ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Федеральное государственное бюджетное образовательное учреждение высшего образования
Национальный исследовательский ядерный университет «МИФИ»

Институт
интеллектуальных кибернетических систем

Кафедра кибернетики (№ 22)

Направление подготовки 09.03.04 Программная инженерия

Пояснительная записка

к учебно-исследовательской работе студента на тему:

Исследование и разработка прототипа системы генерации документов на основе шаблонов

Группа Б14-506

Студент _____ (подпись) Пономарёв Е.А. (ФИО)

Руководитель _____ (подпись) Немешаев С.А. (ФИО)

Научный консультант _____ (подпись) Рословцев В.В. (ФИО)

Оценка руководителя _____ (0-15 баллов)

Оценка за оформление _____ (0-10 баллов)

Оценка за структуру _____ (0-10 баллов)

Члены комиссии

Председатель _____ _____

_____ _____ _____ _____

_____ _____ _____ _____

Москва 2017

Реферат

Пояснительная записка содержит 43 страницы. Количество использованных источников –15. Количество приложений – 1.

Ключевые слова: шаблон, генерация файлов, Word-документ, LaTex, PDF-документ, .NET, веб-сервис, веб-интерфейс.

Целью данной учебно-исследовательской работы является исследование и разработка прототипа системы генерации документов на основе шаблонов.

В первом разделе описывается изучение и сравнительный анализ формальных средств описания шаблонов объектов. Изучение правил подстановки при формировании конечных файлов.

Во втором разделе описан процесс разработки модели объектов данных и метаданных для описания шаблонов документов и механизмов связывания формальных параметров с фактическими информационными объектами. Также описана разработка концептуальной модели тестовой предметной области – генерации протоколов для ВКР.

В третьем разделе описывается инженерная часть научно-исследовательской работы, а именно проектирование архитектуры разрабатываемой системы и разработка функциональной структуры информационной системы.

В четвёртом разделе описывается программная реализация механизмов генерации Word-файлов с простейшим пользовательским интерфейсом, а также приводится пример создания тестового документа протокола ВКР.

Оглавление

Введение

В настоящее время непрерывно растёт объём документации работ на различных видов предприятиях, офисах и т.д. В подобных сферах, будь то разработка корпоративных приложений, бухгалтерия или образовательная система, очень часто приходится решать задачу выгрузки данных в документы — от небольших справок до больших отчетов. Следовательно становится всё более актуальным вопрос о создании решения для генерации docx/pdf документов, которое позволяет заполнять документы по шаблону, оформление которого можно менять в соответствующей среде без переписывания кода.

В первом разделе описывается изучение и сравнительный анализ формальных средств описания шаблонов объектов. Изучение правил подстановки при формировании конечных файлов.

Во втором разделе описан процесс разработки модели объектов данных и метаданных для описания шаблонов документов и механизмов связывания формальных параметров с фактическими информационными объектами. Также описана разработка концептуальной модели тестовой предметной области – генерации протоколов для ВКР.

В третьем разделе описывается инженерная часть научно-исследовательской работы, а именно проектирование архитектуры системы и разработка функциональной структуры информационной системы.

В четвёртом разделе описывается программная реализация механизмов генерации Word-файлов, приводится пример генерации тестового докум ента.

1. Анализ проблематики генерации документов на основе шаблонов

В данном разделе приводятся теоретические аспекты проблемной области решаемой задачи. Приведен сравнительный анализ формальных средств описания шаблонов объектов и изучены правила подстановки при формировании конечных файлов. Также описаны технологии генерации Word- и PDF-документов. В подразделе «Выводы» подведены итоги анализа предметной области. В конце раздела формулируется цель научно-исследовательской работы, а также перечисляются задачи, которые необходимо выполнить для достижения поставленной цели.

1.1. Изучение и сравнительный анализ формальных средств описания шаблонов объектов. Изучение правил подстановки при формировании конечных файлов

В данном разделе приводятся теоретические аспекты проблемной области решаемой задачи. Приводится сравнительный анализ средств описания шаблонов объектов. Также приводятся изученные правила подстановки при формировании конечных файлов.

Исчисление шаблонов является результатом глубокого пересмотра 50-летней разработки. Основная цель — обеспечить унифицирующий подход, связывающий различные стили программирования и парадигмы при помощи сопоставления шаблонов. Полная оценка эффективности данного представления требует глубокого знания многочисленных теоретических областей информатики, такие как лямбда-исчисление, терминологическая перепись и теория типов.

Исчисление на основе шаблонов является новым основанием для вычислений, в которых выражения мощностей функций и структур данных плодотворно сочетаются в рамках функций сопоставления шаблонов. Лучшие из существующих основ сосредоточены на любых функциях (в λ - исчислении) или на структурах данных (в машинах Тьюринга), или же на обоих перечисленных (как в объектной ориентации).

Чтобы найти баланс между функциями и структурами, начнем с теории функций, λ - исчисления. Короче говоря, каждый член чистого λ -исчисления является либо переменным, либо применением, либо λ -абстракцией. Оценка дается одним правилом, которое заменяет аргумент функ-

ции связанной переменной. Чистое λ -исчисление способно кодировать структуры данных как абстракции. Но такое кодирование не является однородным, поскольку нет средств для выделения атомов из соединений. Таким образом, первым шагом является добавление некоторых конструкторов, которые являются атомами. Из таких атомов могут быть созданы структуры данных, посредством применения. Затем добавляются некоторые операции, которые действуют на структуры данных, например, для сравнения конструкторов или восстановления компонентов. Результирующего исчисления должно быть достаточно для определения основных интересующих запросов.

Альтернативным средством воздействия на структуры данных является использование сопоставления шаблонов. Шаблоны могут использоваться для описания внутренней структуры данных, то есть ее формы, и для обозначения ее частей, которые затем доступны для использования.

Концептуальный шаблон имеет набор входных и выходных параметров, а также набор начальных значений. После инициализации этих параметров получается частная модель некоторой предметной области. Таким образом, технология концептуальных шаблонов позволяет экспертам создавать модели сложных систем, не углубляясь в язык системной динамики.

Формальное понятие шаблона на языке теории множеств представлено в виде следующего множества формальных шаблонов $P = \{E, Sign2, R, Fn\}$. $E = \{L, A, C\}$ — множество элементов шаблонов, где L — множество уровней шаблонов (наличия уровня подразумевает наличие входящих и исходящих потоков); A — множество переменных шаблонов; C — множество констант шаблонов. $Sign2$ — множество вторичных признаков распознавания (термины, описывающие шаблон). R — множество отношений внутри шаблонов. Fn — множество законов функционирования переменных в шаблонах.

Поскольку шаблоны могут иметь достаточно сложную внутреннюю структуру, то на разработку модели сложной системы, без технологии концептуальных шаблонов, экспертам может понадобиться достаточно большой промежуток времени. Кроме того, если эксперты не обладают достаточными знаниями о языке системной динамики и опытом построения системно-динамических моделей, то понадобится привлечение группы экспертов в области системно-динамического моделирования. Как следствие, возникает проблема взаимопонимания экспертов, а также увеличение финансовых затрат. Использование технологии концептуальных шаблонов позволяет решить выше указанные проблемы.[1]

Далее рассматриваются правила подстановки в λ — исчислении.

Процесс вычисления термов заключается в подстановке аргументов во все функции. Выражения вида:

$$(\lambda x.M)N$$

Заменяются на

$$M[x = N]$$

Эта запись означает, что в терме M все вхождения x заменяются на терм N. Этот процесс называется редукцией терма. А выражения вида $(\lambda x.M)N$ называются редексами.

При подстановке необходимо следить за тем, чтобы у нас не появлялись лишние связывания переменных. Например рассмотрим такой редекс:

$$(\lambda xy.x)y$$

После подстановки за счёт совпадения имён переменных мы получим тождественную функцию:

$$\lambda y.y$$

Переменная y была свободной, но после подстановки стала связанной. Необходимо исключить такие случаи. Поскольку с ними получается, что имена связанных переменных в определении функции влияют на её смысл. Например смысл такого выражения

$$(\lambda xz.x)y$$

После подстановки будет совсем другим. Но мы всего лишь изменили обозначение локальной переменной y на z. И смысл изменился, для того чтобы исключить такие случаи пользуются переименованием переменных или α -преобразованием. Для корректной работы функций необходимо следить за тем, чтобы все переменные, которые были свободными в аргументе, остались свободными и после подстановки.

