

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

<i>Факультет</i>	<i>Кафедра</i>
вычислительной техники	МОиПЭВМ
Направление	09.03.02 «Информационные системы и технологии»
подготовки	
Направленность	Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программные средства управления графическими объектами в
текстовых документах**

Студент _____ Соколов А.В.
(подпись, дата) (ФИО полностью)

Руководитель _____
(подпись, дата) (фамилия, инициалы)

Нормоконтролер _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от
_____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от
_____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2025

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
«20» декабря 2024 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Соколов Артем Вячеславович гр. 21ВИ1 факультета ВТ
(Фамилия, имя, отчество полностью)

2. Тема работы Программные средства управления графическими объектами в текстовых документах

Тема утверждена приказом ПГУ № _____ от « » _____ 2025 г.

3. Руководитель работы _____ к.т.н., доцент кафедры «МоиПЭВМ» Самуйлов С.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Назначение:* приложение реализует обтекание тестом графических объектов в текстовом редакторе «Глагол»

2) *Среда выполнения:* ОС Windows

3) *Языковые средства разработки:* C#

4) *Инструментальные средства:* Microsoft Visual Studio, Visual Paradigm

5) *Основные функции приложения:*

а) обтекание графических объектов текстом. Виды обтекания: квадрат, сверху и снизу, по контуру, сквозное, за текстом, перед текстом

б) отображение графических объектов внутри текстового документа

в) доступ к обтеканиям через панель инструментов

в) доступ к обтеканиям через контекстное меню графического объекта

г) открытие страницы с обтеканиями

д) сохранение страницы с обтеканиями

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи
2) Проектирование программных средств
3) Реализация программных средств
4) Контроль качества программного обеспечения
5) План разработки и оценка бюджета
6) Оформление пояснительной записки

6. Календарный график выполнения работы

№ пп	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	08.04.2025 – 14.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Контроль качества программного обеспечения	15%	13.05.2025 – 20.05.2025	
5	План разработки и оценка бюджета	5%	21.05.2025 – 25.05.2025	
6	Оформление пояснительной записки	20%	26.05.2025 – 03.06.2025	

Дата выдачи задания «25»__декабря__2024 г.

Руководитель бакалаврской работы

(подпись, дата)

Задание к исполнению принял студент

(подпись, дата)

Работу к защите допустить

Декан факультета

(подпись, дата)

Реферат

Пояснительная записка содержит 103 листа, 20 рисунков, 8 таблиц, 20 использованных источников, 5 приложений.

УПРАВЛЕНИЕ ГРАФИЧЕСКИМИ ОБЪЕКТАМИ, ТЕКСТОВЫЙ РЕДАКТОР, ВИДЫ ОБТЕКАНИЯ, ОБТЕКАНИЕ ГРАФИЧЕСКИХ ОБЪЕКТОВ.

Объектом разработки является текстовый редактор «Глагол» от научно-технического предприятия «Криптософт», а именно работа с обтеканием текста.

Цель проекта – разработать программное средство для обтекания текстом графических объектов на языке программирования C# с использованием библиотеки GUI.

В процессе работы был произведен анализ требований и условий функционирования разрабатываемых программных средств, определены структура ПС, технология их разработки и тестирования.

В результате выполнения проекта было спроектировано, разработано и отлажено программное средство, способное автоматизировать обтекание графических объектов в текстовом редакторе. Программное средство способно применять к объектам типы обтекания «В тексте», «Квадрат», «По контуру», «Сквозное», «Сверху и снизу», «За текстом», «Перед текстом».

					ПГУ 09.03.02 21ВИ1.24 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Программные средства управления графическими объектами в текстовых документах	Лит.	Лист	Листов
Разраб.		Соколов А.В.					4	103
Провер.		Самуйлов С.В.						
Репенз.								
Н. контр.		Такташкин Д.В.						
Утв.		Козлов А.Ю.				Гр. 21ВИ1		

Содержание

Введение.....	7
1 Анализ предметной области и постановка задачи на разработку программного средства управления графическими объектами в текстовом редакторе.....	8
1.1 Анализ предметной области разработки	8
1.1.1 Обзор графических объектов текстовых документов	8
1.1.2 Основные направления деятельности НТП «Криптософт».....	11
1.2 Обзор аналогичных программных средств	13
1.2.1 Описание известных аналогичных программных средств	13
1.2.2 Сравнительный анализ и обоснование выбора собственной разработки.....	18
1.3 Модель предметной области.....	19
1.4 Постановка задачи.....	21
2 Анализ требований к программному обеспечению.....	25
2.1. Функциональные требования к системе	25
2.2 Требования к пользовательскому интерфейсу.....	29
2.3 Требования к надежности информационной системы.....	29
2.4 Требования к аппаратно-программным средствам	30
3 Анализ информационных технологий разработки программного обеспечения.....	32
3.1 Методологии проектирования информационной системы.....	32
3.2 Средства проектирования и разработки приложения	33
3.3 Базовая архитектура приложения.....	34
4 Проектирование программного обеспечения.....	36

	6
4.1 Проектирование пользовательского интерфейса.....	36
4.2 Структура программного средства.....	38
5 Реализация приложения	44
5.1 Тестирование программного средства.....	44
5.1.1 Тестирование черным ящиком	44
5.1.2 Модульное тестирование (Unit-тестирование)	46
5.2 Анализ качества кода.....	47
5.2.1 Метрика цикломатической сложности	47
5.2.2 Метрика Холстеда.....	49
5.3. Планирование разработки и оценка бюджета.....	51
Заключение	54
Список использованных источников	55
Приложение А (справочное) Перечень сокращений и базовых понятий ...	57
Приложение Б (обязательное) Листинг программы	59
Приложение В (рекомендуемое) Графическая часть	92
Приложение Г (рекомендуемое) Код Unit-тестов.....	99
Приложение Д (рекомендуемое) Акт о внедрении.....	102

Введение

Одними из важнейших задач управления графическими объектами в текстовых редакторах является возможность быстро, удобно, а также интуитивно понятно разместить графический объект в нужном пользователю месте. В современном мире, пользователи текстовых редакторов могут занимать абсолютно разные профессии и должности, имеют разные потребности и стандарты работы, как с размещением графических объектов, так и работой с текстовым редактором в целом.

Существование графических объектов и работа с таковыми является в современном мире реальной потребностью в текстовых редакторах. Исходя из приведенных выше пользовательских потребностей, возникла необходимость в реализации программного средства, отвечающего за расположение и обтекание текстом графических объектов в текстовом редакторе «Глагол», который способен функционировать под двумя операционными системами Windows и QP OS.

При выполнении работы необходимо решить приведенные задачи:

- анализ предметной области разработки;
- анализ требований к программному обеспечению;
- анализ информационных технологий разработки программного обеспечения;
- проектирование программного обеспечения;
- реализация приложения.

В результате выполнения работы было разработано и протестировано программное обеспечение, отвечающее за расположение и обтекание текстов графических объектов.

При реализации использовалась среда разработки Microsoft Visual Studio, а также библиотека пользовательского интерфейса Guido.

1 Анализ предметной области и постановка задачи на разработку программного средства управления графическими объектами в текстовом редакторе

1.1 Анализ предметной области разработки

1.1.1 Обзор графических объектов текстовых документов

В современных реалиях чернила, перья, ручки, линейки, листы бумаги и многие другие помощники перекочевали в цифровое пространство и предстали перед нами в виде офисных приложений. Написание и редактирование текста, построение диаграмм, построение таблиц и составление презентаций теперь в мире двоичного кода и это не воспринимается как фантастика и шутка.

Развитие офисных проектов стартовало с MS Word 1989 года [1]. В настоящее время существует множество версий и вариантов программного обеспечения для офисных программ. Можно выделить три основных направления в работе офисного программного обеспечения.

Ими являются работа с текстовыми документами, для которых применяется наиболее известная программа Word, работа с графическими таблицами, с программой Excel, а также программа для презентаций под названием PowerPoint.

Нельзя представить работу с современными офисными программами без использования графических объектов. Такими объектами могут быть как загруженные пользователем изображения, так и созданные им же диаграммы, или рисунки. Все эти объекты уже давно введены в пользование, так как зачастую пользователи наиболее концентрируются на работе с тем форматом информации, который предоставляет им яркие визуальные образы.

Рассмотрим работу с графическими объектами в текстовых документах более подробно.

Для более детального представления о графическом объекте стоит ввести следующую формулировку.

Графический объект – это любая видимая глазом форма, которую сознание человека определяет среди прочих форм как самостоятельную сущность.

К графическим объектам могут относиться как фотографии, так и рисунки, графические изображения, чертежи, графики, визуальные таблицы, трехмерные модели и так далее. Для более простого понимания визуальный объект следует определять, как графическое изображение для визуализации информации о чем-либо.

Для разных видов графических объектов следует понимать разные задачи, которые таковые выполняют. Чертеж чаще всего служит для детального отображения информации об объекте разработки, или функционировании его частей. График служит для отображения зависимости одной величины от другой. Изображения и картинки используются для явного визуального представления.

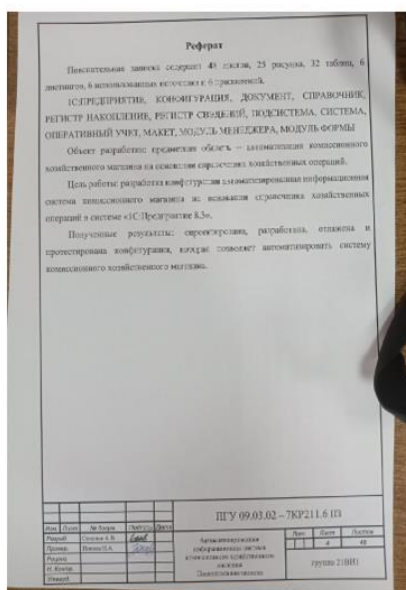
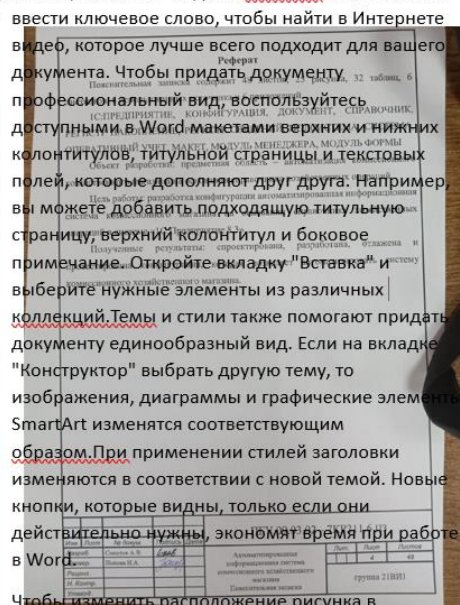
Стоит отметить, что графический объект, имея ряд достоинств перед обычным текстом в плане доступности и выраженности также обладает существенными недостатками. Графические объекты зачастую более громоздки, как по части занимаемого места, так и по части занимаемой памяти, но помимо этого картинка более громоздкая в плане отображения информации. Встречаются ситуации, где отобразить что-либо является менее практичным выбором, чем описать это текстом. Может возникнуть такая ситуация, где пользователь извлекает из изображения иной смысл, чем тот, что был заложен создателем изображения.

Исходя из этого стоит отметить, что наиболее удачным представлением будет связка графического объекта, предоставляющего яркий визуальный образ с текстом, наиболее подробно описывающим происходящее на изображении и дающий правильное представление о смысле, заложенном в изображении.

При разнообразии текста и изображений создается путаница с обилием визуальных образов и нагромождением текста. В таких случаях следует выбирать правильное обтекание текста, верно структурирующего изображения и текст.

В разных ситуациях может возникнуть потребность в использовании разных видов обтекания. Например, в ситуации, когда следует акцентировать внимание смотрящего на изображении наиболее лаконичным решением будет использовать вид обтекания «Квадрат», нежели вид обтекания «За текстом», вносящим более хаотичное отображение информации, чем первый вариант. На рисунке 1 приведен пример разного употребления видов обтекания с одинаковыми изображениями.

видео предоставляет прекрасную возможность подтвердить свою точку зрения. Чтобы вставить видео в документ, нажмите "Вставить видео". Вы также можете вставить видео, которое вы хотите добавить, нажав "Вставить видео".



документе, щелкните его, и рядом с ним появится кнопка для доступа к параметрам разметки. При работе с таблицей щелкните то место, куда нужно добавить строку или столбец, и щелкните знак "..."

Рисунок 1 – Использование двух видов обтекания

В представленном изображении видно, что использование вида обтекания «За текстом» менее удачно, чем использование обтекания «Квадрат», предоставляющего наиболее полное понимание о том, что отображено на изображении.

1.1.2 Основные направления деятельности НТП «Криптософт»

Реализация программного средства будет в приложении «Глагол» от научно-технического предприятия (НТП) «Криптософт». «Криптософт» – ведущее предприятие России в сфере информационной безопасности, разработчик аппаратных и программных средств защиты информации [2].

Научно-техническое предприятие «Криптософт»:

- проводит научно-исследовательские и опытно-конструкторские работы (НИОКР) в различных направлениях;
- выполняет изготовление, единичные и серийные поставки изделий различного назначения;
- ведет опережающую инициативную разработку средства криптографической защиты информации (СКЗИ) новейших поколений;
- соответствует требованиям к лицензиатам ФСБ РФ, ФСТЭК РФ и других ведомств на осуществление работ по созданию средств защиты информации (СЗИ) и оказание услуг в области защиты информации.

НТП «Криптософт» ведет научные исследования по нескольким направлениям:

- защита информации как создание аппаратно-программных решений, исследования и практические разработки в области квантовой криптографии;
- операционные системы в виде защищенной операционной системы QR ОС включена в единый реестр российских программ;
- технология виртуализации отечественный гипервизор QR VMM;
- квантовые технологии как создание кванто-криптографических систем и их интеграция со средствами криптографической защиты информации.

Офисные программы поддерживают формат документов Open xml (DOCX, XLSX и PPTX). Это дает возможность работать с документами как в Microsoft Office, так и в офисном ПО компании.

Документы «старых» форматов DOC, XLS и PPT также поддерживаются, благодаря наличию встроенного конвертера, который выполняет свою функцию при открытии документа в «старом» формате. По завершении конвертации документ сохраняется в формате Open xml и в дальнейшем работа ведется уже с обновленным документом.

Разрабатываемое офисное ПО может работать как на платформе Windows, так и в операционной системе QP ОС.

QP OS – полностью отечественная операционная система с ядром собственной разработки, которая поддерживает платформы x86, x64, ARMv7, ARMv8, Эльбрус. Варианты использования системы: рабочая станция, сервер, встраиваемая платформа. Система соответствует требованиям ФСБ России.

Интерфейс офисных приложений выполнен не только в современном, но и в привычном для пользователей стиле.

Помимо этого, НПП «Криптософт» ведет работу в направлении разработки собственного офисного пакета, в состав которого входят программы «Глагол», «Ведомость» и «Доклад» в разработке которых я принимаю участие. Моя работа будет проходить с программой «Глагол».

Редактор текстовых документов «Глагол» позволяет открывать и редактировать текстовые документы в общепринятых форматах .doc, .docx, .txt, .rtf.

«Глагол» содержит такие функции как создание, открытие, сохранение документа, автоматическая конвертация документов формата .doc в формат .docx, форматирование текста документа (в том числе выравнивание, задание шрифта, цвета и прочих атрибутов), поддержка стилей текста и отображение таблицы стилей документа, работа со списками различного формата внутри документа, поиск/замена в тексте документа, выполнение надстроек документа (в том числе создание резервных копий с заданным интервалом, проверка орфографии и прочее, редактирование колонтитулов документа, вывод документа в печать,

отображение, создание и редактирование таблиц, форматирование стилей таблиц, объединение и деление ячеек, построение графиков и диаграмм, просмотр последних документов, отредактированных в программе «Глагол», создание цифровой подписи [3].

Одной из основных функций редактора текстовых документов станет функция перемещения графических объектов внутри текстового документа, а также автоматизация процесса обтекания графического объекта текстом.

В результате реализации программного средства был получен акт о внедрении приведенный в приложении Д.

1.2 Обзор аналогичных программных средств

1.2.1 Описание известных аналогичных программных средств

В настоящее время можно выделить несколько известнейших компаний, предоставляющих услуги в сфере цифровых офисных приложений. Список крупнейших компаний возглавляет компания Microsoft с программой Word. Пример работы программы Word изображен на рисунке 2.

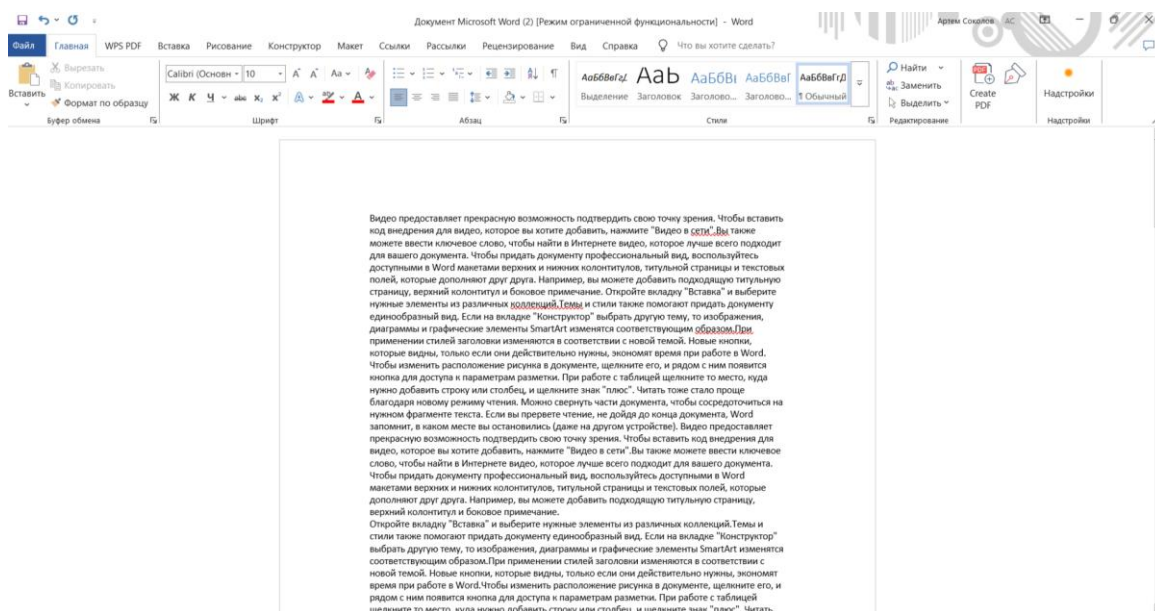


Рисунок 2 – Текстовый редактор Word



Новый документ.docx - Яндекс Документы

Узнать. Чтобы вставить код видеоряда для видео, которое вы хотите добавить, нажмите "Видео в сети". Вы также можете ввести ключевое слово, чтобы найти в Интернете видео, которое лучше всего подходит для вашего документа. Чтобы придать документу профессиональный вид, воспользуйтесь доступными в Word шаблонами заголовка и нижних колонтитулов, титульной страницы и текстовых полей, которые дополняют друг друга. Например, вы можете добавить подзаголовок титульную страницу, верхний колонтитул и боковой примечание.

Откройте вкладку "Вставка" и выберите нужные элементы из различных коллекций. Темы и стили также помогут придать документу единообразный вид. Если на вкладке "Конструктор" выбрать другую тему, то изображения, диаграммы и графические элементы изменятся в соответствии с новой темой. Новые элементы, которые вы видите, только если они действительно нужны, экономят время при работе в Word. Чтобы изменить расположение рисунка в документе, щелкните его, и рядом с ним появится меню для доступа к параметрам рисунка. При работе с таблицей щелкните то место, куда нужно добавить строку или столбец, и щелкните знак "плюс". Начать стало проще благодаря новому режиму чтения. Можно свернуть часть документа, чтобы сосредоточиться на конкретном фрагменте текста. Если вы переключитесь, не выйдя до конца документа, Word запомнит, в каком месте вы находитесь, даже на другом устройстве).

