка системы после выпуска, включая исправление ошибок и обработку запросов на изменение;
 - 5) оценка проекта и извлечение уроков для будущих проектов.

RUP предлагает разработку через серию итераций, что позволяет команде получать обратную связь от заказчика и пользователей на ранних стадиях проекта. Это позволяет быстро внести корректировки и уточнения в требования и архитектуру, улучшая конечный результат. Также выбранная методология предоставляет гибкую рамку для разработки, позволяя адаптировать процесс к конкретным потребностям проекта. Это позволяет выбирать и масштабировать фазы, активности и роли в соответствии с требованиями и ограничениями. RUP акцентирует внимание на идентификации и управлении рисками в проекте.

Предлагаются подходы к анализу и оценке рисков. Это позволяет сосредоточиться на критических аспектах проекта и своевременно реагировать на возникающие проблемы. Методология обладает механизмами для управления изменениями в проекте. Это помогает контролировать и управлять изменениями в требованиях, архитектуре и планах проекта, чтобы минимизировать негативные последствия и обеспечить согласованность и качество продукта. Акцентируется особое внимание на качество разрабатываемого программного

обеспечения. Также RUP акцентирует внимание на контроле качества во время всего процесса разработки.

При планировании проекта было решено использовать Microsoft Project для эффективного управления всеми стадиями его выполнения. Это приложение позволяет структурировать проект, составить его календарный план и рассчитать затраты труда, материальных и финансовых ресурсов в процессе работы над проектом.

Для реализации программы будут использованы Visual Studio и программные надстройки Visual Studio Tools for Office (VSTO), представляющие собой набор средств разработки, доступных в виде Visual Studio add-in (шаблоны проектов) и runtime, что позволяет Microsoft Office 2003 и более поздние версии приложений Office использовать функциональность .NET Framework Common Language Runtime (CLR) с помощью .NET type system. Реализация проекта будет выполнена на языке программирования C#.

2.2 Проектирование программного средства

В ходе проектирования необходимо определить общую структуру системы, компоненты, их взаимодействие и данные, которые будут использоваться. Важно разработать архитектуру таким образом, чтобы она была гибкой, масштабируемой и легко поддерживаемой. Проектирование программного средства направлено на создание надежного, эффективного и удобного в использовании продукта, который соответствует требованиям пользователей и решает поставленные задачи. Проектирование является важным этапом разработки, который позволяет создать детальный план и структуру программного продукта [8].

Для понимания взаимодействия программного средства с системой необходима диаграмма развертывания. Диаграмма развертывания является полезным инструментом для моделирования и планирования размещения компонентов программной системы. Она помогает визуализировать инфраструктуру, оптимизировать производительность, планировать

развертывание и обслуживание, а также облегчает коммуникацию и согласование между участниками проекта [9].

На рисунке 9 представлена диаграмма развертывания, которая показывает, как расположены компоненты системы в физическом отношении.

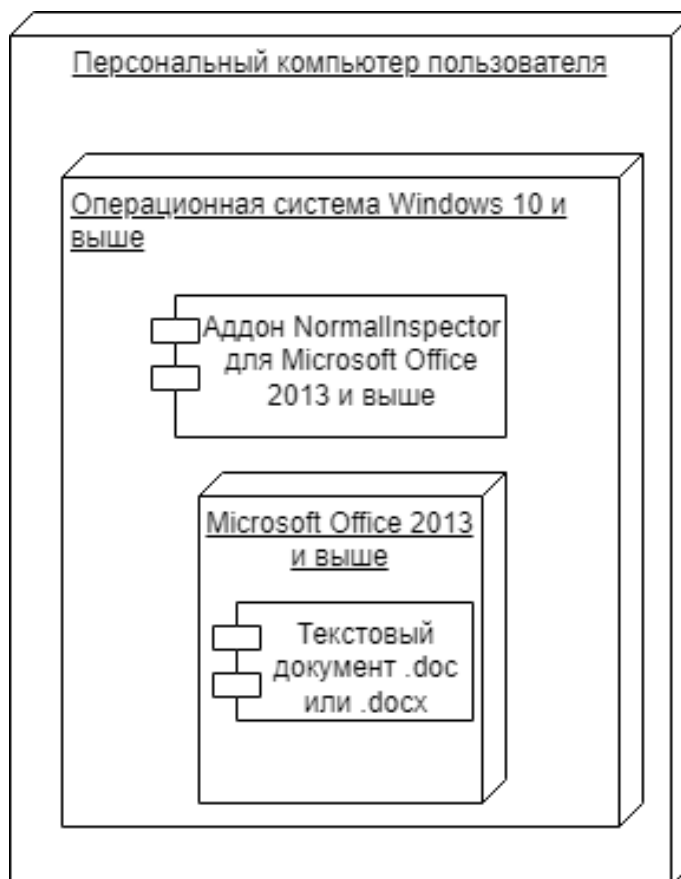


Рисунок 9 – Диаграмма развертывания для программы

При реализации данная диаграмма используется для программы и связи компонентов между собой.

2.3 Проектирование интерфейса пользователя

На основе результатов анализа аналогичных программных средств и требований к ним был разработан интерфейс.

Главная вкладка для взаимодействия со средством представлена на рисунке 10. На ней имеются следующие элементы: 1 – кнопка запуска проверки, 2 – кнопка очистки заметок.

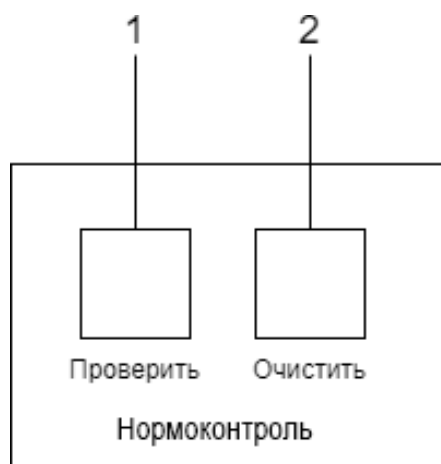


Рисунок 10 – Вкладка для взаимодействия

При нажатии на кнопку проверки запускается анализ текстового документа на наличие ошибок в оформлении. Кнопка очистки производит очистку текстового документа от результатов проверки, содержащихся в заметках. Окно с информацией о состоянии проверки текстового документа представлено на рисунке 11. На нем имеются следующие элементы: 1 – кнопка прерывания проверки, 2 – текст, содержащий информацию о статусе и этапе проверки, 3 – полоса загрузки, отображающая процент анализа текстового документа.

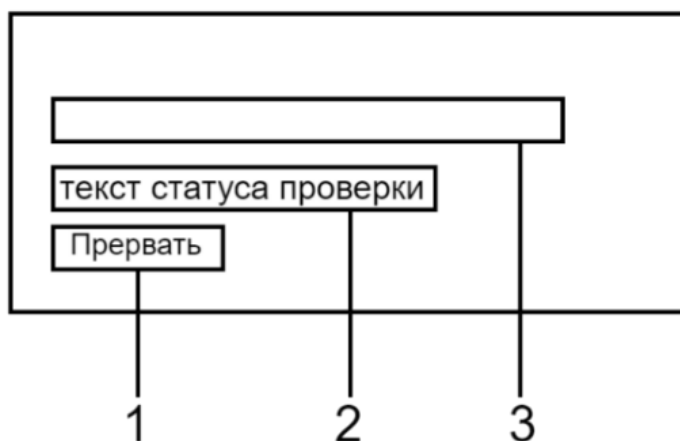


Рисунок 11 – Окно состояния проверки

При нажатии на кнопку «Прервать» прерывается анализ текстового документа. Текст статуса проверки предоставляет информацию по текущему этапу контроля документа. Полоса загрузки отображает состояние проверки текста на каждом этапе.

Результат проверки выводится в примечании в рабочем пространстве Microsoft Office Word (рисунок 12). Включает в себя следующие элементы: 1 – основное рабочее пространство Microsoft Office Word, 2 – текстовый документ, в котором был проведен контроль оформления, 3 – поле примечаний, содержащее результаты проверки.

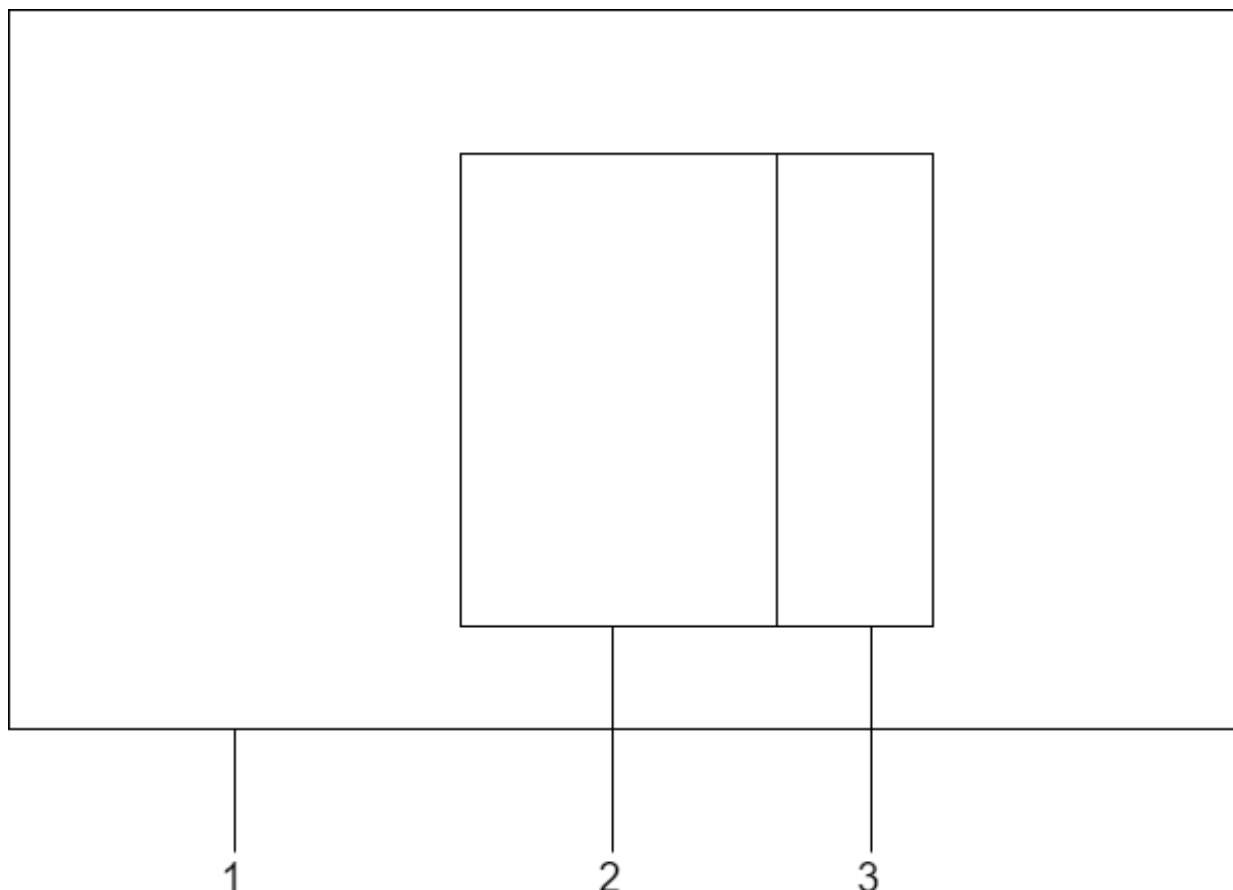


Рисунок 12 – Результирующее представление после проверки документа

После завершения анализа оформления текстового документа результаты выводятся в примечании к документу выделяя в тексте ошибки. Также выводится объяснение к каждой ошибке. С помощью кнопки «Очистить» удалятся все созданные примечания.

3 Реализация средства автоматизации контроля оформления текстовых документов

В процессе проектирования были определены функции и возможности, которые были реализованы в программном продукте. Это включает в себя определение основных функций, алгоритмов, обработки данных и взаимодействия с пользователем.

Диаграмма классов – это структурная диаграмма, используемая в языке моделирования UML (Unified Modeling Language) для визуализации структуры классов и их взаимосвязей в программной системе.

Диаграмма классов позволяет представить основные классы системы, их атрибуты (переменные) и методы (функции), а также связи между классами. Она помогает разработчикам и аналитикам лучше понять структуру системы, ее компоненты и взаимодействие между ними.

Некоторые основные элементы диаграммы классов включают:

- классы;
- атрибуты;
- методы;
- ассоциации;
- наследование;
- интерфейсы.

