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 | | |
|  | 1.007.00.00 - ПЗ |  |

обозначение документа

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ИСМб-19-1 |  |  |  | Ю.А. Михейко |
|  |  | 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 | Михейко Ю.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 SWRL в файл формата EKB (согласно варианту 7). | | |

|  |
| --- |
| 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](#_Toc121483573)

[Введение 4](#_Toc121483574)

[1 Общая часть 5](#_Toc121483575)

[1.1 Описание предметной области 5](#_Toc121483576)

[1.2 Постановка задачи 6](#_Toc121483577)

[2 Специальная часть 7](#_Toc121483578)

[2.1 Техническое задание 7](#_Toc121483579)

[2.1.1 Введение 7](#_Toc121483580)

[2.1.2 Общие сведения 7](#_Toc121483581)

[2.1.3 Назначение и цели создания 8](#_Toc121483582)

[2.1.4 Характеристика объектов автоматизации 9](#_Toc121483583)

[2.1.5 Требования к программе или программному изделию 9](#_Toc121483584)

[2.1.6 Состав и содержание работ по созданию системы 10](#_Toc121483585)

[2.1.7 Порядок контроля и приемки 10](#_Toc121483586)

[2.1.8 Требования к программной документации 11](#_Toc121483587)

[2.1.9 Приложения 11](#_Toc121483588)

[2.2 Описание проекта в Trello 12](#_Toc121483589)

[2.3 Проектирование 12](#_Toc121483590)

[2.3.1 Диаграммы вариантов использования 12](#_Toc121483591)

[2.3.2 Диаграмма последовательностей 17](#_Toc121483592)

[2.3.3 Алгоритмическое обеспечение 18](#_Toc121483593)

[2.3.4 Диаграмма классов 19](#_Toc121483594)

[2.4 Программная реализация 20](#_Toc121483595)

[2.4.1 Автоматически сгенерированный код 20](#_Toc121483596)

[2.4.2 Описание интерфейса 23](#_Toc121483597)

[2.4.3 Тестирование 25](#_Toc121483598)

[Заключение 34](#_Toc121483599)

[Список использованных источников 35](#_Toc121483600)

[Приложение А. Фрагмент сгенерированной документации 36](#_Toc121483601)

[Приложение Б. Листинг программы 37](#_Toc121483603)

Введение

В рамках проекта осуществляется проектирование и разработка программного обеспечения, обеспечивающего автоматизацию процесса конвертации файлов структуры SWRL в EKB – формат редактора продукционных баз знаний Personal Knowledge Base Designer.

Актуальность разрабатываемого программного обеспечения определяется временными затратами в рамках ручного переноса проектов между системами и необходимости нахождения способа автоматизации данного процесса.

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 форматов файлов.

1 Общая часть

1.1 Описание предметной области

Программный компонент осуществляет процесс конвертации файлов SWRL формата редактора продукционных баз знаний EKB.

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



Рисунок 1 – Фрагмент структуры файла .ekb

Язык правил семантической сети (SWRL) — это предлагаемый язык для семантической сети, который можно использовать для выражения правил, а также логики, объединяя OWL DL или OWL Lite с подмножеством языка разметки правил.



Рисунок 2 – Структура файла SWRL

Дополнительно программный компонент предоставляет возможность выбора пользователем места хранения файла .xmi 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 SWRL 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 SWRL 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 Михейко Ю. 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 SWRL 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 SWRL;

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 SWRL;

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. | 12.09.2022-09.10.2022 | Реализация системы на уровне ядра. Описание методов и алгоритмов. |
| 3 | Разработка графического интерфейса приложения. | 10.10.2022-27.11.2022 | Описание базового графического интерфейса. |
| 4 | Тестирование программ­ного продукта и составление программной документации. | 28.11.2022- 11.12.2022 | Тесты. Про­граммный продукт. |
| 5 | Написание сопроводительной документации | 12.12.2022-25.12.2022 | 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