Видео предоставляет прекрасную возможность подтвердить свою точку зрения. Чтобы вставить код видеоряда для видео, которое вы хотите добавить, нажмите "Видео в сети". Вы также можете ввести ключевое слово, чтобы найти в Интернете видео, которое лучше всего подходит для вашего документа. Чтобы придать документу профессиональный вид, воспользуйтесь доступными в Word шаблонами заголовка и нижних колонтитулов, титульной страницы и текстовых полей, которые дополняют друг друга. Например, вы можете добавить подзаголовок титульную страницу, верхний колонтитул и боковой примечание. Откройте вкладку "Вставка" и выберите нужные элементы из различных коллекций. Темы и стили также помогут придать документу единообразный вид. Если на вкладке "Конструктор" выбрать другую тему, то изображения, диаграммы и

Из приведенных вариантов видно сходство во внешнем виде офисных программ. Связано это с тем, что пользователь привык к интерфейсу как программы Word, так и к интерфейсу всего офисного пакета этой компании,

являющейся эталоном в сфере офисного программного обеспечения. Таким образом, дизайн и интерфейс программ остается одинаковым у большинства разрабатываемых компаний. Так, как функционал программ схожий, для более подробного описания конкретного функционала я буду использовать наиболее известный всем пример программу Word от компании Microsoft. Офисный пакет Microsoft использует общий формат файлов под названием Docx [4, 5].

DOCX – это формат файла, используемый во многих приложениях для обработки текстов. Это стандартный формат Microsoft Word, который пришел на смену устаревшему формату .doc в 2007 году.

Формат включает в себя текстовое содержание документа, картинки, нарисованные объекты, стили и иные элементы. Он написан на языке разметки xml и включает в себя множество элементов документа, хранящихся как отдельные файлы, которые объединяются в один архивированный ZIP-файл.

Формат предназначен для создания, изменения и форматирования текстовых документов. Он универсален и подходит для различных целей, таких как создание и редактирование писем, отчетов и эссе, составление резюме и сопроводительных писем, разработка листовок, брошюр и информационных бюллетеней, подготовка академических работ и исследовательских документов, совместная работа над командными проектами.

Общая структура формата xml представляет из себя совокупность xml тегов со своими уникальными значениями. Каждому тегу свойственно свое пространство имен. Также, обычно между xml-документами есть ссылки, находящиеся с самого начала документа для корректного взаимодействия нескольких xml-страниц.

На рисунке 5 представлено отображение документа в формате xml.

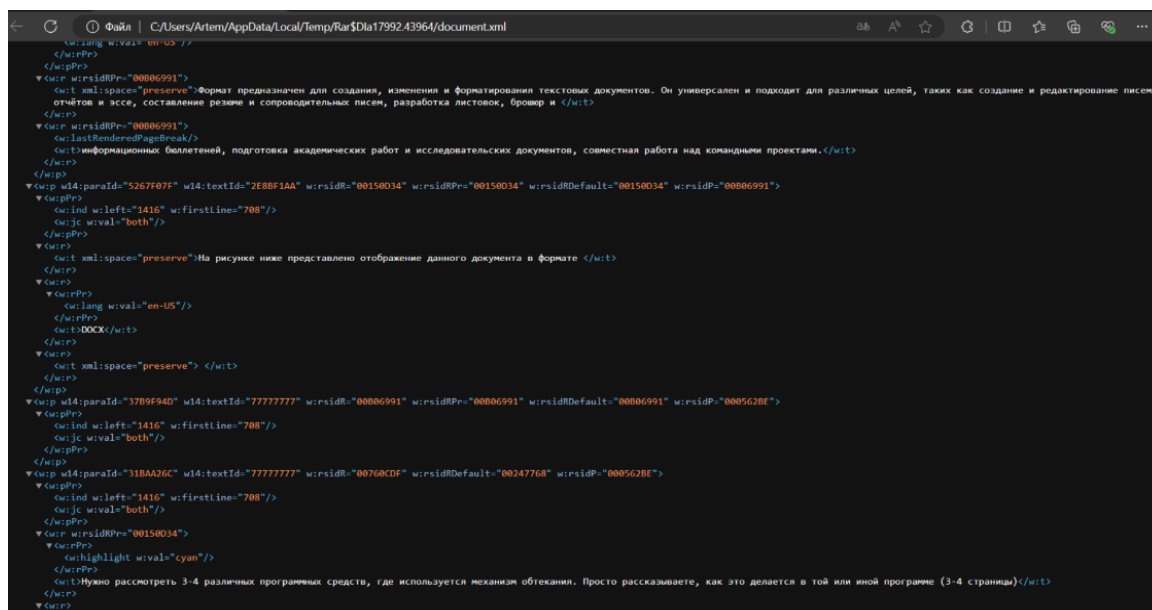


Рисунок 5 – Отображение документа в формате xml

Ниже приведен пример интерфейса работы с объектами в трех программах Word, Яндекс Документы и WPSOffice, а также рисунок с примером работы обтекания текста по параметру «За текстом».

Классический интерфейс Word с примером отображения обтекания «За текстом» изображен на рисунке 6.

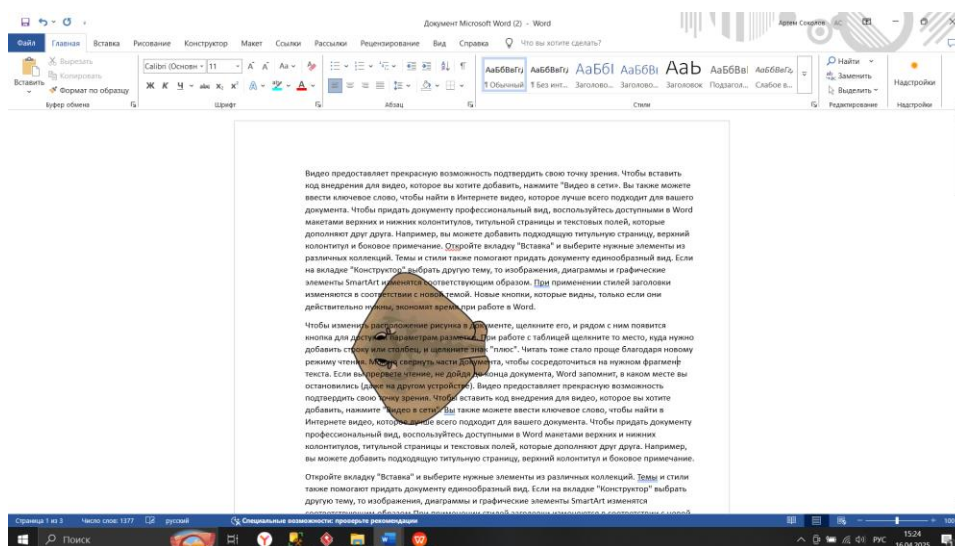


Рисунок 6 – Применение вида обтекания «За текстом» в программе Word

Пример работы с параметром обтекания «За текстом» в приложении Яндекс Документы изображен на рисунке 7.

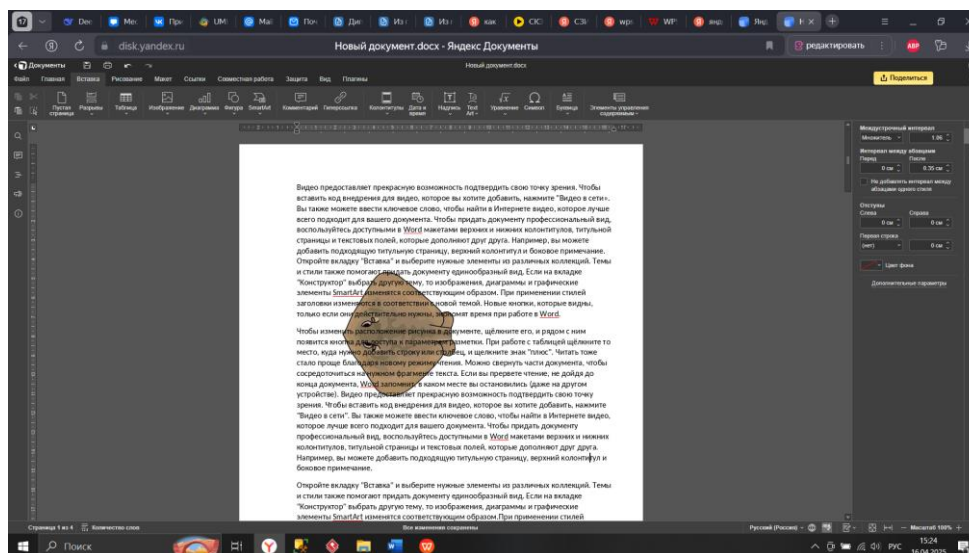


Рисунок 7 – Применение вида обтекания «За текстом» в программе Яндекс Документы

Реализованное обтекание от компании Яндекс с их приложением Яндекс документ, имеет другие названия по сравнению с Word, но схожий функционал.

Также стоит привести в пример продукт от компании Kingsoft Corporation с продуктом WPS Office, изображенном на рисунке 8.

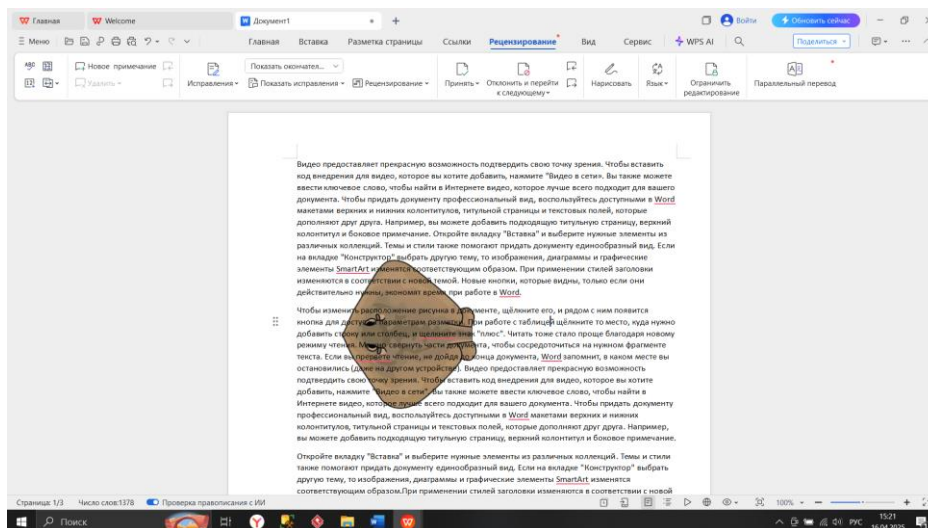


Рисунок 8 – Применение вида обтекания «За текстом» в программе WPS Office

Способ реализации обтекания у всех приведенных в пример продуктов идентичен. Все они работают по принципу, где встроенный в

программу обработчик считывает xml-страницу документа, будь она только создана, или открыт документ, созданный заранее. Значения считываются и обрабатываются, показывая результат в привычном пользователю виде. При изменении документа каким либо образом, xml-страница обновляет свои значения и цикл продолжается снова.

1.2.2 Сравнительный анализ и обоснование выбора собственной разработки

Перед началом разработки следует выделить отличительные критерии разработки, те критерии, которые станут достоинствами при сравнении с продуктами других компаний, а также смогут использоваться как отличительные особенности при рекламе всего основного продукта.

Программа должна обрабатывать все типы обтекания, имеющиеся в аналогичных продуктах, но также иметь собственные отличительные особенности. Первым отличительным признаком будет интеграция программы под две операционные системы. Так, как инструменты разработки приложения «Глагол» располагают к интеграции программы, как под операционную систему Windows, так и под операционную систему QP OS, то разрабатываемая мной программа также будет обладать этим свойством, дающим значительное преимущество перед другими аналогичными программами.

Вторым критерием к сравнению продуктов будет их доступность пользователю, что в условиях политической обстановки ограничивает как использование, так и покупку крупных иностранных аналогов, что осложняет как физическим, так и юридическим лицам работу с продуктами иностранных компаний. В свою очередь программа «Глагол» разрабатывается для использования внутри страны, а значит проблемы с приобретением и использованием, а также долгосрочной поддержкой отсутствуют для юридических и физических лиц, что также является положительным критерием.

Третьим критерием к сравнению станет стоимость разработки программы по сравнению с аналогами. Разработку собственного текстового редактора можно считать наиболее экономичным по отношению к другим за счет курса валют и размера штата разработчиков.

Также критерием к сравнению выступит наличие в приложении закладок, позволяющих злоумышленникам осуществить несанкционированный доступ к ресурсам системы [6].

Более явное сравнение программ с приведением критериев сравнения и аналогичными продуктами других компаний приведено на таблице 1.

Таблица 1 – Сравнение программ по приведенным критериям

Программа	Возможность работы под QP OS и Windows	Доступность российскому пользователю	Стоимость разработки	Отсутствие программных закладок	Возможность менять тип обтекания графических объектов
Глагол	+	+	22048	+	+
Microsoft Word	—	—	220000	—	+
WPS Office	—	+	145000	?	+
Яндекс Документы	—	+	120000	+	+

Исходя из приведенных критериев были выявлены особенности, опираясь на которые будет происходить разработка программного средства.

1.3 Модель предметной области

Предметной областью содержит документ, с которым непосредственно работает пользователь, который включает текст с разнообразными шрифтами и размерами графические объекты с именем и типом обтекания. Модель предметной области приведена на рисунке 9.



Рисунок 9 – Модель предметной области разработки

Документ, который представляет файл и с которым непосредственно работает пользователь, визуализирует, как сохраненный файл, так и резервную копию. «Глагол» корректно функционирует с файлами под расширением doc и docx. Помимо этого, документ содержит ссылки на используемые элементы, совокупность xml-тегов.

Текст документа имеет шрифт, размер и другие атрибуты. Для описания текста в xml-представлении используются теги:

`</w: p>` – тег, применяемый для определения абзаца. Внутренние теги между открывающим `<w: p>` и закрывающим `</w: p>` тегами считаются абзацами.

`<w: u w: val="single"/>` – тег для подчеркивания текста в документе.

`<w: sz w: val="36"/>` – тег для определения размера шрифта.

`<w: color w: val="0000FF"/>` – тег для определения цвета шрифта.

Графический объект является растровым графическим объектом с расширением PNG или JPG. У таких объектов есть возможность изменять тип обтекания текста. Для описания объекта и его обтекания в xml-странице используются такие теги как:

`allowOverlap="1"` – тег, предназначенный для возможности перекрывать другие объекты изображением.

<wp: positionH>, <wp: positionV> – теги для размещения по x и y координатам на странице документа.

<wp:wrapTopAndBottom/> – тег для определения типа обтекания «Сверху и снизу».

Эти и еще множество тегов используются для определения и работы со шрифтами, графическими объектами и текстом в документе. Наличие и обилие тегов дают возможность к использованию разнообразных инструментов и возможности к комфортной работе со всеми элементами документа.

1.4 Постановка задачи

Проведя сравнительный анализ и определив критерии для сравнения, следует разработать требования к реализуемому средству.

Следует разработать систему, осуществляющую корректную работу с различными видами обтекания графических объектов текстом, функционирующую в приложении текстового редактора «Глагол». Также стоит определить основные требования к разрабатываемой системе.

Программное средство (ПС) – это совокупность программ, данных и инструкций, предназначенных для выполнения определенных задач. ПС должно иметь возможность применять один из нижеизложенных видов обтекания к графическому объекту. В программе «Глагол» работа с обтеканием текста графических объектов должна быть реализована в шести вариантах, которые делятся на два подхода работы под названием Anchor и Inline [7].

Встраивание (Inline) означает, что объект рассматривается как часть текста. Он размещается на одной линии с текстом и перемещается вместе с ним.

Когда объект вставляется как встраиваемый, он ведет себя как текстовый символ. Например, если вы вставите изображение как

встраиваемый объект, оно будет находиться на одной строке с текстом и будет перемещаться, если вы добавите или удалите текст.

Встраивание удобно для простых документов, где не требуется сложное форматирование или обтекание текста вокруг объектов. Используется для небольших иконок, формул, элементов, которые должны следовать за текстом.

Якорь – это невидимая метка, которая связывает объект с определенным местом в тексте. Когда вы перемещаете текст, связанный с якорем, объект также перемещается вместе с ним.

Якорь позволяет вам контролировать, как объект будет перемещаться в документе. Например, если вы вставляете изображение и устанавливаете его положение с помощью якоря, изображение будет оставаться на месте относительно текста, даже если вы добавляете или удаляете текст выше или ниже.

Объекты с якорем могут иметь различные параметры обтекания текста (например, «Сверху и снизу», «Квадрат», «Сквозное»). Разберем их подробнее.

Тип Square (Квадрат) обтекает объект прямоугольником, даже если сам объект имеет неправильную форму. Тип используется для фотографий, иллюстраций, когда требуется четкое выравнивание.

Tight (По контуру) обтекает реальные границы объекта, если такие распознаются. Этот тип подходит для изображений с прозрачным фоном или изображениями сложной формы. Используется для логотипов, векторных рисунков, где важно плотное обтекание.

Through (Сквозное) обтекает объект максимально близко к контуру, включая пустые области объекта. В отличие от типа Tight может заполнять полости внутри объекта, если такие имеются. Подходит для сложных векторных изображений с отсутствием части изображения в объекте.

Top and Bottom (Сверху и снизу) не обтекает объект текстом по бокам, а располагается только сверху и снизу. Объект занимает всю ширину

страницы. Данный тип обтекания используется для широких таблиц, диаграмм, в тех случаях, когда боковое обтекание не нужно.

Behind Text (За текстом) полезно для фоновых изображений или водяных знаков, когда объект находится под текстом, а текст накладывается сверху. Тип обтекания полезен для фонов и декоративных элементов.

Последний реализуемый тип обтекания In Front of Text (Перед текстом) перекрывает текст объектом, располагаясь поверх него. Данный тип обтекания может мешать чтению, если не настроена прозрачность. Этот тип обтекания применяется для всплывающих подсказок, маркеров и элементов поверх текста.

Программное средство должно иметь возможность сохранять изменения, произведенные с обтеканием графического объекта.

Программное средство должно корректно отображать текст и графические объекты как после загрузки и печати таковых, так и после открытия сохраненного документа.

Должна обеспечиваться корректная загрузка и отображение графического элемента.

Программное средство должно быть корректно интегрирована в приложение «Глагол».

В результате проектирования системы требуется:

- выполнить анализ предметной области;
- определить функциональные возможности системы;
- выполнить проектирование структуры своеобразной базы данных в виде xml-страницы для хранения сведений о результате пользовательской сессии;
- реализовать возможность сохранения данных при помощи занесения данных о пользовательской сессии в xml-страницу и последующей загрузки из нее;
- разработать обработчик xml-документа, отвечающий за грамотное построение и расположение пользовательских данных в интерфейсе;

- разработать интерфейс программы;
- проанализировать существующие технологии разработки программного обеспечения;
- выбрать инструментальные средства для разработки программного обеспечения;
- выполнить реализацию программного средства;
- протестировать разработанную программу.

2 Анализ требований к программному обеспечению

2.1. Функциональные требования к системе

Для правильной реализации информационной системы нужно провести соответствие между реализуемыми функциями системы и функциональными требованиями. Функциональные требования – это те требования, которые ПС должна выполнять так, как это было предписано в постановке задачи на разработку [8]. Через функции системы реализуются функциональные требования. Функции – это совокупность действий системы, направленные на реализацию какой-либо задачи из задач на разработку.

Для наглядного отображения того, что программное средство должно выполнять в процессе своего функционирования, применяется диаграмма вариантов использования, или, другими словами, диаграмма прецедентов. Диаграмма показывает отношения пользователей текстового редактора с вариантами использования системы. Диаграмма описывает работу на концептуальном уровне.

Диаграмма вариантов использования состоит из актера, роль которого взаимодействие с системой, вариантов использования – отдельных функций системы, с которыми взаимодействует пользователь и связями, показывающими принцип взаимодействия между вариантами использования и актером.

Так как программное средство задумывается для широкого применения, нельзя точно предугадать все виды пользователей системы, поэтому на диаграмме все виды пользователи описаны под абстрактным наименованием «Пользователь».

На основании ранее проведенного анализа предметной области, а также функциональных требований была разработана диаграмма вариантов использования, изображенная на рисунке 10.