Диаграмма классов является важным инструментом для анализа, проектирования и документирования программных систем. Она помогает визуализировать структуру классов, их атрибуты, методы и взаимосвязи, что способствует более четкому пониманию системы и облегчает коммуникацию между разработчиками и заинтересованными сторонами.

Реализованные классы программного средства отображены на диаграмме классов (рисунок 13).



Рисунок 13 – Диаграмма классов

Ниже распишем названия и методы каждого класса.

Название класса: ThisAddIn.

Назначение: Класс служит для выполнения таких задач, как получение доступа к объектной модели ведущего приложения Microsoft Office, настройка пользовательского интерфейса приложения, а также предоставление объектов созданной надстройки VSTO другим решениям Office.

Члены класса:

- Application – объект модели ведущего приложения;
- CancelAnalysis – объект состояния анализа;
- CustomTaskPaneCollection – коллекция настраиваемых областей задач в приложении Microsoft Office;
- Missing – объект отсутствующего значения в данных объекта Type;
- VstoSmartTag – коллекция смарт-тегов в решении Word, созданных с помощью инструментов разработчика Office в Visual Studio.

Функции класса:

- ActiveDocumentCommentsAdd(): void – добавляет к параграфу примечание;
- ApplicationLeft(): int – возвращает параметр Left приложения;
- ApplicationTop(): int – возвращает параметр Top приложения;
- BeginInitialization(): void – метод вызываемый, когда инициализация в XAML начинается;
- BindToData(): void – метод быстрого доступа для set_dataSource(данные);
- DeleteAllComments(): void – удалить все примечания в работе;
- BeginInitialization(): void – метод вызываемый, когда инициализация в XAML заканчивается;
- ErrorsMainText(): void – метод определения ошибок параметров абзаца у основного текста;
- ErrorsPagest(): void – метод определения ошибок параметров страниц у документа;
- ErrorsPictures(): void – метод определения ошибок параметров абзаца для рисунков и подрисуночных надписей;
- ErrorsQuotationMarks(): void – метод определения ошибок кавычек в тексте параграфа;
- ErrorsTableName(): void – метод определения ошибок параметров абзаца названий таблиц;

- `ErrorsTitles(): void` – метод определения ошибок параметров абзаца для заголовков;
- `FindAll(): void` – метод определения совпадений с запросом в параграфе;
- `FinishInitialization(): void` – метод завершающий инициализацию;
- `Initialization(): void` – метод начинающий инициализацию;
- `InitializeCachedData(): void` – метод инициализации кэширования данных;
- `InitializeComponent(): void` – метод инициализации компонентов;
- `InitializeControls(): void` – инициализирует пользовательский элемент управления свойством с указанным объектом;
- `InitializeData(): void` – метод просмотра данных, предоставленные для `RowsManager`, и создает объекты `ColumnBase`;
- `InitializeDataBindings(): void` – метод инициализации кэширования данных;
- `InitializeCachedData(): void` – метод инициализации кэширования данных;
- `InitializeComponents(): void` – метод загрузки откомпилированной страницы компонента;
- `InitializeControls(): void` – метод чтобы установить начальные значения для элементов управления;
- `InitializeData(): void` – метод инициализации данных;
- `InitializeDataBindings(): void` – метод инициализации связи с данными;
- `InternalStartup(): void` – метод инициализации и привязки обработчиков событий;
- `Iscached(): bool` – метод возвращает `true`, если для этого шаблона существует действительный кеш;
- `IsPictureTest(): bool` – метод проверяет является ли параграф рисунком или подрисуночной надписью;
- `IsTableNameTest(): bool` – метод проверяет является ли параграф названием таблицы;
- `IsTableTest(): bool` – метод проверяет является ли параграф частью таблицы;

- `IsTitleTest()`: `bool` – метод проверяющий является ли параграф заголовком;
- `NeedsFill()`: `bool` – метод возвращает `false`, если объект был загружен из кеша автоматически, в противном случае – `true`;
- `OnShutdown()`: `void` – метод задает действия, выполняемые непосредственно перед завершением работы системы;
- `PageAnalysis()`: `void` – метод проверки параметров страниц;
- `ParagraphsAnalysis()`: `void` – метод проверки параметров абзацев;
- `StartCaching()`: `void` – метод для добавления объекта данных в кэш данных;
- `StopCaching()`: `void` – метод удаление объекта данных из кэша данных;
- `TextAnalysis()`: `void` – метод проверки на типовые ошибки;
- `ThisAddIn_Shutdown()`: `void` – метод выполнения кода для очистки используемых надстройкой VSTO ресурсов перед выгрузкой надстройки VSTO;
- `ThisAddIn_Shutdown()`: `void` – метод выполнения кода для инициализации надстройки VSTO при ее загрузке.

Название класса: `Ribbon`.

Назначение: Этот класс отображается как разделяемый класс, разделенный между файлом кода ленты и файлом кода конструктора ленты.

Члены класса:

- `ButtonAnalysis` – объект модели ведущего приложения;
- `ButtonClear` – объект кнопки удаления созданных заметок;
- `Components` – обязательная переменная конструктора;
- `Group1` – представляет группу элементов управления на вкладке ленты;
- `Tab1` – объект, содержащий одну или несколько групп элементов управления на ленте;
- `Progressform` – объект, отображающий `progressBar`.

Функции класса:

- `Button1_Click()`: `void` – метод для вызова анализа текстового документа;
- `ButtonClear_Click()`: `void` – метод для удаления созданных заметок;

- Dispose(): void – метод очистки всех используемых ресурсов;
- InitializeComponent(): void – метод для поддержки конструктора;
- LabelUpdate(): void – метод обновления информации в label;
- ProgressBarSettings(): void – метод определяющий параметры progressBar;
- ProgressBarUpdate(): void – метод обновления прогресса progressBar;
- Ribbon(): void – метод объявления элемента Лента (визуальный конструктор);
- Ribbon_Load(): void – метод загружает элемент Лента, созданный VSTO.

Название класса: ProgressForm.

Назначение: Позволяет создавать различные типы форм .NET в сеансе ABL.

Члены класса:

- ButtonCancelAnalysis – объект модели ведущего приложения;
- Components – обязательная переменная конструктора;
- LabelCurrentOperation – объект, отображающий название текущей операции в progressBar;
- LabelOperationNumber – объект, отображающий номер текущей операции в progressBar;
- ProgressBar – пользовательский интерфейс мультиплатформенного приложения.

Функции класса:

- ButtonCancelAnalysis_Click(): void – метод для кнопки отмены анализа текстового документа;
- Dispose(): void – метод очистки всех используемых ресурсов;
- InitializeComponent(): void – метод для поддержки конструктора;
- ProgressForm() – инициализирует новый экземпляр класса Form.

Название класса: Global.

Назначение: Абстрактный класс необходимый для предоставления общего определения для базового класса, которое могут совместно использовать несколько производных классов.

Члены класса:

– ApplicationFactory – предоставляет методы, которые можно использовать для решения различных задач в надстройках уровня приложения, например, расширение собственных документов Word и создание элементов управления ленты во время выполнения;

– ThisAddIn – объект класса ThisAddIn;

– ThisRibbonCollection – объект коллекции элемента Лента.

Функции класса:

– Globals() – метод определения объекта для абстрактного класса.

Название класса: Resources.

Назначение: Класс ресурсов со строгой типизацией для поиска локализованных строк.

Члены класса:

– ResourceCulture – Переопределяет свойство CurrentUICulture текущего потока для всех;

– ResourceMan – Возвращает кэшированный экземпляр ResourceManager, используемый этим классом.

Функции класса:

– Resources() – метод определения объекта для абстрактного класса.

Название класса: Settings.

Назначение: Представляет параметры конфигурации.

Члены класса:

– defaultInstance – экземпляр, который может использоваться, когда Настройка загрузки сборок не требуется.

Функции класса:

– Default() – инициализирует новый экземпляр класса.

Название класса: ErrorPattern.

Назначение: структура для хранения шаблона ошибки.

Члены класса:

– SearchTemplate – шаблон поиска;

– `ErrorDescription` – описание ошибки.

Функции класса:

– `ErrorPattern ()` – функция поиска ошибок в текстовом документе.

Название класса: `ThisRibbonCollection`.

Назначение: Представляет параметры конфигурации.

Члены класса:

– `Ribbon` – экземпляр, который может использоваться, когда Настройка загрузки сборок не требуется.

Функции класса:

– `ThisRibbonCollection()` – метод определения объекта для абстрактного класса.

Название класса: `TypicalErrors`.

Назначение: Представляет параметры конфигурации.

Члены класса:

– `Templates` – список, содержащий описание ошибок.

Функции класса:

– `TypicalErrors()` – метод определения ошибок и их добавления в список найденных ошибок.

Листинг реализованного программного средства представлен в приложении А.

Также была составлена диаграмма компонентов (рисунок 14).

Диаграмма компонентов – это структурная диаграмма, которая показывает разбиение системы на компоненты и связи между ними и используется в процессе проектирования программных и информационных систем. Ее цель – визуализировать архитектуру системы и помочь разработчикам лучше понять, как различные компоненты взаимодействуют друг с другом и как они могут быть организованы.

Преимуществами от использования диаграмм компонентов является визуализация архитектуры, модульность и повторное использование, управление сложностью.

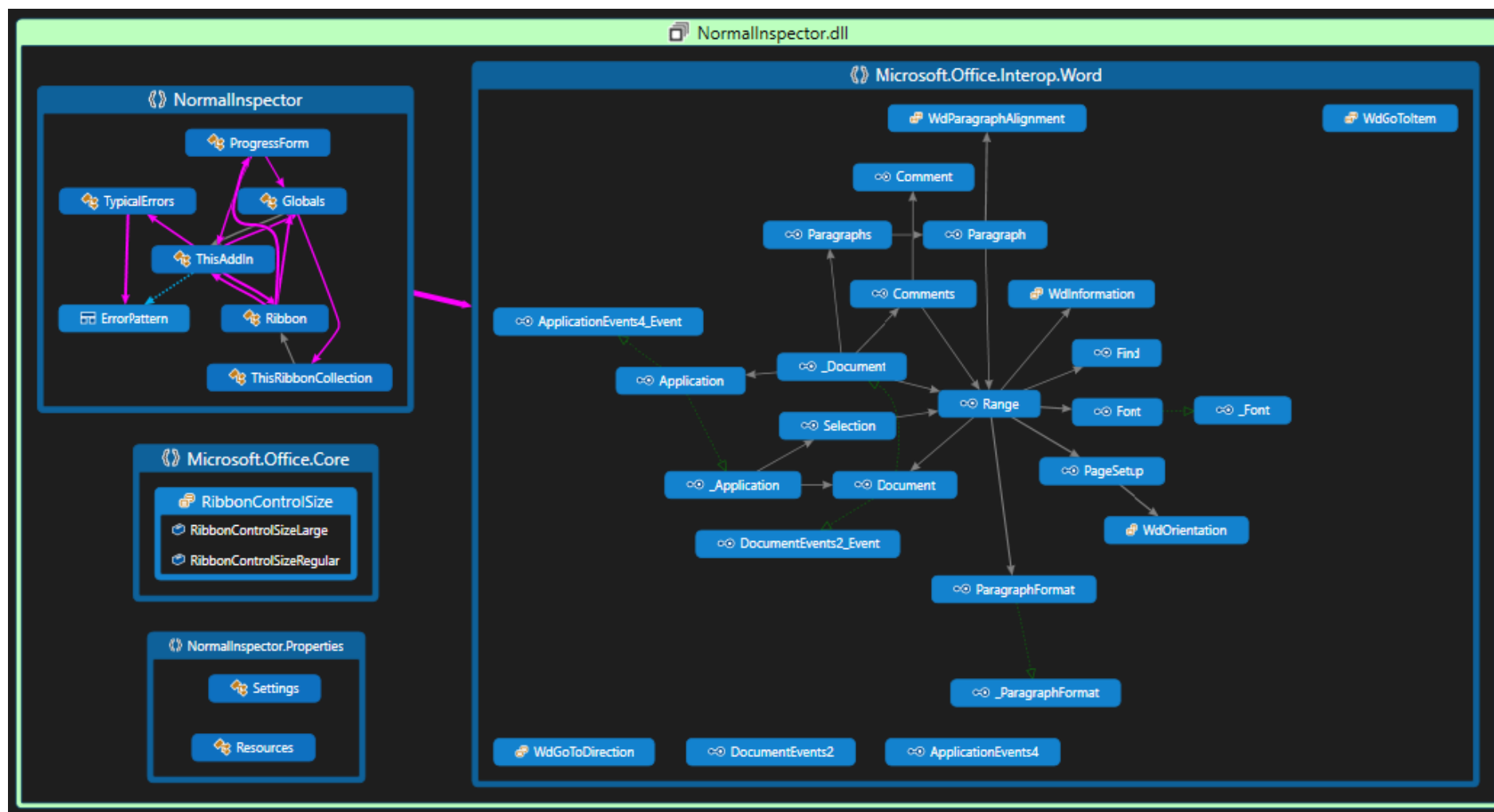


Рисунок 14 – Диаграмма компонентов

Используя данную диаграмму, можно определить, какие компоненты являются узкими местами и могут привести к снижению производительности.