Процесс подстановки аргументов в функции называется β -редукцией. В редексе $(\lambda x.M)N$ вместо свободных вхождений x в M мы подставляем N. Посмотрим на правила подстановки:

$$x[x = N] \Rightarrow N$$

$$y[x = N] \Rightarrow y$$

$$(PQ)[x = N] \Rightarrow (P[x = N]Q[x = N])$$

$$(\lambda y.P)[x = N] \Rightarrow (\lambda y.P[x = N]), y \notin FV(N)$$

$$(\lambda x.P)[x = N] \Rightarrow (\lambda x.P)$$

Первые два правила определяют подстановку вместо переменных. Если переменная совпадает с той, на место которой мы подставляем терм N , то мы возвращаем терм N, иначе мы возвращаем переменную:

$$x[x = N] \Rightarrow N$$

$$y[x = N] \Rightarrow y$$

Подстановка применения термов равна применению термов, в которых произведена подстановка:

$$(PQ)[x = N] \Rightarrow (P[x = N]Q[x = N])$$

При подстановке в лямбда-функции необходимо учитывать связность переменных. Если переменная аргумента отличается от той переменной на место которой происходит подстановка, то мы заменяем в теле функции все вхождения этой переменной на N:

$$(\lambda y.P)[x = N] \Rightarrow (\lambda y.P[x = N]), y \notin FV(N)$$

Условие $y \notin FV(N)$ означает, что необходимо следить за тем, чтобы в N не оказалось свободной переменной с именем y, иначе после подстановки она окажется связанной. Если такая переменная в N всё-таки окажется мы проведём α -преобразование в терме $\lambda y.M$ и заменим y на какую-нибудь другую переменную.

В последнем правиле мы ничего не меняем, поскольку переменная x оказывается связанной. А мы проводим подстановку только вместо свободных переменных:

$$(\lambda x.P)[x = N] \Rightarrow (\lambda x.P)$$

[2]

1.2. Изучение технологии компиляции PDF-файлов из исходных текстов на LaTeX

В данном разделе описывается процесс компиляции PDF-файлов из исходных текстов на LaTeX. Также приводится структура и особенности вёрстки документов на LaTex.

LaTeX-документ (расширение tex) — это обычный текстовый файл, в котором содержится и некоторый объём команд для LaTeX процессора. В каком-то смысле это программа, по выполнении которой получается качественно оформленная печатная или электронная копия документа. [3]

Типичный LaTeX-документ имеет следующую структуру:

Листинг 1. Структура LaTeX-документа

```
\documentclass{article} %Определение класса документа
    %Заголовок документа
\begin{document}
    %Содержание документа
\end{document}
```

Классы LaTeX-документов

Класс - это некоторый базовый набор команд определяющий внешний вид будущего документа. Файлы классов в LaTeX имеют расширение .sty . В дистрибутивах доступны некоторые стандартные классы в частности article, report и некоторые другие. Практически все классы принимают аргументы, например команда:

```
\documentclass[11pt,a4paper,oneside]{report}
```

создаст документ класса report, с форматом бумаги A4, базовым размером шрифта в 11pt и полями для односторонней печати. Как правило пользователь использует один из стандартных стилей и модифицирует внешний вид документа командами, которые он добавляет в заголовке документа, однако если таких команд много, то можно создать собственный стилевой файл.[4]

Листинг 2. Исходный файл HelloWorld.tex

```
\documentclass[12pt,a4paper]{scrartcl}
\usepackage[utf8]{inputenc}
\usepackage[english,russian]{babel}
\usepackage{indentfirst}
```

```
\usepackage{misccorr}
\usepackage{graphicx}
\usepackage{amsmath}
\begin{document}
Здравствуй, Мир!!!
\end{document}
```

На первой строке загружается класс документа scrartcl. Этот класс входит в набор KOMA-Script — современный пакет с отличной документацией и богатыми возможностями. На следующих строках загружаются стилевые файлы, необходимые для ”русификации” документа:

- inputenc — для выбора кодировки текста;
- babel — пакет для локализации;
- indentfirst — красная строка для первого параграфа;
- misccorr — пакет с дополнительными настройками для соответствия правилам отечественной полиграфии.

Стили graphicx и amsmath отвечают за вставку картинок и отображение математической нотации.

Сам текст документа набирается внутри окружения document, которое начинается с команды `\begin{document}` и заканчивается конструкцией `\end{document}`. Параграфы в тексте разделяются друг от друга пустой строкой. После создания файла HelloWorld.tex, его можно скомпилировать с помощью программы pdflatex и посмотреть полученный в результате PDF-файл HelloWorld.pdf, как показано ниже

```
pdflatex HelloWorld.tex
okular HelloWorld.pdf
```

Создание титульного листа

Перед началом работы следует попытаться найти готовый класс LaTeX, который учитывает все правила к оформлению научных публикаций, установленные в ВУЗе. Если такой файл найдётся (у других студентов или в администрации ВУЗа), то задачу по вёрстке документа можно считать решённой, что позволит сразу перейти к набору текста. К сожалению, для российских ВУЗов такие файлы встречаются крайне редко, поэтому мы выполним оформление титульного листа ”вручную”, вставляя в него пробелы, выверенные линейкой на твёрдой копии образца.

После оформления титульного листа можно переходить к набору текста. Пакет LaTeX берёт на себя работу по оформлению заголовков разделов и их автоматической нумерации. Достаточно только указать, что в данном месте начался новый раздел с помощью команд `\section` (раздел), `\subsection` (подраздел) и `\subsubsection` (подподраздел).

Весь текст, который находится за символом % , считается комментарием, и поэтому не выводится при печати. Символ процента можно вывести с помощью команды `\%` , а символ ~ формирует неразрывный пробел. Кроме символа процента необходимо экранировать символы { } $ & # _. Также специальным является и символ `\`.

Окружение enumerate формирует нумерованное перечисление. Аналогично ненумерованное перечисление создаётся с помощью окружения `itemize`. Обратите внимание на метки, поставленные с помощью команды `\label` вслед за заголовками. Используя эти метки, можно с помощью команд `\ref` и `\pageref` сослаться на номер и страницу соответствующего раздела. Для выставления правильной нумерации в ссылках компилятору потребуется выполнить два прохода:

```
pdflatex Kurs.tex
pdflatex Kurs.tex
```

Математика в LaTeX

Как неоднократно говорилось в статьях цикла ”Каталог классов и стилей LaTeX” этот издательский пакет был изначально оптимизирован для набора математических публикаций. Но перед тем как начать набирать математические выражения, нам предстоит изучить соответствующую TeX-нотацию.

Листинг 3. Пример математической нотации

Решение квадратного уравнения

```
\(ax^2+bx+c=0\):
\begin{equation}\label{eq:solv}
 x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}
\end{equation}
```

Можно сослаться на уравнение ~`\eqref{eq:solv}`.

Вставка кода

Для добавления неформатируемых фрагментов текста (например, программного кода) в LaTeX-лучше всего использовать окружение `verbatim`, как показано в листинге 4.

Листинг 4. Пример включения неформатируемого текста в LaTeX-документ

```
\begin{verbatim}
for alpha:=-90 step 3 until 0:
  label(btex IBM developerWorks etex
    scaled (5*(1+alpha/100)) rotated alpha,(0,0))
  withcolor (max(1+alpha/45,0)*red+
    min(-alpha/45,2+alpha/45)*green+
    max(-alpha/45-1,0)*blue);
endfor;

\end{verbatim}
```

Библиография

В конце любой научной работы обязательно должна присутствовать библиография, которую проще всего создать с помощью окружения thebibliography, как показано в листинге 5.

