Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

|  |  |
| --- | --- |
| Допускаю к защите |  |
| Руководитель |  |
|  | подпись |
|  | А.Ю. Юрин |
|  | И.О. Фамилия |

|  |
| --- |
| Разработка программного обеспечения с использованием |
| методов и средств программной инженерии |

наименование темы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

|  |  |  |
| --- | --- | --- |
| Методы и средства проектирования информационных систем и технологий | | |
|  | 1.005.00.00 - ПЗ |  |

обозначение документа

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ИСМб-19-1 |  |  |  | И.А. Малиновцев |
|  |  | шифр группы |  | подпись |  | И.О. Фамилия |
| Нормоконтроль |  |  |  |  |  | А.Ю. Юрин |
|  |  |  |  | подпись |  | И.О. Фамилия |

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Иркутск 2022 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

ЗАДАНИЕ

НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

|  |  |  |
| --- | --- | --- |
| По курсу | Методы и средства проектирования информационных систем и технологий | |
| Студенту | Малиновцев И.А. | |
|  | (фамилия, инициалы) | |
| Тема проекта | Разработка программного обеспечения с использованием методов и средств программной инженерии | |
| Исходные данные | |  |
| 1. Произвести проектирование компонента для преобразования (конвертации) файла в формате RuleMLв файл формата EKB. | | |

|  |
| --- |
| Рекомендуемая литература: |
| 1. Дородных Н.О., Юрин А.Ю. Технология создания продукционных экспертных систем на основе модельных трансформаций. Новосибирск: СО РАН, 2019. 144 стр. 2. Юрин А.Ю., Грищенко М.А. Редактор баз знаний в формате CLIPS // Программные продукты и системы. 2012. № 4. С. 83–87. 3. Юрин А.Ю. CASE-средства: Методические указания по выполнению лабораторных работ. Иркутск: ИРНИТУ, 2018. 87 c. |

Графическая часть на \_\_\_\_\_\_\_\_\_\_\_\_\_\_ листах.

Дата выдачи задания « » 20 г.

|  |  |  |  |
| --- | --- | --- | --- |
| Задание получил |  |  |  |
|  | подпись |  | И.О. Фамилия |

Дата представления проекта руководителю « » 20 г.

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель курсового проектирования |  |  | А.Ю. Юрин |

Содержание

[Содержание 3](#_Toc121404369)

[Введение 4](#_Toc121404370)

[1 Общая часть 5](#_Toc121404371)

[1.1 Описание предметной области 5](#_Toc121404372)

[1.2 Постановка задачи 6](#_Toc121404373)

[2 Специальная часть 7](#_Toc121404374)

[2.1 Техническое задание 7](#_Toc121404375)

[2.1.1 Введение 7](#_Toc121404376)

[2.1.2 Общие сведения 7](#_Toc121404377)

[2.1.3 Назначение и цели создания 8](#_Toc121404378)

[2.1.4 Характеристика объектов автоматизации 9](#_Toc121404379)

[2.1.5 Требования к программе или программному изделию 9](#_Toc121404380)

[2.1.6 Состав и содержание работ по созданию системы 10](#_Toc121404381)

[2.1.7 Порядок контроля и приемки 10](#_Toc121404382)

[2.1.8 Требования к программной документации 11](#_Toc121404383)

[2.1.9 Приложения 11](#_Toc121404384)

[2.2 Описание проекта в Yougile 11](#_Toc121404385)

[2.3 Проектирование 12](#_Toc121404386)

[2.3.1 Диаграммы вариантов использования 12](#_Toc121404387)

[2.3.2 Диаграмма последовательностей 16](#_Toc121404388)

[2.3.3 Алгоритмическое обеспечение 17](#_Toc121404389)

[2.3.4 Диаграмма классов 18](#_Toc121404390)

[2.4 Программная реализация 19](#_Toc121404391)

[2.4.1 Автоматически сгенерированный код 19](#_Toc121404392)

[2.4.2 Описание интерфейса 22](#_Toc121404393)

[2.4.3 Тестирование 25](#_Toc121404394)

[Заключение 32](#_Toc121404395)

[Список использованных источников 33](#_Toc121404396)

[Приложение А 34](#_Toc121404397)

[Фрагмент сгенерированной документации 34](#_Toc121404398)

[Листинг программы 34](#_Toc121404399)

Введение

Курсовой проект в рамках учебной дисциплины «Методы и средства проектирования информационных систем и технологий» подразумевает проектирование и разработку программного обеспечения, осуществляющего конвертацию файлов RuleML в формат редактора продукционных баз знаний EKB инструментального средства Personal Knowledge Base Designer.

Актуальность разрабатываемого программного обеспечения определяется необходимостью создания данного приложения для автоматизации процесса конвертации.

Разрабатываемый программное обеспечение должно автоматизировать деятельность специалистов, в области проектирования информационных систем, сокращая их временные затраты при переносе и поддержании в актуальном состоянии проектов разной структуры в рамках представленных систем.

1 Общая часть

* 1. Описание предметной области

Программный компонент осуществляет процесс конвертации файлов RuleML в формат редактора продукционных баз знаний EKB.

Редактор баз знаний EKB сохраняет проекты с расширением .ekb и содержит многоуровневую структуру.



Рисунок 1 – Фрагмент структуры файла .ekb

RuleML — это глобальная инициатива, возглавляемая некоммерческой организацией RuleML Inc., которая посвящена продвижению исследований и деятельности по разработке отраслевых стандартов в технической области семантических и интероперабельных правил. Дизайн стандартов в основном принимает форму языка разметки, также известного как RuleML.

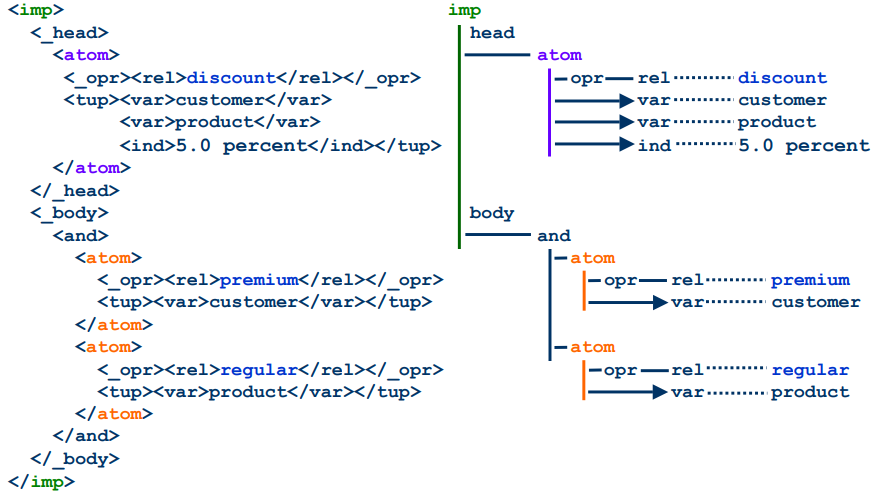


Рисунок 2 – Фрагмент структуры файла RuleML

Дополнительно программный компонент предоставляет возможность выбора пользователем места хранения файла RuleML и места сохранения конвертированного файла .ekb.

1.2 Постановка задачи

Проектирование и программная реализация компонента для преобразования (конвертации) XML-файлов определенной структуры (согласно варианту задания).

Выполнение работы предполагает решение следующих задач:

* Проектирование, включая:
  + Описание предметной области.
  + Формулировка цели работы.
  + Проектирование компонента.
* Программная реализация компонента, включая автоматизированную генерацию программного кода.
* Тестирование.
* Документирование (описание), включая автоматизированную генерацию документации.

2 Специальная часть

2.1 Техническое задание

2.1.1 Введение

Настоящее техническое задание распространяется на разработку программного компонента для конвертации файлов RuleML в формат редактора продукционных баз знаний EKB.

При проектировании баз знаний в определенном программном обеспечении может потребоваться продолжить работу в другом прикладном средстве. Экспортирование наработок в данной ситуации не всегда представляется возможным.

Разрабатываемая программа позволит экспортировать проекты продукционных баз знаний EKB в файлы для работы в RuleML в автоматическом режиме.

2.1.2 Общие сведения

**2.1.2.1 Наименование и шифры**

1. Полное название системы

Программный компонент «Транслятор файлов структуры».

1. Шифр системы

Шифр системы – ПК ТФС.

**2.1.2.2 Сведения о заказчиках и исполнителях**

1. Заказчик

Иркутский Национальный Исследовательский Технический университет.

Адрес: 664074 г. Иркутск, ул. Лермонтова, 83.

1. Исполнитель

Студент Малиновцев И. А.

**2.1.2.3 Основание для разработки**

1. Основания для разработки программного компонента

Основанием для исполнения работ по созданию программного компонента, предусмотренных в настоящем ТЗ, является учебный план «09.03.02 Информационные системы и технологии на предприятиях/в машиностроении» и методические указания курсовому проекту по курсу «Методы и средства проектирования информационных систем и технологий».