4 Тестирование программного продукта

4.1 Тестирование приложения методом черного ящика

Для разработанного проекта будет проведено тестирование методом черного ящика, так как оно в достаточной мере позволяет протестировать приложение и не требует больших затрат для реализации.

Тестирование методом черного ящика – это метод тестирования программного обеспечения, при котором тестируются входы и выходы программы без знания внутренней структуры и реализации системы. В этом методе тестируется только внешнее поведение программы, а не ее внутренняя логика [10].

Для примера будут приведены тесты для метода по определению заголовка и его параметров. Табличное представление классов эквивалентности приведены в таблице 8.

Таблица 8 – Классы эквивалентности

	Правильные КЭ	Неправильные КЭ
Размер шрифта	14 (1)	0 – 13 (2) 15 – ∞ (3)
Используемый шрифт	Используется Times New Roman (4)	Используется не Times New Roman (5)
Начертание	Полужирный (6)	Курсив (7) Подчеркнутый (8)
Выравнивание	По центру (9) По левому краю (11)	По правому (10) По ширине (12)
Отступ слева	Равен 0 см (13)	Не равен 0 см (14)
Отступ справа	Равен 0 см (15)	Не равен 0 см (16)
Интервал «после»	Равен 10 пунктам (17)	Не равен 10 пунктам (18)
Интервал «перед»	Равен 10 пунктам (19)	Не равен 10 пунктам (20)
Междустрочный интервал	Равен 1,5 строки (21)	Не равен 1,5 строки (22)
Отступ первой/красной строки	Равен 0 см (23) Равен 1,25 см (25)	Остальные отступы (24)

После определения структуры для определения заголовка и его параметров необходимо привести примеры, которые будут содержать уникальную последовательность классов эквивалентности. Данные примеры в полной мере будут покрывать метод по созданию логина. Примеры приведены в таблице 9.

Таблица 9 – Тест – кейсы определения заголовка и его параметров

№	Классы эквивалентности	Входные данные	Выходные данные
1	Все правильные 1,4,6,9,13,15,17,19,21,23	Введение	Успешно
2	Все правильные 1,4,6,11,13,15,17,19,21,25	4 Тестирование программного продукта	Успешно
3	2,4,6,9,13,15,17,19,21,23	Заключение	Ошибка: Размер шрифта должен быть равен 14
4	3,4,6,9,13,15,17,19,21,23	Заключение	Ошибка: Размер шрифта должен быть равен 14
5	1,5,6,11,13,15,17,19,21,23	1 Анализ требований	Ошибка: Шрифт должен быть равен Times New Roman
6	1,4,7,9,13,15,17,19,21,23	Список литературы	Ошибка: Шрифт должен быть равен Times New Roman
7	1,4,7,10,13,15,17,19,21,23	Проектирование	Ошибка: Некорректное выравнивание заголовка!
8	1,4,7,12,13,15,17,19,21,23	2 Проектирование программы	Ошибка: Некорректное выравнивание заголовка!
9	1,4,7,9,14,15,17,19,21,23	Список литературы	Ошибка: Отступ слева должен быть равен 0 см.
10	1,4,7,9,13,16,17,19,21,23	Список литературы	Ошибка: Отступ справа должен быть равен 0 см.
11	1,4,7,9,13,15,18,19,21,23	1 Анализ требований	Ошибка: Интервал «После» должен быть равен 10 пт.
12	1,4,7,9,13,15,17,20,21,23	1 Анализ требований	Ошибка: Интервал «Перед» должен быть равен 10 пт.
13	1,4,7,9,13,15,17,19,22,23	3 Реализация программы	Ошибка: Междустрочный интервал должен быть равен 1,5 строки
15	1,4,7,9,13,15,17,19,22,25	3 Реализация программы	Ошибка: Отступ первой/красной строки у нумерованного заголовка должен быть равен 0 см.

Как видно из таблицы примеры полностью покрывают условия для определения заголовка и его параметров.

4.2 Функциональное тестирование

На данном этапе было выполнено функциональное тестирование в целях проверки правильности работы программы. Результаты функционального тестирования представлены в таблице 10.

Таблица 10 – Результаты функционального тестирования

№	Описание теста	Тест и параметры	Результат
1	Запустить проверку на нормоконтроль	Нажата кнопка «Проверить»	Тест пройден. Проверка запущена и открыто окно прогресса проверки (рисунок Б.1)
2	Прервать проверку на нормоконтроль	Нажата кнопка «Прервать»	Тест пройден. Проверка прервана в процессе (рисунок Б.2, рисунок Б.3)
3	Вывести результаты нормоконтроля	Прогресс проверки завершен успешно	Тест пройден. Примечания с результатами выведены (рисунок Б.4)
4	Очистить результаты нормоконтроля	Выведены результаты проверки, нажата кнопка «Очистить»	Тест пройден. Примечания удалены (рисунок Б.5, рисунок Б.6)
5	Проверка параметров страниц	При проверке документа были обнаружены неправильные параметры страниц	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.7)
6	Проверка параметров абзацев основного текста	При проверке документа были обнаружены неправильные параметры абзацев основного текста	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.8)
7	Проверка параметров абзацев для заголовков	При проверке документа были обнаружены неправильные параметры абзацев для заголовков	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.9)
8	Проверка параметров абзацев названий таблиц	При проверке документа были обнаружены неправильные параметры абзацев названий таблиц	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.10)
9	Проверка параметров абзацев для рисунков и подписей	При проверке документа были обнаружены неправильные параметры абзацев для рисунков и подписей	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.11)
10	Проверка нумерованных списков	При проверке документа были обнаружены неправильные параметры нумерованных списков	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.12)

Продолжение таблицы 10

№	Описание теста	Тест и параметры	Результат
11	Проверка маркированных списков	При проверке документа были обнаружены неправильные параметры маркированных списков	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.13)
12	Проверка параметров абзаца названий фрагмента кода	При проверке документа были обнаружены неправильные параметры абзаца названий фрагмента кода	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.14)
13	Проверка параметров абзаца фрагмента кода	При проверке документа были обнаружены неправильные параметры абзаца фрагмента кода	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.15)
14	Проверка тире и кавычек	При проверке документа были обнаружены недопустимые символы	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.16)
15	Проверка неразрывных пробелов	При проверке документа были обнаружены элементы текста, где необходим неразрывный пробел	Тест пройден. Сгенерировано примечание с соответствующим текстом (рисунок Б.17)

Все тесты были выполнены успешно. Следовательно, можно говорить о корректной работе программного средства.

4.3 Метрики кода

Метрики кода отображают количественные величины программного обеспечения, которые предоставляют разработчику доступ к таким свойствам, как эффективность, оптимизацию и масштабируемость написанного кода приложения [11].

Способы, которые используют для получения метрик код, существует огромное количество. Данные способы различны по своим алгоритмам и по информации, полученной в виде результата. Для разработанного приложения будут проведены следующие группы метрик:

- метрики размера программы (SLOC-метрики);
- метрики сложности потока управления программ (метрики Холстеда).

Для оценки возьмем метод `ErrorsMainText` класса `ThisAddIn`, который используется для проверки ошибки параметров абзаца для основного текста. Код для метода приведен ниже в листинге 1.

Листинг 1 – Проверка параметров абзаца основного текста

```
private void ErrorsMainText(int IndexParagraph)
{
    string ErrorText = string.Empty;
    //отступ слева
    float LeftIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LeftIndent;
    if (LeftIndent != 0)
        ErrorText += "\nОтступ слева должен быть равен 0
см.";

    //отступ справа
    float RightIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.RightIndent;
    if (RightIndent != 0)
        ErrorText += "\nОтступ справа должен быть равен 0
см.";

    //отступ красной строки
    float FirstLineIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.FirstLineIndent;
    if (FirstLineIndent != 35.45f)
        ErrorText += "\nОтступ первой/красной строки
должен быть равен 1,25 см.";
    //интервал перед
    float SpaceBefore =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceBefore;
    if (SpaceBefore != 0)
        ErrorText += "\nИнтервал «Перед» должен быть равен
0 пт.";

    //интервал после
    float SpaceAfter =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceAfter;
    if (SpaceAfter != 0)
        ErrorText += "\nИнтервал «После» должен быть равен
0 пт.";

    //междустрочный интервал
    float LineSpacing =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LineSpacing;
    if (LineSpacing != 18f)
        ErrorText += "\nМеждустрочный интервал должен быть
равен 1,5 строки.";
```

```

        //выравнивание
        if
        (this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent != Word.WdParagraphAlignment.wdAlignParagraphJustify)
            ErrorText += "\nВыравнивание у основного текста
должно быть по ширине.";
        //вывод ошибок в абзаце
        if (!String.IsNullOrEmpty(ErrorText))
        {
            object text = "Ошибка настройки параметров абзаца
у основного текста!" + ErrorText;

ActiveDocumentCommentsAdd(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range, text);
        }
    }
}

```

Метод принимает на вход индекс и текст текущего абзаца. Задача метода перебрать основные параметры текста в абзаце и проверить отступ слева, отступ справа, отступ первой строки, интервал «перед» и интервал «после», а также междустрочный интервал и выравнивание текста. В случае обнаружения ошибок вывести примечание в документ в абзаце.

Для данного метода будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

4.3.1 Метрики размера программы

Первой проведенной метрикой будет подсчет размера программы, где будет вычислено количество строк программного кода, а также число комментариев. Метрики будут найдены на основе выше приведенного кода. Результат представлен в таблице 11.

Таблица 11 – Метрики размера программы

Метрика	Количество
Физических строк кода	37
Логических строк кода	20
Строки-комментарии	8

Приведенные количественные значения дают возможность вычислить процент комментариев для рассматриваемого участка кода. Процент комментариев равен $(8/37) * 100 = 21$. Полученный результат свидетельствует о

достаточности комментирования кода. Это полезно для облегчения понимания кода другими разработчиками и обеспечения его поддержки в будущем.

4.3.2 Метрики Холстеда

Для вычисления метрик Холстеда необходимо измерить 4 характеристики программы

- число уникальных операторов;
- число уникальных операндов;
- общее число операторов;
- общее число операндов.

Для вычисления характеристик были составлены словарь операторов, которой отображен в таблице 10 и словарь операндов, который описан в таблице 11.

Таблица 10 – Словарь операторов

Оператор	Число операторов
if	8
+=	7
!=	6
.	35
()	10
{}	2
;	16
[]	8
=	2
+	1

Таблица 11 – Словарь операндов

Операнд	Число операндов
ErrorText	8
LeftIndent	2
IndexParagraph	8
RightIndent	2

Продолжение таблицы 11

FirstLineIndent	2
SpaceBefore	2
SpaceAfter	2
LineSpacing	2
text	2

Таким образом, число уникальных операторов программы $n1 = 10$, число уникальных операндов программы $n2 = 9$, общее число операторов в программе $N1 = 95$, общее число операндов в программе $N2 = 30$.

На основе метрик Холстеда можно вычислить целый ряд характеристик программы. Рассмотрим некоторые из них:

- словарь программы $n = n1 + n2$;
- длина программы $N = N1 + N2$;
- объем программы $V = N * \log_2 n$;
- сложность программы $D = (n1/2) * (N2/n2)$;
- оценка интеллектуальных усилий $E = D * V$.

Рассчитаем словарь программы: $n = 10 + 9 = 19$.

Рассчитаем длину программы: $N = 95 + 30 = 125$.

Рассчитаем объем программы: $V = 125 * \log_2 19 = 530$

Рассчитаем сложность программы: $D = (10/2) * (30/9) = 16,6$.

Рассчитаем оценку интеллектуальных усилий: $E = 16,6 * 530 = 8798$.

Подсчитанные значения предоставляют информацию о том, что в проекте количество используемых переменных не велико, а значит данный проект будет проще поддерживать в будущем.

Также длина программы и ее объем показывают, что код является обширным, но при этом он разбит на логические функциональные блоки, которые легко масштабировать.

Сложность программы и оценка интеллектуальных усилий демонстрирует, что имеется множества высокоуровневых связей.

Полученные характеристики показывают, что программный код имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки.

Заключение

В ходе выполнения выпускной квалификационной работы были созданы программные средства автоматизации контроля оформления текстовых документов.