Листинг 5. Создание библиографии

```
\begin{thebibliography}{9}
\bibitem{Knuth-2003}Кнут Д.Э. Всё про \TeX. \newblock --- Москва:
Изд. Вильямс, 2003. 550~с.
\bibitem{Baldin-2008}Балдин Е.М. Компьютерная типография
\LaTeX. \newblock --- Санкт-Петербург: Изд. БХВ-Петербург,
2008. 302~с.
\end{thebibliography}
```

Команды `\bibitem` формируют библиографические ссылки, на которые можно ссылаться с помощью команды `\cite`, как показано ниже (ссылаться можно даже из тех фрагментов текста, которые располагаются выше определения ссылки):

Для изучения «внутренностей» `\TeX{}` необходимо изучить `~\cite{Knuth-2003}`, а для использования `\LaTeX{}` лучше почитать `~\cite{Baldin-2008}`. Как и в случае с перекрёстными ссылками для правильного отображения библиографических ссылок, исходный документ необходимо скомпилировать дважды. В заключении можно сказать, что сверстать документ так, чтобы его было приятно и удобно читать – это далеко не простая задача. Пакет LaTeX позволяет получить приемлемый результат за разумный промежуток времени без необходимости привлечения специалиста-верстальщика. Однако создание сложных текстов требует временных затрат на изучение возможностей LaTeX.

1.3. Изучение средств автоматизации MS Word. Изучение возможностей генерации Word-документов программным путем

В данном разделе описываются основные средства и возможности генерации документов Word при помощи программных средств. Приводятся результаты автоматического создания документов по заданному шаблону на основе массива имеющихся данных.

1.3.1. Генерация документов Word по шаблону с использованием технологии COM Interoperability

В данном разделе описывается процесс генерации документов Word по заданному шаблону с использованием технологии COM Interoperability, позволяющей получить доступ к объектам Word.

Существует множество способов вывести данные в документ Word. Количество подходов для решения данной проблемы порождает необъятную тему возможностей как самого Word, так и технологий, способных взаимодействовать с ним.

Если свести задачу к простейшему варианту - работе с простыми строками (типовая задача в крупных предприятиях - вставка дат, цифр, фио и тому подобных вещей) - самым простым решением на языке C будет являться механизм COM Interoperability(сокращенно Interop). Для этого потребуется открыть из кода C шаблон Word и что-то в него вставить. Далее необходимо приготовить шаблон .dot, используя тот же самый Word, и обращаться к нему, используя Interop. То есть необходимо запускать отдельный exe-процесс самого Word и через специальный интерфейс управлять им. Такой интерфейс находится в списке подключаемых библиотеках, поставляемых вместе с Office. Ниже перечислены следующие этапы, необходимые для реализации описанного процесса:

1. Подключить необходимые библиотеки (C)
2. Открыть шаблон Word
3. Определить необходимое место для подстановки
4. Вставить в необходимое место строку с информацией

Далее будет немного подробней описан процесс реализации. В начале в код добавляются необходимые библиотеки:

```
using Word = Microsoft.Office.Interop.Word;
using System.Reflection;
```

Далее, по вышеописанной инструкции, идёт работа с самим интерфейсом Word. При этом стоит учитывать следующие принципиально важные моменты:

Самый простой и примитивный вариант работы с шаблоном - поиск и замена строки в документе Word. Весь процесс заключается в том, чтобы добавить текстовую метку вроде `@@nowDate` и заменить её на необходимое значение.

Далее основная часть работы ведется с объектом Range. Данный объект представляет некую смежную область в документе, определяется начальным и конечным символами и может включать в себя все что угодно - от пары символов, до таблиц, закладок и т.д. Подобно тому, как закладки используются в документе, объекты Range используются в процедурах Visual Basic для идентификации определенных частей документа. Однако, в отличие от закладки, объект Range существует только при запущенной процедуре. Объекты диапазона не зависят от выбора. Соответственно есть возможность определять и манипулировать диапазоном без изменения выбора. Вы также можете определить несколько диапазонов в документе, в то время как для каждой панели может быть только один выбор.

Соответственно необходимо получить Range для всего документа, найти нужную строку внутри него, получить Range для этой строки и уже внутри этого последнего диапазона заменить текст на требуемый.

Если строку надо просто заменить, то сойдет простейший способ:

```
_range.Text = "Это текст заменит содержимое Range"
```

Но так как Range является универсальный контейнером для любого куска документа Word, то его возможности неизмеримо шире и могут быть рассмотрены на более высоких уровнях.

Если же необходимо просто вставить в начало документа:

```
object start = 0;
object end = 0
_currentRange = _document.Range(ref start, ref end);
```

Таким образом работает один из подходов генерации документов Word по шаблону, используя доступ к объектам Word.

1.3.2. Генерация документов Word по шаблону с использованием библиотеки C#

В данном разделе описывается возможность генерации документов Word по заданному шаблону при помощи подключаемой библиотеки на языке C#

EasyDox.dll – это бесплатная .NET библиотека для генерации вордовских документов (docx) по шаблону. Использовать ее очень просто:

```
var fieldValues = new Dictionary <string, string> {
    {"№ договора", "123-456/АГ"},
    {"Сторона 1",  "ООО «Ромашка»"},
    {"Сторона 2",  "ЗАО «Тюльпан»"},
    {"Подписант 1", "Иванов И.П."},
    {"Должность 1", "генеральный директор"},
    {"Основание 1", "Устав"},
};
var engine = new Engine ();

engine.Merge ("c:\\template.docx", fieldValues,"c:\\output.docx");
```

Функция Merge читает указанный файл шаблона и подставляет в него значения полей, заданных параметром `fieldValues`, и затем сохраняет результат в `"c:\output.docx"`.

Ниже представлен список преобразований:

Преобразование	Что делает
(родительный)	Ставит предшествующую позицию в родительный падеж.
(цифрами и прописью)	Преобразует число в запись суммы в рублях цифрами и прописью.

Текущая (первая) версия библиотеки включает в себя минимум преобразований. В последующих версиях будут добавлены остальные падежи для русского и украинского, а также возможность склонения по родам.

Библиотека расширяема и позволяет добавлять собственные преобразования. Набор пользовательских преобразований (функций) передается в конструктор класса Engine. Поля в колонтитулах (верхних и нижних) пока не обрабатываются.

Библиотека собрана под платформу AnyCPU (MSIL). Не требует установки OpenXml SDK. Все классы потокобезопасны.

1.4. Изучение средств автоматизации MS Word. Изучение возможностей генерации Word-документов программным путем

В данном разделе описываются основные функции и преимущества выбранной программной технологии для разработки веб-приложений на платформе .NET. Также описываются функциональные возможности ASP.NET Core MVC, упрощающие создание веб-API и веб-приложений.

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету. ASP.NET Core позволяет выполнять следующие задачи:

- Создавать веб-приложения и службы, приложения IoT и серверные части для мобильных приложений.
- Использовать избранные средства разработки в Windows, macOS и Linux.
- Выполнять развертывания в облаке и локальной среде.
- Работать в .NET Core или .NET Framework.

1.4.1. Преимущества использования ASP.NET

В данном разделе описываются основные преимущества использования ASP.NET по сравнению с известными аналогами. Обосновывается выбор данного программного средства для реализации поставленной задачи.

Миллионы разработчиков использовали ASP.NET (и продолжают использовать) для создания веб-приложений. ASP.NET Core является модификацией ASP.NET с внесенными архитектурными изменениями, формирующими более рациональную и модульную платформу.

ASP.NET Core предоставляет следующие преимущества:

- Единое решение для создания пользовательского веб-интерфейса и веб-API.
- Интеграция современных клиентских платформ и рабочих процессов разработки.
- Облачная система конфигурации на основе среды.
- Встроенное введение зависимостей.
- Упрощенный, высокопроизводительный и модульный конвейер HTTP-запросов.

- Возможность размещения в IIS или в собственном процессе.
- Возможность запуска на платформе .NET Core, которая поддерживает параллельное управление версиями приложения.
- Инструментарий, упрощающий процесс современной веб-разработки.
- Возможность сборки и запуска в ОС Windows, macOS и Linux.
- Открытый исходный код и ориентация на сообщество.

