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

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

02.03.03 МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ И АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

БАКАЛАВРСКАЯ РАБОТА

на тему

«Разработка приложения для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR»

Студентка Е.Ш. Млукова

Руководитель Т.Г. Султанов

Консультант А.В. Кириллова

Допустить к защите
Заведующий кафедрой А.В. Очеповский

«_____» _____ 2017 г.

Тольятти 2017

АННОТАЦИЯ

Тема выпускной работы - Разработка приложения для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR. В выпускной работе предлагается программа для облегчения процесса создания UML-диаграмм на основе текстовой информации с использованием методов обработки естественного языка.

Автор проводит анализ технологий консорциума стандартов компьютерной индустрии, который разрабатывает и поддерживает спецификации компьютерной индустрии для совместимых, переносимых и многоразовых корпоративных приложений в распределенных гетерогенных средах. Ключевой проблемой выпускной работы является разработка приложения для генерации UML-моделей на основе анализа текстовой информации в соответствии со стандартом SBVR. В работе рассматривается анализ технологий OMG для визуального моделирования программного обеспечения, создания текстового документа с использованием стилей расширяемого языка разметки, с использованием компьютерной разработки программного обеспечения и анализа документа.

Бакалаврскую работу можно разделить на несколько логически связанных частей: анализ технологий OMG для создания UML моделей, разработка программного обеспечения для создания UML моделей, тестирование и анализ программного обеспечения.

Была проведена работа по созданию метода помогающему анализировать текстовые требования, находить основные понятия и отношения и извлекать диаграммы UML.

Бакалаврская работа состоит из пояснительной записки на 41 странице, введения, включая 29 рисунков, 3 таблиц, список из 20 ссылок, включая 5 иностранных источников и 3 приложения.

ABSTRACT

The title of the graduation work is Development of an Application for Generation of UML Models Based on the Analysis of Textual Information in Accordance with the SBVR Standard. This graduation work is about creating of the method to facilitate the creation of UML diagrams based on textual information using natural language processing methods.

The author conducts technology analysis of computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. The key issue of the graduation work is the development of an application for generation of UML models based on the analysis of textual information in accordance with the SBVR standard. The work touches upon analysis OMG's technologies for visual software modelling, creating a text document using styles of Extensible Markup Language, using computer-aided software engineering and, how the document is parsed.

The graduation work may be divided into several logically connected parts which are: analysis of OMG technologies for generation of UML models, designing software for generating UML models, implementation and testing of software.

Project was made to create a method to assist requirements analysts and Software Engineering students to analyze textual requirements, finding core concepts and its relationships, and extraction UML diagrams.

The graduation work consists of an explanatory note on 41 pages, introduction, including 29 figures, 3 tables, the list of 20 references including 5 foreign sources and 3 appendices.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Объектно-ориентированная разработка программного обеспечения связана с применением объектно-ориентированных моделей при разработке программных систем и их компонентов. Составными частями объектно-ориентированной методологии (ООМ) являются:

- объектно-ориентированный анализ – направлен на создание моделей, более близких к реальности, с использованием объектно-ориентированного подхода;
- объектно-ориентированное проектирование – это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления как логической и физической, так статической и динамической моделей проектируемой системы;
- объектно-ориентированное программирование – методология программирования, которая основана на представлении программы в виде совокупности объектов, каждый из которых является реализацией определенного класса, а классы образуют иерархию на принципах наследования [11].

Эти этапы объектно-ориентированной методологии необходимы при разработке качественного программного обеспечения.

The Object Management Group (OMG) является открытым некоммерческим консорциумом по стандартам компьютерной индустрии, который разрабатывает и поддерживает спецификации компьютерной индустрии для совместимых, портативных и многоразовых корпоративных приложений.

В последнее время были созданы различные методы представления бизнес-правил [10]. Некоторые из них пытаются представить бизнес-правила визуально. Однако, из-за сложности проблемы, графические представления, которые предлагаются, иногда далеки от совершенства. В работе

рассматривается разбор и анализ текстового документа и процесс генерации диаграмм UML из спецификации естественного языка.

Процесс генерации диаграмм UML из спецификации естественного языка является очень сложной задачей. В работе предлагается метод для облегчения процесса создания UML-диаграмм на основе текстовой информации с использованием методов обработки естественного языка. Инженеры анализируют информацию вручную, чтобы понять сферу действия системы. Время, затраченное на анализ и низкое качество человеческого анализа, оправдывает потребность в инструменте для лучшего понимания системы. Данный метод помогает анализировать текстовые требования, находить основные понятия и отношения и извлекать диаграммы UML.

Объект бакалаврской работы: Объектно-ориентированные методы разработки программного обеспечения.

Предмет бакалаврской работы: автоматическая генерация UML моделей из файлов текстовых форматов в соответствии с технологией SBVR.

Целью бакалаврской работы является разработка приложения для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR.

Задачами бакалаврской работы, исходя из поставленной цели, являются:

- анализ OMG стандарта SBVR;
- разработка приложения для генерации UML моделей;
- тестирование приложения для генерации UML моделей;

Бакалаврская работа состоит из трех глав:

Первая глава работы посвящена анализу OMG технологий MDA, UML, XML, спецификациям бизнес правил SBVR и анализу текстового файла.

Вторая глава описывает проектирование и разработку приложения для генерации UML моделей.

В третьей главе проводится тестирование и анализ полученного решения приложения для генерации UML моделей.

ГЛАВА 1 АНАЛИЗ ТЕХНОЛОГИЙ OMG ДЛЯ ГЕНЕРАЦИИ UML МОДЕЛЕЙ

1.1 Анализ технологий OMG

The Object Management Group (OMG) – это международное открытое членство, не коммерческая компьютерная технология стандартов консорциума, была основана в 1989 году. OMG стандарты управляются поставщиками, конечными пользователями, учебными заведениями и правительственными учреждениями [7].

Стандарты моделирования OMG, включают унифицированный язык моделирования – Unified Modeling Language® (UML®) и архитектуру, управляемой моделью – Model Driven Architecture® (MDA®), включает мощный визуальный дизайн, оформление и сопровождение программного обеспечения и других процессов.

OMG также заведует организациями, такие, как совет клиентов по облачным стандартам (CSCC™) и IT-индустрии качества программного обеспечения группы стандартизации, консорциум по качеству программного обеспечения информационных технологий (CISQ™). OMG также управляет консорциумом промышленного интернета (IIC), государственно-частным партнёрством (PPP), которое было сформировано в 2014 году с американским транснациональным телекоммуникационным конгломератом (AT&T), американской транснациональной компанией, разрабатывающей и продающей сетевое оборудование (Cisco), американской многоотраслевой корпорацией (GE), американской компанией производителей и поставщиков аппаратного и программного обеспечения, а также ИТ-сервисов и консалтинговых услуг (IBM) и производителем электронных устройств и компьютерных компонентов – Intel, направленное на развитие, адаптацию, и инновацию промышленных интернет вещей.

OMG поддерживает связи с десятками других организаций, включая ISO, Health Level Seven (HL7) и Коалицию прозрачности данных (Data Transparency Coalition).

На рисунке 1.1. представлены некоторые технологии, курируемые консорциумом Object Management Group [7] .

Рисунок 1.1 – Технологии OMG

Рассмотрим некоторые из технологий, которые будут необходимы при объектно- ориентированном анализе:

1. Model Driven Architecture – это архитектурный подход к построению многокомпонентного ПО. Разработка MDA началась в 2000 году. К концу 90-х годов прошлого столетия было создано большое количество технологий и протоколов для создания распределенных приложений. В качестве примеров можно привести COM/DCOM, CORBA, Java/RMI, XML/SOAP/RPC и др. Большинство таких технологий созданы зависимыми от платформы, не все из них были описаны в виде стандарта.

Основной целью разработчиков MDA являлось создание архитектурного подхода, позволяющего снизить риск, вызванный различиями между и технологиями разработки программных систем и платформами. Для решения этой задачи в MDA вводятся понятия модели, зависимой от платформы (Platform Specific Model (PSM)), и модели, независимой от платформы (Platform Independent Model (PIM)). MDA предполагает создание PIM и последующий переход к PSM с использованием специализированных средств. Фактически, MDA – это концепция разработки, поддержанная группой стандартов (UML, MOF, CWM и XMI), разработанных OMG. Эти стандарты накладывают требования, которым должны удовлетворять модели, и предоставляют рекомендации для разработчиков сред создания ПО по MDA [9].

PIM демонстрирует определенную степень независимости от платформы, чтобы быть пригодным для использования с несколькими платформами подобного типа. Очень распространенным методом достижения независимости является нацеливание на модель системы для нейтральной, с технологической точки зрения, виртуальной машины. Виртуальная машина определяется как набор частей и услуг (связь, планирование, именование и т. д.).

Платформенно-зависимые модели - это вид системы с точки зрения конкретной платформы. PSM объединяет спецификации в PIM с деталями, которые определяют, как эта система использует определенный тип платформы.

MDA обеспечивает подход и предоставляет инструменты для обеспечения:

- указания системы независимо от платформы, которая ее поддерживает,
- определения платформ,
- выбор конкретной платформы для системы, и преобразование спецификации системы в одну, для конкретной платформы.

Три основные цели MDA - мобильность, функциональная совместимость и возможность повторного использования путем архитектурного разделения проблем.

Преобразование модели - это процесс преобразования одной модели в другую модель той же системы на рисунке 1.2.

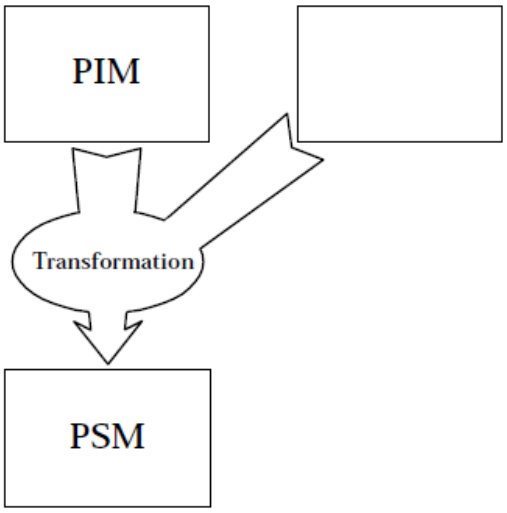

Рисунок 1.2 – Схема преобразования модели

Платформенно-независимая модель и другая информация объединяются путем преобразования, для создания модели, зависящей от платформы.