На этапе анализа предметной области и требований был выполнен анализ программного обеспечения аналогичного назначения, в результате которого были выявлены их основные достоинства и недостатки, определены функциональные и нефункциональные требования.

Также было осуществлено планирование разработки и оценки бюджета. Из результатов планирования следует, что проект займет 45 дней на реализацию, а расчетные затраты составят 108193 рубля, откуда можно сделать вывод об его экономической целесообразности.

В процессе проектирования были определены технологии и инструментальные средства разработки, составлена диаграмма развертывания, а также спроектирован пользовательский интерфейс. В разработке будет использоваться язык программирования C# и IDE Microsoft Visual Studio. Для взаимодействия с программным продуктом Microsoft Office Word будут подключены программные надстройки Visual Studio Tools for Office, представляющие собой набор средств разработки, доступных в виде Visual Studio add-in (шаблоны проектов) и runtime.

На этапе реализации была составлена диаграмма классов и диаграмма компонентов, которые реализованы в программе. Также был реализован весь спроектированный функционал.

В перспективе планируется использовать разработанное программное средство для осуществления нормоконтроля выпускных квалификационных работ на кафедре «Математическое обеспечение и применение ЭВМ». Также будет осуществляться не только сопровождение и поддержка актуальности программы, но и расширение функциональности.

Таким образом, в ходе выполнения выпускной квалификационной работы были успешно выполнены все поставленные задачи.

Список использованных источников

1. Бережков, А.В. Опыт повышения качества оформления выпускных квалификационных работ студентов технического вуза/ Бережков, А.В. Валитова Ю.О., // Педагогический журнал. Т. 10. 2020. № 1А. С. 367–375.
2. Манкевич, О.В. Особенности автоматизации нормоконтроля текстовых документов / Манкевич О.В., Семеняк П.А. // Информационные технологии и системы. 2015. С. 124–125.
3. Стариченко, Б.Е. Программа автоматизации контроля оформления текстовых документов / Стариченко Б.Е., Устинов М.А. // Педагогическое образование в России. 2018. №. 8. С. 163–168.
4. СТО ПГУ 3.12 – 2018. ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры. – Введ. 27–09–2018. Пенза. : ПГУ, 2018. – 33 с.
5. Начало программирования надстроек VSTO [Электронный ресурс]. URL:<https://learn.microsoft.com/ru-RU/visualstudio/vsto/getting-startedprogramming-vsto-add-ins?view=vs-2022> (дата обращения: 02.05.2023).
6. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб. : Лань, 2019. – 244 с.
7. Кролл, П. Rational Unified Process – это легко. Руководство по RUP для практиков: Пер. с англ. / М.: КУДИЦ-ОБРАЗ, 2004. – 432 с.
8. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – 2-е изд. – СПб. : Питер, 2007. – 544 с.
9. Буч, Г. Язык UML. Руководство пользователя, 2-е издание / Г. Буч, Дж. Рамбо, И. Якобсон; пер. с англ.: Н. Мухин. – М. : Издательский дом «ДМК Пресс», 2006. – 496 с.
10. Куликов, С. Тестирование программного обеспечения. Базовый курс, 3-е издание / С. Куликов. – М. : Издательство МГТУ им. Н. Э. Баумана, 2020. – 301 с.

11. Программный код и его метрики. [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 18.05.2023).

Приложение А
ИСХОДНЫЙ КОД ПРОГРАММЫ
(рекомендуемое)

Листинг А.1 – Файл ThisAddIn.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml.Linq;
using Word = Microsoft.Office.Interop.Word;
using Office = Microsoft.Office.Core;
using Microsoft.Office.Tools.Word;

namespace NormalInspector
{
    public partial class ThisAddIn
    {
        public static bool CancelAnalysis;

        private void ThisAddIn_Startup(object sender,
System.EventArgs e)
        {
        }

        private void ThisAddIn_Shutdown(object sender,
System.EventArgs e)
        {
        }

        //Возвращает параметр Left приложения
        public int ApplicationLeft()
        {
            return this.Application.Left + (this.Application.Width
/ 2);
        }

        //Возвращает параметр Top приложения
        public int ApplicationTop()
        {
            return this.Application.Top + (this.Application.Height
/ 2);
        }

        //Удалить все примечания в работе
        public void DeleteAllComments()
        {
            if (this.Application.ActiveDocument.Comments.Count !=
0)
this.Application.ActiveDocument.DeleteAllComments();
        }

        //Найти все совпадения с запросом в параграфе
        private List<Word.Range> FindAll(Word.Range range, string
findText)

```

```

    {
        int start = range.Start;
        int end = range.End;
        List<Word.Range> ranges = new List<Word.Range>();
        range.Find.Execute(FindText: findText, MatchCase:
true);
        while (range.Find.Found)
        {
            //The search will change the range, here is an
out-of-range judgment
            if (range.Start > end)
                break;
            ranges.Add(range.Document.Range(range.Start,
range.End));
            range.Find.Execute(FindText: findText, MatchCase:
true);
        }
        //Restore the original range
        range.SetRange(start, end);
        return ranges;
    }

    //Добавить к параграфу примечание
    private void ActiveDocumentCommentsAdd(Word.Range range,
object comments)
    {
        this.Application.ActiveDocument.Comments.Add(range,
ref comments);
    }

    //Метод проверяет является ли параграф частью таблицы
    private bool IsTableTest(Word.Range range)
    {
        object vobjTable =
range.get_Information(Word.WdInformation.wdWithInTable);
        if (vobjTable.ToString() != "False")
            return true;
        else
            return false;
    }

    //Метод проверяет является ли параграф названием таблицы
    private bool IsTableNameTest(int IndexParagraph)
    {
        if (IndexParagraph + 1 <
this.Application.ActiveDocument.Paragraphs.Count)
        {
            Word.Range next =
this.Application.ActiveDocument.Paragraphs[IndexParagraph +
1].Range;
            if (IsTableTest(next))
            {
                return true;
            }
        }
    }

```

```

        }
        else
        {
            return false;
        }
    }
    else
        return false;
}

//Метод проверяющий является ли параграф заголовком
private bool IsTitleTest(int IndexParagraph)
{
    var document = this.Application.ActiveDocument;
    //признак нумерованного заголовка - частично
    полужирный и выравнено по центру
    if (document.Paragraphs[IndexParagraph].Range.Bold ==
    9999999 &&
        document.Paragraphs[IndexParagraph].Alignment ==
    Word.WdParagraphAlignment.wdAlignParagraphCenter)
        return true;
    //основными признаками заголовка является полужирный
    шрифт
    if (document.Paragraphs[IndexParagraph].Range.Bold ==
    -1)
        return true;
    else
        return false;
}

//Метод проверяет является ли параграф рисунком или
    подрисуночной надписью
private bool IsPictureTest(int IndexParagraph)
{
    var document = this.Application.ActiveDocument;
    //признак рисунка или подрисуночной надписи - не
    полужирный и выравнено по центру
    if (document.Paragraphs[IndexParagraph].Range.Bold ==
    0 &&
        document.Paragraphs[IndexParagraph].Alignment ==
    Word.WdParagraphAlignment.wdAlignParagraphCenter)
        return true;
    else
        return false;
}

//Метод проверяет является ли параграф маркированным
    списком
private bool IsBulletedlist(int IndexParagraph)
{
    var document = this.Application.ActiveDocument;
    //признак маркированного списка - в начале стоит
    маркер

```

```

        if
        (document.Paragraphs[IndexParagraph].Range.ListFormat.ListType ==
Word.WdListType.wdListBullet)
            return true;
        else
            return false;
    }

    //Метод проверяет является ли параграф нумерованным
СПИСКОМ
    private bool IsNumberedlist(int IndexParagraph)
    {
        var document = this.Application.ActiveDocument;
        //признак нумерованного списка - в начале стоит
нумерация
        if
        (document.Paragraphs[IndexParagraph].Range.ListFormat.ListType ==
Word.WdListType.wdListOutlineNumbering)
            return true;
        else
            return false;
    }

    //Метод проверяет является ли параграф названием листинга
    private bool IsListingNameTest(int IndexParagraph)
    {
        var document = this.Application.ActiveDocument;
        //признак названия листинга - первое слово строки
"Листинг"
        if
        (document.Paragraphs[IndexParagraph].Range.Find.Text == "Листинг")
            return true;
        else
            return false;
    }

    //Метод проверяет является ли параграф фрагментом кода
    private bool IsCodeListing(int IndexParagraph)
    {
        var document = this.Application.ActiveDocument;
        //признак листинга - шрифт Courier New, Consolas
        if
        (document.Paragraphs[IndexParagraph].Range.Font.Name == "Courier
New" ||
document.Paragraphs[IndexParagraph].Range.Font.Name == "Consolas")
            return true;
        else
            return false;
    }

```

```

//Ошибки параметров страниц для документа
private void ErrorsPages(Word.Range range)
{
    //Верхнее поле
    float TopMargin = range.PageSetup.TopMargin;
    //Нижнее поле
    float BottomMargin = range.PageSetup.BottomMargin;
    //Левое поле
    float LeftMargin = range.PageSetup.LeftMargin;
    //Правое поле
    float RightMargin = range.PageSetup.RightMargin;
    string ErrorText = string.Empty;
    if (range.PageSetup.Orientation ==
Word.WdOrientation.wdOrientPortrait) //книжная ориентация
    {
        if (TopMargin != 56.7f)
            ErrorText += "\nПри книжной ориентации
страницы верхнее поле должно быть равно 2 см.";
        if (BottomMargin != 56.7f)
            ErrorText += "\nПри книжной ориентации
страницы нижнее поле должно быть равно 2 см.";
        if (LeftMargin != 85.05f)
            ErrorText += "\nПри книжной ориентации
страницы левое поле должно быть равно 3 см.";
        if (RightMargin != 28.35f)
            ErrorText += "\nПри книжной ориентации
страницы правое поле должно быть равно 1 см.";
    }
    else //альбомная ориентация
    {
        if (TopMargin != 85.05f)
            ErrorText += "\nПри альбомной ориентации
страницы верхнее поле должно быть равно 3 см.";
        if (BottomMargin != 28.35f)
            ErrorText += "\nПри альбомной ориентации
страницы нижнее поле должно быть равно 1 см.";
        if (LeftMargin != 56.7f)
            ErrorText += "\nПри альбомной ориентации
страницы левое поле должно быть равно 2 см.";
        if (RightMargin != 56.7f)
            ErrorText += "\nПри альбомной ориентации
страницы правое поле должно быть равно 2 см.";
    }
    if (!string.IsNullOrEmpty(ErrorText))
    {
        object text = "Ошибка настройки параметров
страницы!" + ErrorText;
        ActiveDocumentCommentsAdd(range, text);
    }
}

```

```

//Ошибки параметров абзаца для основного текста
private void ErrorsMainText(int IndexParagraph)
{
    string ErrorText = string.Empty;
    //отступ слева
    float LeftIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LeftIndent;
    if (LeftIndent != 0)
        ErrorText += "\nОтступ слева должен быть равен 0
см.";
    //отступ справа
    float RightIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.RightIndent;
    if (RightIndent != 0)
        ErrorText += "\nОтступ справа должен быть равен 0
см.";
    //отступ красной строки
    float FirstLineIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.FirstLineIndent;
    if (FirstLineIndent != 35.45f)
        ErrorText += "\nОтступ первой/красной строки
должен быть равен 1,25 см.";
    //интервал перед
    float SpaceBefore =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceBefore;
    if (SpaceBefore != 0)
        ErrorText += "\nИнтервал «Перед» должен быть равен
0 пт.";
    //интервал после
    float SpaceAfter =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceAfter;
    if (SpaceAfter != 0)
        ErrorText += "\nИнтервал «После» должен быть равен
0 пт.";
    //междустрочный интервал
    float LineSpacing =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LineSpacing;
    if (LineSpacing != 18f)
        ErrorText += "\nМеждустрочный интервал должен быть
равен 1,5 строки.";
    //выравнивание
    if
(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent != Word.WdParagraphAlignment.wdAlignParagraphJustify)
        ErrorText += "\nВыравнивание у основного текста
должно быть по ширине.";
}

```

```

        //Вывод ошибок в абзаце
        if (!String.IsNullOrEmpty(ErrorText))
        {
            object text = "Ошибка настройки параметров абзаца
у основного текста!" + ErrorText;

ActiveDocumentCommentsAdd(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range, text);
        }
    }