[5]

ASP.NET Core поставляется полностью в виде пакетов NuGet. Это позволяет оптимизировать приложения для включения только необходимых пакетов NuGet. За счет небольшого размера контактной зоны приложения доступны такие преимущества, как более высокий уровень безопасности, минимальное обслуживание и улучшенная производительность.

WCF и ASP.NET Web API

WCF является единой моделью программирования (Майкрософт) для построения ориентированных на службы приложений.Она позволяет разработчикам построить безопасные надежные решения с поддержкой транзакций и возможностью межплатформенной интеграции и взаимодействия с существующими инвестициями.

ASP.NET Web API — платформа .NET Framework, которая позволяет легко создавать службы HTTP, доступные для широкого круга клиентов, включая браузеры и мобильные устройства.ASP.NET Web API — это идеальная платформа для сборки REST-приложений на базе .NET Framework. Далее рассматривается сравнительная характеристика обоих технологий с целью решить, какая из них лучше подходит под конкретные требования.

В следующей таблице описаны основные возможности каждой из технологий:

WCF	ASP.NET Web API
Включает службы сборки, которые поддерживают несколько транспортных протоколов (HTTP, TCP, UDP и пользовательский транспорт) и допускают переключение между ними.	Только HTTP. Идеальная модель программирования для HTTP. Наиболее подходит для доступа из различных браузеров, мобильных устройств и т. д., обеспечивая более широкий охват.
Включает службы сборки, которые поддерживают разные кодирования (текст, MTOM и двоичные) одного типа сообщений и допускают переключение между ними	Позволяет создавать сетевые API-интерфейсы, которые поддерживают большое количество различных типов содержимого, в том числе XML, JSON и т. д.
Поддерживает создание служб по таким стандартам WS-*, как надежный обмен сообщениями, транзакции и безопасность сообщений.	Использует основные протоколы и форматы, такие как HTTP, WebSockets, SSL, JQuery, JSON и XML.Отсутствует поддержка протоколов высокого уровня, таких как надежный обмен сообщениями и транзакции.
Поддерживает шаблоны обмена сообщениями «запрос-ответ», «односторонний» и «дуплексный».	HTTP работает через «запрос-ответ», однако поддерживаются дополнительные шаблоны через интеграцию SignalR и WebSockets.
Службы WCF SOAP могут быть описаны в языке WSDL, что позволяет автоматическим средствам создавать прокси клиентов даже для служб со сложными схемами.	Имеются различные способы описания Web API — от автоматически формируемых html-страниц справки с описанием фрагментов до структурированных метаданных для интеграции API в OData.
Поставляется вместе с платформой .NET Framework.	Поставляется вместе с платформой .NET Framework, но имеет открытый код и доступна также по внешним каналам как независимая загрузка.

Исходя из этого можно сделать выводы о выборе подходящей технологии. Как видно, WCF лучше использовать для создания безопасных и надежных веб-служб. При этом ASP.NET Web

API лучше использовать для создания служб на основе HTTP, доступных для множества клиентов. ASP.NET Web API следует использовать при создании и разработке новых служб в стиле REST. Хотя WCF предоставляет некоторую поддержку написания служб в стиле REST, поддержка REST в ASP.NET Web API более полная и все последующие улучшения функций REST будут вноситься в ASP.NET Web API. [6] [7]

1.4.2. Создание веб-API и пользовательского веб-интерфейса с помощью ASP.NET Core MVC

В данном разделе описываются функциональные возможности ASP .NET Core MVC, играющие важную роль в выборе средств для создания веб-API и различного рода веб-приложений.

ASP.NET Core MVC предоставляет функциональные возможности, упрощающие создание веб-API и веб-приложений:

- Шаблон Model-View-Controller (MVC) помогает сделать веб-API и веб-приложения тестируемыми.
- Страницы Razor (новинка в версии 2.0) — это основанная на страницах модель программирования, которая упрощает и повышает эффективность создания пользовательского веб-интерфейса.
- Синтаксис Razor предоставляет эффективный язык для страниц Razor и представлений MVC.
- Вспомогательные функции тегов позволяют серверному коду участвовать в создании и отображении HTML-элементов в файлах Razor.
- Благодаря встроенной поддержке нескольких форматов данных и согласованию содержимого веб-API становятся доступными для множества клиентов, включая браузеры и мобильные устройства.
- Привязка модели автоматически сопоставляет данные из HTTP-запросы с параметрами метода действия.
- Проверка модели автоматически выполняет проверку на стороне сервера и клиента.
- Помимо прочего, ASP.NET Core легко интегрируется с широким спектром клиентских платформ, включая AngularJS, KnockoutJS и Bootstrap. [8]

1.5. Выводы

В данном разделе приводятся заключения относительно анализа предметной области и существующих методов решения поставленной задачи. Сформулированы основные выводы насчёт практического подхода к решению поставленной задачи.

В ходе аналитической части учебно-исследовательской работы были сделаны следующие заключения:

1. Для генерации PDF файлов будет применяться стандартная технология компиляции из исходных текстов на LaTeX с использованием дополнительно подключаемых пакетов.
2. Для генерации Word-документов на основе шаблонов будет использована технология Interop, позволяющая работать непосредственно с объектами Word.
3. Для разработки веб-сервиса была выбрана технология ASP.NET Web API 2.0.

1.6. Формальная постановка задачи

Целями курсового проекта являются:

1. Реализация механизмов генерации Word-файлов.
2. Реализация механизмов генерации PDF-файлов.
3. Реализация веб-сервиса, работающего с разработанными механизмами генерации документов, создание простейшего пользовательского интерфейса.

Для достижения первой цели необходимо изучить литературу, связанную с формальными средствами описания шаблонов объектов и правилами подстановки. Необходимо изучить и провести сравнительный анализ технологий генерации Word файлов программным путём. В конце необходимо провести тестирование и оценку производительности разработанного механизма.

Для достижения второй цели необходимо изучить литературу, описывающую процесс компиляции PDF файлов из исходных текстов на LaTeX. Также необходимо автоматизировать данный процесс. В конце необходимо провести тестирование и оценку производительности разработанного механизма.