1. Основания для разработки документа

Основанием для разработки настоящего документа является п.3 к методическим указаниям.

1. Нормативные документы

Настоящее Техническое Задание разработано в соответствии с требованиями ГОСТ 34.602 89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы».

При разработке и вводе в эксплуатацию автоматизированной системы Исполнитель должен руководствоваться требованиями следующих нормативных документов Госстандарта:

* ГОСТ 34.601.90 – стандарт, устанавливающий стадии и этапы создания АС, а также содержание работ на каждом этапе;
* ГОСТ 34.603.92 – стандарт, устанавливающий виды испытаний АС и общие требования к их проведению.

При создании проектно-эксплуатационной документации Исполнитель должен руководствоваться требованиями ГОСТ РД 50-34.698-90 «Автоматизированные системы. Требования к содержанию документов».

**2.1.2.4 Сроки исполнения работ**

Начало разработки – «01» сентября 2022 г.

Окончание разработки – «25» декабря 2022 г.

**2.1.2.5 Сведения об источниках и порядке финансирования работ**

Источник финансирования – бюджет Иркутского Национального Исследовательского Технического университета.

Порядок финансирования определяется условиями методического указания.

**2.1.2.6 Порядок оформления и представления заказчику результатов работ**

Порядок оформления и предъявления работы Заказчику ПК ТФС производится согласно методическому указанию. Приемка системы осуществляется комиссией, созданной Заказчиком.

2.1.3 Назначение и цели создания

**2.1.3.1 Назначение**

Программный компонент ПК ТФС предназначен для автоматической конвертации файлов RuleML в формат редактора продукционных баз знаний EKB.

**2.1.3.2 Цели создания**

1. Цели создания и внедрения ПК ТФС:

автоматизация конвертации файлов структуры;

1. Цели текущего этапа

Целью работ по данному этапу проекта является создание первой экспериментальной версии программного компонента.

2.1.4 Характеристика объектов автоматизации

**2.1.4.1 Краткие сведения об объекте автоматизации**

Объектом автоматизации является учебный процесс Иркутского Национального Исследовательского Технического университета.

Предметом автоматизации является процесс конвертации файлов структуры.

**2.1.4.2 Сведения об условиях эксплуатации**

Программный компонент будет эксплуатироваться в составе программного комплекса Заказчика.

2.1.5 Требования к программе или программному изделию

**2.1.5.1 Требования к функциональным характеристикам**

1. Программный компонент должен обеспечивать возможность выполнения следующих функций:

указание пути к файлу RuleML;

указание пути к месту сохранения конвертированного файла;

оповещение пользователя об успешной процедуре конвертации файла;

оповещение пользователя об ошибке при конвертации файла.

1. Исходные данные:

путь к файлу RuleML;

пусть к месту сохранения конвертированного файла.

**2.1.5.2 Требования к надежности**

Предусмотреть контроль вводимой информации.

Предусмотреть блокировку некорректных действий пользователя при работе с системой.

**2.1.5.3 Требования к составу и параметрам технических средств**

Система должна работать на IBM совместимых персональных компьютерах.

Минимальная конфигурация:

тип процессора: Celeron и выше;

объем оперативного запоминающего устройства: 2048 Мб и более.

**2.1.5.4 Требования к информационной и программной совместимости**

Система должна работать под управлением семейства операционных систем Windows NT.

**2.1.5.5 Требования к организационному обеспечению**

В ходе разработки должно обеспечиваться постоянное взаимодействие между сторонами, для чего ими должны быть сформированы рабочие группы, решающие следующие вопросы:

- административные вопросы;

- инженерно-технические вопросы;

- вопросы методического обеспечения.

2.1.6 Состав и содержание работ по созданию системы

Таблица 1 – Состав и содержание работ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название этапа | Срок | Отчетность |
| 1 | Выбор языка программирования и подходящей среды для разработки. | 01.09.2022-11.09.2022 | Обоснование выбора языка программирования. |
| 2 | Разработка ядра программного компонента. | 12.09.2022-30.10.2022 | Реализация системы на уровне ядра. Описание методов и алгоритмов. |
| 3 | Разработка базового графического интерфейса приложения. | 31.10.2022- 13.11.2022 | Описание базового графического интерфейса. |
| 4 | Тестирование базового функционала приложения. | 14.11.2022-20.11.2022 | Тесты. Документация. |
| 5 | Разработка модуля выбора места нахождения файла и пути для сохранения результата. | 21.11.2022- 04.12.2022 | Описание методов и алгоритмов модуля. |
| 6 | Тестирование программ­ного продукта и составление программной документации. | 05.12.2022-13.12.2022 | Тесты. |
| 7 | Написание сопроводительной и документации | 14.12.2022-25.12.2022 | Документация. Про­граммный продукт. |

По окончанию каждого этапа заказчику будет предоставлен отчёт о выполненной работе.

2.1.7 Порядок контроля и приемки

**2.1.7.1 Сдача-приемка работ**

Сдача-приёмка работ производится поэтапно, в соответствии с Календарным графиком. Основанием для сдачи-приёмки работ служит Отчёт о завершении работ по этапу.

**2.1.7.2 Предварительные испытания**

При сдаче-приёмке готовых подсистем ПК ТФС, создаваемых в рамках настоящей работы, проводятся предварительные испытания с целью подтверждения работоспособности соответствующей подсистемы и соответствия требованиям ТЗ. Предварительные испытания должны проводиться представителями Заказчика по программе тестирования, составленной Исполнителем и согласованной с Заказчиком.

По итогам испытаний по мере необходимости Исполнителем проводится корректировка проектной и эксплуатационной документации и устранение выявленных недостатков программного обеспечения.

**2.1.7.3 Опытная эксплуатация**

Передача системы в опытную эксплуатацию производится приемо-сдаточной комиссией на основании положительного заключения, сделанного по итогам предварительных испытаний.

Опытная эксплуатация должна проводиться на технических средствах Заказчика.

2.1.8 Требования к программной документации

Разрабатываемые программные модули должны быть самодокументированы, т. е. тексты программ должны содержать все необходимые комментарии.

В состав сопровождающей документации должны входить:

Пояснительная записка на 25-30 листах, содержащая описание разработки.

Руководство пользователя.

2.1.9 Приложения

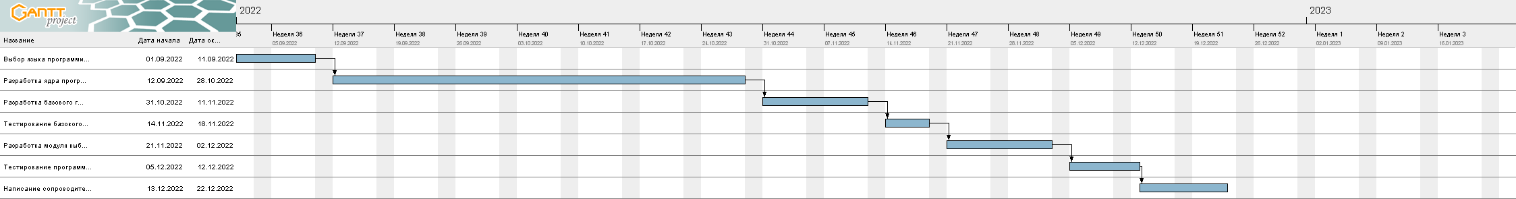


Рисунок 3 – Диаграмма Ганта

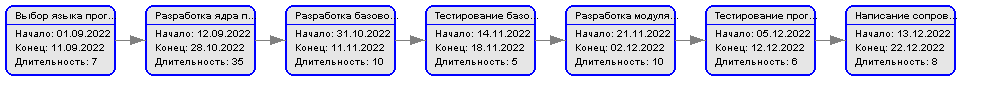


Рисунок 4 – Перт диаграмма

2.2 Описание проекта в Yougile

В качестве программы для управления проектом был выбрана Yougile. Для отслеживания прогресса выполнения курсового проекта был создан проект Yougile. Далее было создано пять колонок для задач. Колонка «Задачи» содержит в себе карточки с этапами выполнения работы. Колонка «В процессе» предназначена для хранения выполняемых в данный момент задач. Блок «Проверяется» создан для содержания отправленных на проверку частей курсового проекта. «На доработке» создан для хранения задач, отправленных на доработку. Колонка «Сделано» содержит выполненные и проверенные задачи.