    //Ошибки параметров абзаца для заголовков
    private void ErrorsTitles(int IndexParagraph)
    {
        string ErrorText = string.Empty;
        //размер шрифта текста
        float FontIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.Font.Size;
        if (FontIndent != 14)
            ErrorText += "\nРазмер шрифта должен быть равен
14";
        //шрифт текста
        float FontTypeIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.Font.Name;
        if (FontTypeIndent != "Times New Roman")
            ErrorText += "\nШрифт должен быть равен Times New
Roman";
        //отступ слева
        float LeftIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.LeftIndent;
        if (LeftIndent != 0)
            ErrorText += "\nОтступ слева должен быть равен 0
см.";
        //отступ справа
        float RightIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.RightIndent;
        if (RightIndent != 0)
            ErrorText += "\nОтступ справа должен быть равен 0
см.";
        //интервал перед
        float SpaceBefore =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.SpaceBefore;
        if (SpaceBefore != 10)
            ErrorText += "\nИнтервал «Перед» должен быть равен
10 пт.";
        //интервал после

```

```

        float SpaceAfter =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceAfter;
        if (SpaceAfter != 10)
            ErrorText += "\nИнтервал «После» должен быть равен
10 пт.";
        //междустрочный интервал
        float LineSpacing =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LineSpacing;
        if (LineSpacing != 18f)
            ErrorText += "\nМеждустрочный интервал должен быть
равен 1,5 строки.";
        //выравнивание и отступ красной строки
        float FirstLineIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.FirstLineIndent;
        if
(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent == Word.WdParagraphAlignment.wdAlignParagraphLeft)
        {
            if (FirstLineIndent != 35.45f)
                ErrorText += "\nОтступ первой/красной строки у
нумерованного заголовка должен быть равен 1,25 см.";
        }
        else
        {
            if
(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent == Word.WdParagraphAlignment.wdAlignParagraphCenter)
            {
                if (FirstLineIndent != 0f)
                    ErrorText += "\nОтступ первой/красной
строки у нумерованного заголовка должен быть равен 0 см.";
            }
            else
            {
                ErrorText += "\nНекорректное выравнивание
заголовка!";
            }
        }
        if (!String.IsNullOrEmpty(ErrorText))
        {
            object text = "Ошибка настройки параметров абзаца
у заголовка!" + ErrorText;

ActiveDocumentCommentsAdd(this.Application.ActiveDocument.Paragrap
hs[IndexParagraph].Range, text);
        }
    }

//Ошибки параметров абзаца для рисунков и подрисуночных
надписей

```

```

        private void ErrorsPictures(int IndexParagraph)
        //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        {
            string ErrorText = string.Empty;
            //отступ слева
            float LeftIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LeftIndent;
            if (LeftIndent != 0)
                ErrorText += "\nОтступ слева должен быть равен 0
см.";

            //отступ справа
            float RightIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.RightIndent;
            if (RightIndent != 0)
                ErrorText += "\nОтступ справа должен быть равен 0
см.";

            //отступ красной строки
            float FirstLineIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.FirstLineIndent;
            if (FirstLineIndent != 0)
                ErrorText += "\nОтступ первой/красной строки
должен быть равен 0 см.";
            //интервал перед
            float SpaceBefore =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceBefore;
            if (SpaceBefore != 0)
                ErrorText += "\nИнтервал «Перед» должен быть равен
0 пт.";

            //интервал после
            float SpaceAfter =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceAfter;
            if (SpaceAfter != 10)
                ErrorText += "\nИнтервал «После» должен быть равен
10 пт.";

            //междустрочный интервал
            float LineSpacing =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LineSpacing;
            if (LineSpacing != 18f)
                ErrorText += "\nМеждустрочный интервал должен быть
равен 1,5 строки.";
            //выравнивание
            if
(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent != Word.WdParagraphAlignment.wdAlignParagraphCenter)
                ErrorText += "\nВыравнивание у рисунков и
подрисуночных надписей должно быть по центру.";
            if (!String.IsNullOrEmpty(ErrorText))

```

```

        {
            object text = "Ошибка настройки параметров абзаца
у рисунка / подрисуночной надписи!" + ErrorText;

ActiveDocumentCommentsAdd(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range, text);
        }
    }

    //Ошибки параметров абзаца названий таблиц
    private void ErrorsTableName(int IndexParagraph)
    {
        string ErrorText = string.Empty;
        //отступ слева
        float LeftIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.LeftIndent;
        if (LeftIndent != 0)
            ErrorText += "\nОтступ слева должен быть равен 0
см.";
        //отступ справа
        float RightIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.RightIndent;
        if (RightIndent != 0)
            ErrorText += "\nОтступ справа должен быть равен 0
см.";
        //отступ красной строки
        float FirstLineIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.FirstLineIndent;
        if (FirstLineIndent != 0)
            ErrorText += "\nОтступ первой/красной строки
должен быть равен 0 см.";
        //интервал перед
        float SpaceBefore =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.SpaceBefore;
        if (SpaceBefore != 10)
            ErrorText += "\nИнтервал «Перед» должен быть равен
10 пт.";
        //интервал после
        float SpaceAfter =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.SpaceAfter;
        if (SpaceAfter != 0)
            ErrorText += "\nИнтервал «После» должен быть равен
0 пт.";
        //междустрочный интервал
        float LineSpacing =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.LineSpacing;
        if (LineSpacing != 18f)

```

```

        ErrorText += "\nМеждустрочный интервал должен быть
равен 1,5 строки.";
        //выравнивание
        if
        (this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent != Word.WdParagraphAlignment.wdAlignParagraphLeft)
            ErrorText += "\nВыравнивание у названий таблиц
должно быть по левому краю.";
        if (!String.IsNullOrEmpty(ErrorText))
        {
            object text = "Ошибка настройки параметров абзаца
у названия таблицы!" + ErrorText;

ActiveDocumentCommentsAdd(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range, text);
        }
    }

//Ошибки кавычек в тексте параграфа
private void ErrorsQuotationMarks(Word.Range range)
{
    List<Word.Range> Find = FindAll(range, "^34");
    if (Find.Count != 0)
    {
        for (int j = 0; j < Find.Count; j++)
        {
            if (Find[j].Font.Name == "Times New Roman")
            {
                object text = "ВНИМАНИЕ!\nВ тексте работы
должны применяться только французские кавычки (елочки) - «...».";
                ActiveDocumentCommentsAdd(Find[j], text);
            }
        }
    }
}

//Ошибки кавычек в тексте параграфа
private void ErrorsQuotationMarks1(Word.Range range)
{
    List<Word.Range> Find = FindAll(range, "^0147");
    if (Find.Count != 0)
    {
        for (int j = 0; j < Find.Count; j++)
        {
            if (Find[j].Font.Name == "Times New Roman")
            {
                object text = "ВНИМАНИЕ!\nВ тексте работы
должны применяться только французские кавычки (елочки) - «...».";
                ActiveDocumentCommentsAdd(Find[j], text);
            }
        }
    }
}

```

```

private void ErrorsQuotationMarks2(Word.Range range)
{
    List<Word.Range> Find = FindAll(range, "^0148");
    if (Find.Count != 0)
    {
        for (int j = 0; j < Find.Count; j++)
        {
            if (Find[j].Font.Name == "Times New Roman")
            {
                object text = "ВНИМАНИЕ!\nВ тексте работы  
должны применяться только французские кавычки (елочки) - «...».";
                ActiveDocumentCommentsAdd(Find[j], text);
            }
        }
    }
}

//Ошибки маркерowanych списков в тексте параграфа
private void ErrorsBulletedlist(int IndexParagraph)
{
    string ErrorText = string.Empty;
    //отступ слева
    var document = this.Application.ActiveDocument;
    if
(document.Paragraphs[IndexParagraph].Range.Characters.Count >= 2)
    {
        if
(document.Paragraphs[IndexParagraph].Range.Characters.Last.Text ==
";" ||

document.Paragraphs[IndexParagraph].Range.Characters.Last.Text ==
"." &&

document.Characters[document.Paragraphs[IndexParagraph].Range.Char
acters.Count - 2].Text == "\r")
            ErrorText += "\nКонец абзаца";
        }
        float LeftIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LeftIndent;
        if (LeftIndent != 0)
            ErrorText += "\nОтступ слева должен быть равен 0
см.";
        //отступ справа
        float RightIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.RightIndent;
        if (RightIndent != 0)
            ErrorText += "\nОтступ справа должен быть равен 0
см.";
        //отступ красной строки

```

```

        float FirstLineIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.FirstLineIndent;
        if (FirstLineIndent != 35.45f)
            ErrorText += "\nОтступ первой/красной строки
должен быть равен 1,25 см.";
        //интервал перед
        float SpaceBefore =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceBefore;
        if (SpaceBefore != 0)
            ErrorText += "\nИнтервал «Перед» должен быть равен
0 пт.";
        //интервал после
        float SpaceAfter =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceAfter;
        if (SpaceAfter != 0)
            ErrorText += "\nИнтервал «После» должен быть равен
0 пт.";
        //междустрочный интервал
        float LineSpacing =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LineSpacing;
        if (LineSpacing != 18f)
            ErrorText += "\nМеждустрочный интервал должен быть
равен 1,5 строки.";
        //выравнивание
        if
(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent != Word.WdParagraphAlignment.wdAlignParagraphJustify)
            ErrorText += "\nВыравнивание у маркерowanego
списка должно быть по ширине.";
        if (!String.IsNullOrEmpty(ErrorText))
        {
            object text = "Ошибка настройки параметров абзаца
у маркерowanego списка!" + ErrorText;

ActiveDocumentCommentsAdd(this.Application.ActiveDocument.Paragrap
hs[IndexParagraph].Range, text);
        }
    }

    //Ошибки нумерованных списков в тексте параграфа
    private void ErrorsNumberedlist(int IndexParagraph)
    {
        string ErrorText = string.Empty;

        //отступ слева
        float LeftIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LeftIndent;

```

```

        if (LeftIndent != 0)
            ErrorText += "\nОтступ слева должен быть равен 0
см.";

        //отступ справа
        float RightIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.RightIndent;
        if (RightIndent != 0)
            ErrorText += "\nОтступ справа должен быть равен 0
см.";

        //отступ красной строки
        float FirstLineIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.FirstLineIndent;
        if (FirstLineIndent != 35.45f)
            ErrorText += "\nОтступ первой/красной строки
должен быть равен 1,25 см.";
        //интервал перед
        float SpaceBefore =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceBefore;
        if (SpaceBefore != 0)
            ErrorText += "\nИнтервал «Перед» должен быть равен
0 пт.";

        //интервал после
        float SpaceAfter =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceAfter;
        if (SpaceAfter != 0)
            ErrorText += "\nИнтервал «После» должен быть равен
0 пт.";

        //междустрочный интервал
        float LineSpacing =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LineSpacing;
        if (LineSpacing != 18f)
            ErrorText += "\nМеждустрочный интервал должен быть
равен 1,5 строки.";
        //выравнивание
        if
(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent != Word.WdParagraphAlignment.wdAlignParagraphJustify)
            ErrorText += "\nВыравнивание у нумерованного
списка должно быть по ширине.";
        if (!String.IsNullOrEmpty(ErrorText))
        {
            object text = "Ошибка настройки параметров абзаца
у нумерованного списка!" + ErrorText;

ActiveDocumentCommentsAdd(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range, text);
        }
    }
}

```

```

//Ошибки параметров абзаца названий листинга
private void ErrorsListingName(int IndexParagraph)
{
    string ErrorText = string.Empty;
    //отступ слева
    float LeftIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LeftIndent;
    if (LeftIndent != 0)
        ErrorText += "\nОтступ слева должен быть равен 0
см.";
    //отступ справа
    float RightIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.RightIndent;
    if (RightIndent != 0)
        ErrorText += "\nОтступ справа должен быть равен 0
см.";
    //отступ красной строки
    float FirstLineIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.FirstLineIndent;
    if (FirstLineIndent != 0)
        ErrorText += "\nОтступ первой/красной строки
должен быть равен 0 см.";
    //интервал перед
    float SpaceBefore =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceBefore;
    if (SpaceBefore != 10)
        ErrorText += "\nИнтервал «Перед» должен быть равен
10 пт.";
    //интервал после
    float SpaceAfter =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.SpaceAfter;
    if (SpaceAfter != 15)
        ErrorText += "\nИнтервал «После» должен быть равен
15 пт.";
    //междустрочный интервал
    float LineSpacing =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.P
aragraphFormat.LineSpacing;
    if (LineSpacing != 18f)
        ErrorText += "\nМеждустрочный интервал должен быть
равен 1,5 строки.";
    //выравнивание
    if
(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent != Word.WdParagraphAlignment.wdAlignParagraphLeft)
        ErrorText += "\nВыравнивание у названия листинга
должно быть по левому краю.";
}

```

```

        if (!String.IsNullOrEmpty(ErrorText))
        {
            object text = "Ошибка настройки параметров абзаца
у названия листинга!" + ErrorText;