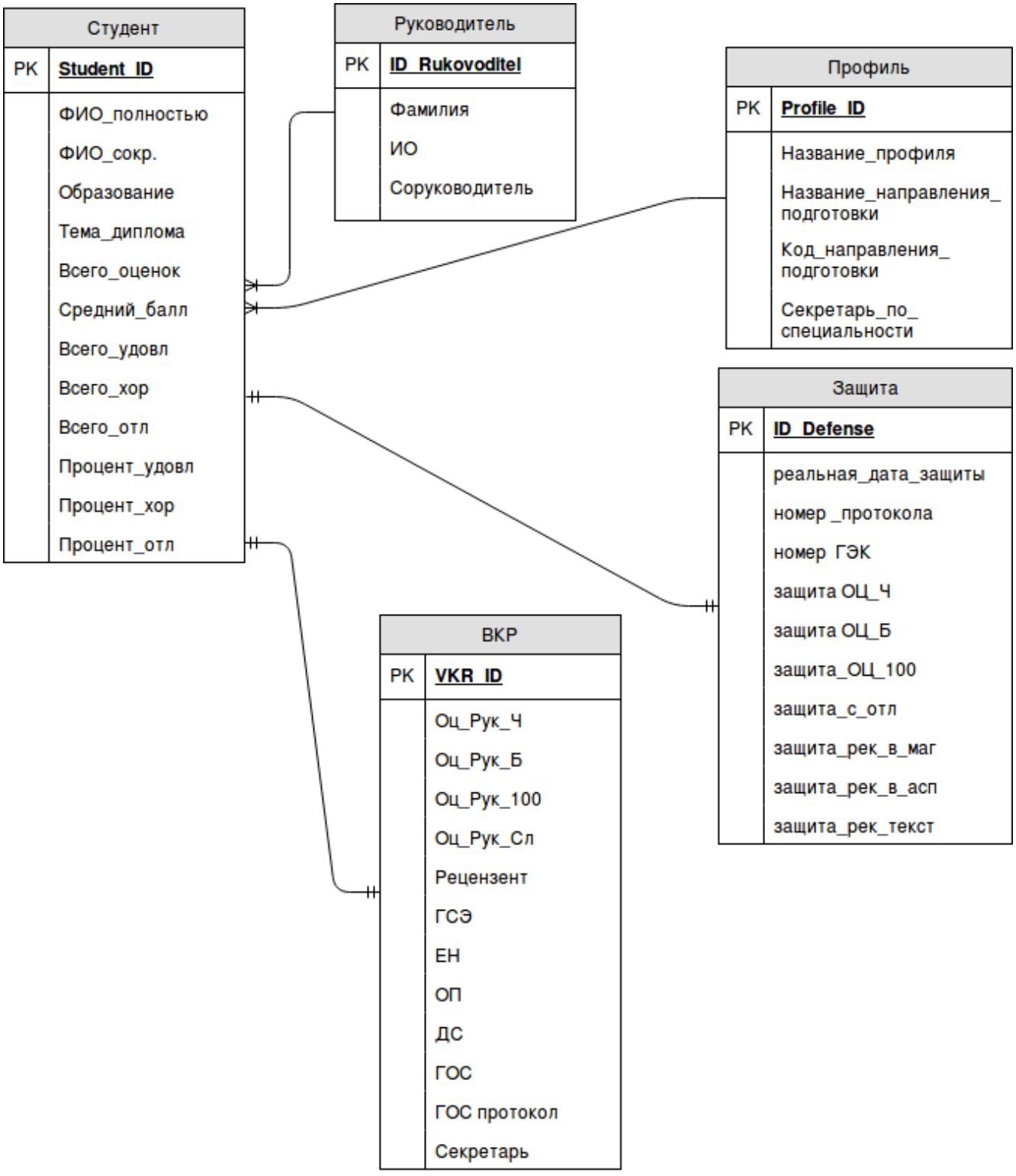
Для достижения третьей цели необходимо изучить литературу и провести сравнительный анализ серверных технологий для разработки веб-приложений на платформе .NET. Необходимо спроектировать систему программных веб-интерфейсов, разработать функциональную структуру информационной системы, а также провести тестирование с помощью выбранных утилит.

2. Разработка концептуальной модели тестовой предметной области – генерация протоколов для ВКР

В данном разделе описывается процесс разработки концептуальной модели тестовой предметной области, а именно генерации протоколов для ВКР.

На рис.2.1 представлена концептуальная модель тестовой предметной области, а именно генерации протоколов для ВКР. [9]

Рис. 2.1 – Концептуальная модель базы данных для генерации протоколов для ВКР



3. Проектирование системы

В данном разделе описывается инженерная часть научно-исследовательской работы, а именно проектирование прототипа системы генерации документов на основе шаблонов. Представлены требования к проектируемому модулю, описан выбор средств для описания результатов проектирования. Также описывается разработка функциональной структуры информационной системы.

3.1. Выбор средств (языка) для описания результатов проектирования

В данном разделе приведен сравнительный анализ и выбор средств (языка) для описания результатов проектирования. Описаны основные достоинства и приведены обоснования сделанного выбора относительно решения конечной задачи.

В данной работе в качестве языка проектирования был выбран язык UML, так как он обладает рядом преимуществ:

1. Визуальная интерпретация

   UML-диаграмма - это визуальное представление отношений между классами и сущностями в компьютерной программе. Класс - это объект в программировании, который организует аналогичные переменные и функции в одном месте. Чтобы понять программу, важно понять, что делает каждый объект класса, хранящаяся в нем информация и как она относится к другим классам в программе. Показывая эту информацию на диаграмме, легко понять и визуализировать отношения программы.

2. Читаемость и повторное использование

   Диаграмма UML выгодна тем, что она очень читаема. Диаграмма предназначена для понимания любым программистом и помогает объяснить отношения в программе простым способом. Традиционно, чтобы понять программу, программист будет читать код напрямую. Это может быть тысячи или миллионы строк кода в очень больших программах. Наличие UML-диаграммы помогает быстро проиллюстрировать эти отношения. Кроме того, используя диаграмму для отображения кода, запущенного в программе, программист может видеть

избыточный код и повторно использовать части кода, которые уже существуют, а не переписывать эти функции.

3. Стандарт

   UML - это текущий стандарт программирования в объектно-ориентированных языках программирования. При создании классов и других объектов с отношениями между собой UML используется для визуального описания этих отношений. Поскольку он используется в качестве стандарта, он широко понимается и хорошо известен. Это позволяет новому программисту вступить в проект и быть продуктивным с первого дня.

[10] UML - унифицированный язык моделирования. Из этих трех слов главным является слово ” язык ”.

Язык - это система знаков, служащая:

1. средством человеческого общения и мыслительной деятельности;
2. способом выражения самосознания личности;
3. средством хранения и передачи информации.

[11] Язык включает в себя набор знаков (словарь) и правила их употребления и интерпретации (грамматику).

Актуальность построения диаграмм объясняется тем, что разработка модели любой системы (не только программной) всегда предшествует ее созданию или обновлению. Это необходимо хотя бы для того, чтобы яснее представить себе решаемую задачу. Продуманные модели очень важны и для взаимодействия внутри команды разработчиков, и для взаимопонимания с заказчиком. В конце концов, это позволяет убедиться в ”архитектурной согласованности” проекта до того, как он будет реализован в коде.

3.2. Проектирование системы программных веб-интерфейсов

В данном разделе описан процесс и представлены результаты проектирования системы программных веб-интерфейсов для разрабатываемого прототипа.

Для проектирования архитектуры модуля был использован объектно- ориентированный подход. Объектно-ориентированное программирование рассматривает абстрактные сущности, называемые объектами. Эти сущности могут пониматься как совокупность выделенных свойств. Опираясь на них и отношения между ними, мы можем реализовывать сложную логику приложения.

Программный модуль, разрабатываемый в данной работе состоит пакетов и классов, составляющих само приложение. Основные пакеты следующие: EasyDox, EasyDox.Morpher, Microsoft.Office.Inte Microsoft.Office.Interop.Word. Первые два пакета необходимы для заполнения шаблона документа данными и склонения определенных словесных конструкций, таких как ФИО, по падежам. Последние два пакета включают в себя функции, упрощающие доступ к объектам API Office. Основными классами являются DataImport, Declination, DataExport.

DataImport отвечает за чтение файла базы данных посредством методов пакета Interop и извлечения необходимых данных. Также , для импорта данных использовалась пользовательская библиотека EasyDox, в которой прописан класс Engine, отвечающий за заполнение шаблонов данными.

Declination отвечает за склонение необходимых словесных конструкций по падежам. Для этого используется библиотека EasyDox.Morpher параметры которой передаются в конструктор класса Engine.

DataExport отвечает за генерирование нового документа Word, с полями шаблона, заполненными необходимой информацией изучаемой предметной области. В результате проектирования программного модуля была получена диаграмма классов, представленная на рисунке 3.1.