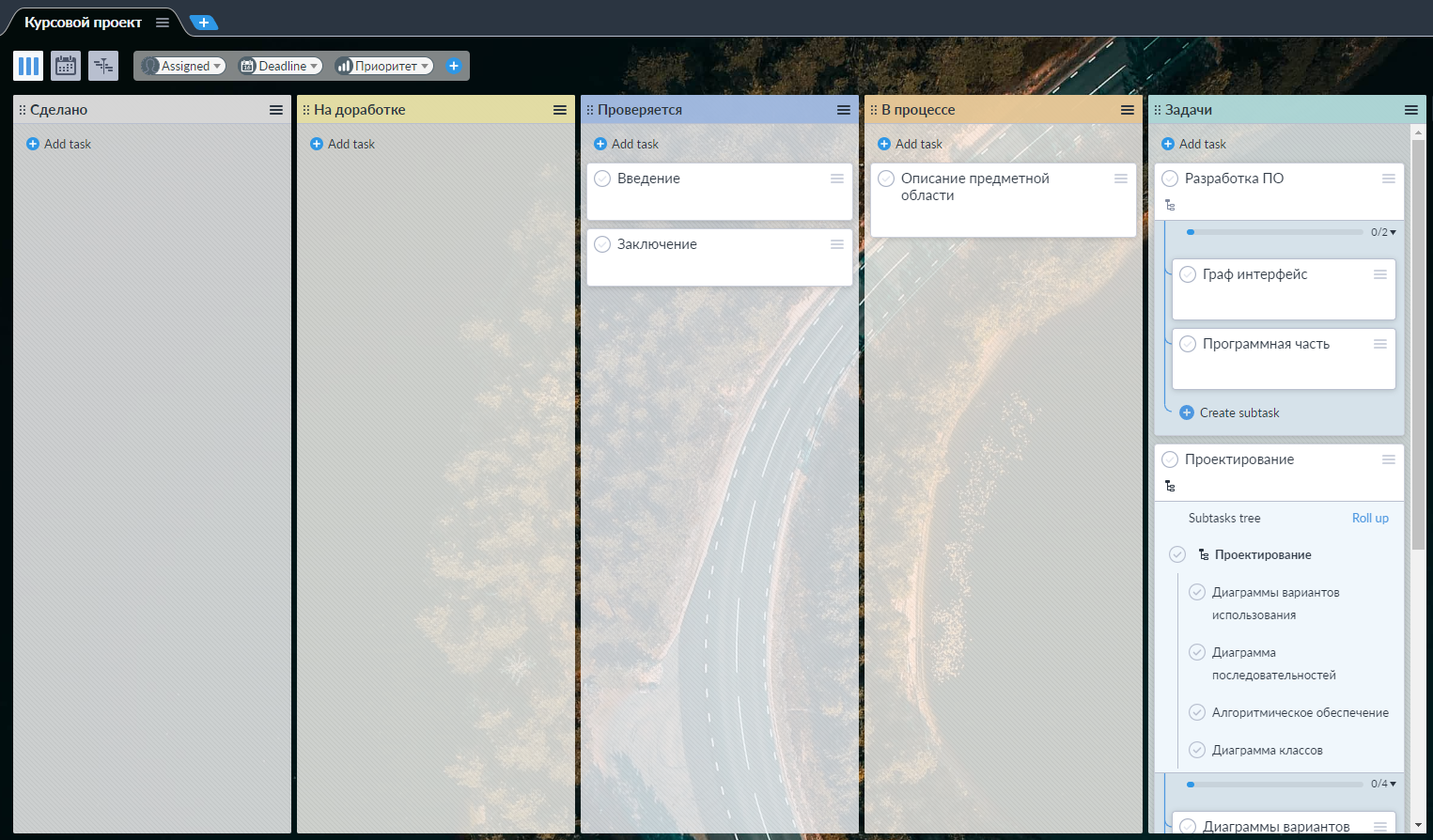


Рисунок 5 – проект в программе Yougile

2.3 Проектирование

В качестве инструмента для проектирования было использовано CASE-средство Enterprise Architect. Enterprise Architect представляет собой уникальный и многофункциональный инструмент, предназначенный для осуществления визуального моделирования и дизайна. Главная особенность данного программного обеспечения заключается в том, что его система основана на OMG UML – одно из самых эффективных и мощных платформ в современном мире. Платформа поддерживает: проектирование и построение программных комплексов, моделирование бизнес-процессов и моделирование отраслевых доменов.

2.3.1 Диаграммы вариантов использования

На основе составленного технического задания была составлена диаграмма вариантов использования программного компонента.

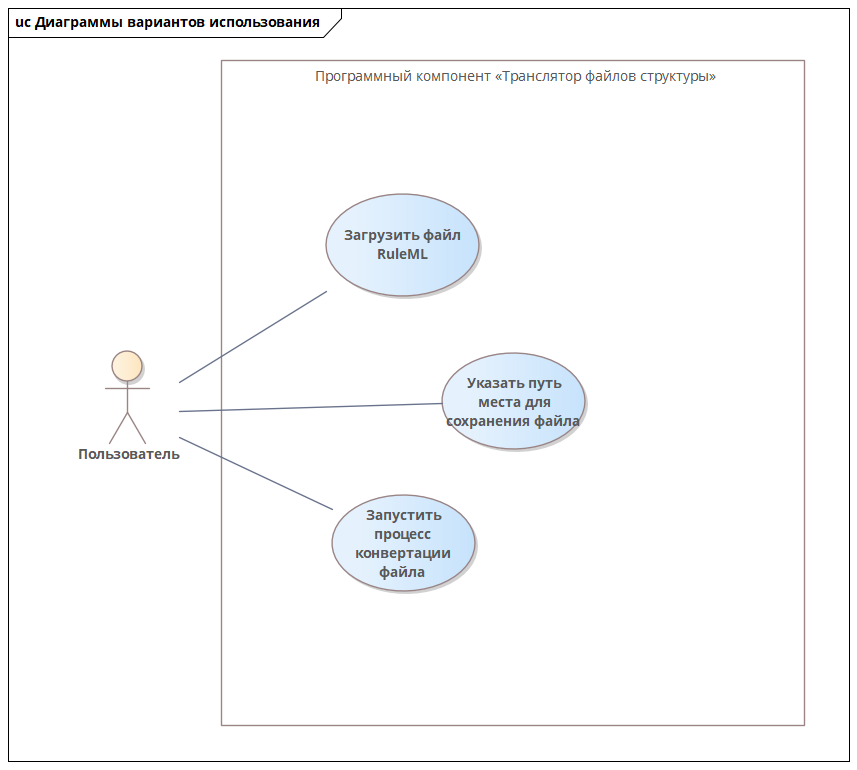


Рисунок 6 – диаграмма вариантов использования программного компонента

Основываясь на разработанной диаграмме, были описаны основные сценарии вариантов использования приложения.

**Спецификация варианта использования «Загрузить файл RuleML»**

**Цель**: загрузка файла RuleML для дальнейшей конвертации

**Активные субъекты**: пользователь

**Краткое описание**: пользователь загружает файл RuleML посредством меню загрузки файла.

**Основной поток событий**:

* 1. Пользователю потребовалось конвертировать файл RuleML в файл приложения EKB.
  2. Пользователь запускает приложение посредством исполняемого EXE-файла.
  3. Пользователь запускает механизм загрузки файла RuleML.
  4. Приложения открывает окно интерфейса-меню выбора файла.
  5. Пользователь указывает путь до необходимого файла с помощью меню загрузки файла.
  6. Система записывает полный пуль до файла в заранее определенную переменную.
  7. Пользователь получает текстовое уведомление об успешной загрузке файла.

**Альтернативные потоки событий:**

1. Активация сценария «Указать путь места для сохранения файла».

**Специальные требования**: нет.

**Предусловия**: нет.

**Постусловия**: после активации варианта должен быть активирован один или несколько сценариев вариантов использования: «Указать путь места для сохранения файла» или «Запустить процесс конвертации файла».

**Дополнительные замечания**: вариант использования «Запустить процесс конвертации файла» не имеет смысла выполнять без завершенных сценариев «Загрузить файл RuleML» и «Указать путь места для сохранения файла».

**Спецификация варианта использования «Указать путь места для сохранения файла»**

**Цель**: указание места сохранения файла.

**Активные субъекты**: пользователь.

**Краткое описание**: пользователь указывает директорию для сохранения файла посредством меню сохранения файла.

**Основной поток событий**:

* 1. Пользователю потребовалось конвертировать файл RuleML в файл приложения EKB.
  2. Пользователь запускает приложение посредством исполняемого EXE-файла.
  3. Пользователь запускает механизм выбора директории для сохранения.
  4. Приложения открывает окно интерфейса-меню выбора директории.
  5. Пользователь указывает путь до необходимой директории с помощью меню.
  6. Система записывает полный пуль до конечной директории в заранее определенную переменную.
  7. Пользователь получает текстовое уведомление об успешном выборе места сохранения файла.

**Альтернативные потоки событий:**

1. Активация сценария «Загрузить файл RuleML».

**Специальные требования**: нет.

**Предусловия**: нет.

**Постусловия**: после активации варианта должен быть активирован один или несколько сценариев вариантов использования: «Загрузить файл RuleML» или «Запустить процесс конвертации файла».