2. Язык моделирования – это нотация которая используется методом для описания проектов. Язык UML служит для определения, представления, проектирования и документирования программных систем, представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем.

Цель UML – обеспечить системных архитекторов, программистов и разработчиков программного обеспечения инструментами для анализа, проектирования и реализации программных систем, а также для моделирования бизнес-процессов и подобных процессов [8].

UML содержит стандартный набор диаграмм и нотаций самых разнообразных видов.

UML описывает девять различных диаграмм, которые перечислены ниже:

• диаграмма классов - отображает статическую структуру системы. Показывает классы с их атрибутами и методами. Также подчеркивается связь между различными классами;

• диаграмма объекта - тесно связана с диаграммой классов. Используется для проверки диаграммы классов;

• диаграмма использования - показывает функциональные возможности системы. Актеры представляют объекты, и их функциональные возможности выделены [17];

• диаграмма последовательностей: показывает взаимодействие между классами через передачу сообщений;

• диаграмма сотрудничества: представляет собой статическую и динамическую структуру системы;

• диаграмма состояний: показывает влияние внешних систем на класс;

• диаграмма активности: отображает поток контроля между действиями. она динамична по своей природе;

• диаграмма компонентов: показывает интеграцию небольших компонентов в строении всей системы;

• диаграмма развертывания: на этой диаграмме изображены физические и материальные ресурсы системы [20];

Вышеупомянутые модели можно разделить на три подкатегории, а именно:

• структурная модель;

• поведенческая модель;

• модель архитектуры;

Структурные модели представляют собой статические компоненты в системе. Они включают следующее:

• диаграмма классов;

• объектная диаграмма;

• диаграмма развертывания;

• компонентная диаграмма;

Как упоминалось ранее, это представления о статичности системы. Структурные диаграммы не связаны с динамическим поведением системы. Классовая диаграмма является наиболее популярной среди них [18].

Поведенческая модель представляет собой динамическое поведение системы. В нем изображены взаимодействия между различными элементами системы. Она состоит из следующего:

• диаграмма активности;

• диаграмма использования;

Архитектурная модель представляет собой весь пакет системы. Таким образом, он состоит из двух вышеупомянутых моделей. Он имеет только одну пакетную диаграмму.

Вместе эти UML-диаграммы дают полное графическое представление системы для разработки. Это упрощает построение программного обеспечения для разработчиков.

Первая новаторская работа в области текстового анализа была сделана Р. Джейбботом, который предложил подход для преодоления разрыва между спецификацией естественного языка и объектно-ориентированным моделированием. Его предложение состояло в том, что классы могут быть получены из существительных в спецификации, а методы из глаголов.

Унифицированный язык моделирования (UML) [6] поддерживает набор диаграмм для визуального представления различных артефактов программного обеспечения. Одной из наиболее распространенных и широко используемых диаграмм является диаграмма классов UML. Диаграммы классов обычно используются для представления структурной информации о программной модели. Ключевым элементом в модели класса является класс. Класс может иметь подэлементы, такие как атрибуты, методы и ассоциации [19].

Чтобы сопоставить SBVR с моделью класса UML, нам нужно извлечь словарь SBVR из заданных правил SBVR, а затем сопоставить словарь SBVR с базовыми элементами модели класса UML (например, классы, ассоциации и т. д.) и, наконец, создать графическое представление модели класса, рисунок 1.3.

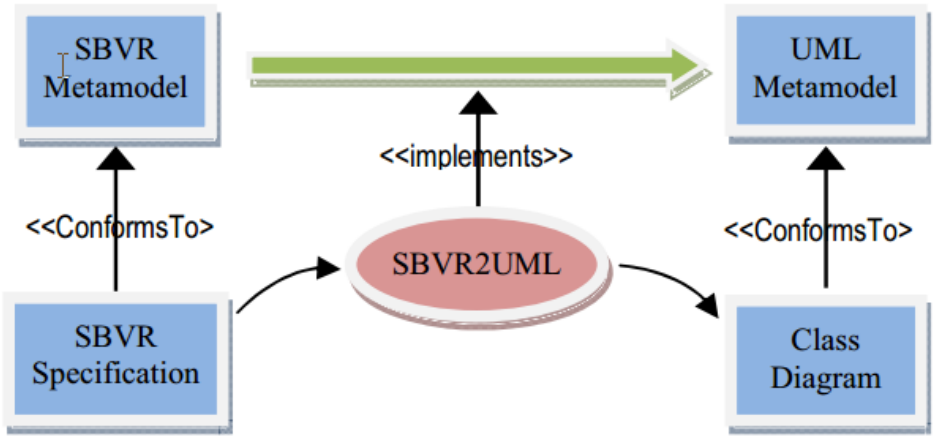

Рисунок 1.3 – От SBVR к UML трансформация фреймворков

3. XML или Язык Расширяемой Маркировки (eXtensible Markup Language) – это язык разметки, описывающий целый класс объектов данных, называемых XML- документами. Фактически XML - это способ разметки документов, предназначенный для формирования в документах какой-либо структуры и определения отношений между различными элементами этой структуры.

XML-документ представляет собой обычный текстовый файл, в котором при помощи специальных маркеров создаются элементы данных. XML предоставляет возможность определять теги и структурные отношения между ними. Так как нет предопределенного набора тегов, не может быть предустановленной семантики. Вся семантика документа XML будет либо определяться приложениями, которые их обрабатывают, либо таблицами стилей [14].

XML-документы состоят из разметки и содержимого. Существует шесть видов разметки, которые могут встречаться в XML-документе:

- элементы;
- ссылки на сущности;
- комментарии;
- инструкции по обработке;
- маркированные разделы;
- объявления типов документов.

1.2. Описание стандартов SBVR

Процесс разработки программного обеспечения неизменно начинается с потребностей и желаний человека в изучении или решении любой бизнес-задачи. На таких ранних этапах разработки программного обеспечения естественный язык используется для описания точной бизнес-проблемы, которую необходимо решить. Но естественный язык часто сложный, неопределенный и двусмысленный, предложения неопределенны, когда они содержат обобщение. Иногда они пропускают важную информацию, такую как предмет или объект, необходимые глаголу для полноты или содержащие местоимения. Все эти трудности возникают, когда кто-либо обсуждает деловую проблему при использовании естественного языка. С другой стороны, программное обеспечение требует большей точности, правильности и чистоты [13].

Анализ - это процесс трансформации задачи, определяющей из нечеткого множества фактов и мифов, в согласованное изложение требований системы. Основная цель объектно-ориентированного анализа (OOA) заключается в том, чтобы охватить целую, определенную и последовательную картину требования к системе и то, что система должна делать для удовлетворения потребностей и пользователя. Естественный язык, такой как английский, переводится в формальные спецификации, такие как UML-модели. Этот перевод состоит из создания структурной модели системы, такой как идентификация участников и связанных с ними вариантов использования, определение классов, их атрибутов, методов и отношений между ними.

Специфика семантического делового словаря и правила (SBVR) - это стандарт, разработанный группой управления объектов, выражающий бизнес знания, которые очевидны и понятны для человека и компьютерных систем. Эта спецификация определяет словарь и правила для документирования семантики деловых словарей, деловой информации и бизнес-правил. Бизнес-правила выражают утверждения по элементам модели, называемым бизнес-лексикой. Таким образом, важно иметь хорошо сформулированную деловую

лексику с точно определенной семантикой [12]. Такие знания захватили бизнес экспертов, аналитиков информационных систем, нуждающихся в инструментах, что позволили бы хранить SBVR спецификации в MOF хранилищах для обменивания и связывания их с другими моделями, основанными на MOF. SBVR полностью интегрирована в OMG (MDA) рисунок 1.4.

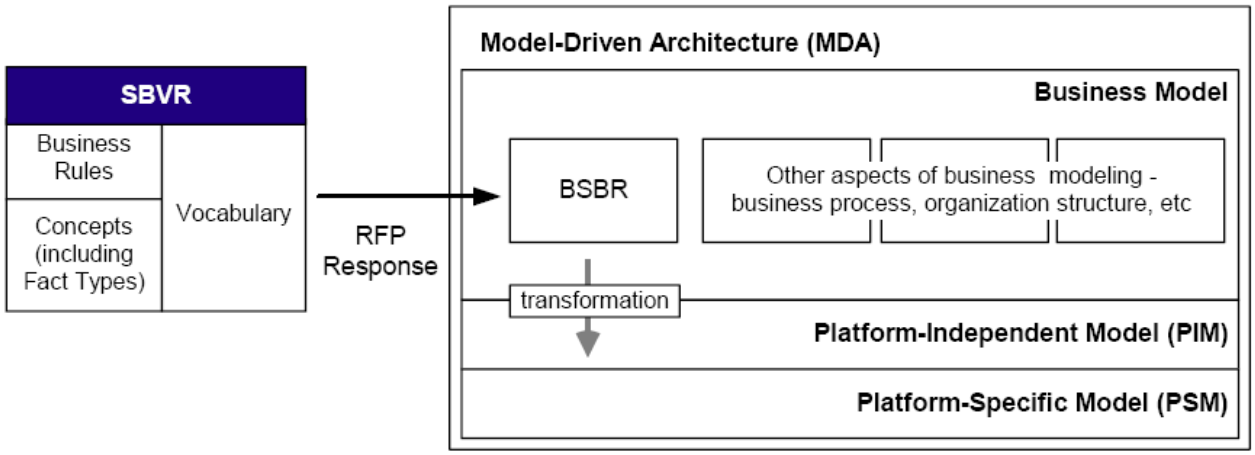

Рисунок 1.4 – Положение SBVR в MDA

Эта спецификация применима к области деловых словарей и бизнес-правил всех видов предпринимательской деятельности и всех видов организаций.

SBVR определяет словарный запас и правила, которые позволяют выражать бизнес-лексику, бизнес-факты и бизнес-правила. Этот стандарт также предоставляет схему XMI для обмена созданными артефактами между различными программными инструментами. Детальная семантика деловой лексики определяется с помощью SBVR в структурированном английском языке, в то время как некоторые аспекты деловой лексики также выражаются в форме диаграмм классов UML [15].

В SBVR все конкретные выражения и определение фактов и понятий, используемых организацией в процессе ведения бизнеса, рассматриваются как словарный запас. Также в SBVR формальная презентация под влиянием

бизнеса рассматривается как правила, которые используются для выражения операции конкретного субъекта бизнеса при определенных условиях [16].