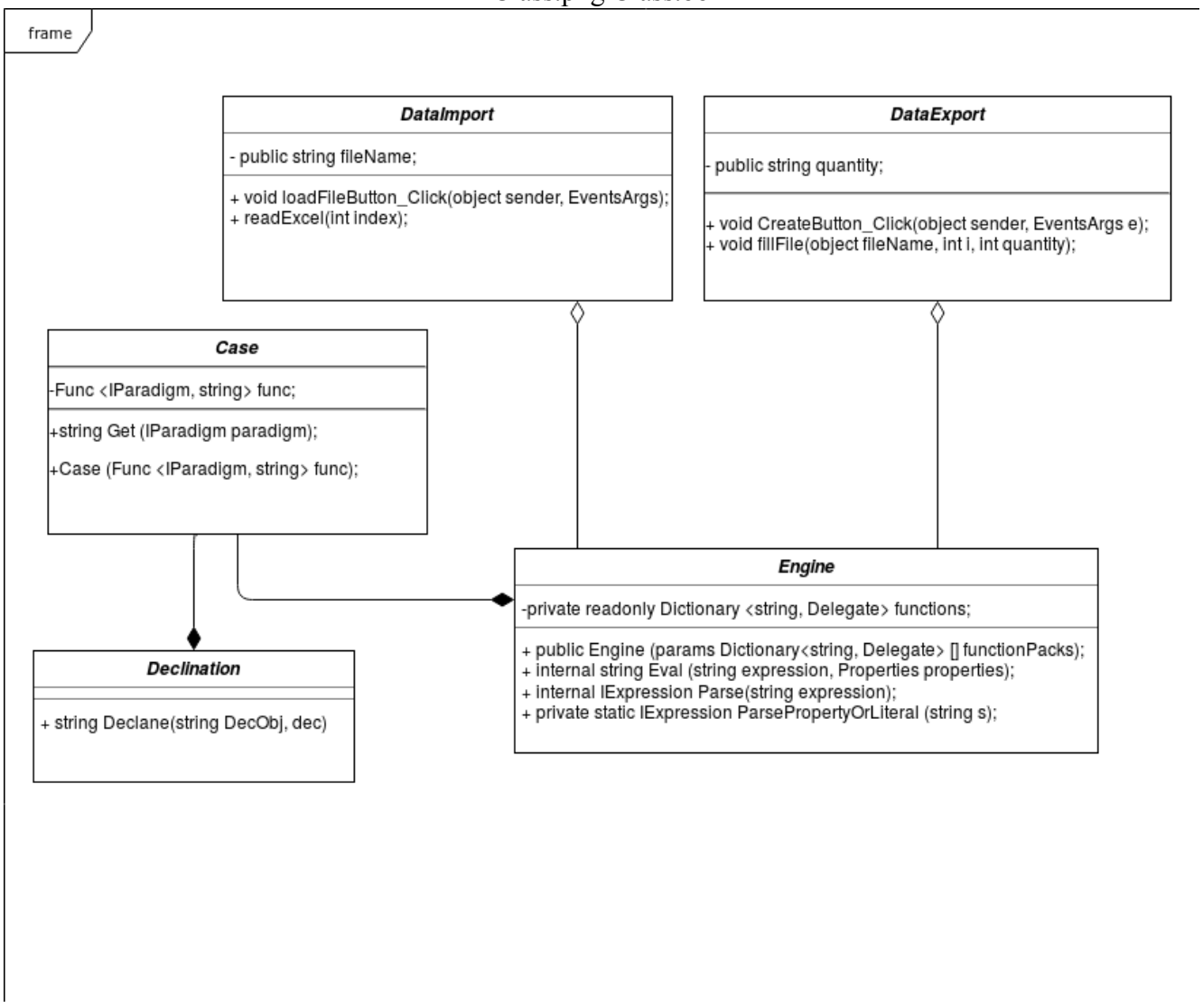
3.3. Разработка функциональной структуры информационной системы

В данном разделе описывается процесс разработки функциональной структуры информационной системы. Описаны основные функции и связи.

Одним из основных методов является метод класса DataImport - ReadExcel, который позволяет при помощи встроенных методов пакета Interop взаимодействовать с объектами Excel, а конкретно получать необходимые файлы из ячеек базы данных. Метод ReadTemplate класса DataExport позволяет открыть заданный шаблон и найти в нем поля для заполнения необходимыми данными. Метод Decline класса Declination склоняет словесные конструкции русского языка по заданным падежам, а метод FillData класса DataExport заполняет поля шаблона готовыми данными в необходимых падежах и создаёт новый документ (отчёт) Word. В простейшей версии пользовательского интерфейса пользователь сможет выбирать базу данных (Excel), шаблон документа (Word), а также задавать число итераций, для получения желаемого количества отчётов. При дальнейшей разработке планируется создание веб-сервиса с доработанным функционалом, а также серверным и клиентским кодом. [12]

Рис. 3.1 – Диаграмма классов

Class.png Class.bb



3.4. Выводы

В данном разделе подводятся итоги проектирования разрабатываемой системы.

В данной главе описаны основные пакеты, которые содержит разрабатываемый программный модуль, а также перечислены классы, включенные в эти пакеты. По итогам данного этапа проектирования получена архитектура системы, которая отражает ключевые моменты реализации модуля.

4. Программная реализация прототипа системы генерации документов на основе шаблонов

В данном разделе описывается техническая сторона выполнения поставленной задачи. Описан принцип работы исходных текстов, а также процесс реализации механизмов генерации Word-документов. Представлены результаты создания пробного Word-документа

4.1. Выбор программных средств для реализации системы

В данном разделе описан процесс выбора инструментальных средств для разработки прототипа генерации Word-документов по шаблону.

В данной работе было принято решение использовать возможности языка C. В центре внимания объектно-ориентированного подхода ставятся объекты (абстрактные сущности). Это позволяет разбивать большие задачи на некоторое множество объектов и описывать отношения между ними. В такой модели объекты — это самостоятельные юниты, т. е. объекты способны получать сообщения друг от друга и реагировать на них, регулируя, тем самым, собственное поведение. Программирование в объектно-ориентированном стиле, являющемся частью императивной парадигмы, представляет собой последовательный переход программы из одного состояния в другое.

C позволяет стартовать разработку быстрее, а это позволяет быстрее получить прототип решения. Скорость разработки на C на начальных этапах проекта значительно выше по сравнению с C++.

C спроектирован быть кросплатформенным, однако его развитие не пошло в этом направлении. Поэтому под Windows образовалась достаточно полная .net инфраструктура; на других же платформах равноценной инфраструктуры не появилось.

Программы C выполняются на платформе .NET Framework, которая интегрирована в Windows и содержит виртуальную общеязыковую среду выполнения (среду CLR) и унифицированный набор библиотек классов. Среда CLR корпорации Майкрософт представляет собой коммерческую реализацию международного стандарта Common Language Infrastructure (CLI), который служит основой для создания сред выполнения и разработки, позволяющих совместно использовать разные языки и библиотеки.

Для работы с объектами Office было принято использовать инструменты пакета Microsoft.Office.Inter

однако также были рассмотрены и использованы функциональные возможности пользовательской библиотеки EasyDox, которая позволяет генерировать документы Word по шаблону, а также содержит пакет Morpher, в котором описаны методы, реализующие склонение по падежам.

4.2. Реализация механизмов генерации Word-файлов.

В данном разделе описан процесс реализации механизмов генерации Word-файлов выбранным при помощи выбранных инструментальных средств. Приводится информация о тестировании разработанных механизмов и даётся оценка производительности.

Для понимания механизмов работы с объектами Word и Excel, было принято решение в качестве инструмента программной реализации использовать пакет Microsoft Office Interop, как универсального средства для решения задачи данного типа.

Несмотря на то, что в базе данных уже имеются вариации словесных конструкций в необходимых падежах, в пакете EasyDox.Morpher присутсвуют классы и методы, позволяющие склонять по падежам, имея только форму именительного.

Был получен простейший пользовательский интерфейс, позволяющий выбрать файл базы данных для чтения, а затем извлечь необходимые данные и сгенерировать документ Word по заданному шаблону.

Для тестирования работоспособности прототипа был построен простейший шаблон страницы отчетности по результатам защиты ВКР и заполнен данными из базы книги Excel. На рис. 4.1 показан результат работы модуля (выделенным цветом обозначаются поля, заполненные программным путем)

Рис. 4.1 – Заполнение тестового документа по шаблону

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Национальный исследовательский ядерный университет «МИФИ»

ВЫПИСКА ИЗ ПРОТОКОЛА

от 03.07.2017 №23

заседания Государственной экзаменационной комиссии

по направлению подготовки (специальности): **09.03.04**

Программная инженерия

1. СЛУШАЛИ: защиту выпускной квалификационной работы **Фильшина Александра Алексеевича**

2. На основании результатов учебной успеваемости, выполнения и защиты выпускной квалификационной работы Государственная экзаменационная комиссия ПОСТАНОВИЛА:

   1) Признать, что студент Фильшин Александр Алексеевич защитил выпускную квалификационную работу с оценкой «отлично», ECTS A (100).