**Дополнительные замечания**: вариант использования «Запустить процесс конвертации файла» не имеет смысла выполнять без завершенных сценариев «Загрузить файл RuleML» и «Указать путь места для сохранения файла».

**Спецификация варианта использования «Запустить процесс конвертации файла»**

**Цель**: конвертация файла.

**Активные субъекты**: пользователь.

**Краткое описание**: пользователь запускает механизм конвертации файла.

**Основной поток событий**:

* 1. Пользователю потребовалось конвертировать файл RuleML в файл приложения EKB.
  2. Пользователь запускает приложение посредством исполняемого EXE-файла.
  3. Пользователь запускает механизм загрузки файла RuleML.
  4. Приложения открывает окно интерфейса-меню выбора файла.
  5. Пользователь указывает путь до необходимого файла с помощью меню загрузки файла.
  6. Система записывает полный пуль до файла в заранее определенную переменную.
  7. Пользователь получает текстовое уведомление об успешной загрузке файла.
  8. Пользователь запускает механизм выбора директории для сохранения.
  9. Приложения открывает окно интерфейса-меню выбора директории.
  10. Пользователь указывает путь до необходимой директории с помощью меню.
  11. Система записывает полный пуль до конечной директории в заранее определенную переменную.
  12. Пользователь получает текстовое уведомление об успешном выборе места сохранения файла.
  13. Пользователь запускает конвертацию файла.
  14. Файл загружается в приложение из указанного пути.
  15. Создается объект класса RuleML с полями файла.
  16. Данные из файла заносятся в соответствующие переменные объекта класса.
  17. Создается объект класса EKB.
  18. Поля класса EKB и RuleML соотносятся и перезаписываются
  19. Конечный объект EKB дополняется необходимыми конструкциями для работоспособности файла.
  20. Объект EKB с дополненными конструкциями сохраняется в формате .ekb в заданной пользователем директории.
  21. Пользователь получает текстовое уведомление об успешной конвертации файла.

**Альтернативные потоки событий:** нет.

**Специальные требования**: нет.

**Предусловия**:

1. Выполнен сценарий «Загрузить файл RuleML».
2. Выполнен сценарий «Указать путь места для сохранения файла».

**Постусловия**: нет.

**Дополнительные замечания**: вариант использования «Запустить процесс конвертации файла» не имеет смысла выполнять без завершенных сценариев «Загрузить файл RuleML» и «Указать путь места для сохранения файла».

2.3.2 Диаграмма последовательностей

На основе диаграммы вариантов использования была составлена диаграмма последовательностей.

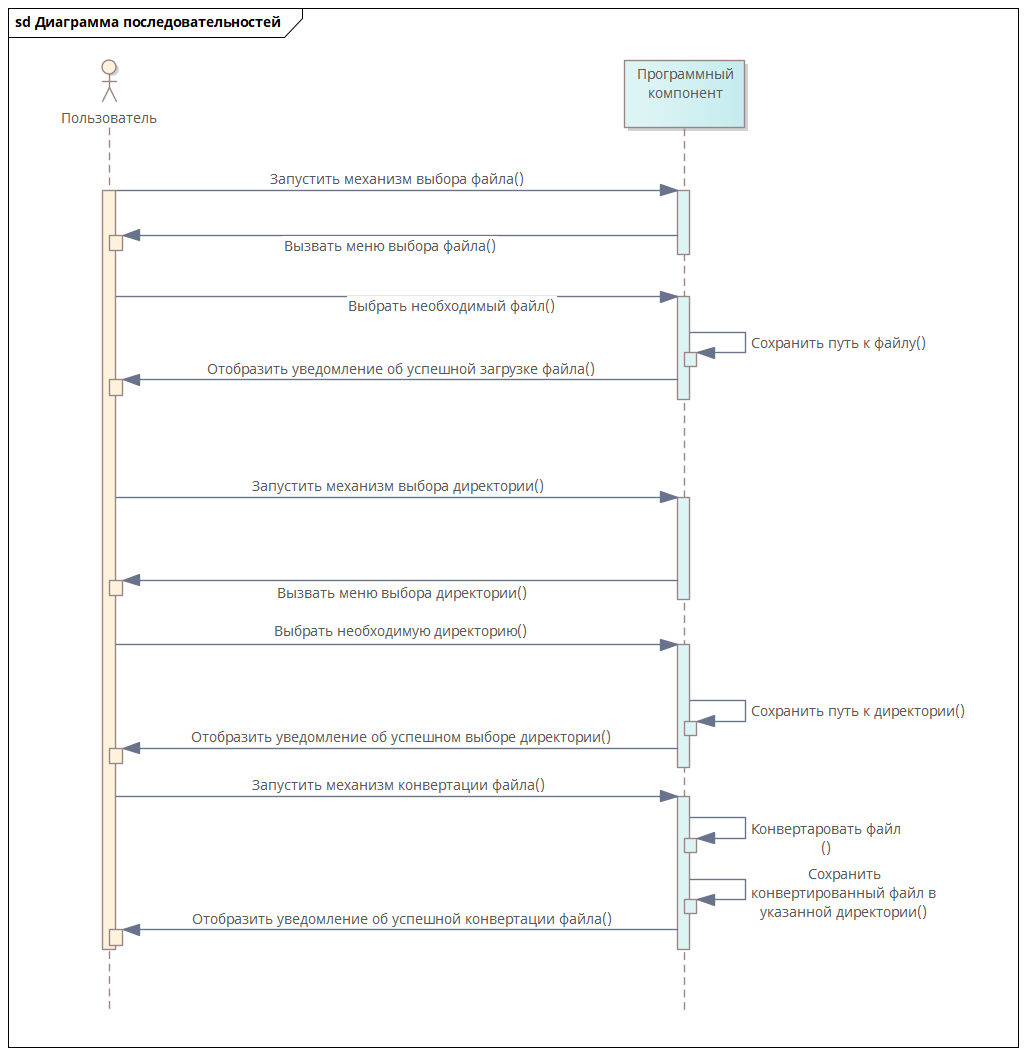


Рисунок 7 – диаграмма последовательностей

Пользователь с помощью меню выбора файла операционной системы задает приложению директорию для загрузки файла и директорию для сохранения результата конвертации. Следующим шагом участник системы запускает механизм конвертации файла посредством нажатия на соответствующий объект графического интерфейса (кнопку). В результате в директории создается файл с соответствующим расширением.

2.3.3 Алгоритмическое обеспечение

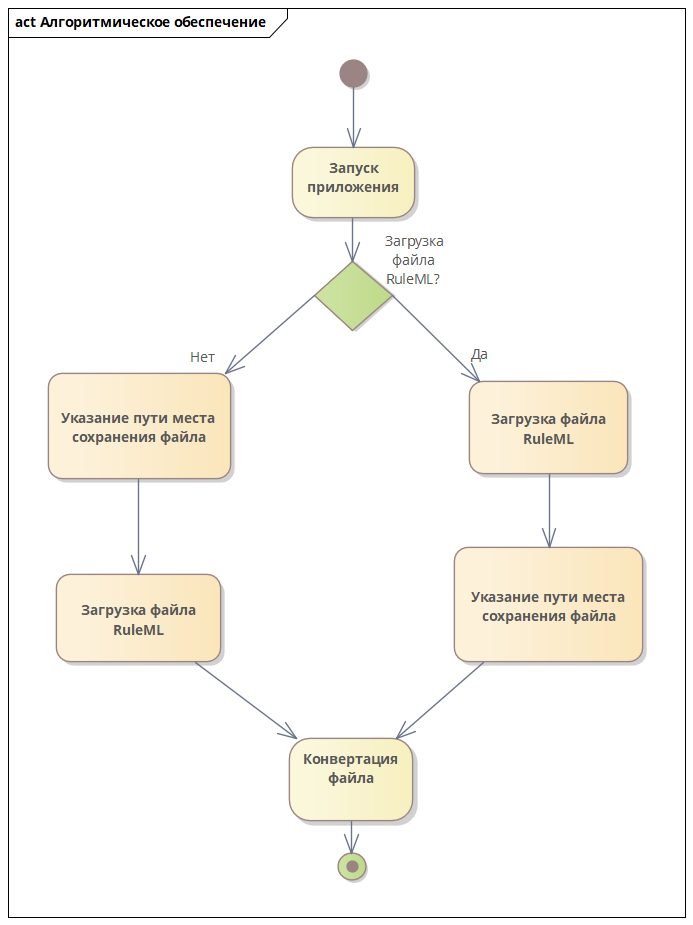


Рисунок 8 – диаграмма деятельности

Первым этапом после инициализации деятельности выступает блок «Запуск приложения». На данном этапе пользователь запускает программный компонент. После участнику системы представляется выбор очередности действия. В случае выполнения первого вариант пользователь укажет путь к директории для сохранения конвертированного файла, далее загрузит файл для конвертации. В противном случае, пользователь сначала загрузит файл для работы, а после укажет путь к месту сохранения. Ветвления сходятся в блоке «Конвертация файла», в котором осуществляется указанный в названии механизм. Данный блок является финальным.