На рисунке 1.5 изображена схема значений и представлений словаря SBVR:

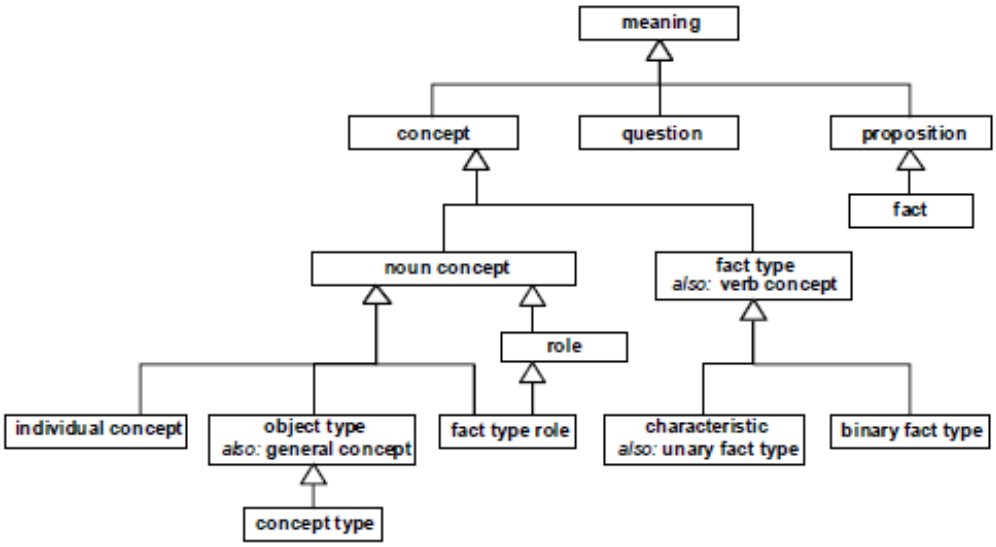

Рисунок 1.5 – Словарь SBVR

Таблица 1.1 – Значение и представление словаря

Meaning	Понимается под словом, знаком, выводом или описанием; Что кто-то намеревается выразить или что кто-то понимает.
Concept	Единица знаний, созданная уникальной комбинацией характеристик.
noun concept	Понятие, которое является смыслом или фразой существительного.
object type	Существительное, которое классифицирует вещи на основе их общих свойств
concept type	Тип объекта, который *специализируется* на понятии «концепция».
Role	Существительное, которое соответствует вещам, основанным на их игре, предполагая функцию или используемое в некоторой ситуации.
fact type role	Роль, которая специфически характеризует свои экземпляры посредством их участия в действительности, являющейся экземпляром данного типа фактов.
fact type	Концепция, которая является смыслом глагольной фразы, которая включает в себя одно или несколько существительных понятий и чьи экземпляры являются действительными.
Characteristic	Тип факта, который *имеет* ровно одну роль.
binary fact type	Тип факта, который имеет ровно 2 роли
individual concept	Концепция, которая соответствует только одному объекту [вещь]

Для определения бизнес-терминов, типов фактов и бизнес-правил на структурированном английском языке SBVR, используются четыре типа стилей шрифтов:

Таблица 1.2 – Стили шрифтов в SBVR

Шрифт	Описание
term	Шрифт «термин» используется для обозначения типов объектов (общие концепции) и роли. Условия определены в единственном числе с использованием строчных букв. Примеры: автомобиль, водитель, кредит и т.д.
Name	Шрифт «Имя» используется для обозначения того, что отдельные понятия как правило, собственные имена. Первая буква названия заглавная. Одним из исключений последнего является представление числовых значений, которые также представлены в этом стиле. Примеры: Россия, Каунас, США, 500 и т. д
Verb	Шрифт «глагол» используется для представления глагола, предлога или их сочетание. Глаголы используются в единственной действительной или пассивных формах - эти два используются как синонимы. Например, для активной формы ассоциативного типа факта «Выражение *представляет* значение», есть синонимичная пассивная форма, «значение, *представленное* выражением» машина управляется водителем ". Типы фактов, представляющие характеристики, всегда используются в пассивной форме, например. «Значение *представлено*».
keyword	Шрифт «ключевое слово» представляет собой лингвистические символы, которые используются для построения инструкций и определений. Используются для выражения логических формулировок. Примеры: каждый, это обязательно, чтобы, больше чем, и т. д.

Деловой лексикон включает в себя все специализированные термины, имена и формы типов фактов, которые данный бизнес-домен или сообщество использует в своих разговорах и письмах в процессе ведения бизнеса. SBVR включает в себя два специализированных словаря: Словарь для описания бизнес-слов, этот элемент спецификации касается всех типов терминов и значений. Словарь для описания бизнес-правил, этот элемент спецификации касается значения бизнес-правил и основывается на предыдущем.

Словарь бизнес-терминов имеет записи в виде глоссария, которые определяют понятия, имеющие представления в лексике. Каждая запись предназначена для одной концепции.

Таблица 1.3 – Словарь бизнес-лексики

Область	**Описание**
Первичное представление:	Первичным представлением записи может быть термин, имя или форма типа факта.
Определение:	Выражение, которое определяет первичное представление. Определение может быть формальным, частично формальным или неформальным.
Источник:	Используется для обозначения исходного словаря или документа для понятия.
Основа словаря:	Определение из общего словаря, который поддерживает использование первичного представления.
Общая концепция:	Поле «Общая концепция:» может использоваться для представления концепции, которая обобщает концепцию входа.
Тип концепции:	Используется для указания типа концепции ввода, если тип понятия не является очевидным из первичного.
Необходимость: Возможность:	Обычно используются для дополнения определения. «Необходимость» утверждает то, что обязательно верно - необходимые условия существования концепция. «Возможность» заявляет то, что возможно и не предотвращено по определению..
Справочная схема:	Показывает структурирование вещей, обозначенных понятием. Она определяет достаточные условия существования понятия.
Заметка:	Некоторые пояснения, которые не помещаются в других подписях, могут быть представлены в поле «Примечание:».
Пример:	Некоторые примеры с концепцией входа могут быть представлены в поле «Пример:».
Синоним:	Синоним - это другое обозначение той же концепции, которая может быть заменена первичным представлением.
Синонимичная форма:	Синонимичная форма - это форма типа факта для одного и того же типа факта. Например, «Клиент *оплачивает* кредит» и «кредит *оплаченный* клиентом» являются двумя синонимичными формами для одного и того же типа факта.
Представление:	Поле «See:» используется, когда первичное представление не является предпочтительным представлением концепции ввода. Поле «See:» вводит предпочтительное представление.

Основных типы понятий, которые могут быть представлены в бизнес-словаре:

- тип объекта (или общая концепция);
- индивидуальная концепция;
- тип факта (или понятие глагола);
- роль.

Тип объекта - понятие существительного, которое классифицирует вещи на основе их общего свойства. Тип объекта представлен в шрифте «Term» и начинается с маленькой буквы.

Индивидуальное понятие - понятие, которое соответствует только одному объекту (вещи).

Тип факта - понятие, которое обозначает некоторый тип отношения между двумя или более понятиями существительного или характеристикой понятия существительного.

В соответствии с определением типы фактов определяются с использованием существующих понятий существительного (типов объектов или отдельных лиц), которые уже были определены в бизнес-лексике. Глаголы представляют типы фактов, создающие отношения между этими существительными или определяющие их характеристики. Глаголы представлены в шрифте «*verb*».

Ассоциативный тип факта - это наиболее распространенный тип факта, который имеет две или более ролей. Как правило, существует два понятия существительных, играющих определенные роли в отношении, определяемые ассоциативным типом фактов. Только в случае рекурсивных отношений существует одна концепция существительного, играющая обе роли.

Ассоциативный тип факта идентифицируется глаголом или фразой-глаголом, который не зарезервирован для других типов фактов.

Частичный тип факта - это двоичный тип факта, в котором утверждается, что одно понятие существительного (все его экземпляры) находятся в составе данного целого, то есть другое понятие существительного.

Тип факта категоризации - это тип факта, который представляет отношение между более общей концепцией существительного и другим (более конкретным) понятием существительного, которое является категорией первой концепции.

Роль - это понятие существительного, которое соответствует вещам, основанным на их игре.

Роль идентифицируется в записи словаря, которая имеет поле «Concept_type:», установленное в «role», и поле «General_concept:», заданное для типа объекта, которое играет эту роль в соответствующем типе фактов.

SBVR является неотъемлемой частью архитектуры управляемой моделью OMG (MDA). SBVR определяет словарь и правила для документирования семантики деловых словарей, бизнес-фактов и бизнес-правил. SBVR включает словарь для концептуального моделирования и фиксирует выражения, основанные на этом словаре, как формальные логические структуры. Словарь SBVR позволяет формально определять представления понятий, определений, экземпляров и правил любой информационной области на естественном языке. Эти функции делают SBVR хорошо подходящим для описания бизнес требований для бизнес-процессов и информационных систем для использования бизнес-моделей.

Элементы SBVR Деловая лексика и бизнес-правила являются основным ключевым компонентом спецификации SBVR OMG.

Преобразование типа объекта. Тип объекта преобразуется в класс UML с тем же именем, что и тип объекта на рисунке 1.6:

Рисунок 1.6 – Преобразование типа объекта

Ассоциативный тип факта преобразуется в ассоциативную связь между двумя классами. Ассоциативный тип факта включает две роли типов объектов, имена которых становятся именами концов свойств, соответствующих концам этой ассоциации. Глагол (или глагольная фраза), используемый типом факта, становится именем этой ассоциации на рисунке 1.7.

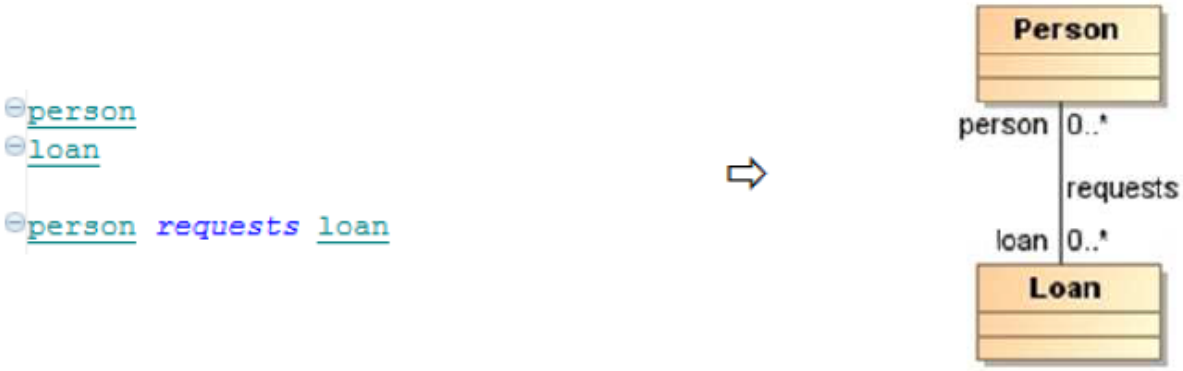

Рисунок 1.7 – Трансформация ассоциативного факта

Типы объектов могут играть определенные роли в ассоциативных типах фактов. В таком случае имена этих ролей сопоставляются с именами свойств, соответствующих концам ассоциации.