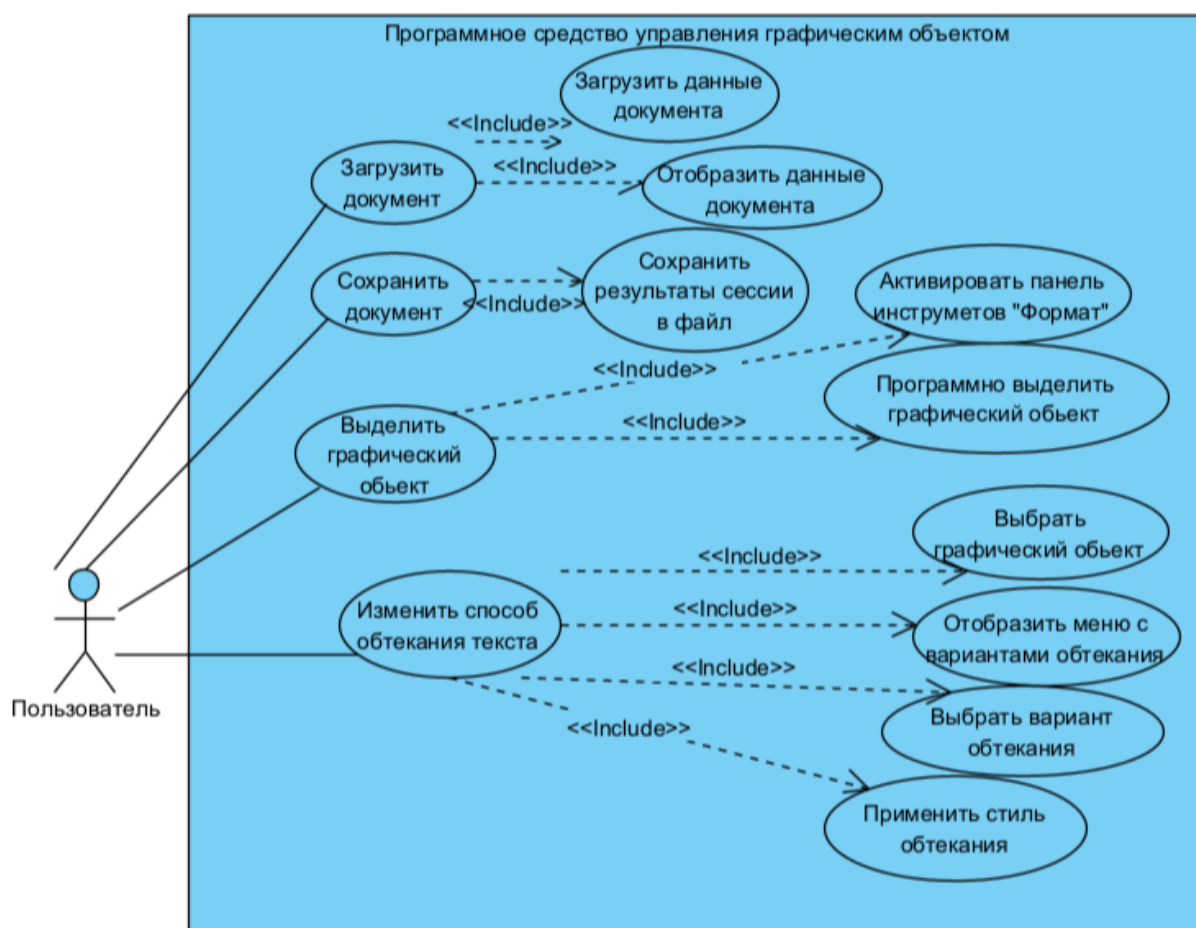


Рисунок 10 – Диаграмма вариантов использования

Из диаграммы вариантов использования следует, что так, как все пользователи для системы имеют одинаковые права, то в системе нет разделения по ролям и все посетители системы имеют доступ к одинаковому числу функций системы.

Более детальное объяснение работы прецедента «Загрузить документ» расположено в таблице 2.

Таблица 2 – Спецификация прецедента «Загрузить документ»

Имя прецедента	Загрузить документ
Идентификатор прецедента	ID1
Краткое описание	Система загружает выбранный пользователем документ
Актеры, вовлеченные в прецедент (главные)	Пользователь
Состояние системы до начала прецедента	Предусловия: Пользователь открыл текстовый редактор

Продолжение таблицы 2

Фактические этапы прецедента	Основной поток: 1. Пользователь выбирает инструмент для открытия сохраненного файла 2. Пользователь выбирает сохраненный файл с расширением DOC или DOCX 3. Система считывает данные из документа 4. Система выстраивает документ и отображает графические объекты с сохранением обтекания
Состояние системы после окончания прецедента	Постусловия: Система отображает открытый документ с графическими объектами и сохраненным обтеканием
Альтернативные потоки	Нет

Объяснение работы прецедента «Сохранить документ» расположено в таблице 3.

Таблица 3 – Спецификация прецедента «Сохранить документ»

Имя прецедента	Сохранить документ
Идентификатор прецедента	ID2
Краткое описание	Система сохраняет данные и информацию текущей сессии в файл
Актеры, вовлеченные в прецедент (главные)	Пользователь
Состояние системы до начала прецедента	Предусловия: Пользователь открыл текстовый редактор и провел работу с документом
Фактические этапы прецедента	Основной поток: 1. Пользователь выбирает инструмент сохранения данных 2. Пользователь сохраняет документ 2.1 Пользователь вводит название файла для сохранения 2.2 Пользователь сохраняет файл с существующим названием 3. Пользователь нажимает на кнопку «Сохранить». Система сохраняет информацию в имеющийся файл или создает новый файл для сохранения
Состояние системы после окончания прецедента	Постусловия: Система предоставляет доступ для дальнейшей загрузки, сохранения и работы с текущей сессии
Альтернативные потоки	Нет

Объяснение работы прецедентов «Выделить графический объект» и «Изменить способ обтекания текста» расположены в таблицах 4 и 5.

Таблица 4 – Спецификация прецедента «Выделить графический объект»

Имя прецедента	Выделить графический объект
Идентификатор прецедента	ID3

Продолжение таблицы 4

Краткое описание	Пользователь выбирает графический объект для каждого варианта обтекания текста
Актеры, вовлеченные в прецедент (главные)	Пользователь
Состояние системы до начала прецедента	Предусловия: Пользователь открыл текстовый редактор и добавил в документ графический объект
Фактические этапы прецедента	Основной поток: 1. Пользователь находит графический объект 2. Пользователь кликает по объекту левой кнопкой мыши 3. Система выделяет графический объект 4. Система предоставляет пользователю контекстную панель «Формат»
Состояние системы после окончания прецедента	Постусловия: Система предоставляет доступ для выбора параметра обтекания текстом графического объекта
Альтернативные потоки	Нет

Таблица 5 – Спецификация прецедента «Изменить способ обтекания текстом»

Имя прецедента	Изменить способ обтекания текстом
Идентификатор прецедента	ID4
Краткое описание	Пользователь изменяет тип обтекания текста графического объекта через инструменты системы
Актеры, вовлеченные в прецедент (главные)	Пользователь
Состояние системы до начала прецедента	Предусловия: Пользователь вошел в текстовый редактор и добавил в документ графический объект
Фактические этапы прецедента	Основной поток: 1. Пользователь выбирает графический объект 2. Пользователь выбирает вид обтекания 2.1 Пользователь выбирает тип обтекания через панель инструментов «Формат» 2.2 Пользователь выбирает тип обтекания через нажатие по объекту правой кнопкой мыши с последующим выбором обтекания текстом через контекстное меню 3. Пользователь выбирает подходящий вид обтекания текстом 4. Система изменяет тип обтекания графического объекта 5. Система динамически перестраивает прилегающий к графическому объекту текст
Состояние системы после окончания прецедента	Постусловия: Система предоставляет доступ для дальнейшего выбора параметра обтекания текстом графического объекта и дальней работы с системой
Альтернативные потоки	Нет

В ходе анализа функциональных требований к системе, была спроектирована диаграмма прецедентов и спецификации основных прецедентов, что дало более детальное понимание о том, что в результате реализации системы пользователь должен получить возможность выбирать у графических объектов тип обтекания текстом, а также загружать и сохранять документы, в которых есть графические объекты с примененным к ним обтеканием.

2.2 Требования к пользовательскому интерфейсу

Первостепенной задачей к разработке интерфейса станет реализация возможности выбора типа обтекания из панели инструментов. Нужно расположить дополнительный раздел с названием «Формат» на панели инструментов. Раздел должен отображаться пользователю после выделения графического объекта. Выбор типа обтекания должен иметь вид нажатия на один из элементов выпадающего списка с названием типа обтекания и изображением, отображающего использования обтекания.

Следующей задачей будет реализация выбора типа обтекания из контекстного меню при нажатии правой кнопкой мыши по графическому объекту. После нажатия должен появляться выпадающий список с названием типов обтекания и иллюстрационными примерами. Типы обтекания должны быть разделены по видам работы с текстом.

2.3 Требования к надежности информационной системы

Одним из немаловажных критериев к разработке программы является ее надежность.

Надежность системы – это ее способность выполнять требуемые функции в заданных условиях в течение определенного времени без сбоев или потери пользовательских данных.

Чем лучше спроектирована и реализована ИС, тем она более надежна в процессе работы и более стабильна при непредвиденных действиях

пользователя. Таким образом, задача проектирования с целью повышения надежности системы является одной из ключевых критериев к разработке системы.

Определены следующие требования к системе при ее реализации по требованиям надежности продукта.

Программное средство должно корректно функционировать при изменении типа обтекания у графических объектов, загрузке и сохранении документа с графическими объектами с примененными к ним обтеканиям, а также при перемещении картинки с примененным к ней типом обтекания. Работа с обтеканием не должна сопровождаться зависанием ПО и возникновением ошибок. Система должна быстро откликаться на действия пользователя.

Также ПС должно отвечать общим требованиям надежности, описанным ниже.

Возникновение ошибок. Некорректные действия пользователя не должны приводить к возникновению критических ошибок.

ПС должно выполнять резервное копирование сессии в течении определенного промежутка времени для защиты уже введенной информации от случайного удаления.

ПС должно корректно обрабатывать сохраненные файлы при загрузке и сессионную информацию при сохранении ее в файл без возникновения ошибок.

Исходя из вышеперечисленного можно делать выводы о том, какими требованиям к надежности должно отвечать разрабатываемое программное средство.

2.4 Требования к аппаратно-программным средствам

Стабильность, надежность и быстродействие являются основными требованиями к аппаратным средствам, которые необходимы при реализации и тестирования ПО.

Аппаратные средства включают в себя физические компоненты компьютера, или другой системы для корректной работы в виде вычислений, хранения, отображения и загрузки данных и взаимодействия с пользователем.

Наиболее известным представителем аппаратных средств является процессор, являющийся своеобразным «мозгом» компьютера, производящий вычисления, оперативная память, предоставляющая временное хранилище данных для быстрого доступа, материнская плата, объединяющая все компоненты системы. Помимо этого, существуют устройства хранения данных такие как жесткий диск HDD и твердый накопитель SSD, являющиеся магнитным хранилищем, как правило, с большим объемом хранимой информации и более быстрой, но менее вместительной по сравнению с HDD flash-памятью [9].

Для наиболее корректной работы программы следует выделить требуемые от пользователя необходимые программные средства.

Рекомендованными параметрами как для корректной работы операционной системы QR OS, так и для функционирования текстового редактора «Глагол» является:

- процессор с тактовой частотой 2 ГГц;
- оперативная память объемом 2ГБ;
- жесткий диск, объемом в 30 ГБ;
- наличие устройств ввода в виде мыши и клавиатуры.

Наличие таких аппаратно-программных средств даст возможность к полноценному функционированию приложения как на операционной системе QR OS, так и на операционной системе Windows.

Улучшение характеристик средств в сторону увеличения производительности даст более комфортные условия для эксплуатации ПО.

Помимо установленной операционной системы для функционирования текстового редактора установка дополнительного программного обеспечения не требуется.

3 Анализ информационных технологий разработки программного обеспечения

3.1 Методологии проектирования информационной системы

Для разработки системы выбрана методология RUP (Rational Unified Process).

Rational Unified Process это методология разработки, при которой вся разработка проекта поделена на небольшие циклы. Каждый из циклов добавляет свою, небольшую часть функционала. После каждого этапа происходит оценка и корректировка планов на остальные этапы разработки.

Разработка в такой методологии строится из повторно используемых компонентов, что придает большую удобность процессу разработки. Архитектура системы разрабатывается на ранних этапах и улучшается в процессе разработки. RUP поддерживает объектно-ориентированную технологию, что в паре с объектно-ориентированным языком программирования C# дает более разнообразные возможности в реализации объектно-ориентированного подхода к разработке. Помимо этого, в качестве стандарта для организации взаимодействия пользователей используется универсальный язык моделирования Unified Modelling Language (UML).

Использование UML диаграмм будет означать, что команда разработки имеет согласованное видение о том, как и по каким принципам, будет создаваться ПО. По сравнению с другими методологиями такими как Waterfall (линейная методология разработки) или Agile (пошаговая разработка), разработка методологией RUP имеет свои преимущества. Так как RUP имеет итеративный подход, где каждая фаза состоит из циклов и где требования уточняются после прохождения итераций в отличие от линейной разработки, то ошибка при тестировании при выборе методологии RUP может обнаружиться уже на первой итерации первого тестирования. Так как методология Agile наиболее удачно подходит для динамичных проектов

и страдает от отсутствия документации и четкой архитектуры, RUP сохранит гибкость, но добавит к разработке определенную структуру, например, сохранит четкие фазы и роли, такие как тестировщик, разработчик, или аналитик, а также с помощью UML диаграмм сможет детально спроектировать как работу разработчиков, так и работу пользователей.

Из других достоинств RUP методологии можно выделить большую фиксируемость на рисках разработки, более оптимальную масштабируемость как на больших, так и на малых проектах, и документирование ключевых артефактов, таких как архитектура или тест-кейсы.

Исходя из приведенных особенностей методологии следует сделать вывод о том, что методология RUP является «золотой серединой» в методологиях разработки за счет идеального баланса между гибкостью и контролем, наличия сложных требований и наличия норм регуляторов и итеративного подхода к разработке [10].

3.2 Средства проектирования и разработки приложения

При проектировании архитектуры приложения и его структуры была использована среда Visual Paradigm Community Edition версии 16.3 поддерживающая унифицированный язык программирования UML.

Использование Visual Paradigm CE позволило эффективно спроектировать архитектуру приложения, обеспечив четкую визуализацию компонентов системы, согласованность между моделью и реализацией и легкость сопровождения.

Выбор средств разработки приложения был обусловлен набором программной разработки, использующейся при разработке текстового редактора «Глагол».

Разработка системы осуществлялась на языке программирования C# под интегрированной средой разработки Visual Studio 2019. Visual Studio

предоставляет мощные инструменты для написания, отладки и тестирования кода, что делает процесс разработки более быстрым и удобным [11].

В процессе разработки применялся инструментарий создания и управления графическими элементами пользовательского интерфейса Guido.

3.3 Базовая архитектура приложения

Как и аналогичные текстовые редакторы без полноценной базы данных, текстовый редактор «Глагол» строится по архитектуре модель-представление-презентатор (MVC), которая включает в себя три основных слоя.

Первый слой Model (Модель) отвечает за данные, бизнес–логику и работу с файловой системой, и такие функции системы как загрузка, сохранение и обработка данных. Примером этого слоя может выступить файловая система текстового редактора «Глагол».

Слой представления View имеет представление в виде интерфейса, который видит сам пользователь и отображает данные пользователю, а также, передает действия пользователя третьему слою Controller, речь о котором пойдет дальше. Слой View содержит только визуализацию, но не логику обработки данных.

Controller выступает посредником между слоями Model и View. Этот слой получает данные из Model и отображает их во View.

Помимо трех слоев есть пользователь, взаимодействующий с системой через слой View и как правило, действия, происходящие во всех трех слоях, происходят по причине взаимодействия пользователя и View.

Шаблон проектирования MVC, как правило используется в разработке пользовательских, что обеспечивает четкое разделение кода на логику, интерфейс и управление.

Представление архитектуры MVC представлено на рисунке 11.

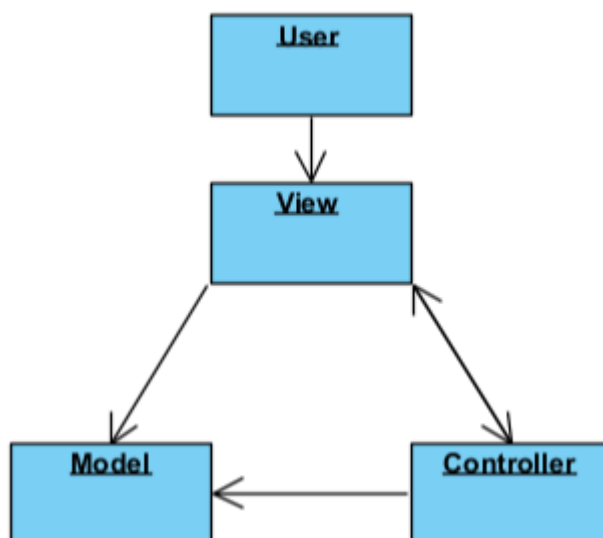


Рисунок 11 – Архитектура MVC

Взаимодействие компонентов происходит по следующему сценарию. Пользователь взаимодействует со слоем View. Слой View передает событие слою Controller. Controller обрабатывает запрос и запрашивает данные у слоя Model. Model возвращает данные из БД, в данном случае из xml-кода документа. Presenter формирует данные и возвращает из в слой View.

Преимущество архитектуры MVC заключается в более простой реализации за счет уменьшения количества кода. Приложение с архитектурой MVC легко тестировать, например, слой Model можно тестировать, не затрагивая слой View, а также ПС с этой архитектурой более гибкое, так как можно менять слой View, не затрагивая остальные слои. Код в такой архитектуре менее связный, так как компоненты в таком случае зависят от интерфейсов, а не от реализаций.

Таким образом, за счет вышеперечисленных достоинств, архитектура MVC удачно применяется как для проектирования текстового редактора, так и для реализации приложения для обтекания текстом.

4 Проектирование программного обеспечения

4.1 Проектирование пользовательского интерфейса

Интерфейс разработанной системы представляется в виде выпадающего списка со всеми видами обтекания. Редактирование типа обтекания графического объекта посредством выбора типа обтекания из выпадающего списка реализовано двумя способами.

При первом способе пользователь должен выделить только графический объект. Тогда в разделе инструментов пользователю станет доступен раздел под названием «Формат». При переходе на этот раздел пользователь сможет выбрать один из типов обтекания при нажатии правой кнопкой мыши по кнопке обтекания текстом. После нажатия появляется выпадающий список со всеми типами обтекания, а также с простой картинкой, представляющей этот тип, для визуального представления об работе конкретного вида обтекания. Нажатие на один из элементов списка обеспечивает применение соответствующего типа обтекания. Пример изменение типа обтекания графического объекта через меню инструментов изображено на рисунке 12.

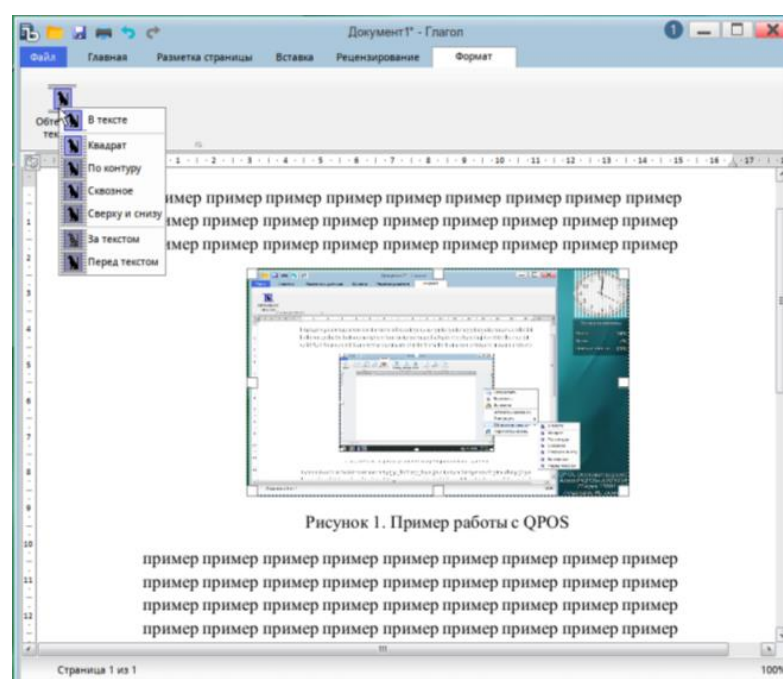


Рисунок 12 – Изменение типа обтекания через панель инструментов

Вторым способом изменения типа обтекания объекта является изменение через нажатие правой кнопкой мыши по графическому объекту. При таком условии появляется выпадающий список с возможностью копировать объект, повернуть его, настроить параметры абзаца и так далее.

Один из подразделов выпадающего списка имеет название «Обтекание текстом» с возможностью выбора дальнейших подразделов. При нажатии левой кнопкой мыши на разделе «Обтекание текстом» появляется подсписок со всеми возможными типами обтекания, которые также имеют прикрепленные поясняющие работу обтекания картинки. При выборе любого типа обтекания из появившегося подсписка применится соответствующий тип обтекания текстом.

Пример изменения типа обтекания объекта через нажатие правой кнопки мыши изображен на рисунке 13.