2.3.4 Диаграмма классов

На основе разработанной проектной документации была создана диаграмма классов программного компонента.

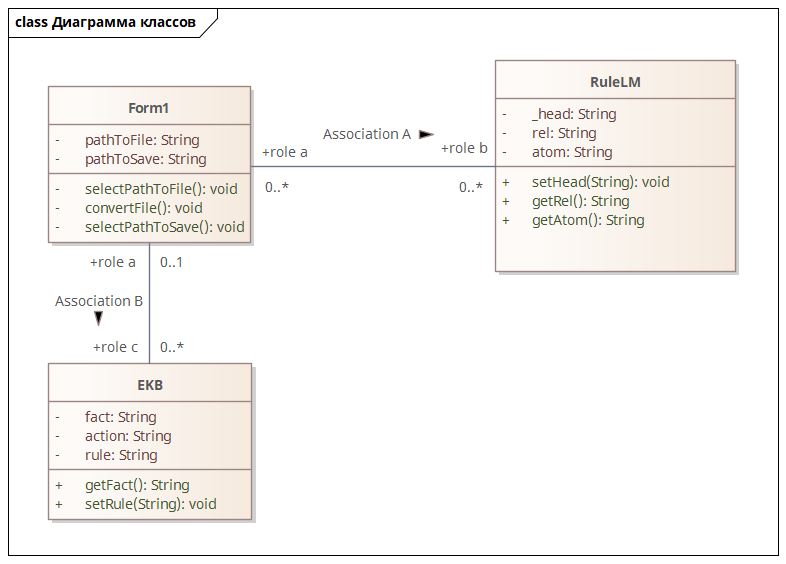


Рисунок 9 – диаграмма классов

Диаграмма представляет из себя три класса, необходимых для полноценного функционирования приложения с целью выполнения поставленных задач. Класс Form1 представляет из себя контроллер графического приложения. Основными переменными являются: путь до директории с файлом и директория для сохранения. Соответствующие методы предназначены для получения необходимых местоположений. Дополнительно спроектирован метод, реализующий механизм конвертации файла.

Классы RuleML и EKB предназначены для хранения данных из файлов. При запуске процесса конвертации файл RuleML будет декомпозирован в отдельные переменные объекта класса. Далее полученные значения будут пересобраны в экземпляре EKB и дополнены необходимыми для структурной целостности файла данными. Спроектированные блоки условных data-классов представлены в виде демонстрационных моделей: отражают неполные наборы переменных и методов getter – setter, так как представленного наполнения достаточно для генерации первичного скелетного кода для последующего расширения.

* 1. Программная реализация

2.4.1 Автоматически сгенерированный код

На основе разработанной диаграммы классов был сгенерирован код. В качестве выбранного языка программирования был установлен C#.

**Код класса Form1**

///////////////////////////////////////////////////////////

// Form1.cs

// Implementation of the Class Form1

// Generated by Enterprise Architect

// Created on: 08-дек-2022 14:45:03

// Original author: worker

///////////////////////////////////////////////////////////

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

public class Form1 {

private String pathToFile;

private String pathToSave;

public EKB role c;

public RuleLM role b;

public Form1(){

}

~Form1(){

}

///

/// <param name="Parameter A"></param>

private void selectPathToFile(Parameter A){

}

private void convertFile(){

}

///

/// <param name="Parameter"></param>

private void selectPathToSave(Parameter){

}

}//end Form1

**Код класса EKB**

///////////////////////////////////////////////////////////

// EKB.cs

// Implementation of the Class EKB

// Generated by Enterprise Architect

// Created on: 08-дек-2022 14:45:06

// Original author: worker

///////////////////////////////////////////////////////////

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

public class EKB {

private String fact;

private String action;

private String rule;

public EKB(){

}

~EKB(){

}

///

/// <param name="Parameter A"></param>

public String getFact(Parameter A){

return "";

}

///

/// <param name="Parameter"></param>

public void setRule(String Parameter){

}

}//end EKB

**Код класса RuleML**

///////////////////////////////////////////////////////////

// RuleLM.cs

// Implementation of the Class RuleLM

// Generated by Enterprise Architect

// Created on: 08-дек-2022 14:45:05

// Original author: worker

///////////////////////////////////////////////////////////

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

public class RuleLM {

private String \_head;

private String rel;

private String atom;

public RuleLM(){

}

~RuleLM(){

}

///

/// <param name="Parameter A"></param>

public void setHead(String Parameter A){

}

///

/// <param name="Parameter"></param>

public String getRel(Parameter){

return "";

}

public String getAtom(){

return "";

}

}//end RuleLM

2.4.2 Описание интерфейса

В качестве среды для создания макетов графического интерфейса был использован онлайн-редактор для создания интерфейсов и прототипов Figma. Основываясь на диаграмме вариантов использования, было спроектировано пять прототипов возможных состояний программного компонента.

Первый фрагмент отображает первоначальное состояние приложения после запуска.

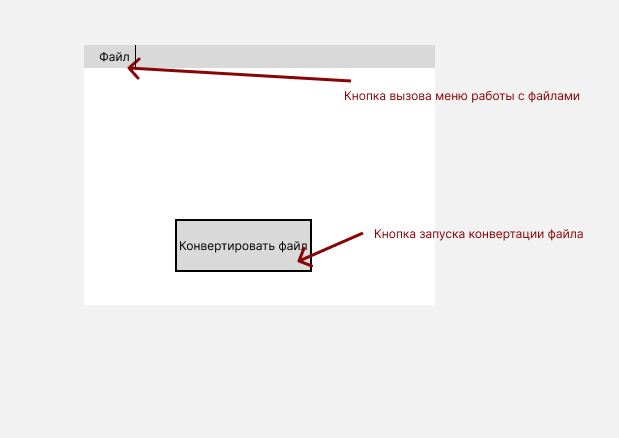


Рисунок 10 – Макет стартового окна

При нажатии на кнопку «Файл» открывается меню, содержащее кнопки для загрузки файла и выбора директории для сохранения.

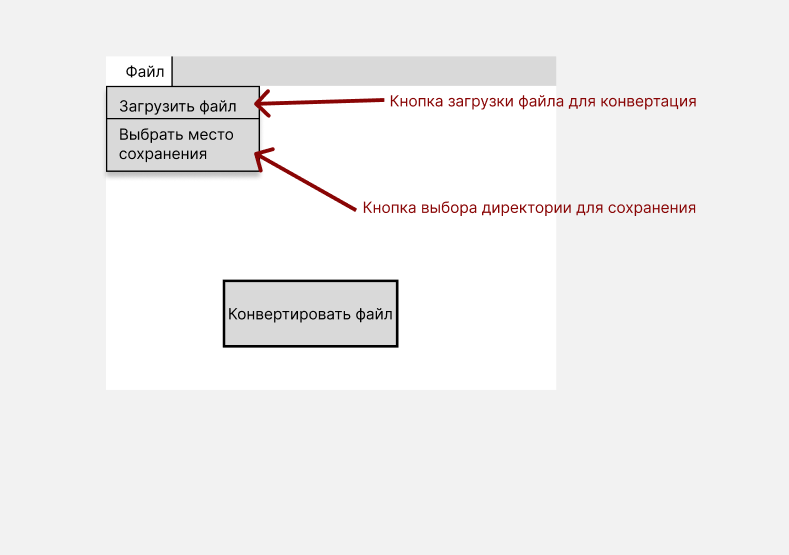


Рисунок 11 – Меню «Файл»

При нажатии на кнопки загрузки или выбора директории открывается окно соответствующего выбора.

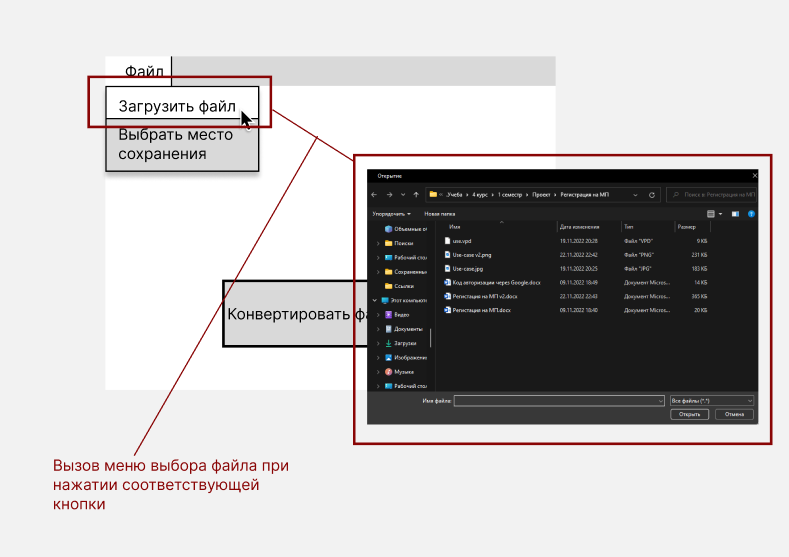


Рисунок 12 – Взаимодействие с кнопками выбора

Текущие состояния работы с файлами и директориями указывают информационные поля. В качестве упрощения визуального восприятия типа состояния текст окрашен соответствующим цветом.