Оперативные бизнес-правила (обязательства и разрешения), сформулированные как закрытые деонтичные формулировки, преобразуются в операции. Глаголы, обозначающие типы фактов, на которых основаны эти формулировки, преобразуются в имена операций, роли - в имена параметров, типы объектов, играющие эти роли, - в типы параметров.

1.3 Формализация требований к приложению на основе текстового анализа для генерации UML моделей со стандартами SBVR

Приложение для генерации UML моделей на основе текстового анализа, должно являться парсером, которое будет анализировать текстовый файл в формате xml, и находить объекты, оформленные в разной стилевой категории, рисунок 1.8.

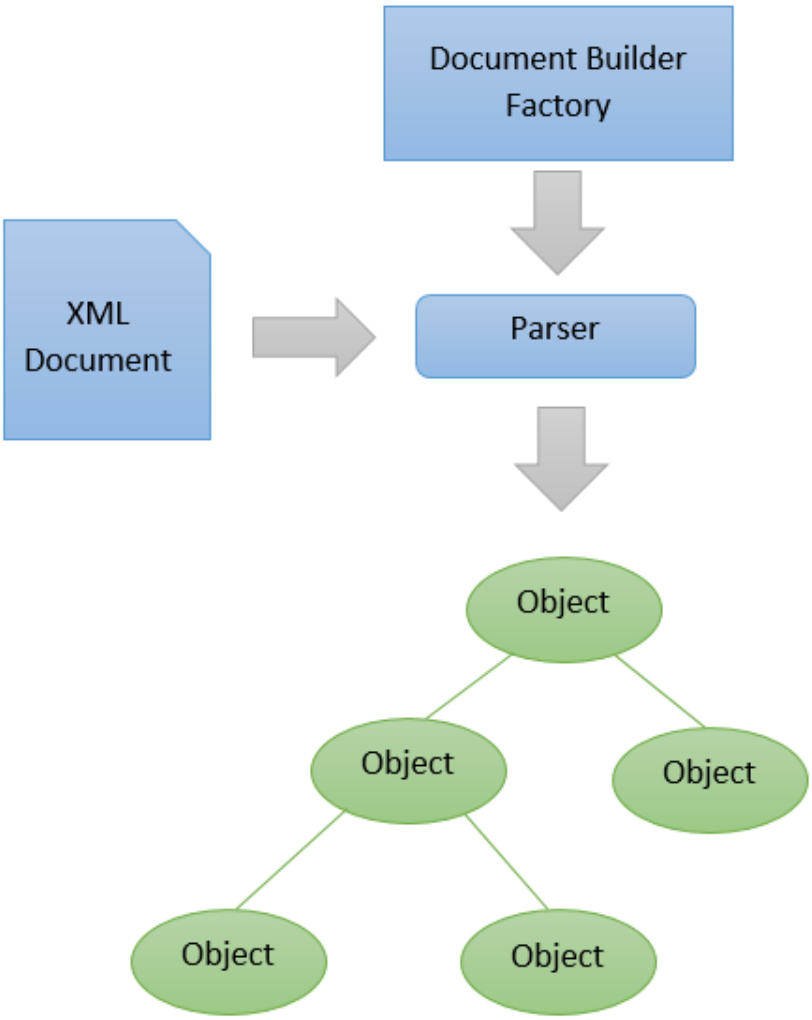

Рисунок 1.8 – Разбор документа

После анализа документа, приложение должно вывести в окно вывода информации эти объекты, и обозначить к какой категории стилей, в соответствие со стандартами SBVR они относятся.

Приложение для поиска стилей, должно иметь пользовательский интерфейс: окно для вывода информации listbox, кнопки: “Файл” - для выбора

файла, “Поиск” – для поиска объектов со стилями и “Очистить” - для удаления имеющейся информации.

После сохранения в формате XMI и загрузки его в любое CASE средство, на выходе должна построиться диаграмма классов UML, где элементы метамодели SBVR, должны соответствовать элементам метамодели UML, рисунок 1.9.

SBVR metamodel element	UML metamodel element
Object Type	Class
Individual Concept	Object
Characteristic	Class Attribute
Verb Concept	Class Method
Fact Type	Association
Partitive Fact ct Type	Generalization
Categorization Fact Type	Aggregation
Quantifications	Cardinalities

,

Рисунок 1.9 – SBVR и UML метамодели

Выводы по первой главе

Сегодня OMG - это основополагающая организация в пространстве моделирования. SBVR и BPMN повторяют успех стандартизации UML, привнося преимущества стандартизированного синтаксиса моделирования в управление бизнесом, позволяя всем аспектам бизнеса извлекать выгоду из точности и простоты понимания, предлагаемых хорошими нотациями моделирования. Опыт моделирования OMG, основанный на универсальной, стандартизированной структуре MOF, позволяет организации предоставлять моделирующие стандарты, которые отвечают за потребности всех аспектов бизнеса и правительства.

В первой главе был проведен анализ OMG технологий: UML, MDA, XML, и спецификации SBVR. Были описаны основные аспекты бизнес словаря и бизнес правил. Выявлены требования на разработку программного продукта.

ГЛАВА 2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Основная цель объектно-ориентированного анализа (ООА) заключается в том, чтобы зафиксировать полную, однозначную и последовательную картину требований к системе и какую систему необходимо выполнить для удовлетворения потребностей пользователя. Это достигается путем создания нескольких моделей системы, таких как модель использования, модель класса.

Таким образом, анализ требований и создание артефактов программного обеспечения для построения модели анализа - это огромная и сложная задача, которая требует автоматической поддержки. Чтобы преодолеть некоторые из проблем при автоматизации этапа анализа, предлагаются методы, которые направлены на автоматическую генерацию программных артефактов на этапе анализа. Первоначально этот метод преобразует требования “natural language” (NL) в язык семантического словаря и правил (SBVR) для повышения точности генерируемых артефактов и моделей. Затем он фокусируется на идентификации артефактов программного обеспечения, таких как актеры, примеры использования, классы, атрибуты, методы, отношения, множественность и многое другое для генерации фазовых моделей анализа, таких как диаграмма использования и диаграмма классов. Наконец, этот метод генерирует файлы XML Metadata Interchange (XMI) для визуализации сгенерированных моделей в инструменте моделирования UML с функцией импорта XMI.

2.1 Разработка алгоритма разбора документа

В этом разделе описывается используемая методология для определения артефактов, которые используются для генерации моделей на этапе анализа естественного языка. Эта методология состоит из автоматического преобразования спецификации программного обеспечения на естественном языке в контролируемый промежуточный формат SBVR, также для идентификации программных артефактов и генерации модели, наконец, визуализации генерируемых моделей.

1. Построение правила. Чтобы сформировать правило SBVR, мы должны сначала создавать типы фактов в виде предложений, которые представляют собой некоторые отношения между концепциями, обозначенными на предыдущем этапе. Для этого используется шаблон, такой как существительное-глагол-существительное, чтобы установить связь между двумя понятиями. Таким образом, тип факта создается путем объединения понятий существительного и глагольных понятий из списка первичных фазовых массивов. Сгенерированный тип факта используется для создания правила SBVR, применяя различные логические формулировки, такие как использование логического выражения AND, OR и NOT и т. д.,

2. Применение обозначений. На этом этапе используются обозначения SBVR для генерации правил, таких как концепции существительных, подчеркнутые понятия глагола, выделенные ключевые слова, отдельные концепции или атрибуты.

3. Извлечение артефактов. На этом этапе созданная лексика и правила SBVR дополнительно обрабатываются для извлечения основных строительных блоков или артефактов моделей, таких как пример использования и диаграмма классов и т. д. Все концепции существительных SBVR и тип объекта имеют тенденцию быть актерами для модели использования и классами для модели класса. Ассоциация между актером и вариантами использования идентифицируется с помощью генерируемого дерева разбора, а также из унарных типов фактов SBVR в виде шаблонных именных глаголов или двоичных фактов. Все характеристики SBVR, связанные с понятиями существительных и Тип объекта сопоставляются с элементами данных в модели класса. SBVR Количественные значения, идентифицированные с соответствующими понятиями существительных, отображаются на множественность между двумя классами.

4. XMI Генерация и визуализация модели. Наконец, на этом этапе вывод вышеперечисленных фаз генерируется в виде формата XML-метаданных

(XMI). Такой файл далее вводится в качестве инструмента для моделирования UML с функцией импорта XMI для визуализации созданных моделей.

На рисунке 2.1, изображен алгоритм для создания приложения для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR [6].

Рисунок 2.1–Алгоритм разбора документа

2.2 Обзор и выбор средств программирования

Прежде чем приступить к реализации приложения, следует рассмотреть некоторые языки программирования.

В наше время существует большая потребность в реализации какой-либо системы, программы, посредством программирования, в связи с чем создано множество языков, которые можно для этого использовать. Самыми широко используемыми среди языков программирования являются языки C# и C++.

C++ – это компилируемый язык со статической типизацией на котором можно создавать программы любого уровня сложности. C++ спроектирован и динамично развивается как язык, поддерживающий различные методы и технологии программирования, но реализующий их на платформе, обеспечивающей высокую техническую эффективность.

Программа на C ++ представляет собой набор команд, которые говорят компьютеру о необходимости «чего-то». Этот набор команд обычно называется исходным кодом C++.

Достоинства:

- предсказуемое выполнение программ является важным достоинством для построения систем реального времени.
- имеется возможность работы на низком уровне с памятью, адресами.
- поддерживаются различные стили и технологии программирования.
- возможность создания встроенных предметно-ориентированных языков программирования.