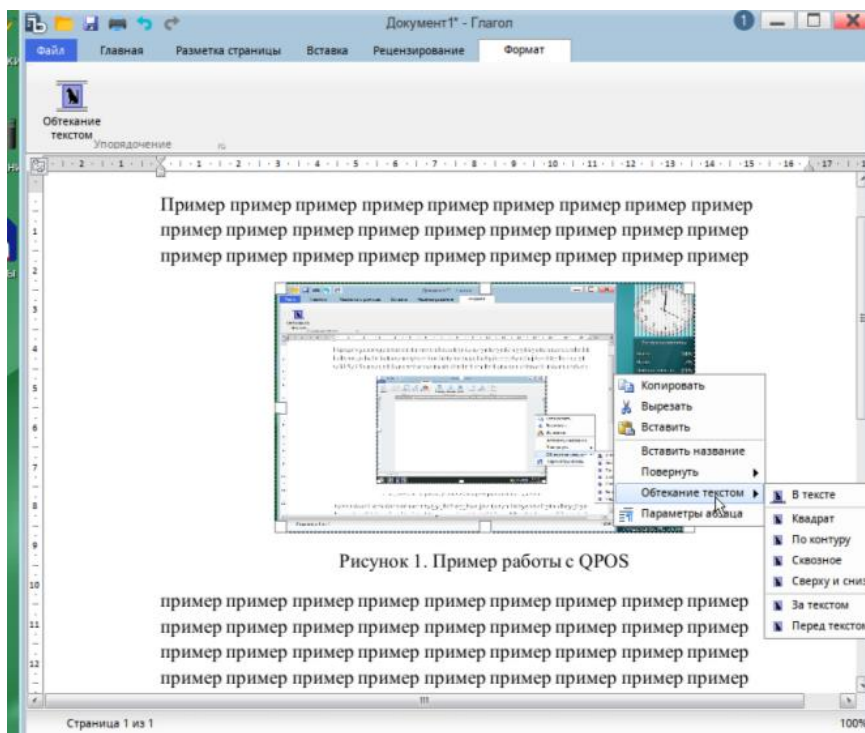


Рисунок 13 – Изменение типа обтекания через контекстное меню объекта

Как можно заметить, оба варианта изменения обтекания имеют одинаковый функционал.

Исходя из приведенных примеров можно сделать вывод о том, что разработка интерфейса программного средства удовлетворяет техническому заданию.

4.2 Структура программного средства

Для визуального представления структуры разработанной системы была разработана диаграмма классов [12, 13]. Диаграмма классов изображена на рисунке 14. Диаграмма отображает классы, используемые при изменении типа обтекания.

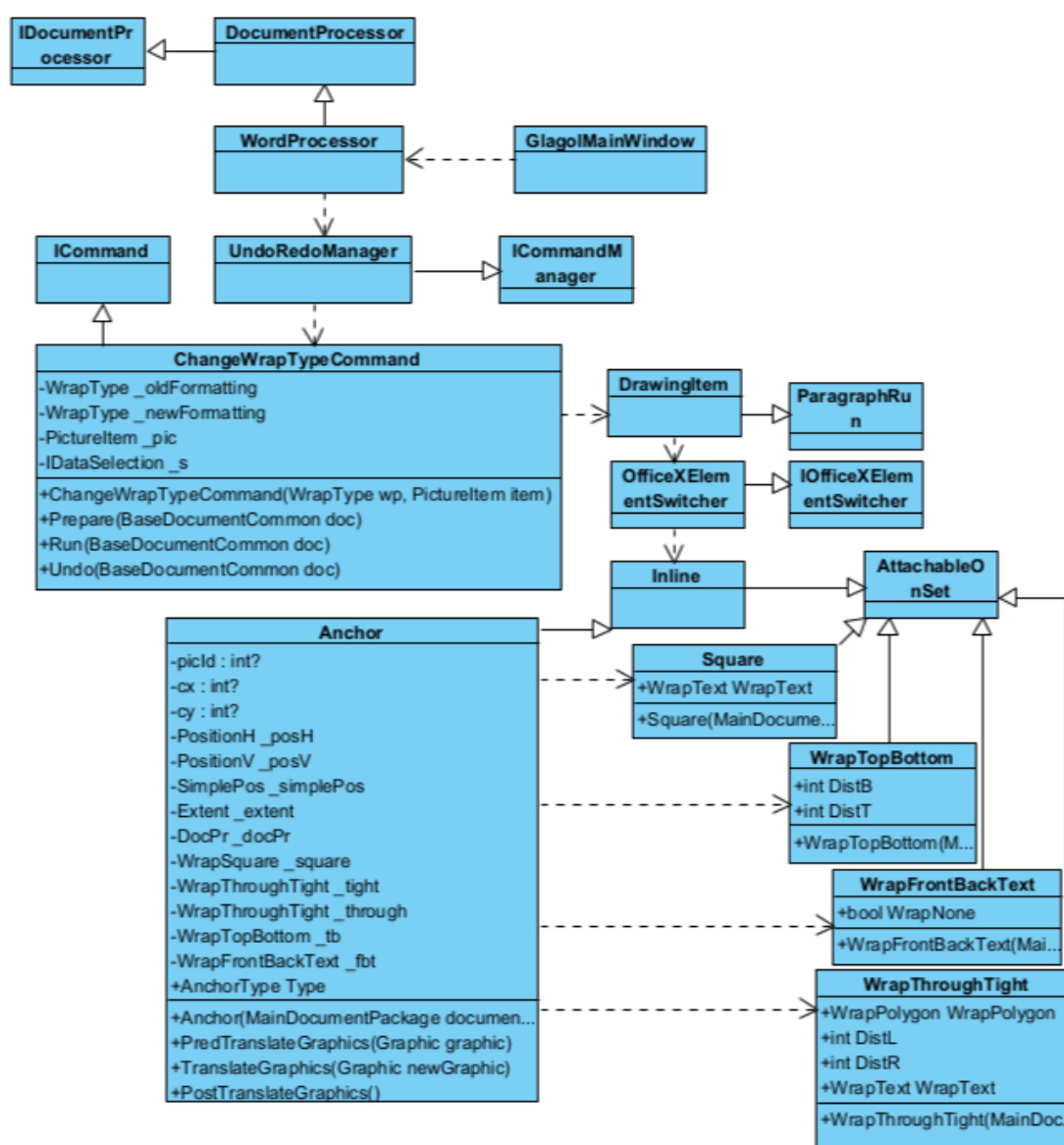


Рисунок 14 – Диаграмма классов приложения

Описание классов и их основной функционал предоставлены ниже.

GlagolMainWindow – класс, представляющий собой обработчик основной формы текстового редактора. Методы этого класса срабатывают при взаимодействии с пользовательским интерфейсом.

WordProcessor – класс, отвечающий за обработку текста, управление командами (в данном случае это команда ChangeWrapTypeCommand), и выполнением операций с данными, класс наследуется от интерфейса IDocumentProcessor.

UndoRedoManager – класс, созданный для управления паттерном, команда в текстовом редакторе, также класс служит для заполнения стеков для возможности отмены действий пользователя. Класс наследуется от интерфейса ICommandManager и перенимает его общедоступные методы.

ChangeWrapTypeCommand – класс, определяющий команду для реализации паттерна, команда, отвечающая за непосредственное изменение вида обтекания у текстового редактора.

Методы класса ChangeWrapTypeCommand:

- конструктор класса ChangeWrapTypeCommand необходим для определения объекта картинки и типа обтекания;
- метод Prepare необходим для замены старого обтекания на новое;
- метод Run необходим для применения обтекания и дальнейшего составления xml-документа;
- метод Undo необходим для отмены последней операции по обтеканию.

Атрибуты класса ChangeWrapTypeCommand:

- атрибут _oldFormatting необходим для сохранения старого форматирования;
- атрибут _newFormatting необходим для сохранения нового форматирования;
- атрибут _picItem необходим для сохранения графического объекта;

– атрибут `_IDataSelection` необходим для сохранения выделения графического объекта;

`ICommand` – интерфейс реализации общедоступных методов для команд, реализованных в текстовом редакторе.

`DrawingItem` – класс, представляющий реализацию графического объекта, в текстовом редакторе с присущим графическому объекту атрибутами.

`OfficeXElementSwitcher` – класс, работающий с xml-представлением документа, класс определяет теги в файле и определяет область имен для тегов и определяет их вложенность.

Классы `Anchor` и `Inline` представляют собой две вариации непосредственной реализации видов обтекания в документе со своими особенными тегами для каждого.

Методы класса `Anchor`:

– конструктор класса `Anchor` отвечающий за наполнение xml-страницы в зависимости от выбранного типа обтекания;

– метод `PredTranslateGraphics` отвечающий за обработку xml-страницы, содержащей данные о картинке;

– метод `TranslateGraphics` отвечающий за парсинг данных о картинке во всем xml-представлении;

– метод `PostTranslateGraphics` создан, для дальнейшего расширения функционала.

Атрибуты класса `Anchor`:

– атрибут `picId` для работы с одноименным тегом, отвечающим за идентификатор картинки во всем документе;

– атрибут `sx` для работы с xml-тегом, отвечающим за расположение картинки в режиме конструирования по оси x;

– атрибут `sy` для работы с xml-тегом, отвечающим за расположение картинки в режиме конструирования по оси y;

- атрибут `posH` для работы с `xml`-тегом, отвечающим за расположение картинки на странице по оси `y`;
- атрибут `posV` для работы с `xml`-тегом, отвечающим за расположение картинки на странице по оси `x`;
- атрибут `SimplePos` для определения позиции картинки на странице;
- атрибут `Extent` для определения размеров графического объекта;
- атрибут `DocPr` для определения общих свойств картинки;
- атрибут `Square` для создания уникальных `xml`-тегов данного типа обтекания;
- атрибут `WrapThroughTight` для создания уникальных `xml`-тегов данного типа обтекания;
- атрибут `WrapTopBottom` для создания уникальных `xml`-тегов данного типа обтекания;
- атрибут `WrapFrontBackText` для создания уникальных `xml`-тегов данного типа обтекания;
- атрибут `WrapType` сохраняет входящий в атрибуты конструктора тип обтекания картинки.

Классы `Square`, `WrapTopBottom`, `WrapFrontBackText`, `WrapThroughTight` реализованы для разбиения класса `Anchor` на подробную реализацию каждого видов обтекания, основанных на принципе «Якорь». Каждому из классов свойственны свои атрибуты и `xml`-теги.

Методом класса `Square` является конструктор класса `Square`, отвечающий за добавление стандартных `xml`-элементов для этого типа обтекания.

Атрибутом класса `Square` является `WrapText` для работы с уникальным `xml`-тегом `wrapSquare`.

Методом класса `WrapTopBottom` является конструктор класса `WrapTopBottom` отвечающий за добавление стандартных `xml`-элементов для этого типа обтекания.

Атрибуты класса `WrapTopBottom`:

- атрибут BistT для работы с уникальным xml-тегом WrapTopBottom;
- атрибут BistB для работы с уникальным xml-тегом WrapTopBottom.

Методом класса FrontBackText является конструктор класса FrontBackText отвечающий за добавление стандартных xml-элементов для этого типа обтекания.

Атрибутом класса FrontBackText является WrapText для работы с уникальным xml-тегом wrapNone.

Методы класса ThroughTight является конструктор класса ThroughTight отвечающий за добавление стандартных xml-элементов для этого типа обтекания;

Атрибуты класса ThroughTight:

- атрибут DistL для работы с уникальным xml-тегом ThroughTight;
- атрибут DistR для работы с уникальным xml-тегом ThroughTight;
- атрибут WrapText для работы с уникальным xml-тегом ThroughTight.

Класс AttachableOnSet реализован для сохранения созданных xml-тегов обтекания к графическому объекту и последующего их прикрепления ко всему xml-файлу.

Класс реализации паттерна команды под названием ChangeWrapTypeCommand создан для буферизации данных о ранее сохраненном и примененном типе обтекания с помощью применения паттерна «Команда» [14]. Сделано это для того, чтобы пользователь мог отменять последние действия, совершенные с обтеканием объекта. Для этого в класс были добавлены атрибуты с названием _oldFormatting и _newFormatting. Атрибут _PictureItem служит представлением графического объекта, для которого применяется изменение типа обтекания, а атрибут IDataSelection представляет собой выделение графического объекта в документе, из которого в свою очередь создается _PictureItem. Методы класса Prepare, Run и Undo переопределяются от интерфейса ICommand, в этих методах, а также в конструкторе класса происходят изменения в типе обтекания графического объекта.

Классы `Square`, `WrapTopAndBottom`, `WrapFrontBackText`, `WrapThroughTight` имеют одинаковое назначение и похожую структуру в реализации. Все из вышеперечисленных классов заполняют xml-документ уникальными элементами и атрибутами. Каждый из видов обтекания имеет уникальный атрибут или элемент, необходимый для корректного определения. Эти значения создаются либо в методах класса, либо, что чаще, в самих конструкторах классов. Эти классы с уникальными значениями создаются из класса `Anchor`.

Класс `Anchor`, в зависимости от пришедшего в его конструктор значения обтекания создает один из классов со своими специальными значениями, например, класс `Square`. Помимо этого, класс `Anchor` добавляет в xml-документ стандартные для каждого вида обтекания значения, такие как `id` графического объекта, его названия, или позицию объекта по ширине или высоте, относительно страницы.

Иногда нельзя самостоятельно задать значения для xml-документа. В таких случаях нужно обращаться к другим частям xml-значений. Примером тому могут служить методы с названием `PredTranslateGraphics`, `TranslateGraphict` и `PostTranslateGraphics`. Эти методы берут значения ширины, высоты, положения на странице у графического объекта, к которому применяются обтекание. От правильного составления xml-части документа зависит то, как будет работать с другими текстовыми редакторами документ, созданный, или отредактированный в приложении «Глагол».

5 Реализация приложения

5.1 Тестирование программного средства

Для уверенности в правильной работе программное средство будет протестировано черным ящиком, и модульным тестированием.

5.1.1 Тестирование черным ящиком

Для тестирования был применен метод классов эквивалентности (КЭ). Такой подход предполагает группировку по входным и выходным значениям по категориям, которые однообразно обрабатывает система. Наборы входных данных вызывают одинаковый результат в программе независимо от того ошибка это, или правильная работа.

Для проведения тестов были составлены классы эквивалентности, представленные на таблице 6.

Таблица 6 – Классы эквивалентности

	Правильные КЭ	Неправильные КЭ
Длина имени	9-255(1)	0-8(4) 256-∞ (5)
Структура	Начинается с заглавной русской буквы, после строчные русские буквы, пробел и положительные цифры (2)	Начинается со строчной буквы (6)
		Начинается с цифры (7)
		Начинается с латинской буквы (8)
		Отсутствует пробел (9)
		Дробное число (10)
		Отрицательное число (11)
		Лишние пробелы (12)
		Начинается с двух и более заглавных русских букв (13)
		Отсутствует число (14)
Содержание	Кириллица Цифры Пробел (3)	Остальное (15)

На таблице 7 показано построение тестов по классам эквивалентности.

Таблица 7 – Тест-кейсы

Номер теста	Классы Эквивалентности	Входные данные	Выходные данные	Реакция программы
1	1, 2, 3	Рисунок 1	Успех: отсутствуют ошибки	
2	1,2,4	Рисунок1	Ошибка: длина значения 8 символов (меньше 9)	System.Exception: "Attribute out of range"
3	2,3,5	Рисунок 1111111111 1111111111 ...	Ошибка: длина значения больше 255 символов	System.Exception: "Attribute out of range"
4	1,3,6	рисунок 1	Ошибка: начинается со строчной буквы	System.Exception: "Incorrect attribute format"
5	1,3,7	1рисунок 1	Ошибка: начинается с цифры	System.Exception: "Incorrect attribute format"
6	1,3,8	Рисунок 1	Ошибка: начинается с латинской буквы	System.Exception: "Incorrect attribute format"
7	1,3,9	Рисунок1	Ошибка: отсутствует пробел	System.Exception: "Incorrect attribute format"
8	1,3,10	Рисунок 1.5	Ошибка: дробное число	System.Exception: "Incorrect attribute format"
9	1,3,11	Рисунок –1	Ошибка: отрицательное число	System.Exception: "Incorrect attribute format"
10	1,3,12	Рисунок 1	Ошибка: лишние пробелы	System.Exception: "Incorrect attribute format"

Продолжение таблицы 7

Номер теста	Классы Эквивалентности	Входные данные	Выходные данные	Реакция программы
11	1,3,13	РИСунок 1	Ошибка: начинается с двух заглавных букв	System.Exception: "Incorrect attribute format"
12	1,3,14	Рисунок	Ошибка: отсутствует число	System.Exception: "Incorrect attribute format"
13	1,2,15	Рисунок №1	Ошибка: содержит спецсимвол	System.Exception: "Incorrect attribute format"

Тестирование черным ящиком показало верную работу программы по отношению ко входящим xml-элементам и их атрибутам.

5.1.2 Модульное тестирование (Unit-тестирование)

Для тестирования корректности работы отдельных компонентов системы был применен способ тестирования Unit-тестами с помощью библиотеки Xunit.

Юнит-тесты позволяют быстро протестировать отдельные компоненты приложения независимо от остальной его части [15 – 17].

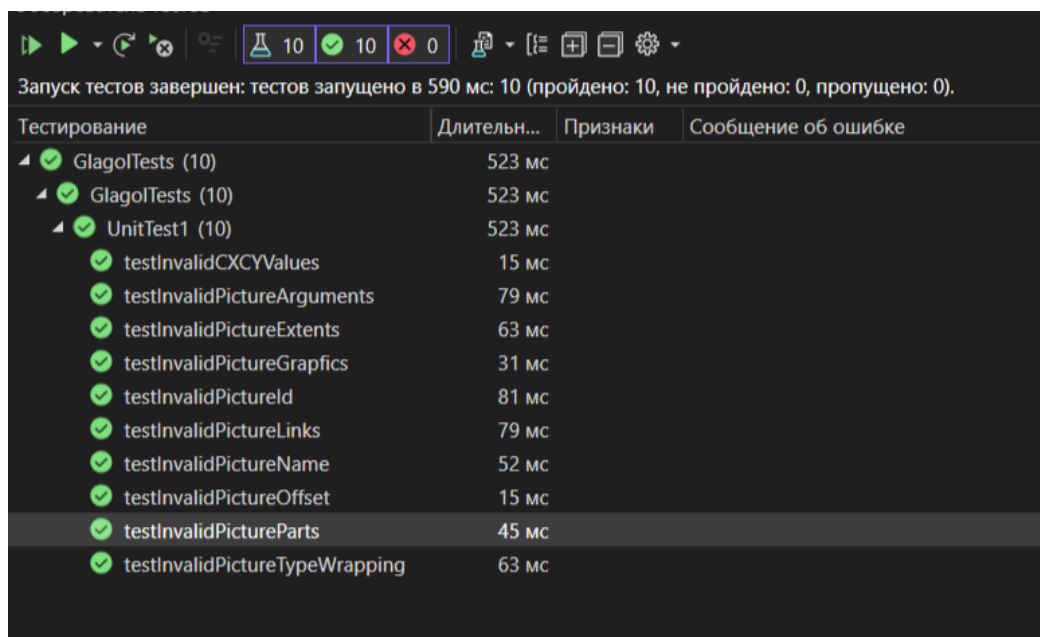
Пример кода для выполнения тестирования предоставлен на 1 листинге.

Листинг 1 – Код выполнения unit-теста, проверяющего корректность имени графического объекта

```
[Fact]
public void testInvalidPictureName(string correctName, string
existingName)
{
    Assert.Equal(correctName, existingName);
}
```

На вход тесту поступают заранее подготовленное правильное значение xml-атрибута, а также значение атрибута, полученное при работе

программы. Значения сравниваются и получается результат теста, показывающий сходство двух xml-атрибутов. Результаты работы тестов приведены на рисунке 15.



Тестирование	Длительн...	Признаки	Сообщение об ошибке
▲ ✓ GlagolTests (10)	523 мс		
▲ ✓ GlagolTests (10)	523 мс		
▲ ✓ UnitTest1 (10)	523 мс		
✓ testInvalidCXCValues	15 мс		
✓ testInvalidPictureArguments	79 мс		
✓ testInvalidPictureExtents	63 мс		
✓ testInvalidPictureGraphics	31 мс		
✓ testInvalidPictureId	81 мс		
✓ testInvalidPictureLinks	79 мс		
✓ testInvalidPictureName	52 мс		
✓ testInvalidPictureOffset	15 мс		
✓ testInvalidPictureParts	45 мс		
✓ testInvalidPictureTypeWrapping	63 мс		

Рисунок 15 – Тестирование unit-тестами

Код остальных unit-тестов приведен в приложении Г.

5.2 Анализ качества кода

Для получения представления о качестве написанного кода использовались сложность потока управления программы (цикломатическая сложность) и количественная метрика Холстеда.