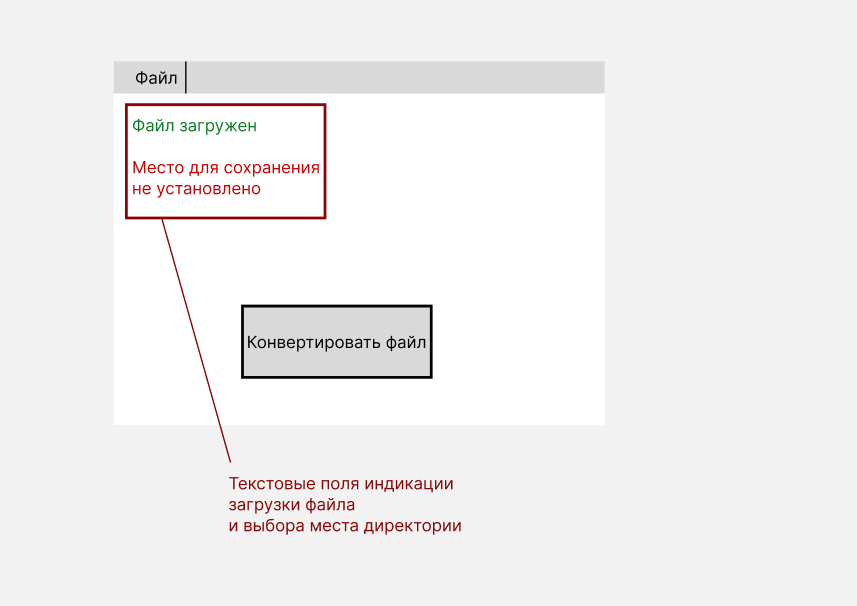


Рисунок 13 – Отображение полей состояния

После завершения данного процесса отображается окно с сообщением об успешной конвертацией или ошибкой.

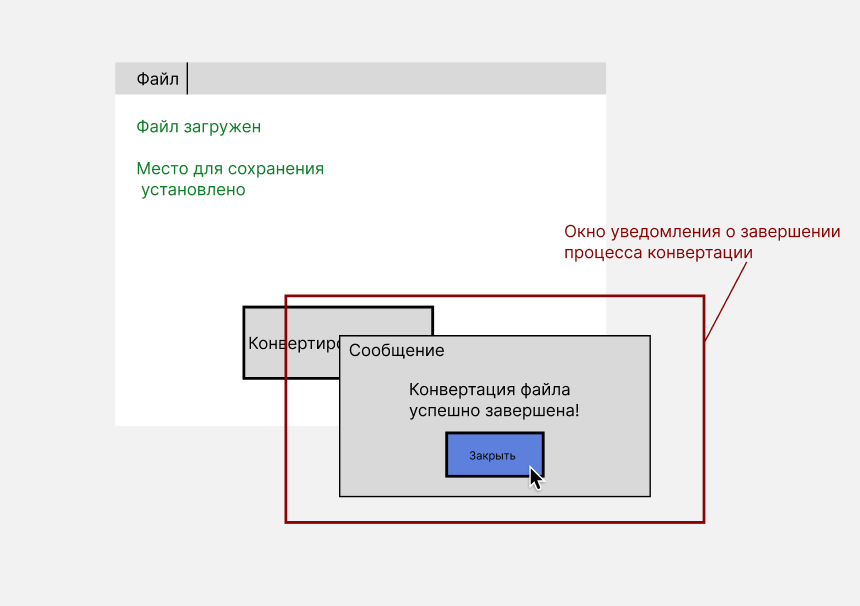


Рисунок 14 – Визуализация завершения процесса

Конечный вариант графического интерфейса создан с помощью элементов окна Windows Forms.

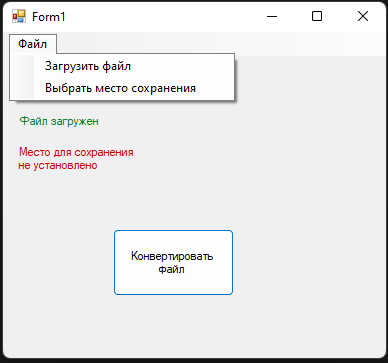


Рисунок 15 – Графический интерфейс в среде разработки

2.4.3 Тестирование

Для тестирования разработанного программного компонента была использована компромиссный стратегия. Компромиссная стратегия – проектирование тестов, исходя из принципов:

1. проверка функции или возможности;
2. проверка каждой области и границы изменения значений какой-либо входной величины;
3. проверка каждого особого случая или исключительной ситуации;
4. каждая команда программы должна проработать хотя бы на одном тесте.

Таблица 2 – Тесты программного компонента

|  |  |  |  |
| --- | --- | --- | --- |
| N | Описание проверяемой ситуации | Тест (входные данные) | Результат |
|  | Проверка работы компонента со стандартным файлом ruleml. | Полученный файл ruleml. | Конвертированный файл с расширением ekb. |
|  | Проверка работы компонента с файлом ruleml нестандартной конфигурации. | Файл ruleml нестандартной конфигурации. | Окно с уведомлением об ошибке конвертации. |
|  | Попытка конвертировать файл без указания директории для сохранения. | Указан путь к файлу ruleml. | Кнопка конвертации сохранила неактивное состояние. |
|  | Попытка конвертировать файл без указания файла для конвертации. | Указана директория для сохранения файла. | Кнопка конвертации сохранила неактивное состояние. |
|  | Проверка работы компонента с файлом нестандартного для приложения компонента. | Файл с расширением ekb. | Окно с уведомлением об ошибке конвертации. |
|  | Проверка работы компонента с пустым файлом ruleml. | Пустой файл ruleml. | Окно с уведомлением об ошибке конвертации. |
|  | Повторная конвертация загруженного файла без изменения директории. | Полученный файл ruleml. | Перезаписанный файл с расширением ekb. |

Результаты тестирования представлены на рисунках 16-29.