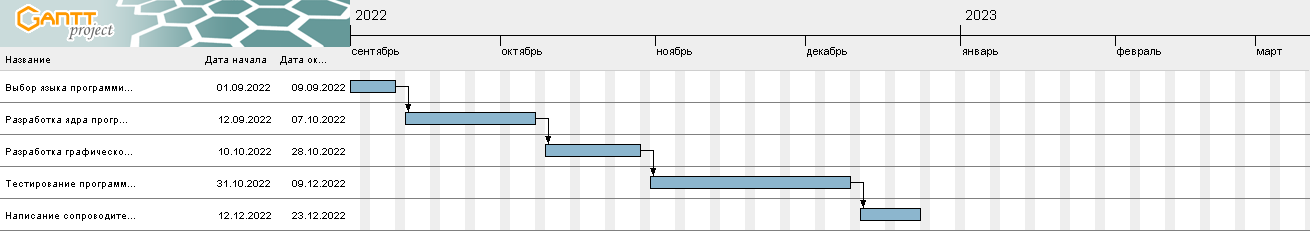


Рисунок 3 – Диаграмма Ганта

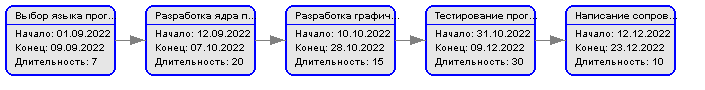


Рисунок 4 – Перт диаграмма

2.2 Описание проекта в Trello

В качестве программы для управления проектом был выбрана Trello. Для ведения прогресса выполнения курсового проекта было создано пять колонок с соответствующими заголовками. Колонка «Выполнено» содержит в себе выполненные и проверенные карточки. Колонка «Проверка» содержит карты с отправленными на проверку частями курсового проекта. Блок «Переделка» содержит отправленные на доработку задачи. Колонка «В работе» предназначена для хранения карт с пунктами текущей разработки. Последняя колонка создана для хранения карт с задачами для выполнения.