5.2.1 Метрика цикломатической сложности

Измерение цикломатической сложности позволяет оценить качество приложения, методом получения всех возможных путей исполнения программы, представленных в виде графа потока управления.

Для примера высчитывания цикломатической сложности был взят код, осуществляющий выбор типа обтекания из графического объекта, подаваемого на вход. Код, участвующий в примере приведен на листинге 2.

Листинг 2 – Код выбора типа обтекания графического объекта класса

DrawingItem

```
public WrapType WrapType{
get{
if (_nestedRun.Drawing.DrawingObject.State is Anchor anchor) {
switch (anchor.Type){
case AnchorType.Square:
return WrapType.Square;
case AnchorType.Tight:
return WrapType.Tight;
case AnchorType.Through:
return WrapType.Through;
case AnchorType.TopAndBottom:
return WrapType.TopAndBottom;
case AnchorType.BehindText:
return WrapType.BehindText;
case AnchorType.InFrontText:
return WrapType.InFrontText;
}
}
return WrapType.Inline;
}
```

Полный код класса DrawingItem приведен в приложении Б.

По приведенному коду был построен граф цикломатической сложности, изображенный на рисунке 16 и приложении В.

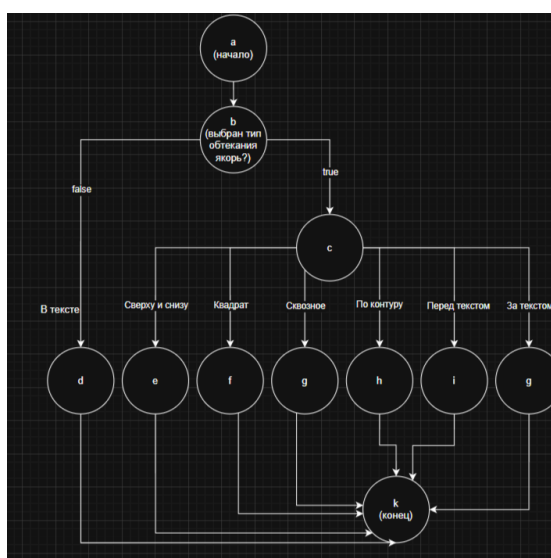


Рисунок 16 – Граф цикломатической сложности

Измерение цикломатической сложности осуществляется по формуле:

$$V(G) = e - n + 2 \times p,$$

где e – количество дуг;

n – количество вершин;

p – число компонент связности.

В данном случае количество дуг 16, а количество вершин 11. По полученным данным можно заполнить формулу цикломатической сложности. $16 - 11 + 2 \times 1 = 7$, что указывает на небольшую цикломатическую сложность. Исходя из значения метрики цикломатической сложности следует сделать вывод о небольшой сложности кода.

5.2.2 Метрика Холстеда

Метрика Холстеда дает получить количественную оценку сложности программного кода через анализ операторов и операндов [18]. В пример будет приведен метод `PredTranslateGraphics` класса `Anchor`, который находится в приложении Б и на листинге 3, приведенном ниже.

Листинг 3 – Код парсинга xml-части `DrawingItem` в классе `Anchor`

```
public void PredTranslateGraphics(Graphic graphic)
{
    switch (Type)
    {
        case AnchorType.None:
            throw new Exception();
            break;
        default:
            var spPr = (graphic.GraphicData.GraphicsObject.State as
Pic).ShapeProperties;
            var xfrm = spPr.Xml.Element(Namespace.a + "xfrm");
            var ext = xfrm.Element(Namespace.a + "ext");
            _extent.Cx = int.Parse(ext.Attribute("cx").Value);
            _extent.Cy = int.Parse(ext.Attribute("cy").Value);
            var docPr = (graphic.GraphicData.GraphicsObject.State as
Pic).NonVisualPictureProperties;
            var cNvPr = docPr.Xml.Element(Namespace.pic + "cNvPr");
            _docPr.Id = long.Parse(cNvPr.Attribute("id").Value);
            _docPr.Name = $"Рисунок {_docPr.Id}";
            break;
    }
}
```

На таблице 8 приведен словарь операторов и операндов, использующихся в методе `PredTranslateGraphics`.

Таблица 8 – Операторы и операнды метода PredTranslateGraphics

Операторы		Операнды	
Название	Количество	Название	Количество
{	2	Type	1
switch	1	AnchorType.None	1
case	1	spPr	2
throw	1	xfrm	3
break	2	ext	4
default	1	_extent.Cx	1
Element	3	_extent.Cy	1
Parse	3	docPr	5
Attribute	3	cNvPr	3
}	2	_docPr.Id	2
var	5	_docPr.Name	1
:	2	(graphic.GraphicData.GraphicsObject.State as Pic).ShapeProperties	1
;	12	(graphic.GraphicData.GraphicsObject.State as Pic).NonVisualPictureProperties	1
=	9	ext.Attribute("cx").Value	1
\$	1	ext.Attribute("cy").Value	1
"	14	Namespaces.a + "xfrm"	1
as	2	Namespaces.a + "ext"	1
+	3	Namespaces.pic + "cNvPr"	1
new	1	long.Parse(cNvPr.Attribute("id").Value)	1
(	11		
)	11		
n1 = 20	N1 = 90	n2 = 18	N2 = 32

В коде содержится 20 операторов и 18 операндов. Исходя из приведенных данных можно вычислить длину программы по формуле:

$$n=n_1+n_2,$$

где n – это длина программы

n_1 – это количество операторов;

n_2 – количество операндов.

В этом случае ответ будет $20 + 18 = 38$. Также можно вычислить объем программы по формуле:

$$V = N \times \log_2(n),$$

где V – это объем программы (бит);

N – общая длина;

n – словарь программы.

В этом случае ответ будет $122 \times \log_2(38) \approx 122 \cdot 5.2479 = 640.24$.

Метрики Холстеда показали малый объем кода, что свидетельствует об отсутствии лишнего кода. Приведенные метрики дали понимание об качестве кода, разработанного программного средства.

5.3. Планирование разработки и оценка бюджета

План разработки состоит из этапов производства программного средства, сроков, и ресурсов, затрачиваемых как на конкретную часть разработки, так и на проект в целом. План разработки необходим для организации работы и оценки рисков производства.

Для планирования разработки и оценки бюджета разрабатываемого программного средства была составлена таблица плана разработки в приложении Microsoft Office Project [19, 20].

Microsoft Office Project – это профессиональное программное обеспечение для управления проектами, разработанное компанией Microsoft. Microsoft Office Project дает возможность планировать задачи на разработку, распределять ресурсы, контролировать сроки выполнения задач и анализировать эффективность проекта. Приложение представляет удобную визуализацию и отчетность. По сравнению с аналогами, Microsoft Office Project требует более сложен в понимании для новичков, но предоставляет более обширный функционал.

План разработки приведен на рисунке 17.

	Название задачи	Длительность	Начало	Окончание
	Оценка бюджета ВКР	54 дней?	Вт 08.04.25	Чт 19.06.25
	Начало проекта	0 дней	Вт 08.04.25	Вт 08.04.25
	Анализ	7 дней	Пт 25.04.25	Пн 05.05.25
	Постановка задачи	1 день	Пт 25.04.25	Пт 25.04.25
	Анализ требований	1 день	Пн 28.04.25	Пн 28.04.25
	Обзор аналогичных программных средств	2 дней	Вт 29.04.25	Ср 30.04.25
	Реализация функциональных требований	1 день	Чт 01.05.25	Чт 01.05.25
	Реализация требований к интерфейсу	1 день	Пт 02.05.25	Пт 02.05.25
	Реализация требований к аппаратно-программным средствам	1 день	Пн 05.05.25	Пн 05.05.25
	Проектирование	2 дней	Вт 06.05.25	Ср 07.05.25
	Проектирование интерфейса программного средства	1 день	Вт 06.05.25	Вт 06.05.25
	Проектирование программного обеспечения	1 день	Ср 07.05.25	Ср 07.05.25
	Реализация	24 дней	Чт 08.05.25	Вт 10.06.25
	Реализация интерфейса	6 дней	Чт 08.05.25	Чт 15.05.25
	Реализация программного обеспечения	18 дней	Пт 16.05.25	Вт 10.06.25
	Тестирование	7 дней	Ср 11.06.25	Чт 19.06.25
	Функциональное тестирование	5 дней	Ср 11.06.25	Вт 17.06.25
	Анализ качества кода	2 дня	Ср 18.06.25	Чт 19.06.25
	Конец проекта	0 дней	Чт 19.06.25	Чт 19.06.25

Рисунок 17 – План разработки ПС

В таблице были описаны все задачи и подзадачи разработки средства. Для каждой задачи и подзадачи определена их длительность и связь между ними, а именно составлена последовательность выполнения задач и подзадач.

На основе созданного плана, была спроектирована диаграмма Ганта, изображенная на рисунке 18.

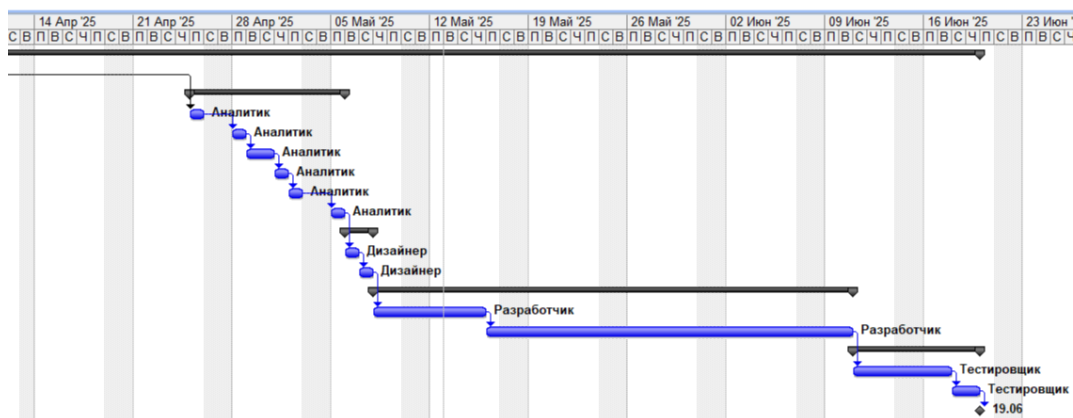


Рисунок 18 – Диаграмма Ганта

Проект разрабатывался только с помощью трудовых ресурсов. При разработке использовались несколько ролей:

- аналитик, ответственный за определение требований к проекту;
- дизайнер, ответственный за разработку интерфейса;

- программист, ответственный за разработку;
- тестировщик, ответственный за отладку.

Для определения стоимости разработки были составлены рабочие спецификации, изображенные на рисунке 19.

Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных
Аналитик	Трудовой		А		100%	77,00 Р/ч	0,00 Р/ч
Дизайнер	Трудовой		Д		100%	62,00 Р/ч	0,00 Р/ч
Разработчик	Трудовой		Р		100%	70,00 Р/ч	0,00 Р/ч
Тестировщик	Трудовой		Т		100%	59,00 Р/ч	0,00 Р/ч

Рисунок 19 – Трудовые ресурсы разработки

На рисунке 20 представлен отчет стоимости разработки программного средства.

Отчет о бюджете от Ср 04.06.25
ДиаграммаГанта

Ид.	Название задачи	Фиксированные затраты	Начисление фикс. затрат	Общие затраты	Базовые	Отклонение
15	Реализация программного обеспече	0,00 Р	Пропорциональное	10 080,00 Р	0,00 Р	10 080,00 Р
14	Реализация интерфейса	0,00 Р	Пропорциональное	3 360,00 Р	0,00 Р	3 360,00 Р
17	Функциональное тестирование	0,00 Р	Пропорциональное	2 360,00 Р	0,00 Р	2 360,00 Р
6	Обзор аналогичных программных ср	0,00 Р	Пропорциональное	1 232,00 Р	0,00 Р	1 232,00 Р
16	Анализ качества кода	0,00 Р	Пропорциональное	944,00 Р	0,00 Р	944,00 Р
4	Постановка задачи	0,00 Р	Пропорциональное	616,00 Р	0,00 Р	616,00 Р
5	Анализ требований	0,00 Р	Пропорциональное	616,00 Р	0,00 Р	616,00 Р
7	Реализация функциональных требо	0,00 Р	Пропорциональное	616,00 Р	0,00 Р	616,00 Р
8	Реализация требований интерфейс	0,00 Р	Пропорциональное	616,00 Р	0,00 Р	616,00 Р
9	Реализация требований к аппаратнс	0,00 Р	Пропорциональное	616,00 Р	0,00 Р	616,00 Р
11	Проектирование интерфейса програ	0,00 Р	Пропорциональное	496,00 Р	0,00 Р	496,00 Р
12	Проектирование программного обес	0,00 Р	Пропорциональное	496,00 Р	0,00 Р	496,00 Р
2	Начало проекта	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
19	Конец проекта	0,00 Р	Пропорциональное	0,00 Р	0,00 Р	0,00 Р
		0,00 Р		22 048,00 Р	0,00 Р	22 048,00 Р

Рисунок 20 – Стоимость разработки программного средства

Исходя из приведенных данных плана разработки и плана трудовых ресурсов, стоимость разработки составила 22048 рублей.

Заключение

Целью выпускной квалификационной работы являлась разработка программного средства для обтекания текстом графического объекта. При выполнении работы были пройдены следующие этапы:

- произведен анализ предметной области и выделены основные задачи, которые необходимо было решить в процессе создания системы;
- построена модель предметной области;
- спроектирован и реализован пользовательский интерфейс;
- реализованы механизмы работы с построением, загрузкой и сохранением html-страницы;
- разработан обработчик html-страницы;
- проведены тестирование и отладка системы;
- проведена оценка стоимости разработки программного средства.

В результате выполнения проекта было спроектировано, разработано и отлажено программное средство, способное автоматизировать обтекание графических объектов в текстовом редакторе. Приложение способно обрабатывать виды обтекания «В тексте», «Квадрат», «По контуру», «Сквозное», «Сверху и снизу», «За текстом», «Перед текстом».

Было проведено тестирование программного средства, в результате которого установилось полное соответствие техническому заданию.

Список использованных источников

1. Microsoft Word Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Microsoft_Word (дата обращения: 17.04.2025).
2. Официальный сайт Cryptosoft [Электронный ресурс]. URL: <https://cryptosoft.ru/> (дата обращения: 17.04.2025).
3. Егоров В.Ю., Гуреев В.О. Администрирование защищенной операционной системы QR ОС. – Пенза, 2024. – 552 с
4. Office Open XML Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Office_Open_XML (дата обращения: 17.04.2025).
5. Что такое .docx файл и чем он отличается от файла .doc? Дзен. [Электронный ресурс]. URL: <https://dzen.ru/a/WuGaugCz3SwbLj7h> (дата обращения: 17.04.2025).
6. Программная закладка Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Программная_закладка (дата обращения: 17.04.2025).
7. Microsoft Office 365: Документация по работе с графическими объектами в Word [Электронный ресурс]. URL: <https://support.microsoft.com/ru-ru/office/добавление-рисунка-в-документ-348a8390-c32e-43d0-942c-b20ad11dea6f> (дата обращения: 17.04.2025).
8. Гурьянов Л.В., Князев В.Н., Балашова И.Ю., Дзюба Е.А., Попова Н.А., Такташкин Д.В., Самуйлов С. В. Выпускная квалификационная работа: стандарты и программные технологии. Под ред. Макарычева П.П. – Пенза: Изд-во ПГУ, 2019. – 182.с.
9. SSD и жесткий диск, в чем разница? WONDERSHARE [Электронный ресурс]. URL: <https://recoverit.wondershare.com.ru/harddrive-recovery/ssd-vs-hdd-whats-the-difference.html> (дата обращения: 17.04.2025).
10. Методологии разработки программного обеспечения Часть 3. Rational Unified Process КомпьютерПресс [Электронный ресурс]. URL: <https://compress.ru/article.aspx?id=9633> (дата обращения: 17.04.2025).

11. Полное руководство по C# METANIT. [Электронный ресурс]. URL: <https://metanit.com/sharp/> (дата обращения: 17.04.2025).
12. Использование диаграммы классов UML при проектировании и документировании программного обеспечения Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/572234/> (дата обращения: 17.04.2025).
13. Диаграмма классов Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Диаграмма_классов (дата обращения: 17.04.2025).
14. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Паттерны объектно-ориентированного проектирования – Пер. с англ. Слинкин А. – СПб: Питер, 2021. – 448 с.
15. Введение в юнит-тесты METANIT [Электронный ресурс]. URL: <https://metanit.com/sharp/aspnet5/22.1.php> (дата обращения: 17.04.2025).
16. Модульное тестирование C# в .NET с помощью dotnet test и xUnit [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/core/testing/unit-testing-csharp-with-xunit> (дата обращения: 17.04.2025).
17. Модульные тесты на C# [Электронный ресурс]. URL: <https://newtechaudit.ru/unit-testi-na-c/> (дата обращения: 17.04.2025).
18. Программный код и его метрики Хабр [Электронный ресурс]. URL: <https://habr.com/ru/companies/intel/articles/106082/> (дата обращения: 17.04.2025).
19. Кудрявцев Е.М. Microsoft Project. Методы сетевого планирования и управления проектом – ДМК Пресс, 2019. – 241 с.
20. Использование MS Project для управления проектами в разработке ПО [Электронный ресурс]. URL: <https://habr.com/ru/articles/151593/> (дата обращения: 17.04.2025).

Приложение А

(справочное)

Перечень сокращений и базовых понятий

Средства криптографической защиты информации (СКЗИ) – программы и устройства, которые шифруют и дешифруют информацию и проверяют, вносились ли в нее изменения.

Средства защиты информации (СЗИ) – совокупность методов и технологий, направленных на сохранение конфиденциальности, целостности и доступности данных.

eXtensible Markup Language (XML) – расширяемый язык разметки, предназначенный для хранения и передачи данных в структурированном виде. Данные находятся внутри тегов, которые помогают организовать информацию в логической иерархии, понятной человеку и компьютеру.

Программное средство (ПС) – это совокупность программ, данных и инструкций, предназначенных для выполнения определенных задач.

Информационная система (ИС) – это организованная совокупность аппаратного и программного обеспечения, данных, людей и процессов, предназначенная для сбора, обработки, хранения и передачи информации.

Методология разработки программного обеспечения (RUP) – это четко определенный процесс (технологическая процедура), описывающая структуру жизненного цикла проекта, роли и ответственности отдельных исполнителей, выполняемые ими задачи и используемые в процессе разработки модели, отчеты.

Унифицированный язык моделирования (UML) – это стандартизированный графический язык для визуализации, проектирования и документирования объектно-ориентированных систем.

Модель–Представление–Контроллер (MVC) – это архитектурный паттерн в разработке ПО, разделяющий приложение на три компонента Model – отвечающий за данные и бизнес–логику, View – который отображает интерфейс и передает действия пользователя и Controller, обрабатывающий данные и управляющий View для упрощения поддержки и тестирования.