**Тест №1 – Проверка работы компонента со стандартным файлом ruleml.**

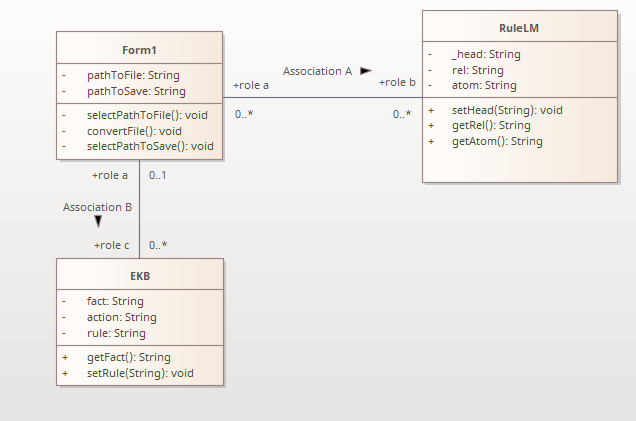


Рисунок 16 – Исходная диаграмма классов



Рисунок 17 – Файл для конвертации

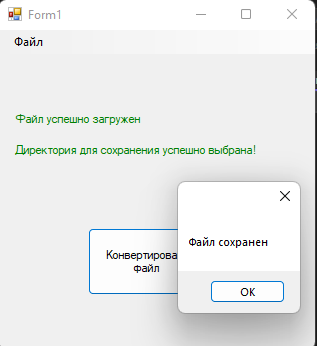


Рисунок 18 – Уведомление об успешной конвертации

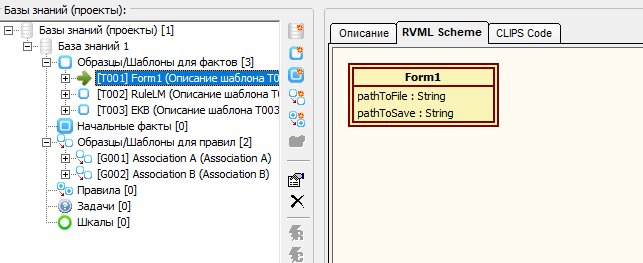


Рисунок 19 – Результат импортирования

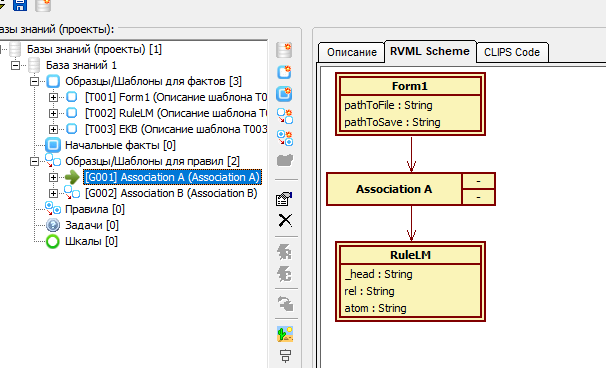


Рисунок 20 – Результат импортирования

**Тест №2 - Проверка работы компонента с файлом ruleml нестандартной конфигурации.**



Рисунок 21 – Файл нестандартной конфигурации

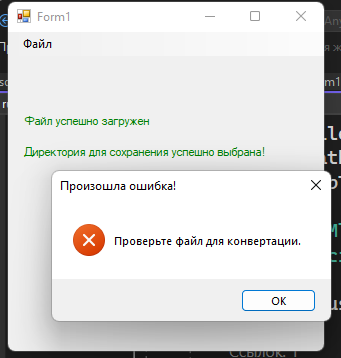


Рисунок 22 – Результат работы приложения

**Тест №3 – Попытка конвертировать файл без указания директории для сохранения.**

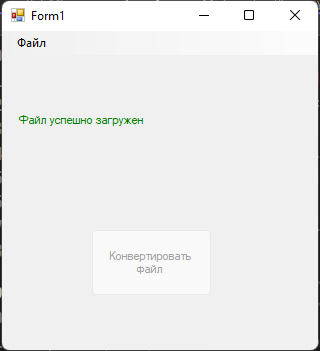


Рисунок 23 – Результат выполнения теста

**Тест 4 – Попытка конвертировать файл без указания файла для конвертации.**

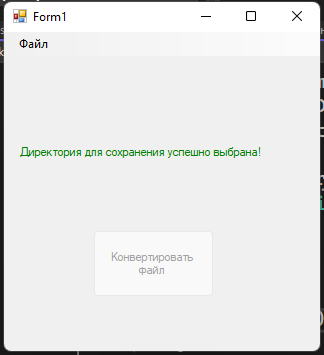


Рисунок 24 – Результат выполнения теста

**Тест №5 – Проверка работы компонента с файлом нестандартного для приложения компонента.**

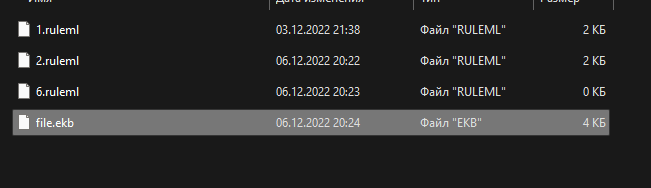


Рисунок 25 – Выбор нестандартного файла

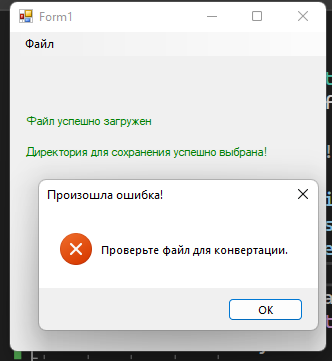


Рисунок 26 – Результат выполнения теста

**Тест №6 – Проверка работы компонента с пустым файлом ruleml.**

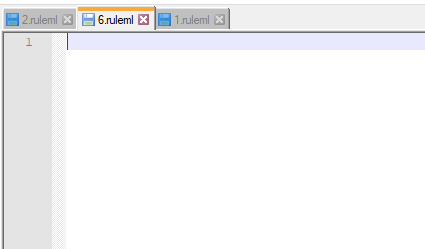


Рисунок 27 – Пустой файл

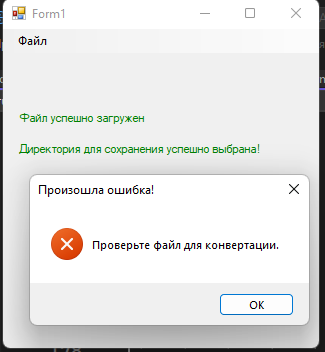


Рисунок 28 – Результат выполнения теста

**Тест №7 – Повторная конвертация загруженного файла без изменения директории.**

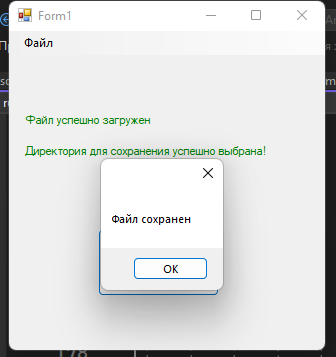


Рисунок 29 – Результат выполнения теста

Заключение

В ходе выполнения проекта было разработано программное обеспечение, обеспечивающее автоматизированное преобразование файлов RuleML в формат инструментального средства Personal Knowledge Base Designer.

Составлено техническое задание для выполнения задания в рамках курсового проекта, отвечающее стандартам и требованиям ГОСТ, описан состав и содержание работы, составлены Перт диаграмма и диаграмма Ганта.

Спроектирована доска в специализированном программном обеспечении Trello для отслеживания процесса выполнения задания в рамках курсового проекта.

С помощью CASE-средства Enterprise Architect спроектированы:

1. Диаграммы вариантов использования в нотации UML.
2. Описаны основные варианты использования разработанного программного компонента.
3. Разработана системная диаграмма последовательности действий.
4. Разработана диаграмма классов уровня проектирования в нотации UML.
5. Произведена генерация скелетного кода на основе диаграмм классов для целевого языка программирования.

Описан графический интерфейс и разработана программная реализация приложения. Разработанное приложение успешно прошло составленные тесты.

Список использованных источников

1. Гутгарц Р.Д. Проектирование автоматизированных систем обработки информации и управления. Методические указания по выполнению курсового проекта. Иркутск, ИрГТУ, 2010. 57 с.
2. Коваленко В.В. Проектирование информационных систем: учеб. пособие. Иркутск: Форум, НИЦ ИНФРА. М, 2014.  320 с.
3. Исаев Г.Н. Проектирование информационных систем: Учебное пособие/ Москва: Изд-во Омега Л, 2015. 424 с.
4. Кватрани Т. RationalRose 2000 и UML. Визуальное моделирование. М.: ДМК Пресс, 2001. 176 с.
5. Дородных Н.О., Юрин А.Ю. Технология создания продукционных экспертных систем на основе модельных трансформаций. Новосибирск: СО РАН, 2019. 144 стр.
6. Юрин А.Ю., Грищенко М.А. Редактор баз знаний в формате CLIPS // Программные продукты и системы. 2012. № 4. С. 83–87.
7. Юрин А.Ю. CASE-средства: Методические указания по выполнению лабораторных работ. Иркутск: ИРНИТУ, 2018. 87 c.

Приложение А

Фрагмент сгенерированной документации

**Пользователь**

Actor in package 'Actors'

Пользователь

Version 1.0 Phase 1.0 Proposed

worker created on 20.11.2022. Last modified 20.11.2022

| ASSOCIATIONS | |
| --- | --- |
| Association (direction: Unspecified) | |
| Source: Public (Actor) Пользователь | Target: Public (UseCase) Закрыть приложение |
| Association (direction: Unspecified) | |
| Source: Public (Actor) Пользователь | Target: Public (UseCase) Загрузить файл RuleML |
| Association (direction: Unspecified) | |
| Source: Public (Actor) Пользователь | Target: Public (UseCase) Указать путь места для сохранения файла |
| Association (direction: Unspecified) | |
| Source: Public (Actor) Пользователь | Target: Public (UseCase) Открыть приложение |
| Association (direction: Unspecified) | |
| Source: Public (Actor) Пользователь | Target: Public (UseCase) Запустить процесс конвертации файла |

**Загрузить файл RuleML**

UseCase in package 'Use Cases'

Загрузить файл RuleML

Version 1.0 Phase 1.0 Proposed

worker created on 20.11.2022. Last modified 28.11.2022

| ASSOCIATIONS | |
| --- | --- |
| Association (direction: Unspecified) | |
| Source: Public (Actor) Пользователь | Target: Public (UseCase) Загрузить файл RuleML |

Листинг программы

**Класс Form1**

public partial class Form1 : Form

{

string filename; //Путь к файлу

string pathToDid; //Путь к директории

string ekbText = "";

//Листы объектов классов и ассоциаций

List<RuleMl> ruleMl;

List<Association> associations;

bool status = false;

public Form1()

{

InitializeComponent();

openFileDialog1.Filter = "Файл ruleml (\*.ruleml)|\*.ruleml|All files(\*.\*)|\*.\*"; //Инициализация фильтра

convertBtn.Enabled = false;

ruleMl = new List<RuleMl>();

associations = new List<Association>();

}

private void loadFileBtn\_Click(object sender, EventArgs e) //Слушитель кнопки выбора файла