ActiveDocumentCommentsAdd(this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range, text);
        }
    }

    //Ошибки параметров абзаца фрагмента кода
    private void ErrorsCodeListing(int IndexParagraph)
    {
        string ErrorText = string.Empty;
        //отступ слева
        float LeftIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.LeftIndent;
        if (LeftIndent != 0)
            ErrorText += "\nОтступ слева должен быть равен 0
см.";
        //отступ справа
        float RightIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.RightIndent;
        if (RightIndent != 0)
            ErrorText += "\nОтступ справа должен быть равен 0
см.";
        //отступ красной строки
        float FirstLineIndent =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.FirstLineIndent;
        if (FirstLineIndent != 0)
            ErrorText += "\nОтступ первой/красной строки
должен быть равен 0 см.";
        //интервал перед
        float SpaceBefore =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.SpaceBefore;
        if (SpaceBefore != 0)
            ErrorText += "\nИнтервал «Перед» должен быть равен
0 пт.";
        //интервал после
        float SpaceAfter =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.SpaceAfter;
        if (SpaceAfter != 0)
            ErrorText += "\nИнтервал «После» должен быть равен
0 пт.";
        //междустрочный интервал
        float LineSpacing =
this.Application.ActiveDocument.Paragraphs[IndexParagraph].Range.ParagraphFormat.LineSpacing;

```

```

        if (LineSpacing != 12f)
            ErrorText += "\nМеждустрочный интервал должен быть
одинарный.";
        //выравнивание
        if
        (this.Application.ActiveDocument.Paragraphs[IndexParagraph].Alignm
ent != Word.WdParagraphAlignment.wdAlignParagraphLeft)
            ErrorText += "\nВыравнивание у фрагмента кода
должно быть по левому краю.";
        if (!String.IsNullOrEmpty(ErrorText))
        {
            object text = "Ошибка настройки параметров абзаца
у фрагмента кода!" + ErrorText;

ActiveDocumentCommentsAdd(this.Application.ActiveDocument.Paragraph
hs[IndexParagraph].Range, text);
        }
    }
}

```

```

//Проверка параметров страниц
public void PageAnalysis()
{
    Ribbon.labelUpdate("1 / 3", "Выполняется: проверка
параметров страниц.");
    var document = this.Application.ActiveDocument;
    Word.Range rangeStart =
document.Application.Selection.GoTo(Word.WdGoToItem.wdGoToPage,
Word.WdGoToDirection.wdGoToAbsolute, 1);
    float TopMargin = rangeStart.PageSetup.TopMargin;
    float BottomMargin =
rangeStart.PageSetup.BottomMargin;
    float LeftMargin = rangeStart.PageSetup.LeftMargin;
    float RightMargin = rangeStart.PageSetup.RightMargin;
    ErrorsPages(rangeStart);
    //задаем начальные значения для шкалы прогресса
    Ribbon.progressBarSettings(1,
document.Paragraphs.Count, 1, 1);
    //постраничный перебор документа
    int Page = 0;
    for (int i = 2; i <= document.Paragraphs.Count; i++)
    {
        //обработка запроса на отмену проверки
        if (CancelAnalysis)
            break;
        //обновляем прогресс
        Ribbon.progressBarUpdate();

        Word.Range range =
document.Range(document.Paragraphs[i].Range.End - 1,
document.Paragraphs[i].Range.End);
    }
}

```

```

        if
        (range.Information[Word.WdInformation.wdActiveEndPageNumber] ==
        Page)
        {
            continue;
        }
        else
        {

            float currentTopMargin =
range.PageSetup.TopMargin;
            float currentBottomMargin =
range.PageSetup.BottomMargin;
            float currentLeftMargin =
range.PageSetup.LeftMargin;
            float currentRightMargin =
range.PageSetup.RightMargin;
            //если значения полей изменились обрабатываем
страницу
            if (TopMargin != currentTopMargin ||
                BottomMargin != currentBottomMargin ||
                LeftMargin != currentLeftMargin ||
                RightMargin != currentRightMargin)
            {
                ErrorsPages(range);
                TopMargin = currentTopMargin;
                BottomMargin = currentBottomMargin;
                LeftMargin = currentLeftMargin;
                RightMargin = currentRightMargin;
            }
        }
    }
}

//Проверка параметров абзацев
public void ParagraphsAnalysis()
{
    Ribbon.labelUpdate("2 / 3", "Выполняется: проверка
параметров абзацев.");
    var document = this.Application.ActiveDocument;
    //задаем начальные значения для шкалы прогресса
    Ribbon.progressBarSettings(1,
document.Paragraphs.Count, 1, 1);
    for (var i = 1; i <= document.Paragraphs.Count; ++i)
    {
        //обработка запроса на отмену проверки
        if (CancelAnalysis)
            break;
        //обновляем прогресс
        Ribbon.progressBarUpdate();
        //Пустой абзац
        if (document.Paragraphs[i].Range.End -
document.Paragraphs[i].Range.Start <= 1)

```

```

{
    continue;
}
//пустая строка заканчивающаяся разрывом страницы
if (document.Paragraphs[i].Range.Text == "\f\r")
{
    continue;
}
//Параграф является частью таблицы
if (IsTableTest(document.Paragraphs[i].Range))
{
    continue;
}

if (IsTitleTest(i)) //является заголовком
{
    ErrorsTitles(i);
}
else //является основным текстом
{
    if (IsPictureTest(i)) //является
рисунком/подписуночной надписью
    {
        ErrorsPictures(i);
    }
    else
    {
        if (IsTableNameTest(i)) //является
заголовком таблицы
        {
            ErrorsTableName(i);
        }
        else
        {
            if (IsBulletedlist(i)) //является
маркерованным списком
            {
                ErrorsBulletedlist(i);
            }
            else
            {
                if (IsNumberedlist(i)) //является
нумерованным списком
                {
                    ErrorsNumberedlist(i);
                }
                else
                {
                    if (IsCodeListing(i))
//является листингом
                    {
                        ErrorsCodeListing(i);
                    }
                }
            }
        }
    }
}

```

```

else
{
    if (IsListingNameTest(i))
    {
        ErrorsListingName(i);
    }
    else
    {
        ErrorsMainText(i);
    }
}
}
}
}
}
}
}
}

//Проверка на типовые ошибки
public void TextAnalysis()
{
    Ribbon.labelUpdate("3 / 3", "Выполняется: проверка
текста работы.");
    var document = this.Application.ActiveDocument;
    TypicalErrors typicalErrors = new TypicalErrors();
    //задаем начальные значения для шкалы прогресса
    Ribbon.progressBarSettings(1,
document.Paragraphs.Count, 1, 1);
    for (int i = 1; i <= document.Paragraphs.Count; i++)
    {
        //обработка запроса на отмену проверки
        if (CancelAnalysis)
            break;
        //обновляем прогресс
        Ribbon.progressBarUpdate();
        Word.Range rng = document.Paragraphs[i].Range;

        //Проверка на ошибки из List
        typicalErrors.Templates
        for (int t = 0; t < typicalErrors.Templates.Count;
t++)
        {
            List<Word.Range> Find = FindAll(rng,
typicalErrors.Templates[t].SearchTemplate);
            for (int j = 0; j < Find.Count; j++)
            {
                ActiveDocumentCommentsAdd(Find[j],
typicalErrors.Templates[t].ErrorDescription);
            }
        }
    }
}

```

```

        ErrorsQuotationMarks(rng);
        ErrorsQuotationMarks1(rng);
        ErrorsQuotationMarks2(rng);
    }
}

#region Код, автоматически созданный VSTO

/// <summary>
/// Требуемый метод для поддержки конструктора — не
изменяйте
/// содержимое этого метода с помощью редактора кода.
/// </summary>
private void InternalStartup()
{
    this.Startup += new
System.EventHandler(ThisAddIn_Startup);
    this.Shutdown += new
System.EventHandler(ThisAddIn_Shutdown);
}

#endregion
}
}

```

Листинг A.2 – Файл ProgressForm.cs

```

namespace NormalInspector
{
    partial class ProgressForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components =
null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources
should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

```

```

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        System.ComponentModel.ComponentResourceManager
resources = new
System.ComponentModel.ComponentResourceManager(typeof(ProgressForm
));
        this.progressBar = new
System.Windows.Forms.ProgressBar();
        this.labelOperationNumber = new
System.Windows.Forms.Label();
        this.labelCurrentOperation = new
System.Windows.Forms.Label();
        this.buttonCancelAnalysis = new
System.Windows.Forms.Button();
        this.SuspendLayout();
        //
        // progressBar
        //
        this.progressBar.Location = new
System.Drawing.Point(12, 63);
        this.progressBar.Name = "progressBar";
        this.progressBar.Size = new System.Drawing.Size(354,
23);
        this.progressBar.TabIndex = 0;
        //
        // labelOperationNumber
        //
        this.labelOperationNumber.AutoSize = true;
        this.labelOperationNumber.Font = new
System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
        this.labelOperationNumber.Location = new
System.Drawing.Point(375, 63);
        this.labelOperationNumber.Name =
"labelOperationNumber";
        this.labelOperationNumber.Size = new
System.Drawing.Size(39, 20);
        this.labelOperationNumber.TabIndex = 1;
        this.labelOperationNumber.Text = "0 / 0";
        //
        // labelCurrentOperation
        //
        this.labelCurrentOperation.AutoSize = true;
        this.labelCurrentOperation.Font = new
System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));

```

```

        this.labelCurrentOperation.Location = new
System.Drawing.Point(8, 110);
        this.labelCurrentOperation.Name =
"labelCurrentOperation";
        this.labelCurrentOperation.Size = new
System.Drawing.Size(116, 20);
        this.labelCurrentOperation.TabIndex = 2;
        this.labelCurrentOperation.Text = "Выполняется:";
        //
        // buttonCancelAnalysis
        //
        this.buttonCancelAnalysis.Location = new
System.Drawing.Point(12, 148);
        this.buttonCancelAnalysis.Name =
"buttonCancelAnalysis";
        this.buttonCancelAnalysis.Size = new
System.Drawing.Size(75, 23);
        this.buttonCancelAnalysis.TabIndex = 3;
        this.buttonCancelAnalysis.Text = "Прервать";
        this.buttonCancelAnalysis.UseVisualStyleBackColor =
true;
        this.buttonCancelAnalysis.Click += new
System.EventHandler(this.buttonCancelAnalysis_Click);
        //
        // ProgressForm
        //
        this.AutoScaleDimensions = new
System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
        this.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink;
        this.ClientSize = new System.Drawing.Size(424, 195);
        this.Controls.Add(this.buttonCancelAnalysis);
        this.Controls.Add(this.labelCurrentOperation);
        this.Controls.Add(this.labelOperationNumber);
        this.Controls.Add(this.progressBar);
        this.Icon =
((System.Drawing.Icon) (resources.GetObject("$this.Icon")));
        this.MaximizeBox = false;
        this.MinimizeBox = false;
        this.Name = "ProgressForm";
        this.Text = "Проверка нормоконтроля";
        this.ResumeLayout(false);
        this.PerformLayout();

    }

#endregion

public System.Windows.Forms.ProgressBar progressBar;
public System.Windows.Forms.Label labelOperationNumber;
public System.Windows.Forms.Label labelCurrentOperation;

```

```

        public System.Windows.Forms.Button buttonCancelAnalysis;
    }

```

Листинг А.3 – Файл ProgressForm.cs

```

namespace NormalInspector
{
    partial class Ribbon :
Microsoft.Office.Tools.Ribbon.RibbonBase
    {
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components =
null;

        public Ribbon()
            : base(Globals.Factory.GetRibbonFactory())
        {
            InitializeComponent();
        }

        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        /// <param name="disposing">истинно, если управляемый
ресурс должен быть удален; иначе ложно.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Код, автоматически созданный конструктором
компонентов

        /// <summary>
        /// Требуемый метод для поддержки конструктора — не
изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager
resources = new
System.ComponentModel.ComponentResourceManager(typeof(Ribbon));
            this.tab1 = this.Factory.CreateRibbonTab();
            this.group1 = this.Factory.CreateRibbonGroup();
            this.buttonAnalysis =
this.Factory.CreateRibbonButton();

```