Приложение Б
(обязательное)
Листинг программы

```
protected void CustomizeRibbon()
{
    tiSections.Click += (sender, args) =>
    {
        if (!LockHandlers)
        {
            Processor.AddSection(Selection,
            SectionBreakType.defaultNextPage);
        }
    };
    InitWrapTypeStrip(tnTextWrapping);
}

```

```
protected void InitWrapTypeStrip(Thumbnail toCaseType)
{
    //tbFormat.Visible = false;
    PopupMenuStrip caseTypeStrip = new PopupMenuStrip(null);
    toCaseType.ContextMenu = caseTypeStrip;
    var ti = caseTypeStrip.AddItem("В тексте", Resource.flow_text);
    ti.Tag = Office.Wordprocessing.Models.WrapType.Inline;
    ti.Click += WrapTypeClick;
    caseTypeStrip.AddSeparator();
    ti = caseTypeStrip.AddItem("Квадрат", Resource.flow_frame);
    ti.Tag = Office.Wordprocessing.Models.WrapType.Square;
    ti.Click += WrapTypeClick;
    ti = caseTypeStrip.AddItem("По контуру", Resource.flow_through);
    ti.Tag = Office.Wordprocessing.Models.WrapType.Tight;
    ti.Click += WrapTypeClick;
    ti = caseTypeStrip.AddItem("Сквозное", Resource.flow_contour);
    ti.Tag = Office.Wordprocessing.Models.WrapType.Through;
    ti.Click += WrapTypeClick;
    ti = caseTypeStrip.AddItem("Сверху и снизу",
Resource.flow_updown);
    ti.Tag = Office.Wordprocessing.Models.WrapType.TopAndBottom;
    ti.Click += WrapTypeClick;
    caseTypeStrip.AddSeparator();
    ti = caseTypeStrip.AddItem("За текстом", Resource.flow_behind);
    ti.Tag = Office.Wordprocessing.Models.WrapType.BehindText;
    ti.Click += WrapTypeClick;
    ti = caseTypeStrip.AddItem("Перед текстом", Resource.flow_front);
    ti.Tag = Office.Wordprocessing.Models.WrapType.InFrontText;
    ti.Click += WrapTypeClick;
}

```

```
public void WrapTypeClick(object sender, EventArgs e)
{
    if (!LockHandlers)
    {
        //поменяли только стиль написания
        if (Document.IsSigned)
            return;
        WordProcessingPictureView pic = null;
        var selection = PageAdapter.SelectionManager;
        if (_SelectedItem is WordProcessingPictureView pv)

```

```

    {
        pic = pv;
    }
    WrapType wrapType = (WrapType)((Control)sender).Tag;
    ChangeWrapTypeCommand(pic.Item as PictureItem, wrapType);
}

public void ChangeWrapTypeCommand(PictureItem item, WrapType
wrapType)
{
    if (item == null)
        throw new Exception("Item не может быть null");
    Processor.ChangeWrapType(item, wrapType);
}

public partial class WordProcessor : DocumentProcessor<Document,
DocX>
{
    public void ChangeWrapType(PictureItem item, WrapType wrapType)
    {
        if (item == null)
            throw new Exception("item не может быть null");
        //Selection = SelectionManager.GetSelection(selection);
        RunCommand(new ChangeWrapTypeCommand(wrapType, item));
    }
}

public abstract partial class DocumentProcessor :
IDocumentProcessor
{
    protected virtual void RunCommand(ICommand command)
    {
        OpsPerformer.RunCommand(command);
    }
}

public class : ICommandManager
{
    public void RunCommand(ICommand command)
    {
        var commandName = command.GetType().Name;
        OfficeLogger.Instance.Info($"Prepare command {commandName}");
        try
        {
            _undoList.Push(new List<ICommand> { command });
            _redoList.Clear();

            _document.SuspendChangeNotifications();
            command.Prepare(_document);
            OfficeLogger.Instance.Info($"Run command {commandName}");
            command.Run(_document);
        }
    }
}

```

```

    }
    catch (Exception e)
    {
        OfficeLogger.Instance.Error(e);
        _undoList.Pop();
        throw;
    }
    finally
    {
        OfficeLogger.Instance.Info($"Leave command {commandName}");
    }
}

public class ChangeWrapTypeCommand : ICommand
{
    WrapType _oldFormatting;
    WrapType _newFormatting;
    private PictureItem _pic;
    private IDataSelection _s;
    public ChangeWrapTypeCommand(WrapType wp, PictureItem item)
    {
        _newFormatting = wp;
        _pic = item;
    }

    public void Prepare(BaseDocumentCommon doc)
    {
        //var index = _s.Start.IndexInParagraph;
        //_pic = _s.Start.Paragraph.Items[index] as PictureItem;
        _oldFormatting = _pic.WrapType;
    }

    public void Run(BaseDocumentCommon doc)
    {
        _pic.WrapType = _newFormatting;
        doc.SetData(Operation.ChangeImage, _s);
    }

    public void Undo(BaseDocumentCommon doc)
    {
        _pic.WrapType = _oldFormatting;
        doc.SetData(Operation.ChangeImage);
    }
}

public abstract class DrawingItem : ParagraphRun<IRunFormatting>,
IDrawingItem
{
    private readonly DrawingRun _nestedRun;

```

```

    protected DrawingItem(BaseDocumentCommon doc, DrawingRun
nestedRun, IParagraphItemContainer par) : base(doc, nestedRun,
par)
    {
        _nestedRun = nestedRun;
    }

public WrapType WrapType
{
    get
    {
        if (_nestedRun.Drawing.DrawingObject.State is Anchor anchor)
        {
            switch (anchor.Type)
            {
                case AnchorType.Square:
                    return WrapType.Square;
                case AnchorType.Tight:
                    return WrapType.Tight;
                case AnchorType.Through:
                    return WrapType.Through;
                case AnchorType.TopAndBottom:
                    return WrapType.TopAndBottom;
                case AnchorType.BehindText:
                    return WrapType.BehindText;
                case AnchorType.InFrontText:
                    return WrapType.InFrontText;
            }
        }
        return WrapType.Inline;
    }
    set
    {
        var graphic = _nestedRun.Drawing.DrawingObject.State.Graphic;

        switch (value)
        {
            case WrapType.Inline:
                _nestedRun.Drawing.DrawingObject.Type =
DrawingObjects.inline;
                break;
            case WrapType.Square:
                _nestedRun.Drawing.DrawingObject.Type =
DrawingObjects.anchor;
                var anchor = (_nestedRun.Drawing.DrawingObject.State as
Anchor);
                anchor.Type = AnchorType.Square;
                anchor.PredTranslateGraphics(graphic);
                anchor.TranslateGraphics(graphic);
                anchor.PostTranslateGraphics();

                break;
            case WrapType.Tight:

```

```

        _nestedRun.Drawing.DrawingObject.Type =
DrawingObjects.anchor;
        var tight = (_nestedRun.Drawing.DrawingObject.State as
Anchor);
        tight.Type = AnchorType.Tight;
        tight.PredTranslateGraphics(graphic);
        tight.TranslateGraphics(graphic);
        tight.PostTranslateGraphics();
        //(_nestedRun.Drawing.DrawingObject.State as Anchor).Type =
AnchorType.Tight;
        break;
        case WrapType.Through:
            _nestedRun.Drawing.DrawingObject.Type =
DrawingObjects.anchor;
            var through = (_nestedRun.Drawing.DrawingObject.State as
Anchor);
            through.Type = AnchorType.Through;
            through.PredTranslateGraphics(graphic);
            through.TranslateGraphics(graphic);
            through.PostTranslateGraphics();
            //(_nestedRun.Drawing.DrawingObject.State as Anchor).Type =
AnchorType.Through;
            break;
            case WrapType.TopAndBottom:
                _nestedRun.Drawing.DrawingObject.Type =
DrawingObjects.anchor;
                var topAndBottom = (_nestedRun.Drawing.DrawingObject.State
as Anchor);
                topAndBottom.Type = AnchorType.TopAndBottom;
topAndBottom.PredTranslateGraphics(graphic);
                topAndBottom.TranslateGraphics(graphic);
                topAndBottom.PostTranslateGraphics();
                //(_nestedRun.Drawing.DrawingObject.State as Anchor).Type =
AnchorType.TopAndBottom;
                break;
                case WrapType.BehindText:
                    _nestedRun.Drawing.DrawingObject.Type =
DrawingObjects.anchor;
                    var behindText = (_nestedRun.Drawing.DrawingObject.State as
Anchor);
                    behindText.Type = AnchorType.BehindText;
                    behindText.PredTranslateGraphics(graphic);
                    behindText.TranslateGraphics(graphic);
                    behindText.PostTranslateGraphics();
                    //(_nestedRun.Drawing.DrawingObject.State as Anchor).Type =
AnchorType.BehindText;
                    break;
                    case WrapType.InFrontText:
                        _nestedRun.Drawing.DrawingObject.Type =
DrawingObjects.anchor;
                        var inFrontText = (_nestedRun.Drawing.DrawingObject.State as
Anchor);

```

```

        inFrontText.Type = AnchorType.InFrontText;
inFrontText.PredTranslateGraphics(graphic);
        inFrontText.TranslateGraphics(graphic);
        inFrontText.PostTranslateGraphics();
        //(_nestedRun.Drawing.DrawingObject.State as Anchor).Type =
AnchorType.InFrontText;
        break;
    }
}
}

public virtual float? Width
{
    get
    {
        var xValue = _nestedRun.Drawing.DrawingObject.State.Extent.Cx;
        if (xValue.HasValue)
        {
            return UnitConverter.EMUToPoint(xValue);
        }

        return null;
    }
    set
    {
        _nestedRun.Drawing.DrawingObject.State.Extent.Cx = value !=
null ? UnitConverter.PointToEMU(value) : null;
    }
}

public virtual float? Height
{
    get
    {
        var xValue = _nestedRun.Drawing.DrawingObject.State.Extent.Cy;
        if (xValue.HasValue)
        {
            return UnitConverter.EMUToPoint(xValue);
        }

        return null;
    }
    set
    {
        _nestedRun.Drawing.DrawingObject.State.Extent.Cy = value != null
? UnitConverter.PointToEMU(value) : null;
    }
}

public string Name
{
    get { return _nestedRun.Drawing.DrawingObject.State is Inline
inline ? inline.DocPr.Name : null; }
}

```

```

    set
    {
        if (_nestedRun.Drawing.DrawingObject.State is Inline inline)
            inline.DocPr.Name = value;
    }
}

public abstract Bitmap GetPreview();
}

public enum WrapType
{
    Inline,
    Square,
    Tight,
    Through,
    TopAndBottom,
    BehindText,
    InFrontText
}
}
{
public interface IOfficeXElementSwitcher
{
}

public abstract class OfficeXElementSwitcher<TEnum, TNestedObject>
: IOfficeXElementSwitcher where TNestedObject : class, IXmlEntity
{
    protected readonly MainDocumentPackage Document;

    protected abstract Dictionary<TEnum, Type> NestedObjects { get; }
    private readonly OfficeXElement _parentElement;
    private readonly PackagePart _part;
    private TEnum _type;

    protected OfficeXElementSwitcher(MainDocumentPackage document,
    OfficeXElement parentElement)
    {
        Document = document;
        _parentElement = parentElement;
        State = Init();
    }

    private TNestedObject Init()
    {
        TNestedObject currentObj = null;
        var t = Nullable.GetUnderlyingType(typeof(TEnum));
        if (t == null)
            throw new Exception("TEnum must be nullable");
        foreach (var enumValue in Enum.GetValues(t))
        {

```

```

        var xName = t.GetField(Enum.GetName(t,
enumValue)).GetCustomAttribute<XmlNameAttribute>()?.XmlName;
        if (xName == null) throw new Exception("Incorrect enum. Add
XmlName Attribute to Enum.");
        var xElement = _parentElement.Xml.Element(GetNamespace(xName)
+ xName);
        if (_parentElement.Xml.Element(GetNamespace(xName) + xName) !=
null)
        {
            if (currentObj == null)
            {
                _type = (TEnum)enumValue;
                currentObj = Switch(_type, xElement);
            }
            else
            {
                throw new Exception("Найдено несколько элементов из этой
группы");
            }
        }
    }
}

return currentObj;
}

```

```

protected OfficeXElementSwitcher(MainDocumentPackage document,
OfficeXElement parentElement, PackagePart part):this(document,
parentElement)
{
    _part = part;
}

```

```

public TNestedObject State { get; private set; }

```

```

public TEnum Type
{
    get { return _type; }
    set
    {
        _type = value;

        if (State?.Xml.Parent != null)
            State.Xml.Remove();

        if (_type != null)
        {
            State = Switch(value);
            if (State is IAttachableOnSet attachable)
            {
                attachable.Attach();
            }
        }
        else
        {

```

```

        _parentElement.Xml.Add(State.Xml);
    }
}
}

protected virtual XNamespace GetNamespace(string xName)
{
    return Document.N;
}

private TNestedObject Switch(TEnum dest)
{
    var t = dest.GetType();
    var xmlName = t.GetField(Enum.GetName(t,
dest)).GetCustomAttribute<XmlNameAttribute>().XmlName;
    return Switch(dest, new
CustomOfficeXElement(GetNamespace(xmlName) + xmlName));
}

private TNestedObject Switch(TEnum dest, IOfficeXElement xElement)
{
    if (!NestedObjects.TryGetValue(dest, out var type))
        return null;
    try
    {
        if (_part != null)
        {
            if (type.IsSubclassOf(typeof(AttachableOnSet)))
            {
                return (TNestedObject)Activator.CreateInstance(type,
Document, xElement, _parentElement, _part);
            }
            else
            {
                return (TNestedObject)Activator.CreateInstance(type,
Document, xElement, _part);
            }
        }
        else
        {
            try
            {
                if (type.IsSubclassOf(typeof(AttachableOnSet)))
                {
                    return (TNestedObject)Activator.CreateInstance(type,
Document, xElement, _parentElement);
                }
                else
                {
                    return (TNestedObject)Activator.CreateInstance(type,
Document, xElement);
                }
            }
        }
    }
}

```

```

    }
    catch (Exception e)
    {
        Console.WriteLine(e);
        throw;
    }
}
}
catch (Exception)
{
    throw;
}
}
}

/// <summary>
/// Уведомляет, что данный член класса является объектом
OfficeXElementSwitcher
/// </summary>
[AttributeUsage(AttributeTargets.Method |
AttributeTargets.Property, Inherited = false)]
sealed class SwitcherElement : Attribute
{
    // See the attribute guidelines at
    // http://go.microsoft.com/fwlink/?LinkId=85236
    public SwitcherElement()
    {
    }
}

public abstract class OfficeXElementSwitcher<TEnum,
TNestedObject> : IOOfficeXElementSwitcher where TNestedObject :
class, IXmlEntity
{
    private TNestedObject Switch(TEnum dest)
    {
        var t = dest.GetType();
        var xmlName = t.GetField(Enum.GetName(t,
dest)).GetCustomAttribute<XmlNameAttribute>().XmlName;
        return Switch(dest, new
CustomOfficeXElement(GetNamespace(xmlName) + xmlName));
    }
}

namespace Office.OOXML.MediaContent
{
    public class Anchor : Inline
    {
        public Anchor(MainDocumentPackage document, IOOfficeXElement xml,
OfficeXElement parentXml) : base(document, xml, parentXml)
        {

```

```

        _square = new WrapSquare(document,
            xml.Element(Namespaces.wp + "wrapSquare") ?? new
CustomOfficeXElement(Namespaces.wp + "wrapSquare"), this);
        _tight = new WrapThroughTight(document,
            xml.Element(Namespaces.wp + "wrapTight") ?? new
CustomOfficeXElement(Namespaces.wp + "wrapTight"), this);
        _through = new WrapThroughTight(document,
            xml.Element(Namespaces.wp + "wrapThrough") ?? new
CustomOfficeXElement(Namespaces.wp + "wrapThrough"), this);
        _tb = new WrapTopBottom(document,
            xml.Element(Namespaces.wp + "wrapTopAndBottom") ?? new
CustomOfficeXElement(Namespaces.wp + "wrapTopAndBottom"), this);
        _fbt = new WrapFrontBackText(document,
            xml.Element(Namespaces.wp + "wrapNone") ?? new
CustomOfficeXElement(Namespaces.wp + "wrapNone"), this);
        _posH = new PositionH(document,
            xml.Element(Namespaces.wp + "positionH") ?? new
CustomOfficeXElement(Namespaces.wp + "positionH"), this);
        _posV = new PositionV(document,
            xml.Element(Namespaces.wp + "positionV") ?? new
CustomOfficeXElement(Namespaces.wp + "positionV"), this);
        _simplePos = new SimplePos(document,
            xml.Element(Namespaces.wp + "simplePos") ?? new
CustomOfficeXElement(Namespaces.wp + "simplePos"), this);
        _extent = new Extent(document,
            xml.Element(Namespaces.wp + "extent") ?? new
CustomOfficeXElement(Namespaces.wp + "extent"), this);
        _docPr = new DocPr(document, xml.Element(Namespaces.wp +
"docPr") ?? new CustomOfficeXElement(Namespaces.wp + "docPr"),
this);
    }

```

```

private int? picId;
private int? cx;
private int? cy;

```

```

private PositionH _posH;
private PositionV _posV;
private SimplePos _simplePos;
private Extent _extent;
private DocPr _docPr;

```

```

private WrapSquare _square { get; set; }

```

```

private WrapThroughTight _tight { get; set; }

```

```

private WrapThroughTight _through { get; set; }

```

```

private WrapTopBottom _tb { get; set; }

```

```

private WrapFrontBackText _fbt { get; set; }

```

```

public AnchorType Type

```

```

{
    get
    {
        if (_square.IsAttached)
        {
            return AnchorType.Square;
        }

        if (_tight.IsAttached)
        {
            return AnchorType.Tight;
        }

        if (_through.IsAttached)
        {
            return AnchorType.Through;
        }

        if (_tb.IsAttached)
        {
            return AnchorType.TopAndBottom;
        }

        if (_fbt.IsAttached)
        {
            return AnchorType.BehindText;
        }

        if (_fnt.IsAttached)
        {
            return AnchorType.InFrontText;
        }

        return AnchorType.None;
    }
    set
    {
        switch (value)
        {
            case AnchorType.Square:
                DistT = 0;
                DistB = 0;
                DistL = 114300;
                DistR = 114300;
                SimplePos = false;
                RelativeHeight = 251658240;
                Locked = false;
                BehindDoc = false;
                LayoutInCell = true;
                AllowOverlap = true;
                _simplePos.X = 0;
                _simplePos.Y = 0;
                _posH.Attach();

```

```

_posH.RelativeFrom = RelativeFromH.Page;
_posH.PosOffset = 0;
_posV.Attach();
_posV.RelativeFrom = RelativeFromV.Page;
_posV.PosOffset = 0;
_extent.Attach();
_square.WrapText = WrapText.BothSides;
if (!_square.IsAttached)
    _square.Attach();
_docPr.Attach();

_tight.Detach();
_tight.WrapText = null;
_tight.WrapPolygon.Detach();
_through.Detach();
_through.WrapText = null;
_through.WrapPolygon.Detach();
_tb.Detach();
_fbt.Detach();
WrapNone = false;
break;
case AnchorType.Tight:
    DistT = 0;
    DistB = 0;
    DistL = 114300;
    DistR = 114300;
    SimplePos = false;
    RelativeHeight = 251658240;
    BehindDoc = true;
    Locked = false;
    LayoutInCell = true;
    AllowOverlap = true;
    _simplePos.X = 0;
    _simplePos.Y = 0;
    _posH.Attach();
    _posH.RelativeFrom = RelativeFromH.Page;
    _posH.PosOffset = 0;
    _posV.Attach();
    _posV.RelativeFrom = RelativeFromV.Page;
    _posV.PosOffset = 0;
    _extent.Attach();
    if (!_tight.IsAttached)
    {
        _tight.WrapText = WrapText.BothSides;
        _tight.WrapPolygon.Attach();
        _tight.Attach();
        _tight.WrapPolygon.Start.X = 0;
        _tight.WrapPolygon.Start.Y = 0;
        _tight.WrapPolygon.AddNewPoint(0, 100);
        _tight.WrapPolygon.AddNewPoint(100, 100);
        _tight.WrapPolygon.AddNewPoint(100, 0);
        _tight.WrapPolygon.AddNewPoint(0, 0);
    }

```

```

}
_docPr.Attach();

_square.Detach();
_square.WrapText = null;
_through.Detach();
_through.WrapText = null;
_through.WrapPolygon.Detach();
_tb.Detach();
_fbt.Detach();
WrapNone = false;
break;
case AnchorType.Through:
    DistT = 0;
    DistB = 0;
    DistL = 114300;
    DistR = 114300;
    SimplePos = false;
    RelativeHeight = 251658240;
    BehindDoc = true;
    Locked = false;
    LayoutInCell = true;
    AllowOverlap = true;
    _simplePos.X = 0;
    _simplePos.Y = 0;
    _posH.Attach();
    _posH.RelativeFrom = RelativeFromH.Page;
    _posH.PosOffset = 0;
    _posV.Attach();
    _posV.RelativeFrom = RelativeFromV.Page;
    _posV.PosOffset = 0;
    _extent.Attach();
    if (!_through.IsAttached)
    {
        _through.WrapText = WrapText.BothSides;
        _through.WrapPolygon.Attach();
        _through.Attach();
        _through.WrapPolygon.Start.X = 0;
        _through.WrapPolygon.Start.Y = 0;
        _through.WrapPolygon.AddNewPoint(0, 100);
        _through.WrapPolygon.AddNewPoint(100, 100);
        _through.WrapPolygon.AddNewPoint(100, 0);
        _through.WrapPolygon.AddNewPoint(0, 0);
    }
    _docPr.Attach();

_square.Detach();
_square.WrapText = null;
_tight.Detach();
_tight.WrapText = null;
_tight.WrapPolygon.Detach();