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

{

fileLbl.Text = "Ошибка загрузки файла!";

fileLbl.ForeColor = Color.Red;

filename = null;

}

else

{

// получаем выбранный файл

filename = openFileDialog1.FileName;

fileLbl.Text = "Файл успешно загружен";

fileLbl.ForeColor = Color.Green;

}

checkToReady();

}

private void checkToReady()//Метод проверки загрузки файла и выбора директории

{

if (filename != null && pathToDid != null)

{

convertBtn.Enabled = true;

}

else

{

convertBtn.Enabled = false;

}

}

private void saveFileBtn\_Click(object sender, EventArgs e) //Слушатель кнопки выбора директории

{

FolderBrowserDialog FBD = new FolderBrowserDialog();

FBD.ShowNewFolderButton = false;

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

{

pathLbl.Text = "Директория для сохранения успешно выбрана!";

pathLbl.ForeColor = Color.Green;

pathToDid = FBD.SelectedPath;

}

else

{

pathLbl.Text = "Ошибка выбора \r\nдиректории для сохранения";

pathToDid = null;

pathLbl.ForeColor = Color.Red;

}

checkToReady();

}

private void convertBtn\_Click(object sender, EventArgs e) //Слушатель кнопки конвертации

{

associations.Clear();

ruleMl.Clear();

parseTheFile();

convertToEKB();

if (status)

{

saveToEkb();

}

else

{

MessageBox.Show("Проверьте файл для конвертации.", "Произошла ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void parseTheFile() //Метод парсинга файла

{

try

{

string attr = "";

string func = "";

string start = "";

string end = "";

string typeText = "";

List<(string, string)> Attribute;

filename = openFileDialog1.FileName;

XmlDocument xDoc = new XmlDocument();

xDoc.Load(filename);

XmlElement xRoot = xDoc.DocumentElement;

if (xRoot != null) //Проверка корневой структуры

{

//Разбор файла

var child = xRoot.ChildNodes;

var assert = child.Item(2);

if(assert == null)

{

status = false;

return;

}

var imp = assert.FirstChild;

var \_head = imp.FirstChild;

var atoms = \_head.ChildNodes;

for (int i = 0; i < atoms.Count; i++) //Разбор классов

{

Attribute = new List<(string, string)>();

var inside = atoms[i];

var atom = inside.ChildNodes;

var op = atom.Item(0);

var className = op.ChildNodes.Item(0).InnerText;

for (int j = 0; j < atom.Count; j++) //Разбор переменных

{

var temp = atom[j];

if (temp.Attributes["type"] != null)

{

typeText = temp.Attributes["type"].Value;

attr = temp.FirstChild.InnerText;

Attribute.Add((typeText, attr));

}

}

ruleMl.Add(new RuleMl(className, Attribute)); //Компановка классов

attr = "";

func = "";

}

var bodyChilds = imp.ChildNodes;

var body = bodyChilds.Item(1);

if(body == null){

status = false;

return;

}

var bodyAtoms = body.ChildNodes;

for (int i = 0; i < bodyAtoms.Count; i++) //Разбор связей

{

Attribute = new List<(string, string)>();

var inside = bodyAtoms[i];

var atom = inside.ChildNodes;

var op = atom.Item(0);

var className = op.ChildNodes.Item(0).InnerText;

start = inside.ChildNodes.Item(1).InnerText;

end = inside.ChildNodes.Item(2).InnerText;

associations.Add(new Association(className, start, end)); //Компановка связей

}

Console.WriteLine("efefefef");

}

status = true;

}

catch (Exception ex)

{

status= false;

}

}

private void saveToEkb() //Сохранение готового файла

{

string path = $"{pathToDid}/file.ekb";

System.IO.File.WriteAllText(path, ekbText);

MessageBox.Show("Файл сохранен");

}

private void convertToEKB() //Создание файла структуры ekb

{

//Заполнение шапки

Random random = new Random();

string id = $"{random.Next(1000000000)}{random.Next(100)}";

string header = $"\r\n<Structure>"

+ $"\r\n<KnowledgeBase>"

+ $"\r\n<ID>{id}</ID>"

+ $"\r\n<Name>База знаний 1</Name>"

+ $"\r\n<ShortName>Baza-znaniy-1</ShortName>"

+ $"\r\n<Kind>0</Kind>"

+ $"\r\n<Description></Description>"

+ $"\r\n<Vars/>"

+ $"\r\n<Templates>";

string templates = "";

string slots = "";

string templateEnd = "";

string templatesEnd = "";

string grules = "";

int l = 0;

ekbText = header;

int count = 1;

while (l < ruleMl.Count) //Заполнение классов

{

string idTempStr = "";

if (count < 10)

{

idTempStr = "";

idTempStr = $"00{count}";

}

if (count >= 10)

{

idTempStr = "";

idTempStr = $"0{count}";

}

templates = $"\r\n<Template>" +

$"\r\n<ID>T{idTempStr}</ID>" +

$"\r\n<Name>{ruleMl[l].ClassName}</Name>" +

$"\r\n<ShortName>{ruleMl[l].ClassName}</ShortName>" +

$"\r\n<Description>Описание шаблона T{idTempStr}</Description>" +

$"\r\n<PackageName></PackageName>" +

$"\r\n<RootPackageName></RootPackageName>" +

$"\r\n<DrawParams>xT{idTempStr}=15" +

$"\r\nyT{idTempStr}=15" +

$"\r\nw=265" +

$"\r\nh=65" +

$"\r\n</DrawParams>" +

$"\r\n<Slots>";

int p = 0;

ekbText += templates;

while (p < ruleMl[l].Attribute.Count) //Заполнение переменных класса

{

slots = $"\r\n<Slot>" +

$"\r\n<Name>{ruleMl[l].Attribute[p].Item2}</Name>" +

$"\r\n<ShortName>{ruleMl[l].Attribute[p].Item2}</ShortName>" +

$"\r\n<Description>{ruleMl[l].Attribute[p].Item2}</Description>" +

$"\r\n<Value></Value>" +

$"\r\n<DataType>{ruleMl[l].Attribute[p].Item1}</DataType>" +

$"\r\n<Constraint></Constraint>" +

$"\r\n</Slot>";

ekbText += slots;

p++;

}

templateEnd = $"\r\n</Slots>" +

"\r\n</Template>";

ekbText += templateEnd;

l++;

count++;

}

templatesEnd = templatesEnd = $"\r\n</Templates>";

ekbText += templatesEnd;

string facts = "\r\n<Facts/>" +

"\r\n<GRules>";

ekbText += facts;

int q = 0;

count = 1;

while (q < associations.Count) //Заполнение связей

{

string idTempStr = "";

if (count < 10)

{

idTempStr = "";

idTempStr = $"00{count}";

}

if (count >= 10)

{

idTempStr = "";

idTempStr = $"0{count}";

}

grules =

$"\r\n<GRule>" +

$"\r\n<ID>G{idTempStr}</ID>" +

$"\r\n<Name>{associations[q].AssotionName}</Name>" +

$"\r\n<ShortName>{associations[q].AssotionName}</ShortName>" +

$"\r\n<Description>{associations[q].AssotionName}</Description>" +

$"\r\n<PackageName></PackageName>" +

$"\r\n<RootPackageName></RootPackageName>" +

$"\r\n<DrawParams>xG{idTempStr}=26" +

$"\r\nyG{idTempStr}=105" +

$"\r\nw=170" +

$"\r\nh=34" +

$"\r\n</DrawParams>" +

$"\r\n<Conditions>" +

$"\r\n<C0>{associations[q].SourceName}</C0>" +

$"\r\n</Conditions>" +

$"\r\n<Actions>" +

$"\r\n<A0>{associations[q].TargetName}</A0>" +

$"\r\n</Actions>" +

$"\r\n</GRule>";

ekbText += grules;

q++;

count++;

}

//Заполнение конца файла

string ekbEnd = $"\r\n</GRules>" +

$"\r\n<Rules/>" +

$"\r\n<Functions/>" +

$"\r\n<Tasks/>" +

$"\r\n<FScales/>" +

$"\r\n<TempPackageList/>" +

$"\r\n<FactPackageList/>" +

$"\r\n<RulePackageList/>" +

$"\r\n<GRulePackageList/>" +

$"\r\n</KnowledgeBase>" +

$"\r\n</Structure>";

ekbText += ekbEnd;

}

**Класс RuleMl**

internal class RuleMl

{

public string ClassName { get; set; }

public List<(string, string)> Attribute { get; set; }

public RuleMl(string className, List<(string, string)> attribute)

{

ClassName = className;

Attribute = attribute;

}

}

**Класс Association**

internal class Association

{

public string AssotionName { get; set; }

public string SourceName { get; set; }

public string TargetName { get; set; }

public Association(string assotionName, string sourceName, string targetName)

{

AssotionName = assotionName;

SourceName = sourceName;

TargetName = targetName;

}

}