Однако, при всем этом, язык программирования C++ имеет ряд недостатков:

- плохая поддержка модульности;
- нехватка информации о типах данных в процессе компиляции;
- язык сложен для изучения и компиляции;
- синтаксис, провоцирующий ошибки.

Другим популярным языком программирования является C# – является объектно-ориентированным и в этом плане много перенял у Java и C++. Например, C# поддерживает полиморфизм, наследование, перегрузку

операторов, статическую типизацию. Также C# имеет указатели на функции-члены классов, атрибуты, события, свойства, исключения и комментарии в XML-формате. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей, как, например, лямбды, динамическое связывание, асинхронные методы и т.д.

Достоинства:

- инкапсуляция;
- имеется компонентно-ориентированный подход;
- язык c# ориентирован на безопасность кода;
- унифицированная система типизации;
- сравнительно простая возможность модификации программ;

Несмотря на перечисленные достоинства, язык C# имеет также некоторые недостатки, такие как:

- довольно сложный синтаксис;
- малое количество свежих концептуальных идей;
- относительно невысокая производительность;

В качестве среды разработки был выбран объектно-ориентированный язык программирования C#, одной из причин выбора языка C#, стало то, что он использует простые встроенные конструкции языка компоненты, которого, могут быть преобразованы в Web сервисы и использование таких технологий, как XML.

2.3 Реализация приложения для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR

Для реализации приложения для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR, в первую очередь были созданы объекты классифицирующие вещи на основе их общего свойства, и связка между ними, с помощью которой набор из объектов и связки

преобразуются в предложение. Типы фактов определяются с использованием существующих понятий существительного.

Типы объектов являются существительными, и обозначаются: “Объект”. Тип факта, является глаголом, который связывает объекты по смыслу, т.е. является отношением между объектами, и обозначается: «*глагол*». Для примера составим предложение, которое будет иметь два объекта и связку между ними, рисунок 2.2

Студент *имеет* зачетную книжку

Рисунок 2.2 –Трансформация NL в SBVR

“Студент” и “зачетную книжку” – являются объектами.

“имеет” – это отношение между объектами, оно связывает их, придавая смысл фразе.

После того, как перевели натуральный язык, в язык семантического бизнес словаря и бизнес правил, нужно сохранить документ в xml формате, рисунок 2.3 и 2.4.

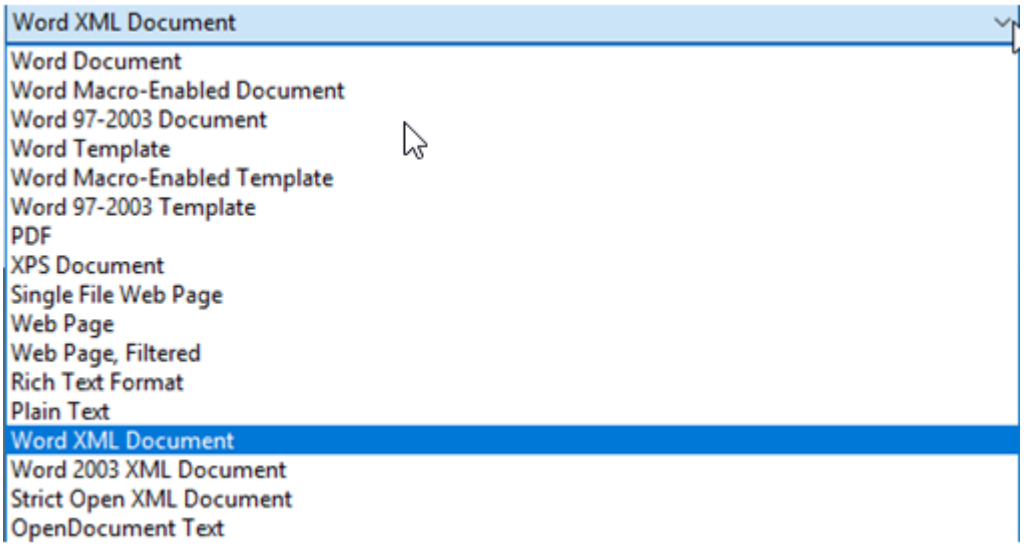

Рисунок 2.3 – Сохранение в xml формате

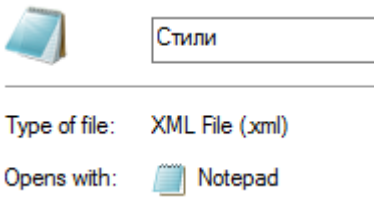

Рисунок 2.4 – Сохраненный в xml формате документ

С 2007 года в системах Microsoft Office используются форматы файлов на основе XML, такие как DOCX, XLSX и PPTX. DOCX представляет собой zip-архив, содержащий целый набор файлов: собственно текст, картинки и т. д.

Сохраненный в XML формате документ, имеет структуру схемы XML, рисунок 2.5.

Рисунок 2.5 – Схема XML файла

Третий шаг – это создание приложения, являющееся парсером, которое будет проводить поиск, анализировать текст, из загруженного в приложение xml файла и искать стили SBVR.

Сначала создадим интерфейс программы, рисунок 2.6, которая будет иметь 3 кнопки действия, для загрузки файла, поиска стилей из XML файла и очистки информации из окна вывода.

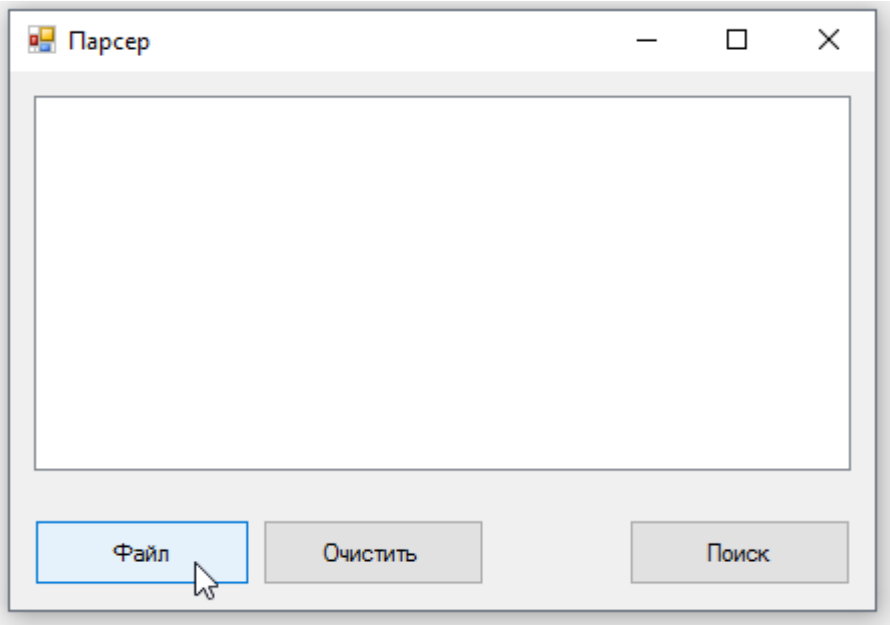

Рисунок 2.6 – Кнопка выбора файлов

При помощи кнопки “Файл”, можно загрузить документ в приложение, рисунок 2.7.

```
private void button1_Click(object sender, EventArgs e)
{
    {
        OpenFileDialog openFileDialog1 = new OpenFileDialog();
        openFileDialog1.Filter = "Текстовые формата (*docx.)|*.docx|Все файлы (*.*)|*.*";
        openFileDialog1.ShowDialog();
        try
        {
            listBox1.Items.AddRange(System.IO.File.ReadAllLines(openFileDialog1.FileName, Encoding.
        }
        catch { }
        GC.Collect();
    }
}
```

Рисунок 2.7 – Код кнопки “Файл”

Кнопка “Поиск” осуществляет поиск стилей из XML файла, рисунок 2.8, после чего заносит их в программу и выводит в окно вывода.

```
var paragraphs =
    from para in xDoc
                 .Root
                 .Element(w + "body")
                 .Descendants(w + "p")
    let styleNode = para
                    .Elements(w + "pPr")
                    .Elements(w + "pStyle")
                    .FirstOrDefault()
    select new
    {
        ParagraphNode = para,
        StyleName = styleNode != null ?
            (string)styleNode.Attribute(w + "val") :
            defaultStyle
    };
```

Рисунок 2.8 – Код кнопки “Поиск”

Кнопка “Очистить”, рисунок 2.9, удаляет содержимое из окна вывода, чтобы можно было загрузить другой файл для извлечения из него стилей.

```
ListBox lb = new ListBox();
lb.Items.Clear();
```

Рисунок 2.9 – Код кнопки “Очистить”

При нажатии на “Файл”, откроется диалог открытия файлов, из которого можно будет выбрать имеющийся XML файл, рисунок 2.10.

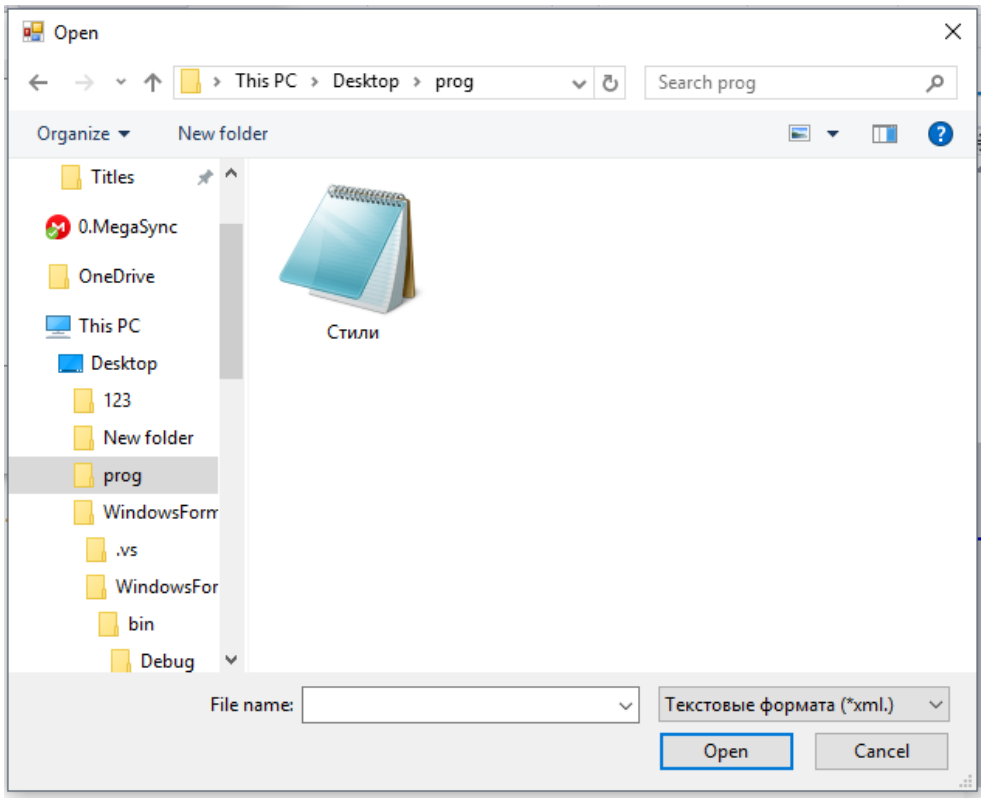

Рисунок 2.10 – Диалог открытия файлов

После успешной загрузки файла, на экран будет выведена надпись, подтверждающая это, рисунок 2.11.