   2) Присвоить выпускнику Фильшину Александру Алексеевичу квалификацию (степень) «бакалавра» по направлению подготовки (специальности) 09.03.04 «Программная инженерия».

   3) Выдать диплом государственного образца без отличия выпускнику НИЯУ МИФИ Фильшину Александру Алексеевичу по направлению подготовки (специальности) 09.03.04 «Программная инженерия».

   4) Другое: рекомендовать Фильшина Александра Алексеевича для поступления в магистратуру.

Секретарь ГЭК _____ Федорец О.В.

Заключение

В данной работе был рассмотрен процесс генерации документов Word программным путем по шаблону. Были изучены аспекты различных технологий, используемых при автоматической генерации содержимого документов, были разобраны основы работы с объектами Microsoft Office, в частности Word и Excel, а также был разобран функционал пользовательских пакетов C, предназначенных для решения описанной задачи. Была построена концептуальная модель преметной области а также спроектирована архитектура разрабатываемой системы. В качестве тестового примера был сгенерирован простейший протокол ВКР, заполненный по шаблону данными, полученными из базы данных. В дальнейшем планируется продолжит рассмотрение механизмов генерации документов Word и PDF программным путем, а также при реализации веб-сервиса. Для построения полноценной системы основным инструментом разработки веб-приложений была выбрана технология ASP.NET Web API 2.0, что позволит реализовать весь необходимый функционал веб-сервиса. В конечном итоге планируется реализовать веб-ориентированную систему генерации документов на основе шаблонов, которая может применяться для автоматического создания необходимых документов на основе имеющегося массива данных.

Список литературы

1. *Henk Barendregt, Erik Barendsen*. Introduction to Lambda Calculus. — Revised edition, December 1998, March 2000. - 53 p.

2. *Barry Jay*. Pattern Calculus. — University of Technology, Sydney. Faculty of Engineering and Information Technology School of Software. Broadway NSW , Australia, 2007. - 214 p.

3. . Работа в LaTex. IBM `https://ru.wikibooks.org/wiki/LaTeX/LaTeX`.

4. *С. М. Львовский*. Набор и вёрстка в система LaTeX. — изд.3. 2003. 448 – с.

5. *Daniel Roth, Rick Anderson, Shaun Luttin*. Introduction to ASP.NET Core with Dynamically Typed Languages. `https://docs.microsoft.com/ru-ru/aspnet/core/`.

6. *Дж.Рихтер*.CLR via C_. Программирование на платформе Microsoft.NET Framework 4.5 на языке C_. — изд. 4-е. – СПб.: Питер, 2013. – 896 с.

7. *А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд.*Язык программирования C_. Классика Computers Science. 4-е изд.. — 4-е изд., 2011. – 784 с.

8. *Adam Freeman*. Pro ASP.NET MVC 5. — publ. 5. 2013. 832 – с.

9. *А.В.Кузовкин, А.А.Цыганов,Б.А.Щукин* Управление Данными. Издательский центр ”Академия”, 2010. – 256 с.

10. *Национальный Открытый Университет «ИНТУИТ»*. Преимущества UML. — URL: `https://www.intuit.ru/studies/courses/1007/229/lecture/5952` (Дата обращения: 10.10.17)

11. *Craig Larman* Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. — Addison Wesley Professional, October 20, 2004. – 736 p.

12. *Пошаговое руководство. Программирование приложений Office (C и Visual Basic)*. URL: `https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/interop/walkthrough-office-programming` (Дата обращения: 17.11.17)

13. Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics — Elsevier, MIT Press, 1990. – 1264 p.

14. *Дейт К. Дж*. Введение в системы баз данных/ 8-е издание. — Издательский дом «Вильяме», 2005. – 1328 с.: ил.

15. *Rozentals N.* Mastering TypeScript. Build enterprise-ready, industrial strength web applications using TypeScript and leading JavaScript frameworks. — Packt Publishing Ltd, Birmingham, UK. – 2015.

Приложение 1. Листинги кода приложения

Класс DataImport

```
namespace WordReport
{
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

using Word = Microsoft.Office.Interop.Word;
Microsoft.Office.Interop.Word.Application app = new Microsoft.Office.Interop.Word.
    Application();

 OpenFileDialog ofd = new OpenFileDialog();

private void button2_Click(object sender, EventArgs e)

{

ofd.Filter = "XLSM| *.xlsm";

if (ofd.ShowDialog() == DialogResult.OK)

{

textBox1.Text = ofd.FileName;

textBox2.Text = ofd.SafeFileName;

}

}

 private string[] readExcel(int index)

{

string res = textBox1.Text;

Excel.Application xlApp;

Excel.Workbook xlWorkBook;

Excel.Worksheet xlWorkSheet;

xlApp = new Excel.Application();

xlWorkBook = xlApp.Workbooks.Open(res, 0, true, 5, "", "", true);

xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

```

```
index += 2;

string[] data = new string[13];

data[0] = xlWorkSheet.get_Range("BA" + index.ToString()).Text;

data[1] = xlWorkSheet.get_Range("AP" + index.ToString()).Text;

data[2] = xlWorkSheet.get_Range("AR" + index.ToString()).Text;

data[3] = xlWorkSheet.get_Range("D" + index.ToString()).Text;

data[4] = xlWorkSheet.get_Range("C" + index.ToString()).Text;

data[5] = xlWorkSheet.get_Range("AD" + index.ToString()).Text;

data[6] = xlWorkSheet.get_Range("AB" + index.ToString()).Text;

data[7] = xlWorkSheet.get_Range("AC" + index.ToString()).Text;

data[8] = xlWorkSheet.get_Range("F" + index.ToString()).Text;

data[9] = xlWorkSheet.get_Range("K" + index.ToString()).Text;

data[10] = xlWorkSheet.get_Range("BH" + index.ToString()).Text;

data[11] = xlWorkSheet.get_Range("BK" + index.ToString()).Text;

data[12] = xlWorkSheet.get_Range("AH" + index.ToString()).Text;

xlWorkBook.Close(false);

xlApp.Quit();

System.Runtime.InteropServices.Marshal.ReleaseComObject(xlWorkSheet);

System.Runtime.InteropServices.Marshal.ReleaseComObject(xlWorkBook);

System.Runtime.InteropServices.Marshal.ReleaseComObject(xlApp);

return data;

}
```

Класс DataExport

```
namespace WordReport
{
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

using Word = Microsoft.Office.Interop.Word;
Microsoft.Office.Interop.Word.Application app = new Microsoft.Office.Interop.Word.
    Application();
```

```
Microsoft.Office.Interop.Word.Document doc = null;

object fileName = "E:\\UIR 2017–2018\\Material\\protocol_extract.doc";

object missing = Type.Missing;

for(int i =2; i<7; i++)

{

doc = app.Documents.Open(fileName, missing, missing);

app.Selection.Find.ClearFormatting();

app.Selection.Find.Replacement.ClearFormatting();

// read excel file

string[] tmp = new string[4];

tmp = readExcel(i);

// fill data to template

app.Selection.Find.Execute("<date>", missing, missing, missing, missing, missing,
    missing, missing, missing, tmp[0], 2);

app.Selection.Find.Execute("<spec_numb>", missing, missing, missing, missing,
    missing, missing, missing, missing, tmp[1], 2);

app.Selection.Find.Execute("<spec_nazv>", missing, missing, missing, missing,
    missing, missing, missing, missing, tmp[2], 2);

app.Selection.Find.Execute("<name RP>", missing, missing, missing, missing,
    missing, missing, missing, missing, tmp[3],2);

app.Selection.Find.Execute("<name IP>", missing, missing, missing, missing,
    missing, missing, missing, missing, tmp[4],2);

app.Selection.Find.Execute("<mark>", missing, missing, missing, missing, missing,
    missing, missing, missing, tmp[5],2);

app.Selection.Find.Execute("<letter>", missing, missing, missing, missing, missing
    , missing, missing, missing, tmp[6],2);

app.Selection.Find.Execute("<number>", missing, missing, missing, missing, missing
    , missing, missing, missing, tmp[7], 2);

app.Selection.Find.Execute("<name DP>", missing, missing, missing, missing,
    missing, missing, missing, missing, tmp[8], 2);

app.Selection.Find.Execute("<obrazov>", missing, missing, missing, missing,
    missing, missing, missing, missing, tmp[9], 2);

app.Selection.Find.Execute("<otlichie>", missing, missing, missing, missing,
    missing, missing, missing, missing, tmp[10], 2);

app.Selection.Find.Execute("<text_postup>", missing, missing, missing, missing,
    missing, missing, missing, missing, tmp[11], 2);

app.Selection.Find.Execute("<recenzent>", missing, missing, missing, missing,
    missing, missing, missing, missing, tmp[12], 2);

// save new file as report

object SaveAsFile = (object)"E:\\reports\\Report" + tmp[4] + ".doc";

doc.SaveAs2(SaveAsFile, missing, missing, missing);

}

MessageBox.Show("Files are created");

doc.Close(false, missing, missing);
```

```
app.Quit(false, false, false);

System.Runtime.InteropServices.Marshal.ReleaseComObject(app);

}
```