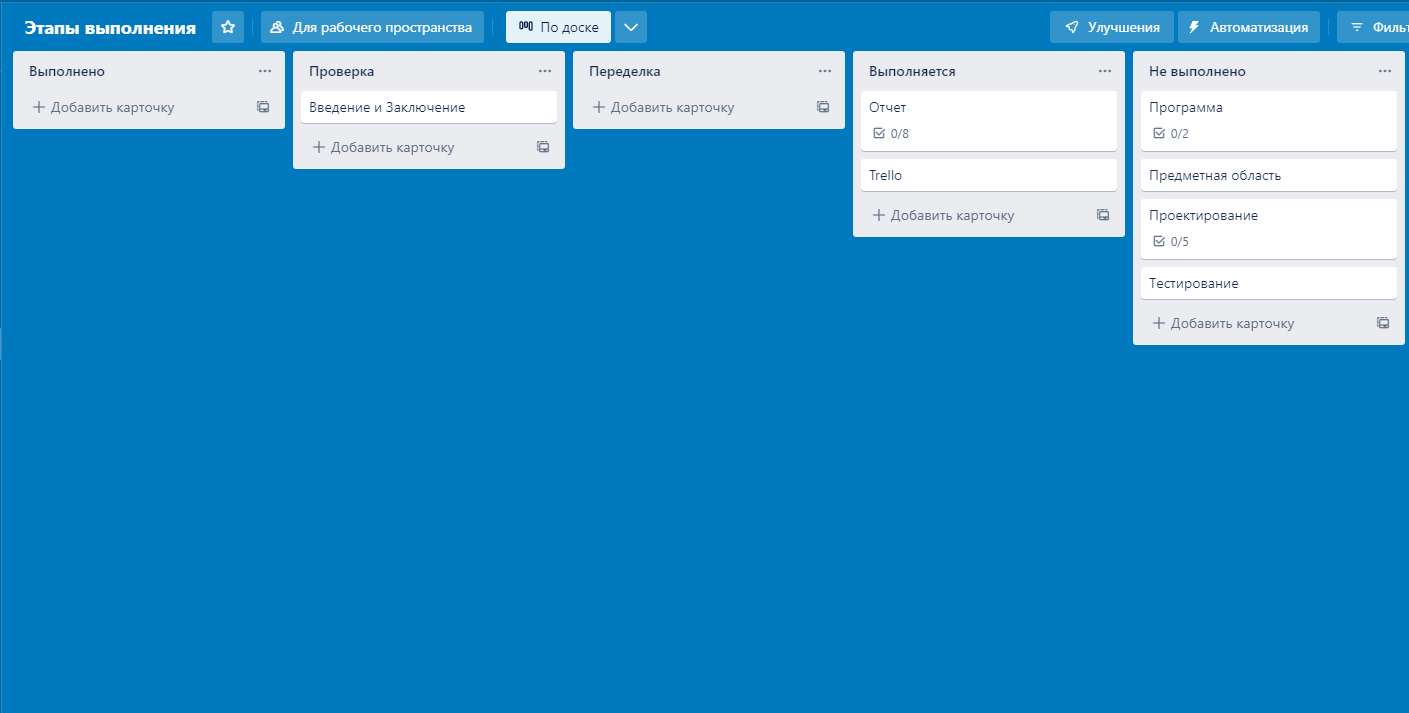


Рисунок 5 – Проект в программе Trello

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.

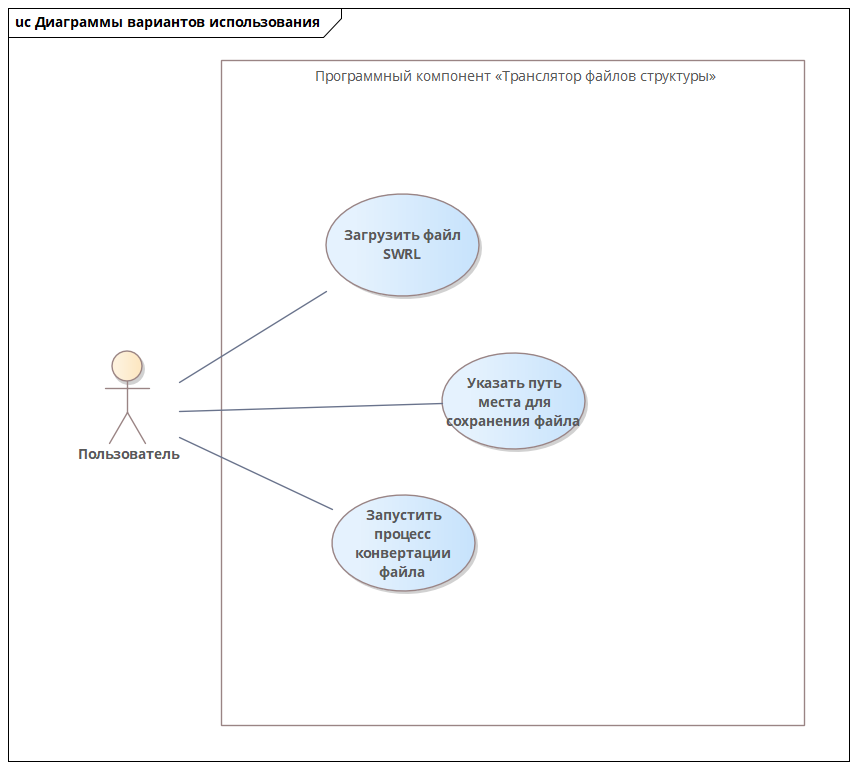


Рисунок 6 – Диаграмма вариантов использования программного компонента

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

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

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

**Активные субъекты**: пользователь

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

**Основной поток событий**:

* 1. Пользователю потребовалось конвертировать файл SWRL в файл приложения EKB.
  2. Пользователь запускает приложение посредством исполняемого EXE-файла.
  3. Пользователь запускает механизм загрузки файла SWRL.
  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 SWRL» 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 SWRL 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 SWRL».

**Специальные требования**: нет.

**Предусловия**: нет.

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

**Дополнительные замечания**: вариант использования «Запустить процесс конвертации файла» не имеет смысла выполнять без завершенных сценариев «Загрузить файл SWRL» 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 SWRL в файл приложения EKB.
  2. Пользователь запускает приложение посредством исполняемого EXE-файла.
  3. Пользователь запускает механизм загрузки файла SWRL.
  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 SWRL с полями файла.
  16. Данные из файла заносятся в соответствующие переменные объекта класса.
  17. Создается объект класса EKB.
  18. Поля класса EKB и SWRL 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 SWRL».
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 SWRL» и «Указать путь места для сохранения файла».