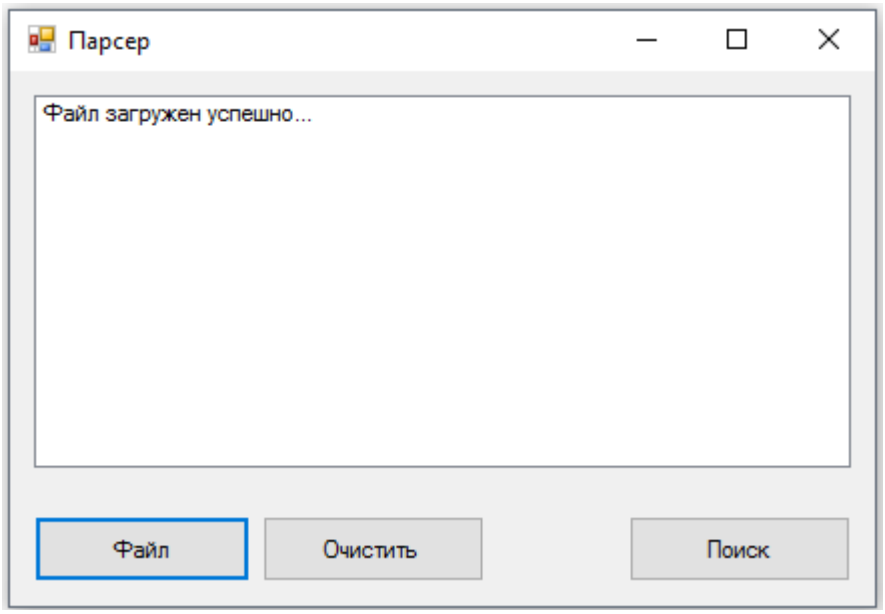

Рисунок 2.11 – Успешная загрузка файла

Если в загруженном XML файле не были найдены объекты со стилями, тогда парсер сообщит об ошибке, и попросит загрузить другой файл, рисунок 2.12.

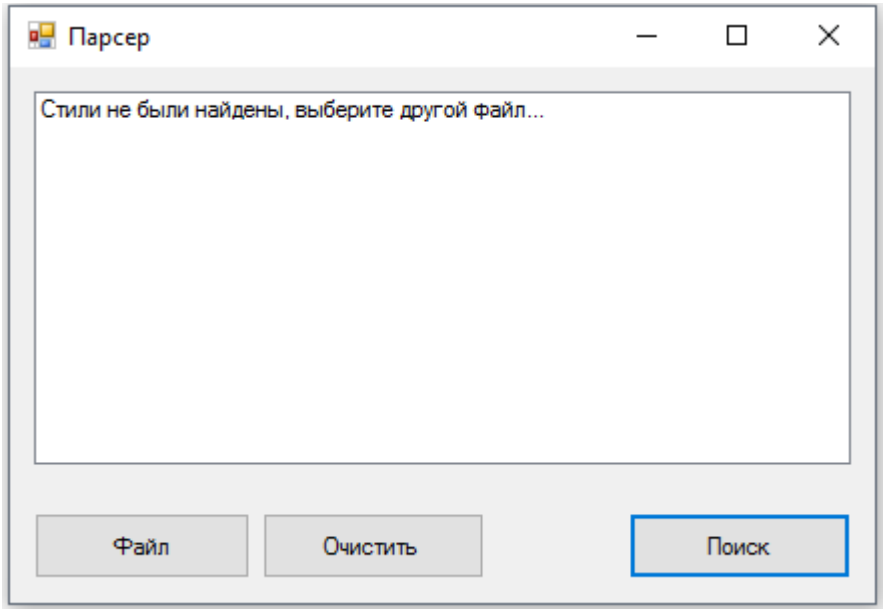

Рисунок 2.12 – Ошибка при поиске стилей

Благодаря кнопки поиск, происходит извлечение слов, выделенных стилем. Кнопка “очистить”, удаляет данные из окна вывода информации.

Источник этого запроса по получению всех абзацев документа и их стилей из xml файла в среде программирования C# таков, рисунок 2.13:

```
xDoc.Root.Element(w + "body").Descendants(w + "p")
```

Рисунок 2.13 – запроса по получению всех абзацев документа и их стилей

Запрос использует ось <xref:System.Xml.Linq.XContainer.Descendants%2A> вместо оси <xref:System.Xml.Linq.XContainer.Elements%2A> поскольку в документах, в которых имеются разделы, абзацы не будут прямыми потомками элемента текста, а будут находиться на два уровня ниже в иерархии.

Или используется предложение let, чтобы определить элемент, содержащий узел стиля. Если элемент не найден, то styleNode устанавливается в значение null, рисунок 2.14:

```
let styleNode = para.Elements(w + "pPr").Elements(w + "pStyle").FirstOrDefault()
```

Рисунок 2.14 – Определение элемента, содержащего узел стиля

Let сначала использует ось <xref:System.Xml.Linq.XContainer.Elements%2A>, чтобы найти все элементы с именем pPr, затем использует метод расширения <xref:System.Xml.Linq.Extensions.Elements%2A>, чтобы найти все дочерние элементы с именем pStyle, и после этого использует стандартный оператор запроса <xref:System.Linq.Enumerable.FirstOrDefault%2A> для преобразования коллекции в один элемент. Если коллекция пуста, styleNode устанавливается в значение null.

Затем все это нужно сохранить в XMI формате, для того, чтобы в итоге можно было получить диаграмму классов.

Выводы по второй главе

Во второй главе был разработан алгоритм разработки приложения, на основе текстового анализа. Также было реализовано создание приложения для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR.

ГЛАВА 3 ТЕСТИРОВАНИЕ И АНАЛИЗ ПОЛУЧЕННЫХ РЕШЕНИЙ

3.1 Соответствие разработанного программного обеспечения формализованным требованиям

Разработанное программное приложение на основе текстового анализа для генерации UML моделей в соответствии со стандартом SBVR, содержит алгоритм загрузки XML файла со стилями, в программное приложение, которое осуществляет поиск слов по стилям, рисунок 3.1.

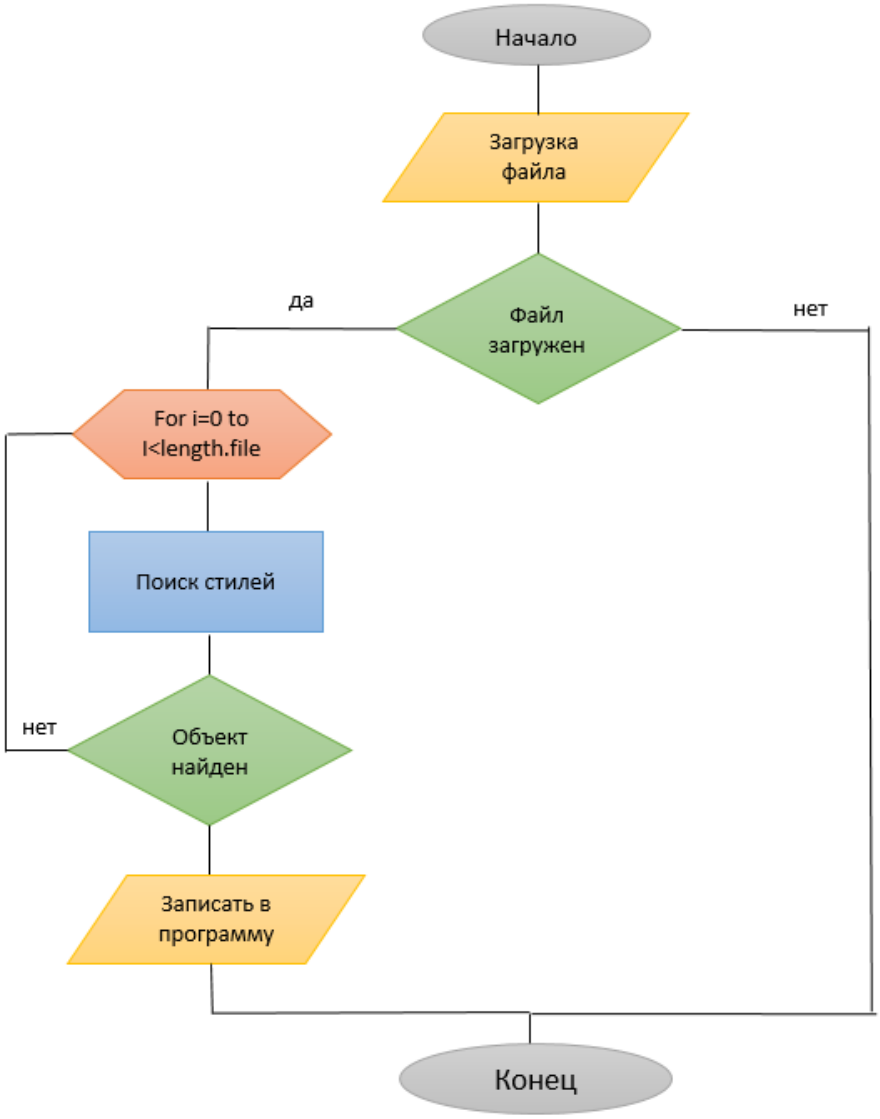

Рисунок 3.1 – Блок схема парсера

После нахождения слов, программа выводит их на экран, разделяя по стилям, в соответствие со стандартами SBVR, где стиль H1, служит для определения объектов, H2 является связкой между объектами, а также H3 в

который входят ключевые слова, для выражения логических формулировок. В итоге исходные файлы были сохранены в формате XMI формате. Файл XMI был загружен в case средство, на выходе была получена диаграмма UML.