Класс Case

```
using System;

namespace Morpher.Russian
{
    public class Case : ICase <IParadigm>
    {
        private readonly Func <IParadigm, string> func;

        Case (Func <IParadigm, string> func)
        {
            this.func = func;
        }

        public string Get (IParadigm paradigm)
        {
            return func (paradigm);
        }

        public static Case Nominative    = new Case (p => p.Nominative   );
        public static Case Genitive      = new Case (p => p.Genitive     );
        public static Case Dative        = new Case (p => p.Dative       );
        public static Case Accusative    = new Case (p => p.Accusative   );
        public static Case Instrumental  = new Case (p => p.Instrumental );
        public static Case Prepositional = new Case (p => p.Prepositional);
        public static Case Locative      = new Case (p => p.Locative     );
    }
}
}
```

Класс IParadigm

```
namespace Morpher.Russian
{
    /// <summary xml:lang="en">
    /// </summary>
    /// <summary xml:lang="ru">
    /// Представляет русскую падежную парадигму набор( падежных форм), где каждая
   форма представлена строковым свойством.
    /// </summary>
    public interface IParadigm : ISlavicParadigm
    {
        string Locative {get;}
    }
}
```

Класс FullParadigm

```
using System;
using Morpher.WebService.V2.MorpherWebService;

namespace Morpher.Russian
{
    class Parse : Paradigm, IParse
    {
        private readonly GetXmlResult result;

        public Parse (GetXmlResult result, string lemma) : base (lemma)
        {
            this.result = result;
        }

        Gender IParse.Gender
        {
            get
            {
                switch (resultрод.)
                {
                    case "Мужской": return Gender.Masculine;
                    case "Женский": return Gender.Feminine;
                    case "Средний": return Gender.Neuter;
                    case "": return Gender.Plural;
                }

                throw new ApplicationException("Вебсервис— вернул неожиданное
   значение рода: " + resultрод.);
            }
        }

        IParadigm IParse.Plural
        {
            get { throw new NotImplementedException (); }
        }

        protected override Forms Forms
        {
            get { return result; }
        }

        protected override string Locative
        {
            get { return resultгде.; }
        }

        bool IParse.IsAnimate
        {
            get { throw new NotImplementedException (); }
        }

        string IParse.Paucal
        {
            get { throw new NotImplementedException (); }
        }
    }
}
```

Класс MorpherFunctionPack

```
using System;
using System.Collections.Generic;
using System.Globalization;
using Morpher.Generic;
using Morpher.Russian;

namespace EasyDox
{
    using System.Linq;
    using System.Text;
```

```
    /// <summary>
    /// Предоставляет функции склонения по падежам и прописи денежных сумм на русском языке.
    /// </summary>
    public class MorpherFunctionPack
    {
        private readonly IDeclension declension;
        private readonly INumberSpelling numberSpelling;

        /// <summary>
        /// Creates a new instance of the <see cref="MorpherFunctionPack"/>.
        /// </summary>
        public MorpherFunctionPack (IDeclension declension, INumberSpelling numberSpelling)
        {
            this.declension = declension;
            this.numberSpelling = numberSpelling;
        }

        /// <summary>
        ///
        /// </summary>
        public Dictionary <string, Delegate> Functions => new Dictionary <string, Delegate>
        {
            {"родительный", new Func<string, stringРодительный>()},
            {"цифрами и прописью", new Func<string, string> Пропись()},
            {"фамилия и. о.", new Func<string, string> ФамилияИнициалы()},
        };

        /// <summary>
        /// Сокращает ФИО до Фамилия И. О.
        /// </summary>
        /// <param name="fieldValueФИО"></param>
        /// <returnsФамилия> И. О.</returns>
        public static string ФамилияИнициалы (string fieldValue)
        {
            if (string.IsNullOrWhiteSpace(fieldValue))
            {
                throw new ArgumentException("String is null or white space", nameof(fieldValue));
            }

            var parts = fieldValue.Split (" .".ToCharArray (), StringSplitOptions.RemoveEmptyEntries).ToList();
            if (parts.Count == 0)
            {
                throw new ArgumentException("Нет частей ФИО");
            }

            if (parts[0].ToLowerInvariant() == "ип")
            {
                parts.RemoveAt(0);
            }

            if (parts.Count == 0)
            {
                throw new ArgumentException("После удаления ИП нет частей ФИО");
            }

            StringBuilder fioBuilder = new StringBuilder();
            fioBuilder.Append(parts[0]);
            if (parts.Count > 1)
            {
                foreach (var part in parts.Skip(1))
                {
                    fioBuilder.AppendFormat(" {0}.", part[0]);
                }
            }

            return fioBuilder.ToString();
        }

        string Родительный (string fieldValue)
```

```
        {
            var paradigm = declension.Parse(fieldValue);

            return paradigm == null
                ? fieldValue // TODO: return a warning / error?
                : paradigm.Genitive;
        }

        string Пропись (string fieldValue)
        {
            return new ДенежнаяЕдиница
                        {
                            ПолноеНаименованиеЦелойЧасти = "рубль",
                            ПолноеНаименованиеДробнойЧасти = "копейка",
                            СокращенноеНаименованиеЦелойЧасти = "руб.",
                            СокращенноеНаименованиеДробнойЧасти = "коп."
                        }
                    СуммаПрописью. (Decimal.Parse (fieldValue, CultureInfo.
    GetCultureInfo ("en-US")), numberSpelling.AsGeneric().DefaultCase());
        }
    }
}
```

Класс UnitParadigm

```
using Morpher.WebService.V2.MorpherWebService;

namespace Morpher.Russian
{
    class UnitParadigm : Paradigm
    {
        private readonly Forms forms;

        public UnitParadigm (Forms forms) : base (formsИ.)
        {
            this.forms = forms;
        }

        protected override Forms Forms
        {
            get { return forms; }
        }

        protected override string Locative
        {
            get { return formsП.; } // Совпадает с предложным.
        }
    }
}
```

Класс UnitTest

```
namespace EasyDox.Morpher.UnitTests
{
    using System;
    using System.Reflection;
    using NUnit.Framework;

    [TestFixture]
    public class UnitTest
    {
        [Test]
        public void ИП_Test() => Assert.AreEqual("Слепов С. Н.",
    MorpherFunctionPackФамилияИнициалы.("Слепов Сергей Николаевич"));

```

```
14          [Test]
            public void Dot_Test() => Assert.AreEqual("Слепов С.", MorpherFunctionPack
    ФамилияИнициалы.("Слепов С. ."));
16
            [Test]
18          public void DoubleDot_Test() => Assert.AreEqual("Слепов С.",
    MorpherFunctionPackФамилияИнициалы.("Слепов С.."));
     }
20  }
```