|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт искусственного интеллекта |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры |

КУРСОВАЯ РАБОТА

по дисциплине «Анализ и проектирование информационных систем»

на тему: «Разработка архитектуры текстового редактора(word)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_\_\_\_ | *Бредихин Виктор Андреевич* | | | |
|  | *Подпись* | *Фамилия Имя Отчество* | | | |
|  | | |  |
| Группа | КМБО-02-21 |  | | |  |
|  |  |  | | |  |
| Руководитель  работы | \_\_\_\_\_\_\_\_\_\_\_\_\_ | *Завьялов Антон Владимирович* | | | |
|  | *Подпись* | *Фамилия Имя Отчество* | | | |

Москва 2024

ОГЛАВЛЕНИЕ

[1 ОБЩИЕ ПОЛОЖЕНИЯ 1](#_Toc169365671)

[2 ОСНОВНЫЕ ТРЕБОВАНИЯ К ПРОГРАММЕ 1](#_Toc169365672)

[2.1 Краткий обзор требований к системе 1](#_Toc169365673)

[2.1.1 Функциональные требования 1](#_Toc169365674)

[2.1.2 Нефункциональные требования 2](#_Toc169365675)

[2.2 Модель предметной области 3](#_Toc169365676)

[2.3 Модель прецедентов 4](#_Toc169365677)

[2.4 Архитектурные мотивы 5](#_Toc169365678)

[3 ОПИСАНИЕ АРХИТЕКТУРЫ 5](#_Toc169365679)

[3.1 Применяемые шаблоны архитектуры 5](#_Toc169365680)

[3.2 Диаграмма компонентов 5](#_Toc169365681)

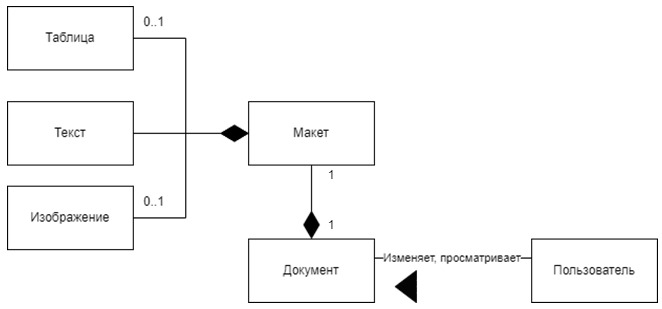
[3.3 Матрица трассирования архитектурных мотивов 7](#_Toc169365682)

[4 Алгоритмы работы программы 7](#_Toc169365683)

[5 ВЫВОДЫ 9](#_Toc169365684)

1. **ОБЩИЕ ПОЛОЖЕНИЯ**
2. **ОСНОВНЫЕ ТРЕБОВАНИЯ К ПРОГРАММЕ**
   1. Краткий обзор требований к системе
      1. Функциональные требования
         1. Система должна предоставлять возможность пользователю создавать, открывать, редактировать, сохранять и отправлять на печать текстовые документы в форматах: doc, docx, rtf, txt, pdf, odt, html.
         2. Система должна предоставлять возможность пользователю форматировать текст документа, изменяя его шрифт, размер, цвет, выравнивание, отступы, интервалы, маркировку, нумерацию и стили.
         3. Система должна предоставлять возможность пользователю вставлять в документ различные объекты, такие как: изображения, таблицы, графики, диаграммы, формулы, ссылки, примечания, сноски, оглавление.
         4. Система должна предоставлять возможность пользователю проверять правописание и грамматику текста документа
         5. Система должна предоставлять возможность пользователю защищать документы от несанкционированного доступа, редактирования или копирования, используя пароли, шифрование или цифровые подписи.
         6. Система должна предоставлять возможность пользователю добавлять в документ специальные символы, такие как: стрелки, знаки валют, математические знаки и смайлики, а также использовать различные алфавиты и языки.
         7. Система должна предоставлять возможность пользователю использовать различные инструменты для работы с документом, такие как: поиск и замена, навигация, сортировка, группировка, фильтрация и выделение.
         8. Система должна предоставлять возможность пользователю использовать различные функции для работы с текстом, такие как: копирование, вставка, вырезание, перенос, удаление, вставка даты и времени.
         9. Система должна предоставлять возможность пользователю использовать различные средства для улучшения читаемости документа, такие как: гиперссылки, закладки, перекрестные ссылки, иерархические заголовки, разрывы страниц и разделов.
         10. Система должна предоставлять возможность пользователю использовать различные средства для улучшения оформления документа, такие как: подписи, нумерация страниц, колонтитулы, сноски, примечания и библиография.
         11. Система должна предоставлять возможность пользователю использовать различные средства для управления документами, такие как: создание, удаление, перемещение, копирование, переименование, архивация и сжатие.
      2. Нефункциональные требования
         1. Система должна иметь интуитивно понятный и удобный графический интерфейс, который позволяет пользователю легко находить и использовать все функции программы.
         2. Система должна иметь встроенную справочную систему, которая предоставляет пользователю подробную информацию о всех функциях программы, а также часто задаваемые вопросы и ответы на них.
         3. Система должна корректно открывать, сохранять и закрывать документы любого размера и формата, а также обрабатывать все внесенные пользователем изменения в документах.
         4. Система должна использовать не более 50% ресурсов процессора, а также не вызывать конфликтов с другими программами или операционной системой.
         5. Система должна обеспечивать стабильную и безошибочную работу во всех ситуациях, а также предотвращать потерю или повреждение данных пользователя в случае сбоев в работе программы, компьютера или сети.
         6. Система должна иметь функцию автоматического сохранения документов в заданных интервалах времени, а также функцию восстановления документов из резервных копий в случае их удаления или повреждения.
         7. Разработка внутренней логики программы должна производиться с помощью языка программирования C++.
         8. Система должна работать в операционных системах: Windows.
   2. Модель предметной области