Алгоритм создания UML диаграмм на основе текстового анализа, будет применяться к XML файлу, созданному с любым текстовым содержимым, использующим стили. Исходный текст и его стили, будут непосредственно влиять на итоговую UML диаграмму.

3.2 Тестирование приложения для генерации UML моделей

При создании приложения для генерации UML моделей, на основе текстовой информации, со стандартом SBVR были:

- оформлены стили слов, используя спецификации бизнес словаря и бизнес правил;
- файл со стилями был сохранен в xml формате;
- было создано приложение, которое производит поиск по стилям, в xml файле, и выделяет эти слова:

При запуске приложения запускается интерфейс программы, имеются 3 кнопки, каждые отвечающие за свои функции, также имеется окно для вывода информации на экран. Выбрав файл со стилями слов, сохраненный в XML формате, и после нажатия кнопки “поиск”, расположенной на интерфейсе программы, начинается поиск слов, оформленных заголовками в Microsoft Word.

Пример предложения, содержащего объекты и связку между ними, оформленными в соответствие со стандартом SBVR:

Студент *имеет* зачетную книжку.

Исходя их рисунка 3.2, можно сделать вывод, что парсер по поиску слов, работает верно.

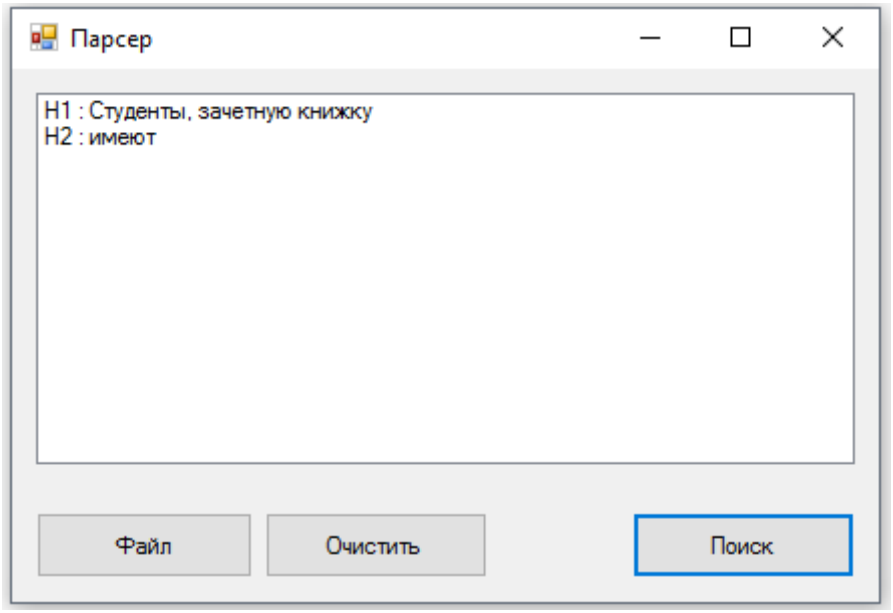

Рисунок 3.2 – Поиск стилей

На рисунке 3.2 видно разделение по стилям, оно было создано для того, чтобы показать, чем являются эти слова, в соответствие со спецификациями SBVR. “H1” содержит в себе объекты, т.е. это понятие существительного, которое классифицирует вещи на основе их общего свойства. “H2” содержит в себе связку, между объектами из стиля H1 и является глаголом, создающим отношение между этими существительными или определяющим их характеристики.

При нажатии на кнопку “очистить”, рисунок 3.3, удаляется содержимое из Listbox, и после этого можно выбрать другой файл, для поиска стилей.

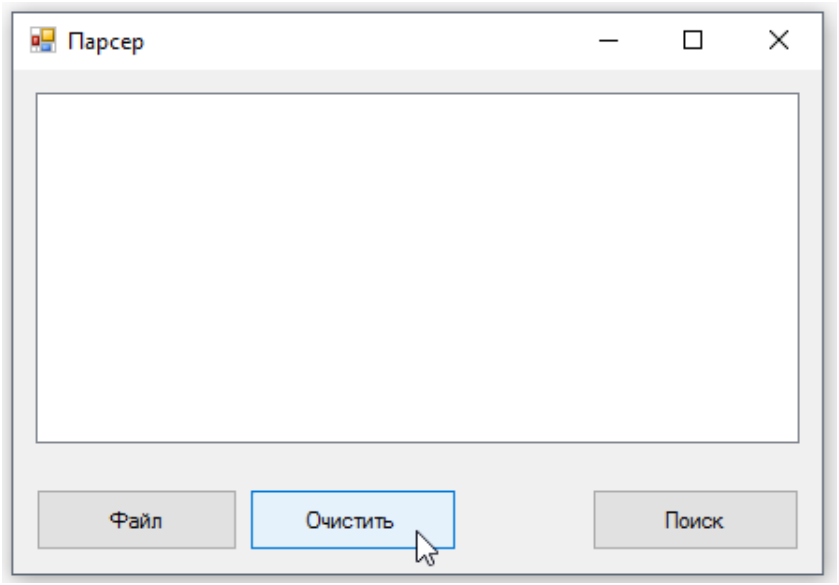

Рисунок 3.3– Поиск стилей из XML

Если использовать пример, в который добавлены ключевые слова, тогда добавляется новая строка стилей, рисунок 3.4.

Самые хорошие студенты *получают* стипендию

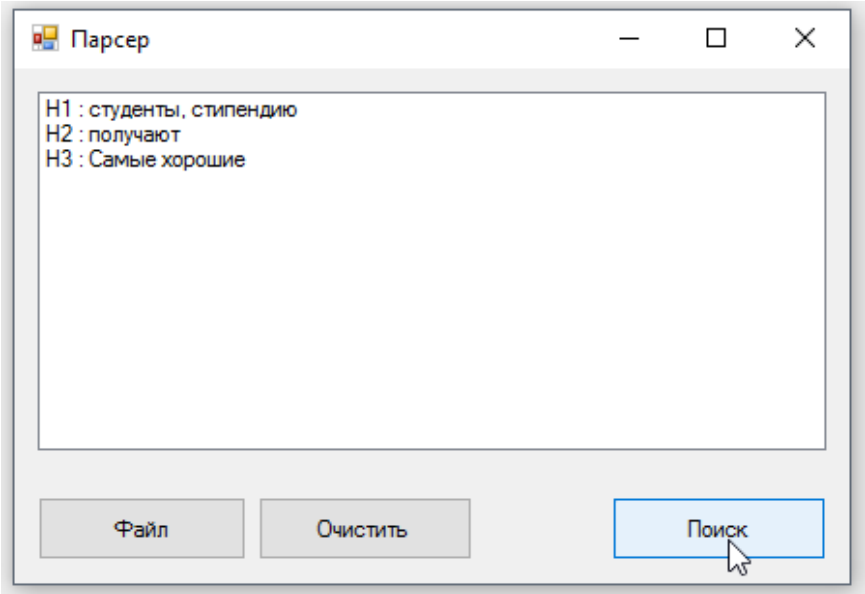

Рисунок 3.4– Поиск стилей из XML

В этом примере, добавляется стиль H3, он характеризует ключевые слова, в стандарте SBVR. Ключевые слова используются для выражения логических формулировок.

Это приложение было сохранено, в специальном формате XMI, чтобы в дальнейшем, его можно было использовать для построения диаграмм UML.

Когда модель XMI генерируется как XML-схема, следуя правилам построения схемы XMI, результатом является набор элементов XML и атрибутов.

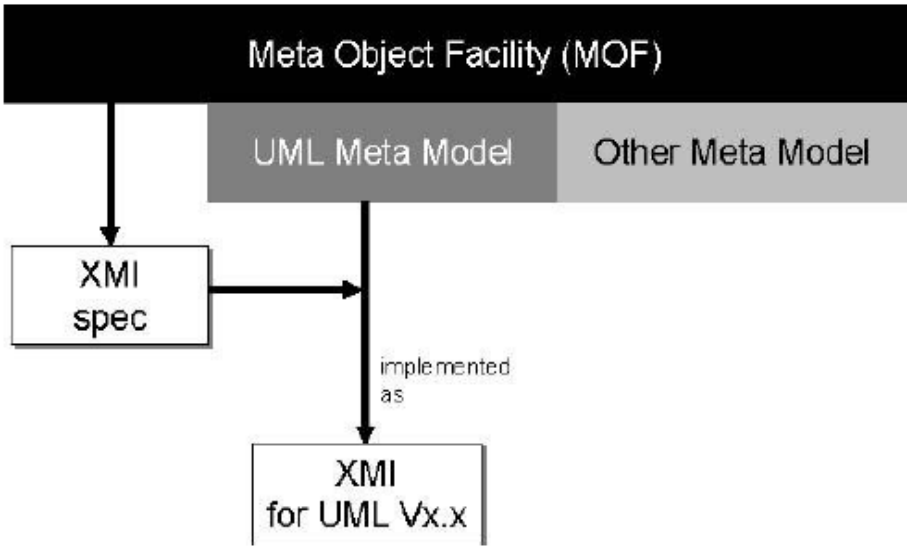

Рисунок 3.5– XMI - XML Metadata Interchange

Основная цель XMI - обеспечить легкий обмен метаданными между инструментами моделирования (основанными на OMG UML) и репозиториями метаданных (OMG MOF, Meta Object Facility) в распределенных гетерогенных средах. UNISYS, IBM, Oracle, Rational Software и Fujitsu в основном способствовали разработке спецификации XMI, и XMI уже реализовал своих продуктах обмен UML-моделями.

Для построения диаграмм из кода, было использовано case средство Magicdraw. В конечном итоге имеется диаграмма, которая показывает связь между объектами, которые имеют разные стилевые параметры, в соответствии, со стандартами SBVR.

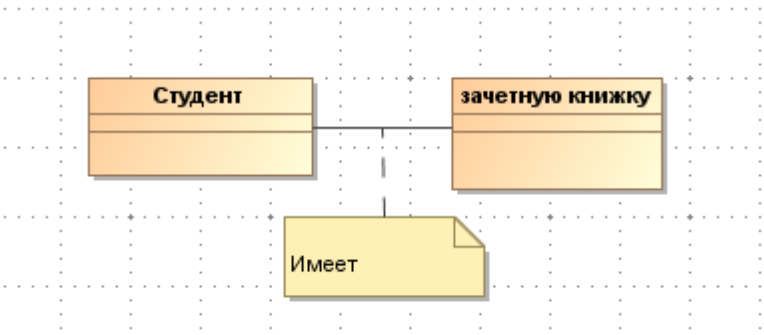

Рисунок 3.6– Генерация UML модели

Выводы по третьей главе

Во третьей главе было протестировано и проанализировано приложение для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR. В конечном итоге, из кода была создана диаграмма классов, которая показывает связь между объектами.