```

```

    _tb.Detach();
    _fbt.Detach();
    WrapNone = false;
    break;
case AnchorType.TopAndBottom:
    DistT = 0;
    DistB = 0;
    DistL = 114300;
    DistR = 114300;
    SimplePos = false;
    RelativeHeight = 251658240;
    BehindDoc = true;
    Locked = false;
    LayoutInCell = true;
    AllowOverlap = true;
    _simplePos.X = 0;
    _simplePos.Y = 0;
    _posH.Attach();
    _posH.RelativeFrom = RelativeFromH.Page;
    _posH.PosOffset = 0;
    _posV.Attach();
    _posV.RelativeFrom = RelativeFromV.Page;
    _posV.PosOffset = 0;
    _extent.Attach();
    if (!_tb.IsAttached)
        _tb.Attach();
    _docPr.Attach();

    _square.Detach();
    _square.WrapText = null;
    _tight.Detach();
    _tight.WrapText = null;
    _tight.WrapPolygon.Detach();
    _through.Detach();
    _through.WrapText = null;
    _through.WrapPolygon.Detach();
    _fbt.Detach();
    WrapNone = false;
    break;
case AnchorType.BehindText:
    DistT = 0;
    DistB = 0;
    DistL = 114300;
    DistR = 114300;
    SimplePos = false;
    RelativeHeight = 251658240;
    BehindDoc = true;
    Locked = false;
    LayoutInCell = true;
    AllowOverlap = true;
    _simplePos.X = 0;
    _simplePos.Y = 0;
    _posH.Attach();

```

```

_posH.RelativeFrom = RelativeFromH.Page;
_posH.PosOffset = 0;
_posV.Attach();
_posV.RelativeFrom = RelativeFromV.Page;
_posV.PosOffset = 0;
_extent.Attach();
if (!_fbt.IsAttached)
    _fbt.Attach();
_docPr.Attach();

_square.Detach();
_square.WrapText = null;
_tight.Detach();
_tight.WrapText = null;
_tight.WrapPolygon.Detach();
_through.Detach();
_through.WrapText = null;
_through.WrapPolygon.Detach();
_tb.Detach();
break;
case AnchorType.InFrontText:
    DistT = 0;
    DistB = 0;
    DistL = 114300;
    DistR = 114300;
    SimplePos = false;
    RelativeHeight = 251658240;
    BehindDoc = false;
    Locked = false;
    LayoutInCell = true;
    AllowOverlap = true;
    _simplePos.X = 0;
    _simplePos.Y = 0;
    _posH.Attach();
    _posH.RelativeFrom = RelativeFromH.Page;
    _posH.PosOffset = 0;
    _posV.Attach();
    _posV.RelativeFrom = RelativeFromV.Page;
    _posV.PosOffset = 0;
    _extent.Attach();
    if (!_fbt.IsAttached)
        _fbt.Attach();
    _docPr.Attach();

_square.Detach();
_square.WrapText = null;
_tight.Detach();
_tight.WrapText = null;
_tight.WrapPolygon.Detach();
_through.Detach();
_through.WrapText = null;

```

```

        _through.WrapPolygon.Detach();
        _tb.Detach();
        break;
    case AnchorType.None:
        break;
    }
}
}

public void PredTranslateGraphics(Graphic graphic)
{
    switch (Type)
    {
        case AnchorType.None:
            throw new Exception();
            break;
        default:
            var spPr = (graphic.GraphicData.GraphicsObject.State as
Pic).ShapeProperties;
            var xfrm = spPr.Xml.Element(Namespaces.a + "xfrm");
            var ext = xfrm.Element(Namespaces.a + "ext");
            _extent.Cx = int.Parse(ext.Attribute("cx").Value);
            _extent.Cy = int.Parse(ext.Attribute("cy").Value);

            var docPr = (graphic.GraphicData.GraphicsObject.State as
Pic).NonVisualPictureProperties;
            var cNvPr = docPr.Xml.Element(Namespaces.pic + "cNvPr");
            _docPr.Id = long.Parse(cNvPr.Attribute("id").Value);
            _docPr.Name = $"Рисунок {_docPr.Id}";
            break;
        }
    }

    public void TranslateGraphics(Graphic newGraphic)
    {
        Graphic g = new Graphic(Document, new
CustomOfficeXElement(newGraphic.Xml), this);
        if (!g.IsAttached)
        {
            g.Attach();
            Graphic = g;
        }

        switch (Type)
        {
            case AnchorType.Square:
                break;
            case AnchorType.Tight:
                break;
            case AnchorType.Through:
                break;
            case AnchorType.TopAndBottom:

```

```

        break;
    case AnchorType.BehindText:
        break;
    case AnchorType.InFrontText:
        break;
    case AnchorType.None:
        break;
    default:
        throw new ArgumentOutOfRangeException();
    }
}

public void PostTranslateGraphics()
{
    switch (Type)
    {
        case AnchorType.Square:
            break;
        case AnchorType.Tight:
            break;
        case AnchorType.Through:
            break;
        case AnchorType.TopAndBottom:
            break;
        case AnchorType.BehindText:
            break;
        case AnchorType.InFrontText:
            break;
        case AnchorType.None:
            break;
        default:
            throw new ArgumentOutOfRangeException();
    }
}

/// <summary>
/// Квадратное обтекание
/// </summary>
public WrapSquare WrapSquare
{
    get
    {
        return _square;
    }

    set
    {
        if (value != null)
        {
            _square = value;
            WrapNone = false;
            _through.Detach();
            _tight.Detach();
        }
    }
}

```

```

        _tb.Detach();
    }
    else
    {
        _square.Detach();
        WrapNone = true;
    }
}
}

```

```

public WrapThroughTight WrapThrough
{
    get
    {
        return _through;
    }

    set
    {
        if (value == null)
        {
            _through.Detach();
            WrapNone = true;
        }
        else
        {
            _through = value;
            WrapNone = false;
            _tight.Detach();
            _square.Detach();
            _tb.Detach();
        }
    }
}

```

```

public WrapThroughTight WrapTight
{
    get
    {
        return _tight;
    }
    set
    {
        if (value == null)
        {
            _tight.Detach();
            WrapNone = true;
        }
        else
        {
            _tight = value;
            WrapNone = false;
        }
    }
}

```

```

        _through.Detach();
        _square.Detach();
        _tb.Detach();
    }
}

public WrapTopBottom WrapTopAndBottom
{
    get
    {
        return _tb;
    }
    set
    {
        if (value == null)
        {
            _tb.Detach();
            WrapNone = true;
        }
        else
        {
            _tb = value;
            WrapNone = false;
            _through.Detach();
            _square.Detach();
            _tight.Detach();
        }
    }
}

public bool? LayoutInCell
{
    get
    {
        var attr = Xml.Attribute("layoutInCell");
        if (attr == null) return null;
        return attr.Value == "1";
    }
    set
    {
        var attr = Xml.Attribute("layoutInCell");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("layoutInCell", value.Value ? "1" :
"0");
        }
    }
}

/// <summary>
/// Можно ли переместить объект

```

```

/// </summary>
public bool? Locked
{
    get
    {
        var attr = Xml.Attribute("locked");
        if (attr == null) return null;
        return attr.Value == "1";
    }
    set
    {
        var attr = Xml.Attribute("locked");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("locked", value.Value ? "1" : "0");
        }
    }
}

/// <summary>
/// Скрыт ли объект
/// </summary>
public bool? Hidden
{
    get
    {
        var attr = Xml.Attribute("hidden");
        if (attr == null) return null;
        return attr.Value == "true";
    }
    set
    {
        var attr = Xml.Attribute("hidden");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("hidden", value.Value ? "true" :
"false");
        }
    }
}

/// <summary>
/// Относительная высота
/// </summary>
public int? RelativeHeight
{
    get
    {
        var attr = Xml.Attribute("relativeHeight");

```

```

    if (attr != null)
    {
        return Convert.ToInt32(attr.Value);
    }
    return null;
}
set
{
    var attr = Xml.Attribute("relativeHeight");
    if (value == null)
        attr?.Remove();
    else
    {
        Xml.SetAttributeValue("relativeHeight", value.Value);
    }
}
}

///

```

```

{
    var attr = Xml.Attribute("behindDoc");
    if (value == null)
        attr?.Remove();
    else
    {
        Xml.SetAttributeValue("behindDoc", value.Value ? "1" : "0");
    }
}
}

/// <summary>
/// Перегружает ли объект другие объекты страницы (но это не
точно)
/// </summary>
public bool? AllowOverlap
{
    get
    {
        var attr = Xml.Attribute("allowOverlap");
        if (attr == null) return null;
        return attr.Value == "1";
    }
    set
    {
        var attr = Xml.Attribute("allowOverlap");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("allowOverlap", value.Value ? "1" :
"0");
        }
    }
}

/// <summary>
/// Параметр, показывающий, что обтекания текстом нет
/// </summary>
public bool WrapNone
{
    get
    {
        var elem = Xml.Element(XName.Get("wrapNone",
Namespaces.wp.NamespaceName));
        return elem != null;
    }
    set
    {
        // TODO: WrapSquare = null
        var elem = Xml.Element(XName.Get("wrapNone",
Namespaces.wp.NamespaceName));
        if (!value)

```

```

        {
            elem?.Remove();
        }
        else if (elem == null)
        {
            WrapSquare = null;
            Xml.Add(new CustomOfficeXElement(XName.Get("wrapNone",
Namespaces.wp.NamespaceName)));
        }
    }
}

}

public enum AnchorType
{
    Square,
    Tight,
    Through,
    TopAndBottom,
    BehindText,
    InFrontText,
    None
}
}

using System.Xml.Linq;

namespace Office.OOXML.MediaContent.AnchorWrappings
{
    public class WrapSquare : Inline
    {
        public WrapSquare(MainDocumentPackage document, IOOfficeXElement
xml, OfficeXElement parentXml) : base(document, xml, parentXml)
        {
        }

        public WrapText? WrapText
        {
            get
            {
                var attr = Xml.Attribute("wrapText");
                if (attr != null)
                {
                    switch (attr.Value)
                    {
                        case "bothSides":
                            return OOXML.WrapText.BothSides;

                        case "largest":
                            return OOXML.WrapText.Largest;
                    }
                }
            }
        }
    }
}

```

```

        case "left":
            return OOXML.WrapText.Left;

        case "right":
            return OOXML.WrapText.Right;
    }
}
return null;
}
set
{
    var attr = Xml.Attribute("wrapText");
    if (value == null)
    {
        attr?.Remove();
    }
    else
    {
        switch (value)
        {
            case OOXML.WrapText.BothSides:
                Xml.SetAttributeValue(XName.Get("wrapText"), "bothSides");
                break;

            case OOXML.WrapText.Largest:
                Xml.SetAttributeValue(XName.Get("wrapText"), "largest");
                break;

            case OOXML.WrapText.Left:
                Xml.SetAttributeValue(XName.Get("wrapText"), "left");
                break;

            case OOXML.WrapText.Right:
                Xml.SetAttributeValue(XName.Get("wrapText"), "right");
                break;
        }
    }
}
}
}
}

public class WrapThroughTight : AttachableOnSet
{
    public WrapThroughTight(MainDocumentPackage document,
        IOOfficeXElement xml, OfficeXElement parentXml) : base(document,
        xml, parentXml)
    {
        WrapPolygon = new WrapPolygon(document,
            Xml.Element(Namespace.wp + "wrapPolygon") ?? new
            CustomOfficeXElement(Namespace.wp + "wrapPolygon"), this);
    }
}

```

```

public WrapPolygon WrapPolygon
{ get; set; }

public override void Attach()
{
    if (!IsAttached)
    {
        base.Attach();
        if (!WrapPolygon.Start.IsAttached)
        {
            WrapPolygon.Start.Attach();
        }
    }
}

/// <summary>
/// Расстояние от левого края страницы
/// </summary>
public int? DistL
{
    get
    {
        var attr = Xml.Attribute("distL");
        if (attr != null)
        {
            return Convert.ToInt32(attr.Value);
        }
        return null;
    }
    set
    {
        var attr = Xml.Attribute("distL");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("distL", value.Value);
        }
    }
}

/// <summary>
/// Расстояние от правого края страницы
/// </summary>
public int? DistR
{
    get
    {
        var attr = Xml.Attribute("distR");
        if (attr != null)
        {
            return Convert.ToInt32(attr.Value);
        }
    }
}

```

```

    return null;
}
set
{
    var attr = Xml.Attribute("distR");
    if (value == null)
        attr?.Remove();
    else
    {
        Xml.SetAttributeValue("distR", value.Value);
    }
}
}

public WrapText? WrapText
{
    get
    {
        var attr = Xml.Attribute("wrapText");
        if (attr != null)
        {
            switch (attr.Value)
            {
                case "bothSides":
                    return OOXML.WrapText.BothSides;

                case "largest":
                    return OOXML.WrapText.Largest;

                case "left":
                    return OOXML.WrapText.Left;

                case "right":
                    return OOXML.WrapText.Right;
            }
        }

        return null;
    }
    set
    {
        var attr = Xml.Attribute("wrapText");
        if (value == null)
        {
            attr?.Remove();
        }
        else
        {
            switch (value)
            {
                case OOXML.WrapText.BothSides:
                    Xml.SetAttributeValue(XName.Get("wrapText"), "bothSides");
            }
        }
    }
}

```

```

        break;

        case OOXML.WrapText.Largest:
            Xml.SetAttributeValue(XName.Get("wrapText"), "largest");
            break;

        case OOXML.WrapText.Left:
            Xml.SetAttributeValue(XName.Get("wrapText"), "left");
            break;

        case OOXML.WrapText.Right:
            Xml.SetAttributeValue(XName.Get("wrapText"), "right");
            break;
    }
}
}
}
}

using System;

namespace Office.OOXML.MediaContent.AnchorWrappings
{
    public class WrapTopBottom : AttachableOnSet
    {
        public WrapTopBottom(MainDocumentPackage document,
            IOfficeXElement xml, IOfficeXElement parentXml) : base(document,
            xml, parentXml)
        {
        }

        /// <summary>
        /// Расстояние от верхнего края страницы
        /// </summary>
        public int? DistT
        {
            get
            {
                var attr = Xml.Attribute("distT");
                if (attr != null)
                {
                    return Convert.ToInt32(attr.Value);
                }
                return null;
            }
            set
            {
                var attr = Xml.Attribute("distT");
                if (value == null)
                    attr?.Remove();
                else
                {
                    Xml.SetAttributeValue("distT", value.Value);
                }
            }
        }
    }
}

```

```

    }
    }
}

/// <summary>
/// Расстояние от нижнего края страницы
/// </summary>
public int? DistB
{
    get
    {
        var attr = Xml.Attribute("distB");
        if (attr != null)
        {
            return Convert.ToInt32(attr.Value);
        }
        return null;
    }
    set
    {
        var attr = Xml.Attribute("distB");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("distB", value.Value);
        }
    }
}
}

namespace Office.OOXML.Model.MediaContent.AnchorWrappings
{
    public class WrapFrontBackText : AttachableOnSet
    {
        public WrapFrontBackText(MainDocumentPackage document,
            IOOfficeXElement xml, IOOfficeXElement parentXml, PackagePart part =
            null) : base(document, xml, parentXml, part)
        {
        }

        public bool? WrapNone
        {
            get
            {
                var elem = Xml.Element("wrapNone");
                if (elem != null)
                {
                    return true;
                }
                return null;
            }
        }
    }
}

```

```

    }
    set
    {
        var elem = Xml.Element("wrapNone");
        if (value == null)
            elem?.Remove();
        else
        {
            elem = new CustomOfficeXElement(Namespaces.wp + "wrapNone");
            Xml.Add(elem);
        }
    }
}
}
}

// Copyright (c) НТП Криптософт
// @file Inline.cs
// @date 06.06.2022

using System;
using Office.OOXML.Pptx;

namespace Office.OOXML.MediaContent
{
    /// <summary>
    /// Обозначает, что элемент имеет привязку к контенту (20.4.2.8)
    /// </summary>
    public class Inline : AttachableOnSet
    {
        public Extents Extent { get; }
        public DrawingObjectProperties DocPr { get; }

        public Graphic Graphic { get; protected set; }
        public Inline(MainDocumentPackage document, IOfficeXElement xml,
            OfficeXElement parentXml)
            : base(document, xml, parentXml)
        {
            Extent = new Extents(document, xml.Element(Namespaces.wp +
                "extent") ?? new CustomOfficeXElement(Namespaces.wp + "extent"),
                this);
            DocPr = new DrawingObjectProperties(document,
                xml.Element(Namespaces.wp + "docPr") ?? new
                CustomOfficeXElement(Namespaces.wp + "docPr"), this);
            Graphic = new Graphic(document, xml.Element(Namespaces.a +
                "graphic") ?? new CustomOfficeXElement(Namespaces.a + "graphic"),
                this);
        }

        /// <summary>
        /// Расстояние от верхнего края страницы

```

```

/// </summary>
public int? DistT
{
    get
    {
        var attr = Xml.Attribute("distT");
        if (attr != null)
        {
            return Convert.ToInt32(attr.Value);
        }
        return null;
    }
    set
    {
        var attr = Xml.Attribute("distT");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("distT", value.Value);
        }
        Attach();
    }
}

```

```

/// <summary>
/// Расстояние от нижнего края страницы
/// </summary>

```

```

public int? DistB
{
    get
    {
        var attr = Xml.Attribute("distB");
        if (attr != null)
        {
            return Convert.ToInt32(attr.Value);
        }
        return null;
    }
    set
    {
        var attr = Xml.Attribute("distB");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("distB", value.Value);
        }
        Attach();
    }
}

```

```

/// <summary>

```

```

/// Расстояние от левого края страницы
/// </summary>
public int? DistL
{
    get
    {
        var attr = Xml.Attribute("distL");
        if (attr != null)
        {
            return Convert.ToInt32(attr.Value);
        }
        return null;
    }
    set
    {
        var attr = Xml.Attribute("distL");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("distL", value.Value);
        }
        Attach();
    }
}
/// <summary>
/// Расстояние от правого края страницы
/// </summary>
public int? DistR
{
    get
    {
        var attr = Xml.Attribute("distR");
        if (attr != null)
        {
            return Convert.ToInt32(attr.Value);
        }
        return null;
    }
    set
    {
        var attr = Xml.Attribute("distR");
        if (value == null)
            attr?.Remove();
        else
        {
            Xml.SetAttributeValue("distR", value.Value);
        }
        Attach();
    }
}
}
}