Предметная область – «Работа с текстом и макетом оформления». Диаграмма предметной области изображена на рисунке 1.

Рисунок 1 Диаграмма предметной области

Макет – часть документа, определяющая то, как этот документ будет выглядеть после печати. Макет может включать в себя текст, изображения и/или таблицы. Пользователь способен определённым образом взаимодействовать с документом, просматривать, изменять, создавать новый документ. В изменение входит как редактирование внутреннего наполнения документа, так и его внешних характеристик, которые отвечают за то, как документ будет выглядеть на бумаге уже после печати.

* 1. Модель прецедентов

Диаграмма прецедентов изображена на рисунке 2.



Рисунок 2 Диаграмма прецедентов

Пользователь способен создавать документ средствами системы, редактировать, экспортировать документ в другие форматы, отличные от основных форматов, с которыми работает система (doc, docx), напечатать документ, подключив принтер к системе.

* 1. Архитектурные мотивы
     1. Требование [2.1.2.1](#Т2121) подразумевает наличие графического интерфейса пользователя, взаимодействие с которым составляет основную функциональность приложения.
     2. Из функциональных требований следует, что должно быть создано приложение, почти вся функциональность которого заключается в работе с одним текущим документом.

1. **ОПИСАНИЕ АРХИТЕКТУРЫ**
   1. Применяемые шаблоны архитектуры

Ввиду требования [2.1.2.1](#Т2121) целесообразно рассматривать шаблоны с отделением пользовательского интерфейса. В данном случае самым подходящим шаблоном является шаблон MVC (Model-View-Controller), являющегося модификацией шаблона BCE (Boundary-Control-Entity). Выбор в пользу именно этого шаблона обосновывается тем фактом, что в контексте шаблона BCЕ приложение подразумевает работу сразу лишь с одной сущностью (в данном случае с документом) и основа функциональности приложения состоит в изменении этой сущности. Это подталкивает к объединению сущностей и бизнес-логики приложения в один модуль – в контексте шаблона MVC называемый моделью.

Компоненты системы и компоненты шаблона соотносятся следующим образом: DocumentHandler – модель, View – представление, Controller – контроллер. Взаимодействие компонентов в рамках шаблона представлено на диаграмме компонентов в разделе 3.2.

* 1. Диаграмма компонентов

Диаграмма компонентов изображена на рисунке 3.

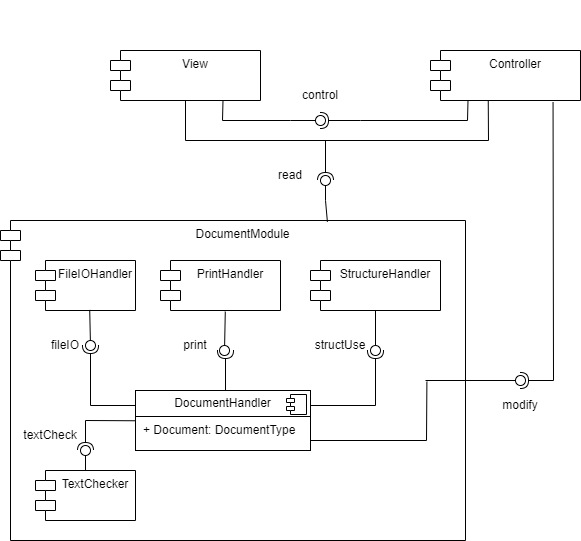


Рисунок 3 Диаграмма компонентов

DocumentModule – модуль реализующий логику работы пользователя с документом.

View – модуль, отвечающий за отображение графического интерфейса, а именно макета документа и инструментов работы с ним. Способен считывать информацию о документе с помощью интерфейса read.

Controller – модуль, который отвечает за:

1. обработку входных данных пользователя и преобразование их в запросы к DocumentModule c помощью интерфейса modify,
2. уведомление модуля View об изменениях в документе.

Также способен к получению информации о документе с помощью интерфейса read.