ЗАКЛЮЧЕНИЕ

Этот подход описывает компьютеризированный способ вывода программного элемента на этапе анализа. Он использует методы обработки естественного языка для рассмотрения требований к программному обеспечению бизнес-уровня и создает встроенную модель уровня анализа. Этот подход может использоваться для идентификации программных элементов, таких как список событий, примеры использования, классы, их атрибуты и статические отношения между ними с повышением точности из-за использования промежуточного формата SBVR. Достигнутый результат показал полезность этого подхода

Был проведен анализ OMG технологий: UML, MDA, XML, и спецификации SBVR. Были описаны основные аспекты бизнес словаря и бизнес правил. Выявлены требования на разработку программного продукта.

Был разработан алгоритм разработки приложения, на основе текстового анализа, реализовано создание приложения для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR.

Было протестировано и проанализировано приложение для генерации UML моделей на основе анализа текстовой информации в соответствии со стандартом SBVR. В конечном итоге, из кода была создана диаграмма классов, которая показывает связь между объектами.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

Нормативно-правовые акты

1. ГОСТ 2.105 – 95. Общие требования к текстовым документам [Текст]. – М.: Изд-во стандартов, 1996. – 29 с. – (Единая система конструкторской документации).

2. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание документа.

3. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. Структура и правила оформления.

4. ГОСТ 7.82-2001. Библиографическая запись. Библиографическое описание электронных ресурсов.

5. ГОСТ 19.701 – 90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения (ИСО 5807–85) [Текст]. Введен 1992–01–01. – М.: Изд-во стандартов, 1992. – 14 с. – (Единая система программной документации).

Электронные ресурсы

6. Генерация артефактов программного обеспечения и моделей на этапе анализа [Электронный ресурс]. – Электрон. дан. – [2012]. – Режим доступа: http://www.ijera.com/papers/Vol2_issue5/JE2516241630.pdf.

7. OMG технологии [Электронный ресурс]. – Электрон. дан. – [2016]. – Режим доступа: http://www.omg.org/.

8. UML [Электронный ресурс]. – Электрон. дан. – [2017]. – Режим доступа: http://www.uml.org/.

9. MDA [Электронный ресурс]. – Электрон. дан. – [2013]. – Режим доступа: http://www.omg.org/mda/.

10. Текстовый анализ [Электронный ресурс]. – Электрон. дан. – [2016]. – Режим доступа: https://fedcsis.org/proceedings/2008/pliks/96.pdf.

11. Технология для визуального программирования [Электронный ресурс]. – Электрон. дан. – [2016]. – Режим доступа: http://www.enterprise-design.eu/downloads/free/TextualAnalysis_English_v1.0.pdf.

12. Заголовки [Электронный ресурс]. – Электрон. дан. – [2016]. – Режим доступа: http://wpnew.ru/raskrutka-bloga/seo_optimizaciya/zagolovki-wordpress-teg-h1-h2-h3.html.

13. SBVR [Электронный ресурс]. – Электрон. дан. – [2016]. – Режим доступа: http://www.omg.org/spec/SBVR.

14. XML [Электронный ресурс]. – Электрон. дан. – [2017]. – Режим доступа: http://www.omg.org/technology/xml.

15. Textual Analysis [Электронный ресурс]. – Электрон. дан. – [2013]. – Режим доступа: http://www.researchgate.net/.

Литература на иностранном языке

16. Mohd Ibrahim, Rodina Ahmad “Class diagram extraction from textual requirements using Natural language processing (NLP) techniques”,IEEE journal Inc., 2010.

17. Deva Kumar Deeptimahanti, Muhammad Ali Babar “An Automated Tool for Generating UML Models from Natural Language Requirements” IEEE journal Inc., 2010.

18. Deeptimahanti Deva Kumar, Ratna Sanyal “Static UML Model Generator from Analysis of Requirements (SUGAR)” IEEE journal Inc., 2011.

19. Farid Meziane, Nikos Athanasakis, Sophia Ananiadou, 2009, Generating Natural Language specifications from UML class diagrams, Springer-Verlag London Limited Inc., 2009.

20. Ahmad Alsaadi ,”UML-Based Representation for Textual Objects”.2008 IEEE. Inc., 2008.

ПРИЛОЖЕНИЕ А

Код стилей XML файла

(полный код приложения на cd диске)

```
<w:body>
  <w:p w:rsidR="009E7A0A" w:rsidRDefault="00FD1862" w:rsidP="009E7A0A">
   <w:bookmarkStart w:id="0" w:name="_GoBack"/>
   <w:bookmarkEnd w:id="0"/>
   <w:r>
    <w:rPr>
     <w:rStyle w:val="H1Char"/>
    </w:rPr>
    <w:t>Студент</w:t>
   </w:r>
   <w:r w:rsidR="001D2BED" w:rsidRPr="001D2BED">
    <w:t xml:space="preserve"></w:t>
   </w:r>
    <w:r>
    <w:rPr>
     <w:rStyle w:val="H2Char"/>
    </w:rPr>
    <w:t>имеет</w:t>
   </w:r>
   <w:r w:rsidR="001D2BED">
    <w:t xml:space="preserve"></w:t>
   </w:r>
   <w:r>
    <w:rPr>
     <w:rStyle w:val="H1Char"/>
    </w:rPr>
    <w:t>зачетную книжку</w:t>
   </w:r>
   <w:r w:rsidR="001D2BED">
    <w:t xml:space="preserve"></w:t>
   </w:r>
  </w:p>
  <w:p w:rsidR="00576104" w:rsidRDefault="00576104" w:rsidP="009E7A0A"/>
  <w:p w:rsidR="00E50413" w:rsidRDefault="00E50413" w:rsidP="001D2BED">
    <w:pPr>
    <w:ind w:firstLine="567"/>
    </w:pPr>
  </w:p>
<w:sectPr w:rsidR="00E50413">
<w:pgSz w:w="11906" w:h="16838"/>
<w:pgMar w:top="1134" w:right="850" w:bottom="1134" w:left="1701" w:header="708" w:footer="708" w:gutter="0"/>
<w:cols w:space="708"/>
<w:docGrid w:linePitch="360"/>
</w:sectPr>
</w:body>
```

ПРИЛОЖЕНИЕ В

Код структуры формы

```
private void InitializeComponent()

        {
            this.button1 = new System.Windows.Forms.Button();
            this.listBox1 = new System.Windows.Forms.ListBox();
            this.button2 = new System.Windows.Forms.Button();
            this.button3 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            // button1
            this.button1.Location = new System.Drawing.Point(12, 222);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(108, 33);
            this.button1.TabIndex = 0;
            this.button1.Text = "Файл";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            // listBox1
            this.listBox1.FormattingEnabled = true;
            this.listBox1.Location = new System.Drawing.Point(12, 12);
            this.listBox1.Name = "listBox1";
            this.listBox1.Size = new System.Drawing.Size(408, 186);
            this.listBox1.TabIndex = 1;
            this.listBox1.SelectedIndexChanged += new
System.EventHandler(this.listBox1_SelectedIndexChanged);
            // button2
            this.button2.Location = new System.Drawing.Point(126, 222);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(111, 33);
            this.button2.TabIndex = 2;
            this.button2.Text = "Очистить";
            this.button2.UseVisualStyleBackColor = true;
            // button3
            this.button3.Location = new System.Drawing.Point(309, 222);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(111, 33);
            this.button3.TabIndex = 3;
            this.button3.Text = "Поиск";
            this.button3.UseVisualStyleBackColor = true;
            // Form1
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.SystemColors.ButtonFace;
            this.ClientSize = new System.Drawing.Size(432, 267);
            this.Controls.Add(this.button3);
```

```
            this.Controls.Add(this.button2);
            this.Controls.Add(this.listBox1);
            this.Controls.Add(this.button1);
            this.ForeColor = System.Drawing.SystemColors.ControlText;
            this.Name = "Form1";
            this.Text = "Парсер";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.ListBox listBox1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button3;
    }
}
```

ПРИЛОЖЕНИЕ С

Код нахождения стиля слова из XML файла

```
using (Package wdPackage = Package.Open(fileName, FileMode.Open, FileAccess.Read))
{
    PackageRelationship docPackageRelationship =
wdPackage.GetRelationshipsByType(documentRelationshipType).FirstOrDefault();
    if (docPackageRelationship != null)
    {
        Uri documentUri = PackUriHelper.ResolvePartUri(new Uri("/",
UriKind.Relative), docPackageRelationship.TargetUri);
        PackagePart documentPart = wdPackage.GetPart(documentUri);
        //  Загрузка документа XML
        xDoc = XDocument.Load(XmlReader.Create(documentPart.GetStream()));
        //  Часть нахождения стилей
        PackageRelationship styleRelation =
documentPart.GetRelationshipsByType(stylesRelationshipType).FirstOrDefault();
        if (styleRelation != null)
        {
            Uri styleUri = PackUriHelper.ResolvePartUri(documentUri,
styleRelation.TargetUri);
            PackagePart stylePart = wdPackage.GetPart(styleUri);
            //  Загрузка стилей XML в часть XDocument.
            styleDoc = XDocument.Load(XmlReader.Create(stylePart.GetStream()));
        }
    }
}
string defaultStyle =
    (string)(
        from style in styleDoc.Root.Elements(w + "style")
        where (string)style.Attribute(w + "type") == "paragraph"&&
              (string)style.Attribute(w + "default") == "1"
              select style
    ).First().Attribute(w + "styleId");
// Запрос, который находит все параграфы в документе и их стили.
var paragraphs =
    from para in xDoc
                 .Root
                 .Element(w + "body")
                 .Descendants(w + "p")
    let styleNode = para
                    .Elements(w + "pPr")
                    .Elements(w + "pStyle")
                    .FirstOrDefault()
    select new
    {
        ParagraphNode = para,
        StyleName = styleNode != null ?
            (string)styleNode.Attribute(w + "val") :
            defaultStyle
    };

foreach (var p in paragraphs)
                Console.WriteLine("StyleName:{0}", p.StyleName);
```