```

Приложение В
(рекомендуемое)
Графическая часть

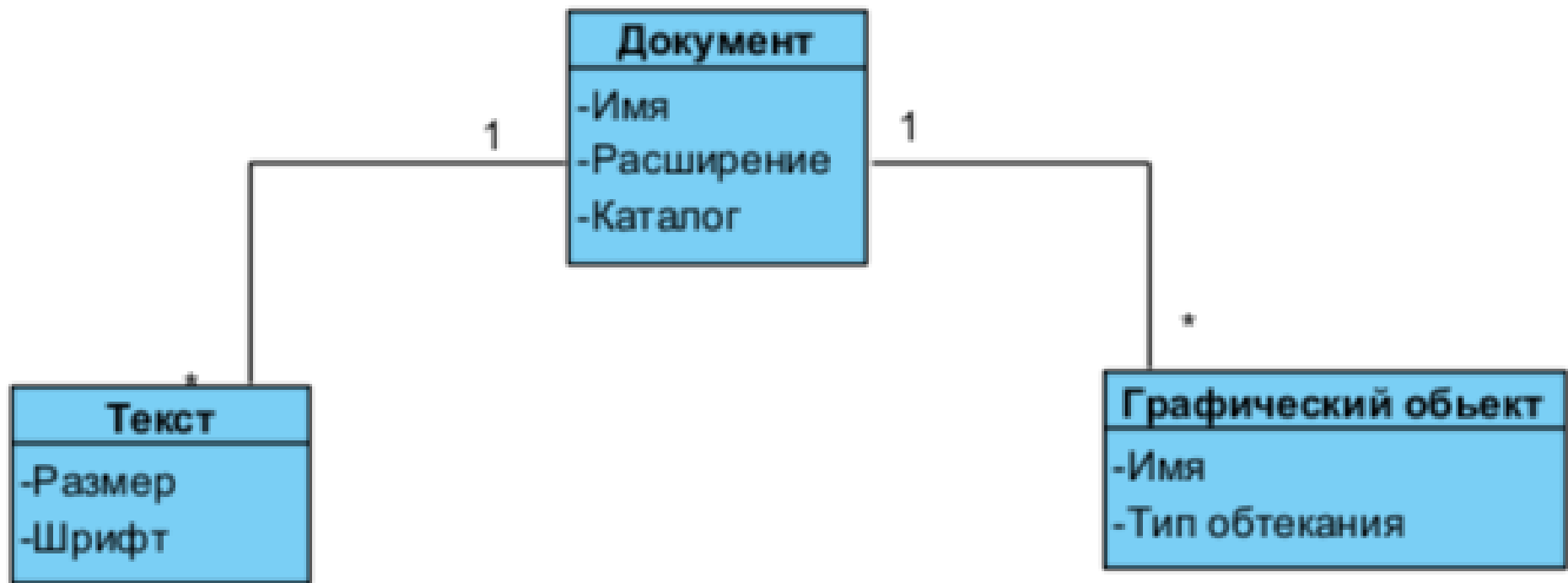


Рисунок В.1 – Модель предметной области разработки

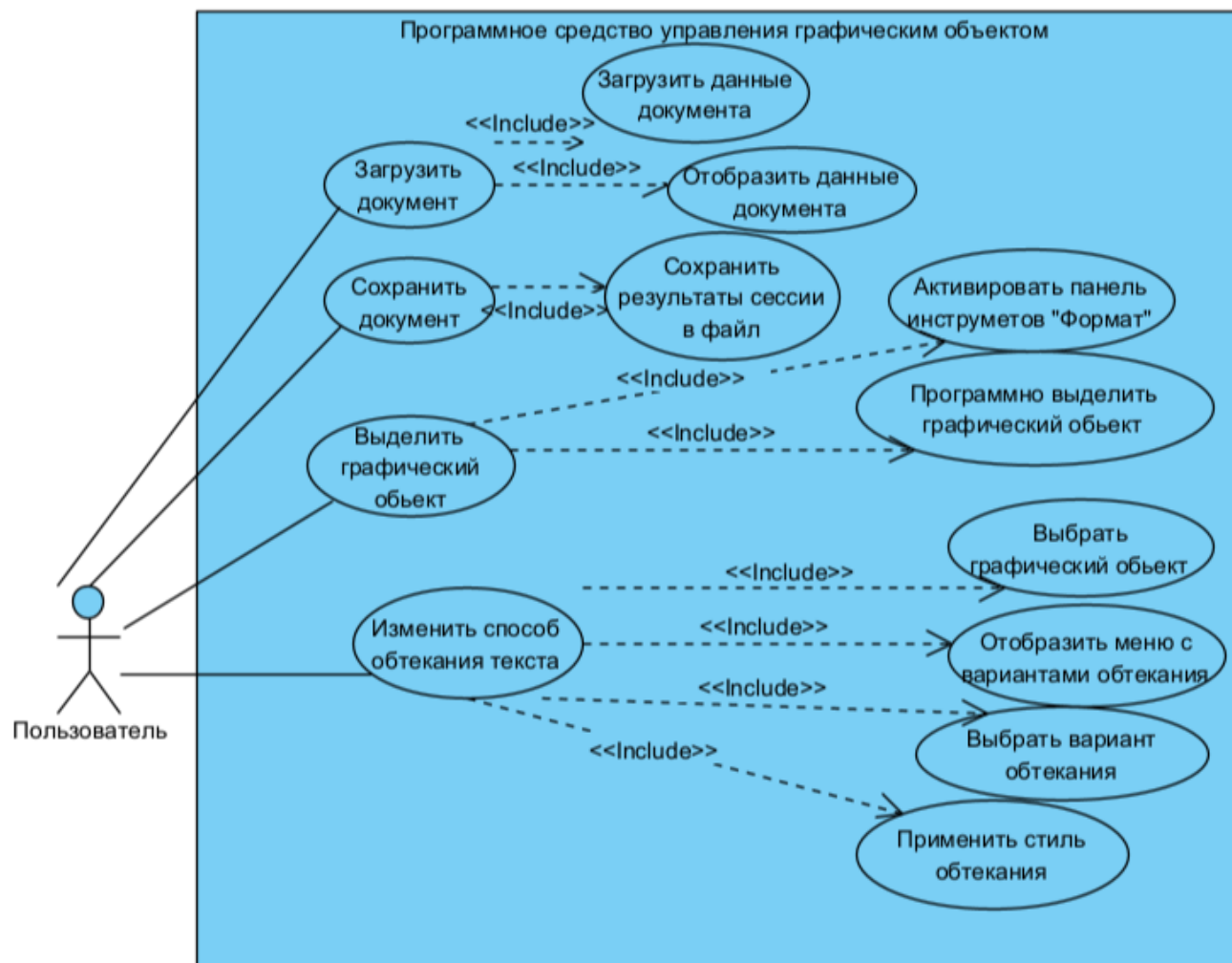


Рисунок В.2 – Диаграмма вариантов использования

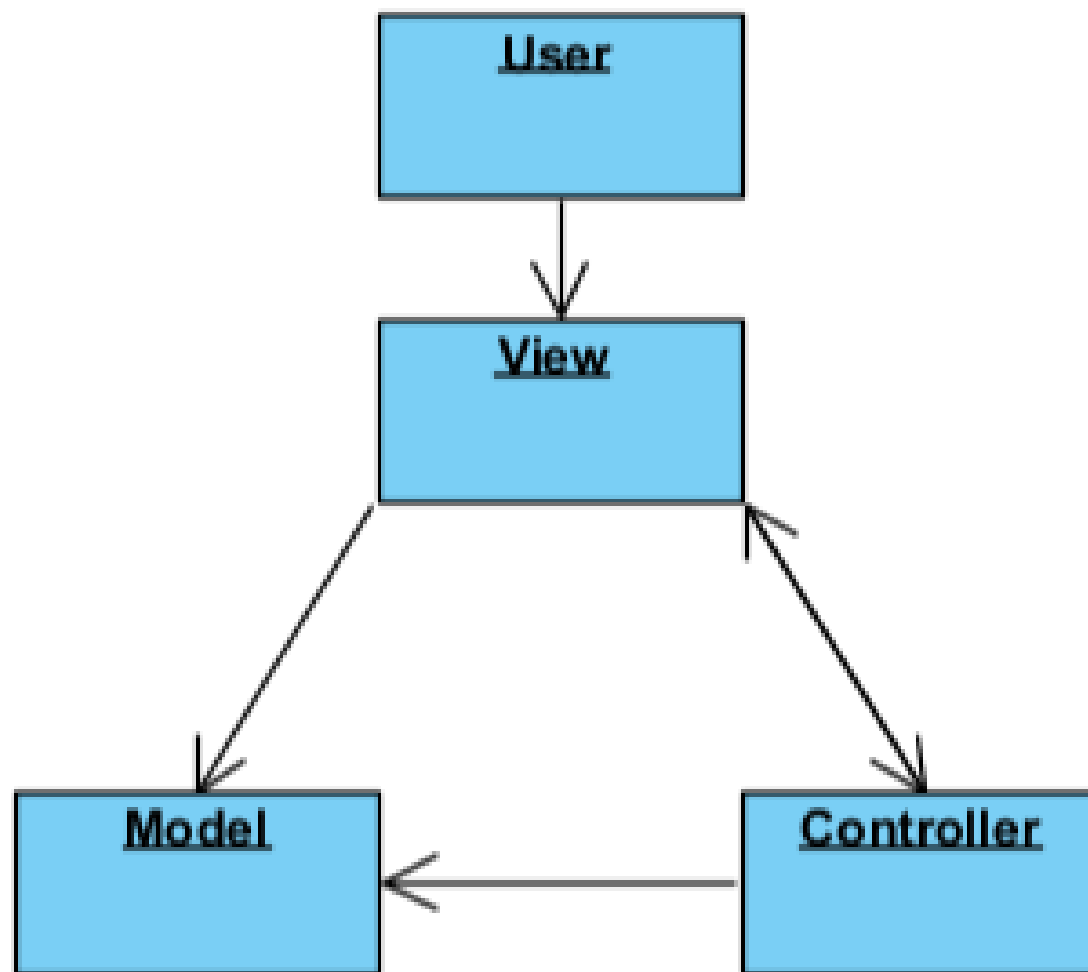


Рисунок В.3 – Модель архитектуры программного средства

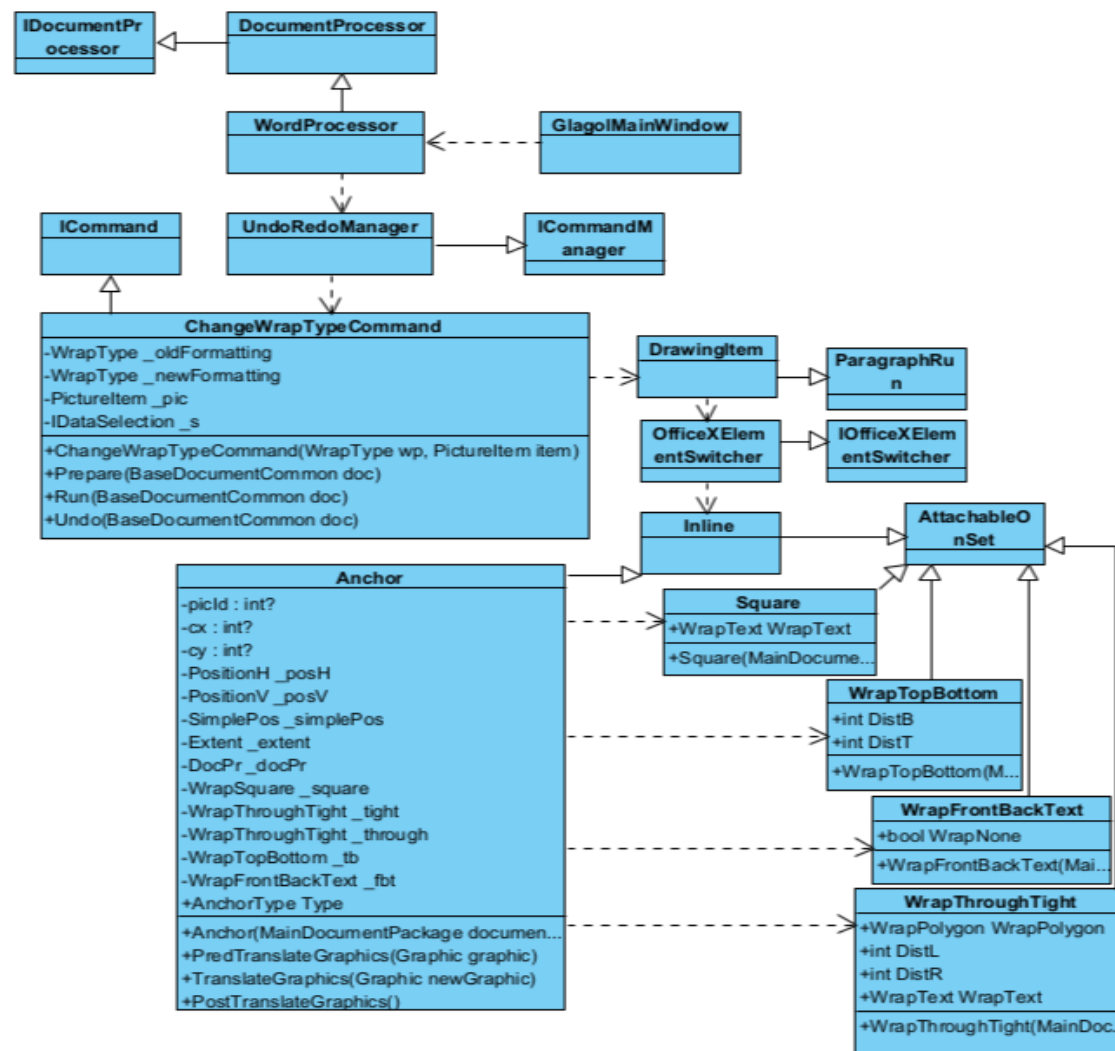


Рисунок В.4 – Диаграмма классов

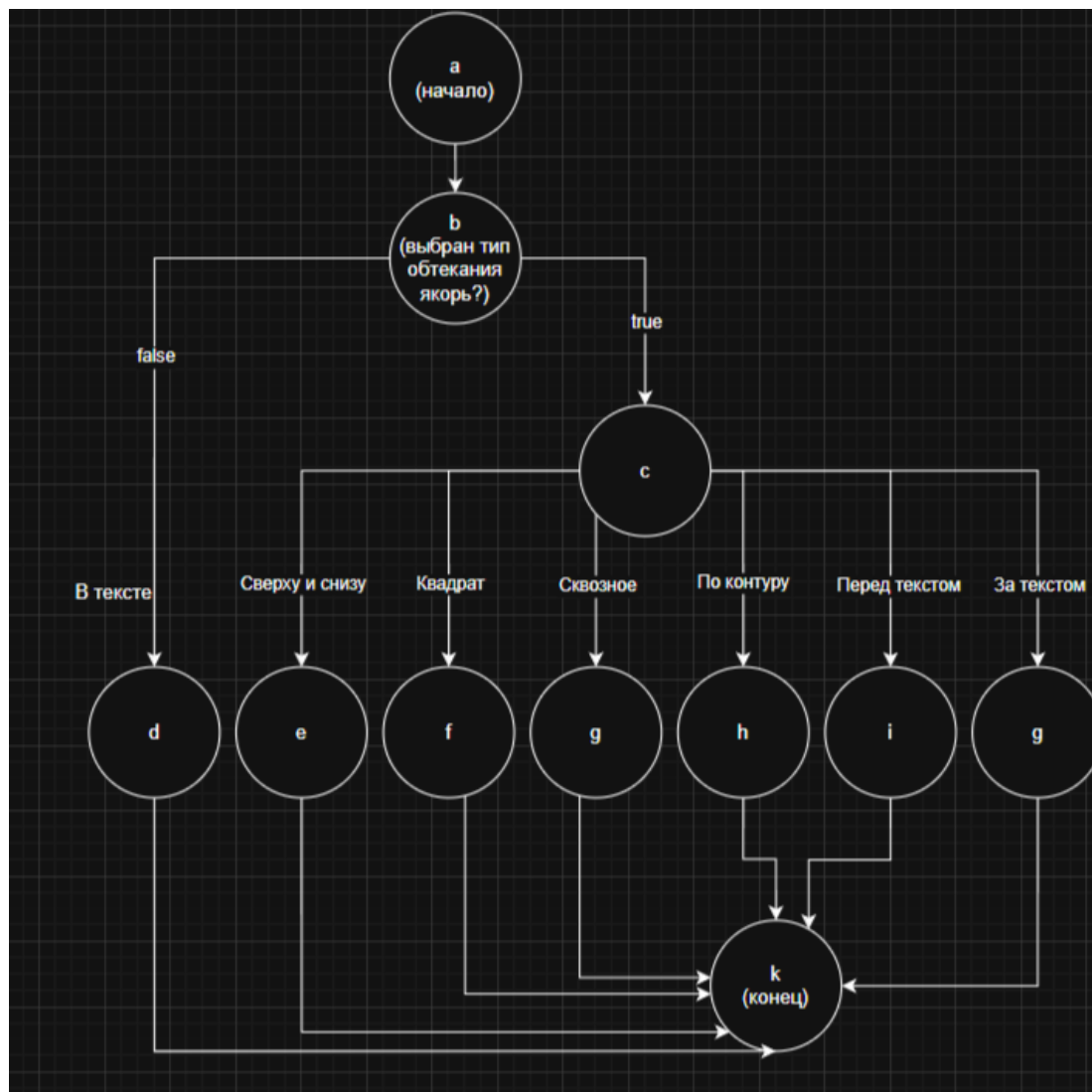


Рисунок В.5 – Граф цикломатической сложности

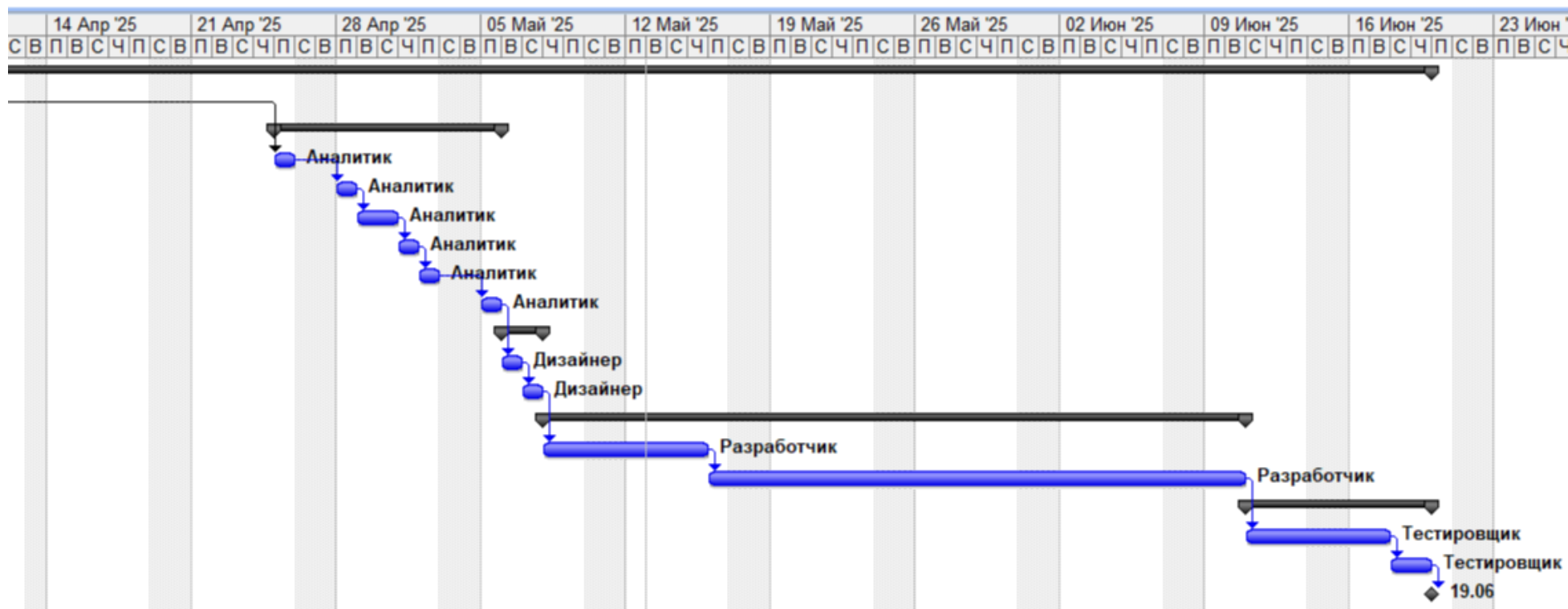


Рисунок В.6 – Диаграмма Ганта

Приложение Г
(рекомендуемое)
Код Unit-тестов

```

using System.Xml.Serialization;
using Xunit;

namespace GlagolTests
{
    public class UnitTest1
    {
        [Fact]
        public void testInvalidPictureName(string correctName,string
existingName)
        {
            Assert.Equal(correctName, existingName);
        }

        [Fact]
        public void testInvalidPictureId(int? correctId,int? existingId)
        {
            Assert.Equal(correctId, existingId);
        }

        [Fact]
        public void testInvalidPictureParts(Xml intryXml,XML correctXml)
        {
            Assert.Equal(intryXml, correctXml);
        }

        [Fact]
        public void testInvalidPictureArguments(List<Pt> first,List<Pt>
second)
        {
            for (int i = 0; i < first.Count; i++)
            {
                Assert.Equal(first[i], second[i]);
            }
        }

        [Fact]
        public void testInvalidPictureGrapfics(Graphic intrGrapf,Graphic
trueGrapg)
        {
            Assert.Equal(intrGrapf, trueGrapg);
        }

        [Fact]
        public void testInvalidPictureOffset(int correctOffset,int
existingOffset)
        {
            Assert.Equal(correctOffset, existingOffset);
        }

        [Fact]

```

```

    public void testInvalidPictureTypeWrapping(WrapType
existingType, WrapType CorrectType)
    {
        Assert.Equal(existingType, CorrectType);
    }

    [Fact]
    public void testInvalidPictureLinks(string correctLink, string
existingLink)
    {
        Assert.Equal(correctLink, existingLink);
    }

    [Fact]
    public void testInvalidCXCValues(List<int> correctValues,
List<int> existingValues)
    {
        for (int id = 0; id < correctValues.Count; id++)
        {
            Assert.Equal(correctValues[id], existingValues[id]);
        }
    }

    [Fact]
    public void testInvalidPictureExtents(List<int> correctExtents,
List<int> existingExtents)
    {
        for (int id = 0; id < correctExtents.Count; id++)
        {
            Assert.Equal(correctExtents[id], existingExtents[id]);
        }
    }
}

```

Приложение Д
(рекомендуемое)
Акт о внедрении



Общество с ограниченной ответственностью
научно-техническое предприятие
«Крипософт»
440026, г. Пенза, ул. Лермонтова, д. 3
тел. 8 (8412) 99-10-91
факс 8 (8412) 99-10-91, доб. 1999
E-mail: mails@cryptosoft.ru

Акт о внедрении

Результаты теоретических и практических исследований, приведенные в дипломной работе Соколова Артем Вячеславович на тему «Программные средства управления графическими объектами в текстовых документах», внедрены в офисный пакет, разрабатываемый в ООО НТП «Крипософт».

Начальник отдела разработки
офисного программного обеспечения

А.П. Чернов

Начальник управления системных
исследований



И.Н. Дорошенко