DocumentModule состоит из нескольких компонентов:

1. DocumentHandler – модуль, который отвечает за работу с физическим представлением типа DocumentType, обработку запросов от модуля Controller и соответственно работу с документом;
2. FileIOHandler – модуль, реализующий работу с файловой системой операционной системы Windows;
3. TextChecker – модуль, реализующий проверку текста на грамматические ошибки;
4. PrintHandler – модуль, отвечающий за печать документа;
5. StructureHandler – модуль, отвечающий за изменение и создание различных структур, которые могут содержаться в документе. Примерами структур являются текст, таблицы, оглавление, диаграммы.
   1. Матрица трассирования архитектурных мотивов

|  |  |
| --- | --- |
| Номер и название архитектурного мотива | Пояснение по реализации |
| [2.4.1](#Т242) Наличие графического интерфейса пользователя, взаимодействие с которым составляет основную функциональность приложения | Наличие графического интерфейса пользователя реализовано с помощью модуля View (см. п. [3.2](#Т32)), взаимодействие с ним с помощью модуля Controller (см. п. [3.2](#Т32)) |
| [2.4.2](#Т242) Почти вся функциональность приложения заключается в редактировании одного текущего документа | Реализовано с помощью модуля DocumentModule (см. п. [3.2](#Т32)) |

1. **АЛГОРИТМЫ РАБОТЫ ПРОГРАММЫ**
   1. Прецедент изменения текста в документе. Диаграмма последовательностей изображена на рисунке 4.

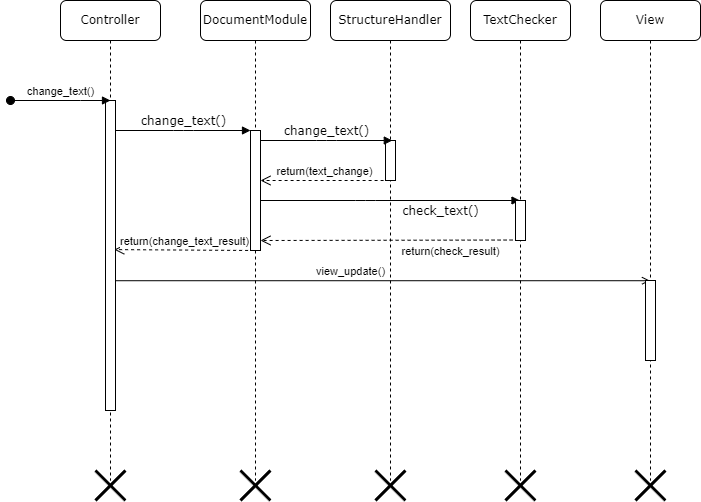


Рисунок 4 Диаграмма последовательностей для прецедента изменения текста в документе

На диаграмме изображен ответ программы на изменение пользователем текста в документе. Изменение текста документа пользователем происходит через модуль, отвечающий за обработку ввода пользователя, Controller. Controller после приема ввода передает этот ввод модулю DocumentModule через интерфейс modify. DocumentModule последовательно отправляет два запроса. Модулю StructureHandler отправляет запрос на изменение текста, модулю TextChecker – на проверку текста на ошибки. Далее Controller уведомляет View о необходимости обновления представления.

* 1. Прецедент сохранения документа. Диаграмма последовательностей изображена на рисунке 5.

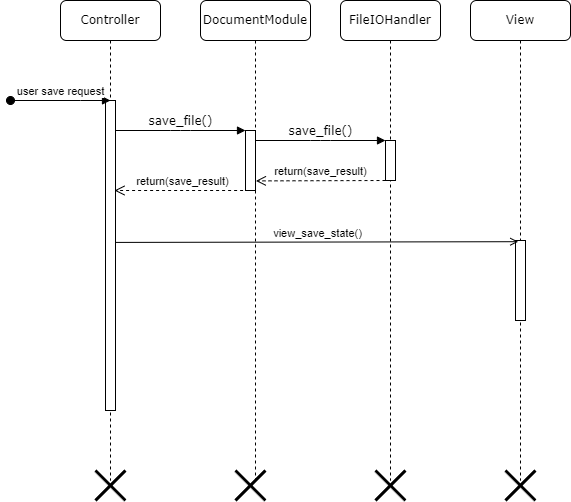


Рисунок 5 Диаграмма последовательностей для прецедента сохранения документа

На диаграмме изображен ответ программы на запрос пользователя на сохранение документа. Обработку запроса производит модуль Controller, передающий запрос модулю DocumentModel. DocumentModule запрашивает сохранение документа у модуля FileIOHandler, который производит сохранение файла, после чего возвращает результат сохранения файла. После обработки запроса DocumentModel возвращает результат сохранения файла. Далее Controller уведомляет View о результатах сохранения файла.

1. **ВЫВОДЫ**

В результате проделанной работы были выведены требования для приложения текстового редактора Microsoft Word, разработаны архитектура и примерные алгоритмы работы программы. В рамках разработки архитектуры были получены навыки в создании различных UML-диаграмм и навыки выбора и обоснования шаблона архитектуры.