```

        this.buttonClear = this.Factory.CreateRibbonButton();
        this.tab1.SuspendLayout();
        this.group1.SuspendLayout();
        this.SuspendLayout();
        //
        // tab1
        //
        this.tab1.ControlId.ControlIdType =
Microsoft.Office.Tools.Ribbon.RibbonControlIdType.Office;
        this.tab1.Groups.Add(this.group1);
        this.tab1.Label = "Нормоконтроль";
        this.tab1.Name = "tab1";
        //
        // group1
        //
        this.group1.Items.Add(this.buttonAnalysis);
        this.group1.Items.Add(this.buttonClear);
        this.group1.Label = "Нормоконтроль";
        this.group1.Name = "group1";
        //
        // buttonAnalysis
        //
        this.buttonAnalysis.ControlSize =
Microsoft.Office.Core.RibbonControlSize.RibbonControlSizeLarge;
        this.buttonAnalysis.Image =
((System.Drawing.Image) (resources.GetObject("buttonAnalysis.Image"
))) );
        this.buttonAnalysis.Label = "Проверить";
        this.buttonAnalysis.Name = "buttonAnalysis";
        this.buttonAnalysis.ShowImage = true;
        this.buttonAnalysis.Click += new
Microsoft.Office.Tools.Ribbon.RibbonControlEventHandler(this.butto
n1_Click);
        //
        // buttonClear
        //
        this.buttonClear.ControlSize =
Microsoft.Office.Core.RibbonControlSize.RibbonControlSizeLarge;
        this.buttonClear.Image =
((System.Drawing.Image) (resources.GetObject("buttonClear.Image")))
;
        this.buttonClear.Label = "Очистить";
        this.buttonClear.Name = "buttonClear";
        this.buttonClear.ShowImage = true;
        this.buttonClear.Click += new
Microsoft.Office.Tools.Ribbon.RibbonControlEventHandler(this.butto
nClear_Click);
        //
        // Ribbon
        //
        this.Name = "Ribbon";
        this.RibbonType = "Microsoft.Word.Document";
        this.Tabs.Add(this.tab1);

```

```

        this.Load += new
Microsoft.Office.Tools.Ribbon.RibbonUIEventHandler(this.Ribbon_Load);

        this.tab1.ResumeLayout(false);
        this.tab1.PerformLayout();
        this.group1.ResumeLayout(false);
        this.group1.PerformLayout();
        this.ResumeLayout(false);

    }

    #endregion

    internal Microsoft.Office.Tools.Ribbon.RibbonTab tab1;
    internal Microsoft.Office.Tools.Ribbon.RibbonGroup group1;
    internal Microsoft.Office.Tools.Ribbon.RibbonButton
buttonAnalysis;
    internal Microsoft.Office.Tools.Ribbon.RibbonButton
buttonClear;
}

partial class ThisRibbonCollection
{
    internal Ribbon Ribbon
    {
        get { return this.GetRibbon<Ribbon>(); }
    }
}
}

```

Листинг А.4 – Файл ProgressForm.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace NormalInspector
{
    struct ErrorPattern // структура для хранения шаблона ошибки
    {
        public string SearchTemplate; //шаблон поиска
        public string ErrorDescription; //описание ошибки

        public ErrorPattern(string _SearchTemplate, string
_ErrorDescription)
        {
            SearchTemplate = _SearchTemplate;
            ErrorDescription = _ErrorDescription;
        }
    }
}

```

```

class TypicalErrors
{
    public List<ErrorPattern> Templates;

    public TypicalErrors()
    {
        Templates = new List<ErrorPattern>();

        Templates.Add(new ErrorPattern(
            "—", "ВНИМАНИЕ!\nВ работе использован недопустимый
символ! На его месте должно стоять обычное тире (-)."));
        Templates.Add(new ErrorPattern(
            " - ", "ВНИМАНИЕ!\nВ данном случае будет более
корректно заменить дефис на тире (Ctrl + Num-)."));

        Templates.Add(new ErrorPattern(
            "т.д.", "ВНИМАНИЕ!\nДанное сокращение должно
писаться через неразрывный пробел (Ctrl + Shift + Пробел)."));
        Templates.Add(new ErrorPattern(
            "т.п.", "ВНИМАНИЕ!\nДанное сокращение должно
писаться через неразрывный пробел (Ctrl + Shift + Пробел)."));
        Templates.Add(new ErrorPattern(
            "т.е.", "ВНИМАНИЕ!\nДанное сокращение должно
писаться через неразрывный пробел (Ctrl + Shift + Пробел)."));
        Templates.Add(new ErrorPattern(
            "т.к.", "ВНИМАНИЕ!\nДанное сокращение должно
писаться через неразрывный пробел (Ctrl + Shift + Пробел)."));
    }
}

```

Приложение Б
РЕЗУЛЬТАТЫ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ
(рекомендуемое)

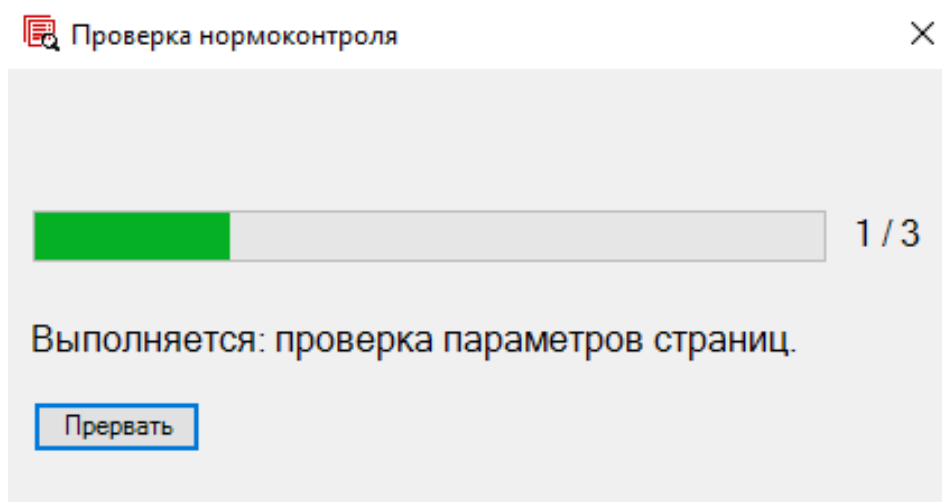


Рисунок Б.1 – Результат тестирования №1

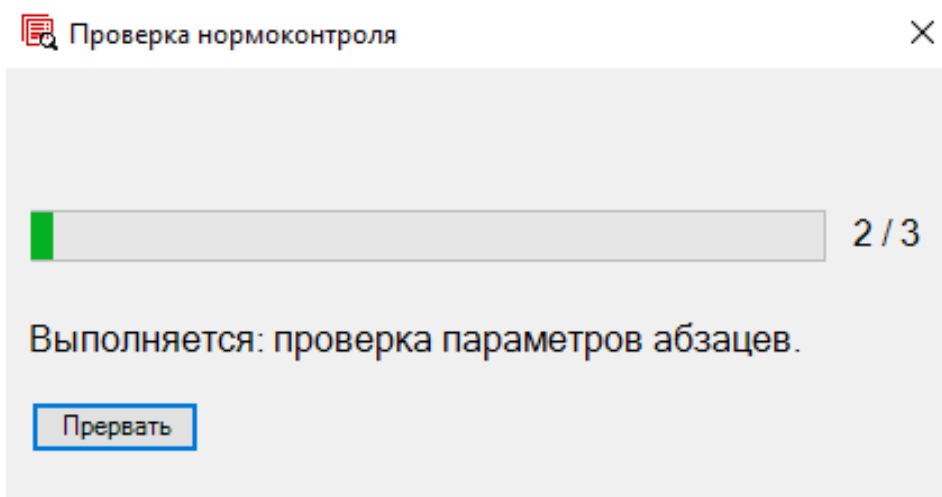


Рисунок Б.2 – Результат тестирования №2

Проектирование интерфейса — это задача, основанная на разработке визуального представления и структуры интерактивных элементов продукта (например, веб-сайта, мобильного приложения, программного обеспечения), с целью обеспечения максимального удобства и эффективности взаимодействия пользователя с продуктом.¶

При «проверке» документов на соответствие ГОСТам присутствуют следующие шаги и этапы:¶

- → идентификация применимых ГОСТов;¶
- → получение актуальных версий ГОСТов;¶
- → анализ требований ГОСТов;¶
- → сопоставление требований и фактического содержания документов;¶

dmitry
ВНИМАНИЕ!¶
В работе использован недопустимый символ! На его месте должно стоять обычное тире (-).¶

dmitry
ВНИМАНИЕ!¶
В тексте работы должны применяться только французские кавычки (ёлочки) — «...».¶

Рисунок Б.3 – Результат тестирования №2

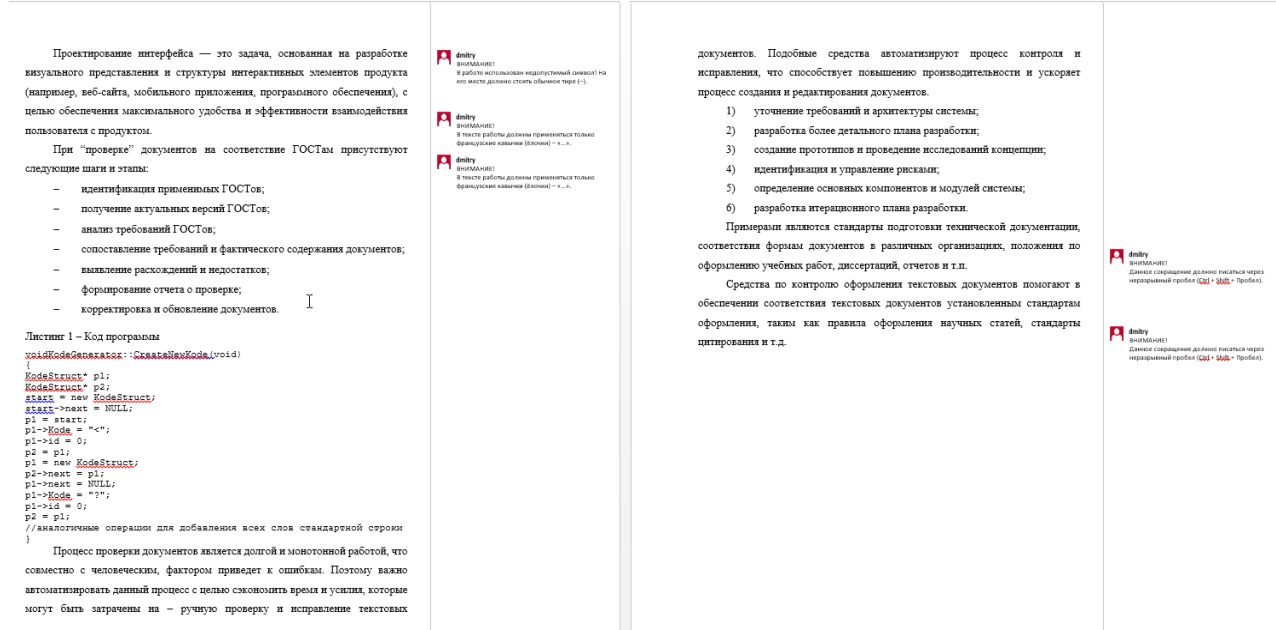


Рисунок Б.4 – Результат тестирования №3

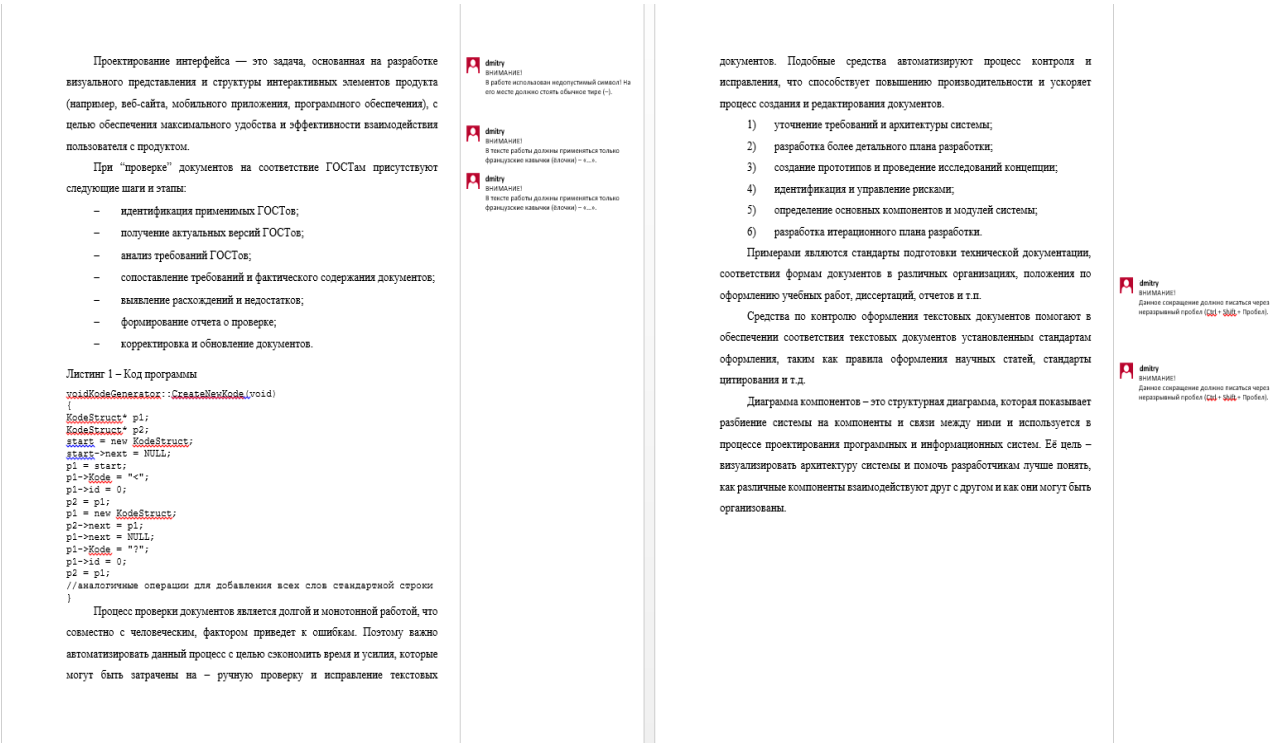


Рисунок Б.5 – Результат тестирования №4

Проектирование интерфейса — это задача, основанная на разработке визуального представления и структуры интерактивных элементов продукта (например, веб-сайта, мобильного приложения, программного обеспечения), с целью обеспечения максимального удобства и эффективности взаимодействия пользователя с продуктом.

При “проверке” документов на соответствие ГОСТам присутствуют следующие шаги и этапы:

- идентификация применимых ГОСТов;
- получение актуальных версий ГОСТов;
- анализ требований ГОСТов;
- сопоставление требований и фактического содержания документов;
- выявление расхождений и недостатков;
- формирование отчета о проверке;
- корректировка и обновление документов.

Листинг 1 – Код программы

```
void KodeGenerator::CreateNewKode(void)
{
    KodeStruct* p1;
    KodeStruct* p2;
    stack = new KodeStruct;
    stack->next = NULL;
    p1 = start;
    p1->Kode = "<";
    p1->id = 0;
    p2 = p1;
    p1 = new KodeStruct;
    p2->next = p1;
    p1->next = NULL;
    p1->Kode = "?";
    p1->id = 0;
    p2 = p1;
    //аналогичные операции для добавления всех слов стандартной строки
}
```

Процесс проверки документов является долгой и монотонной работой, что совместно с человеческим фактором приведет к ошибкам. Поэтому важно автоматизировать данный процесс с целью сэкономить время и усилия, которые могут быть затрачены на – ручную проверку и исправление текстовых

документов. Подобные средства автоматизируют процесс контроля и исправления, что способствует повышению производительности и ускоряет процесс создания и редактирования документов.

- 1) уточнение требований и архитектуры системы;
- 2) разработка более детального плана разработки;
- 3) создание прототипов и проведение исследований концепции;
- 4) идентификация и управление рисками;
- 5) определение основных компонентов и модулей системы;
- 6) разработка итерационного плана разработки.

Примерами являются стандарты подготовки технической документации, соответствия формам документов в различных организациях, положения по оформлению учебных работ, диссертаций, отчетов и т.п.

Средства по контролю оформления текстовых документов помогают в обеспечении соответствия текстовых документов установленным стандартам оформления, таким как правила оформления научных статей, стандарты цитирования и т.д.

Диаграмма компонентов – это структурная диаграмма, которая показывает разбиение системы на компоненты и связи между ними и используется в процессе проектирования программных и информационных систем. Её цель – визуализировать архитектуру системы и помочь разработчикам лучше понять, как различные компоненты взаимодействуют друг с другом и как они могут быть организованы.

Рисунок Б.6 – Результат тестирования №4

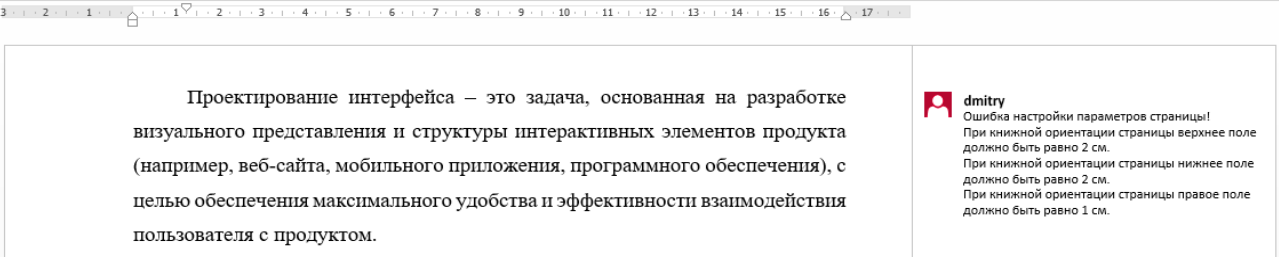


Рисунок Б.7 – Результат тестирования №5

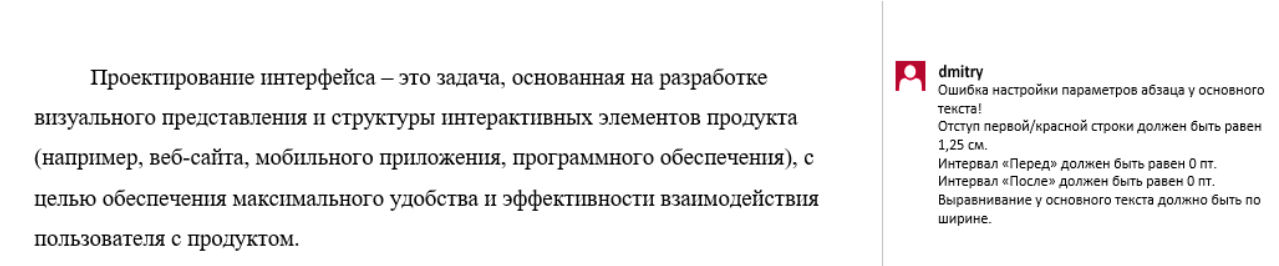


Рисунок Б.8 – Результат тестирования №6

1.1 Анализ функциональных требований

В ходе изучения предметной области и аналогов разрабатываемого программного продукта были определены варианты использования и сформирована диаграмма (рисунок 1).

dmitry
Ошибка настройки параметров абзаца у заголовка!
Интервал «Перед» должен быть равен 10 пт.
Интервал «После» должен быть равен 10 пт.
Междустрочный интервал должен быть равен 1,5 строки.
Некорректное выравнивание заголовка!

Рисунок Б.9 – Результат тестирования №7

Таблица 10 – Результаты функционального тестирования

№	Описание теста	Тест и параметры	Результат
1	Запустить проверку на <u>нормоконтроль</u>	Нажата кнопка «Проверить»	Тест пройден. Проверка запущена и открыто окно прогресса проверки (рисунок Б.1)
2	Прервать проверку	Нажата кнопка	Тест пройден. Проверка

dmitry Неск. сек. назад
Ошибка настройки параметров абзаца у названия таблицы!
Отступ первой/красной строки должен быть равен 0 см.
Интервал «После» должен быть равен 0 пт.

Рисунок Б.10 – Результат тестирования №8

Интерфейс приложения представлен на рисунке 2.

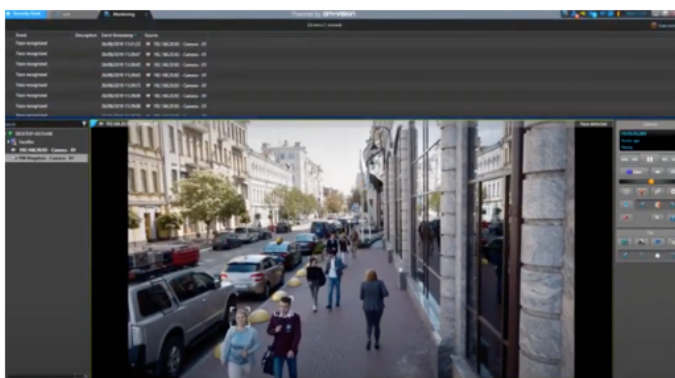


Рисунок 2 – Интерфейс приложения «AnyVision»

dmitry
Ошибка настройки параметров абзаца у рисунка / подрисуночной надписи!
Интервал «После» должен быть равен 10 пт.

dmitry
Ошибка настройки параметров абзаца у рисунка / подрисуночной надписи!
Интервал «После» должен быть равен 10 пт.

Рисунок Б.11 – Результат тестирования №9

- 1) уточнение требований и архитектуры системы;
- 2) разработка более детального плана разработки;
- 3) создание прототипов и проведение исследований концепции;
- 4) идентификация и управление рисками;
- 5) определение основных компонентов и модулей системы;
- 6) разработка итерационного плана разработки.

dmitry
Ошибка настройки параметров абзацев у нумерованного списка!
Отступ первой/красной строки должен быть равен 1,25 см.
Интервал «Перед» должен быть равен 0 пт.
Интервал «После» должен быть равен 0 пт.

Рисунок Б.12 – Результат тестирования №10

В результате анализа программных продуктов по автоматизации контроля доступа были определены следующие наиболее близкие решения:

- SecurOS FaceX от ISS;
- AnyVision;
- Face-Intellect от AxxonSoft;
- Vocord FaceControl.

Эти решения получают высокие оценки от пользователей и пользуются большой популярностью.



dmitry

Ошибка настройки параметров абзацев у маркерowanego списка!
Отступ первой/красной строки должен быть равен 1,25 см.
Интервал «После» должен быть равен 0 пт.
Выравнивание у маркерowanego списка должно быть по ширине.

Рисунок Б.13 – Результат тестирования №11

Листинг 1 – Код программы

```
void KodeGenerator::CreateNewKode(void)
{
    KodeStruct* p1;
    KodeStruct* p2;
    start = new KodeStruct;
    start->next = NULL;
    p1 = start;
    p1->Kode = "<";
    p1->id = 0;
    p2 = p1;
    p1 = new KodeStruct;
    p2->next = p1;
    p1->next = NULL;
    p1->Kode = "?";
    p1->id = 0;
    p2 = p1;
}
```



dmitry

Ошибка настройки параметров абзаца названий фрагмента кода!
Отступ первой/красной строки должен быть равен 0 см.
Интервал «Перед» должен быть равен 10 пт.
Выравнивание у названий фрагмента кода должно быть по левому краю.

Рисунок Б.14 – Результат тестирования №12

```
void KodeGenerator::CreateNewKode(void)
{
    KodeStruct* p1;
    KodeStruct* p2;
    start = new KodeStruct;
    start->next = NULL;
    p1 = start;
    p1->Kode = "<";
    p1->id = 0;
    p2 = p1;
    p1 = new KodeStruct;
    p2->next = p1;
    p1->next = NULL;
    p1->Kode = "?";
    p1->id = 0;
    p2 = p1;
    //аналогичные операции для добавления всех слов стандартной строки
}
```



dmitry

Ошибка настройки параметров абзацев у фрагментов кода!
Отступ первой/красной строки должен быть равен 0 см.

Рисунок Б.15 – Результат тестирования №13

Проектирование интерфейса — это задача, основанная на разработке визуального представления и структуры интерактивных элементов продукта (например, веб-сайта, мобильного приложения, программного обеспечения), с целью обеспечения максимального удобства и эффективности взаимодействия пользователя с продуктом.

При “проверке” документов на соответствие ГОСТам присутствуют следующие шаги и этапы:

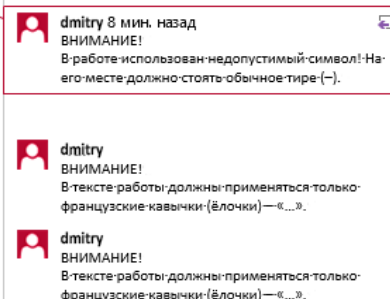


Рисунок Б.16 – Результат тестирования №14

Примерами являются стандарты подготовки технической документации, соответствия формам документов в различных организациях, положения по оформлению учебных работ, диссертаций, отчетов и т.п.

Средства по контролю оформления текстовых документов помогают в обеспечении соответствия текстовых документов установленным стандартам оформления, таким как правила оформления научных статей, стандарты цитирования и т.д.¶

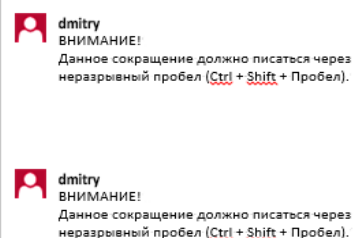


Рисунок Б.17 – Результат тестирования №15