2.3.2 Диаграмма последовательностей

На основе диаграммы вариантов использования была составлена диаграмма последовательностей.

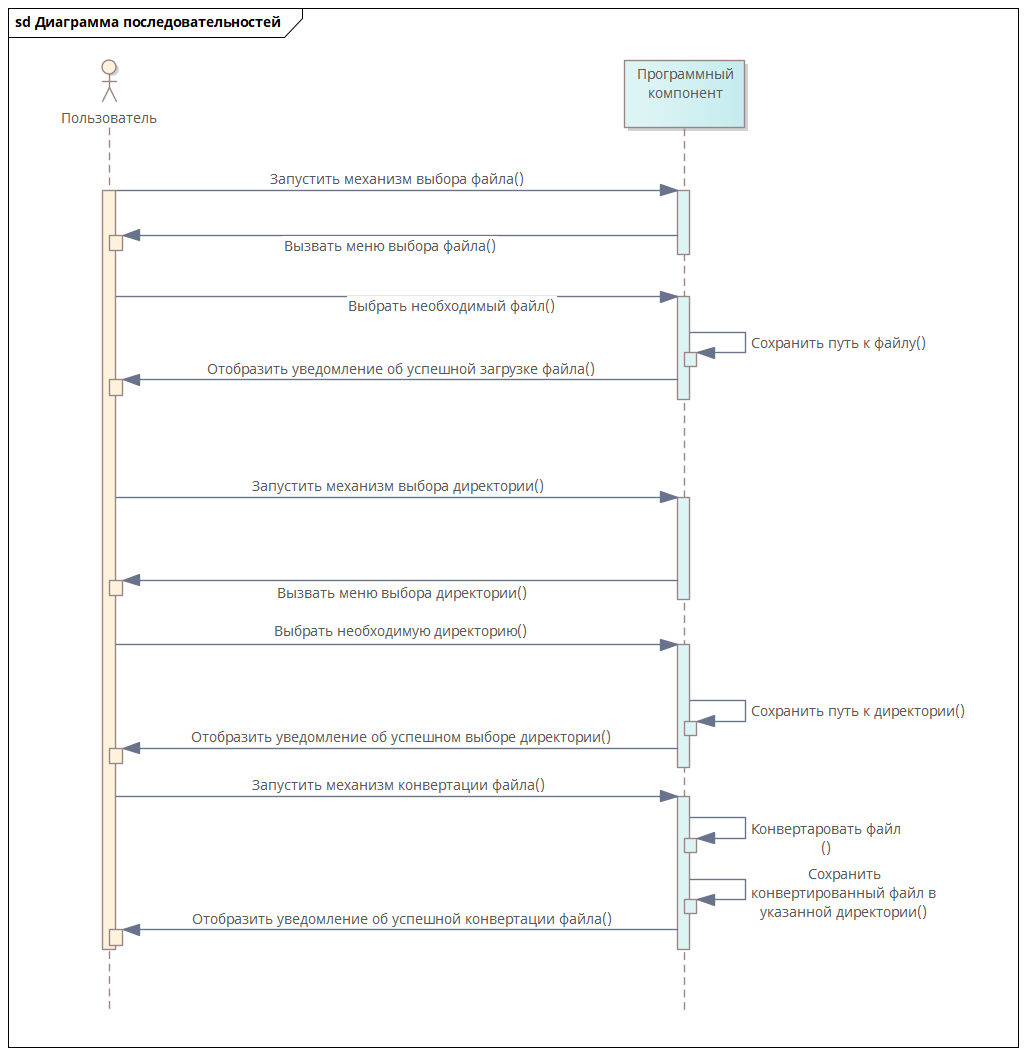


Рисунок 7 – Диаграмма последовательностей

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

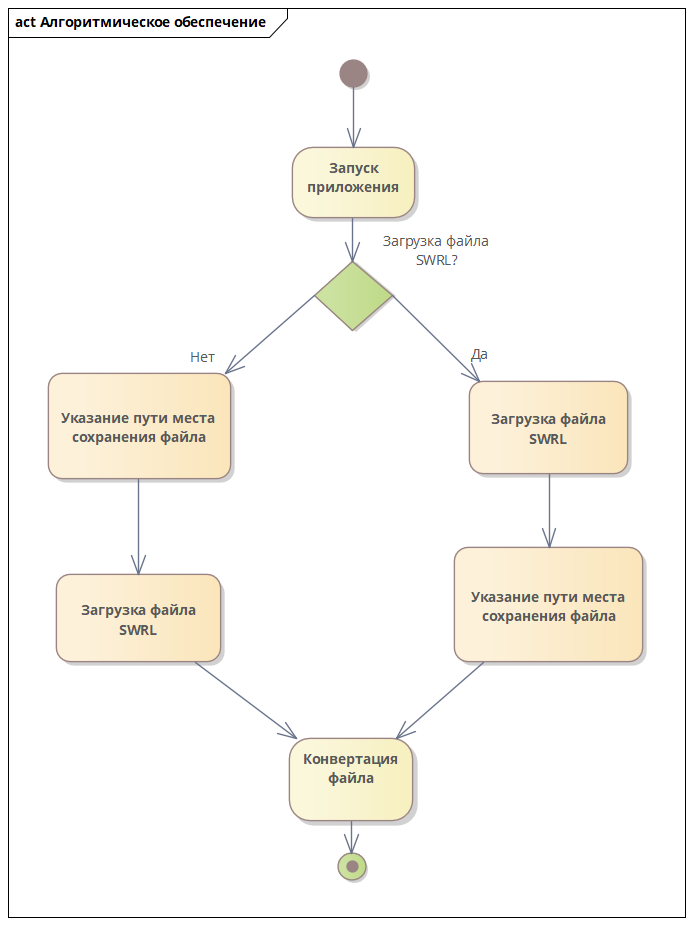


Рисунок 8 – Диаграмма деятельности

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.

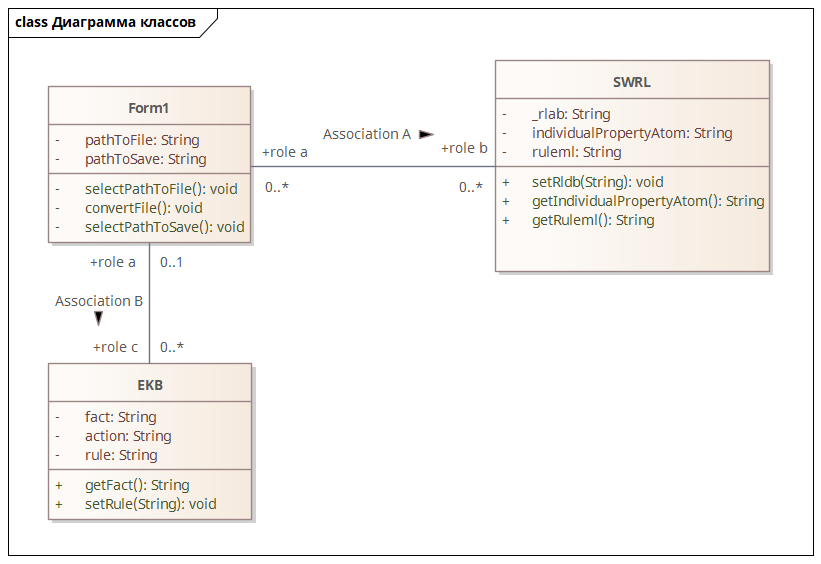


Рисунок 9 – Диаграмма классов

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 SWRL и EKB предназначены для хранения данных из файлов. При запуске процесса конвертации файл SWRL 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.

2.4 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 15:44:03

// 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 SWRL 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 15:44:17

// 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 SWRL**

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

// SWRL.cs

// Implementation of the Class SWRL

// Generated by Enterprise Architect

// Created on: 08-дек-2022 15:44:12

// Original author: worker

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

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

public class SWRL {

private String \_rlab;

private String individualPropertyAtom;

private String ruleml;

public SWRL(){

}

~SWRL(){

}

///

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

public void setRldb(String Parameter A){

}

///

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

public String getIndividualPropertyAtom(Parameter){

return "";

}

public String getRuleml(){

return "";

}

}//end SWRL

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 четыре прототипа возможных состояний программного компонента.

Первый фрагмент отображает первоначальное состояние приложения после запуска.

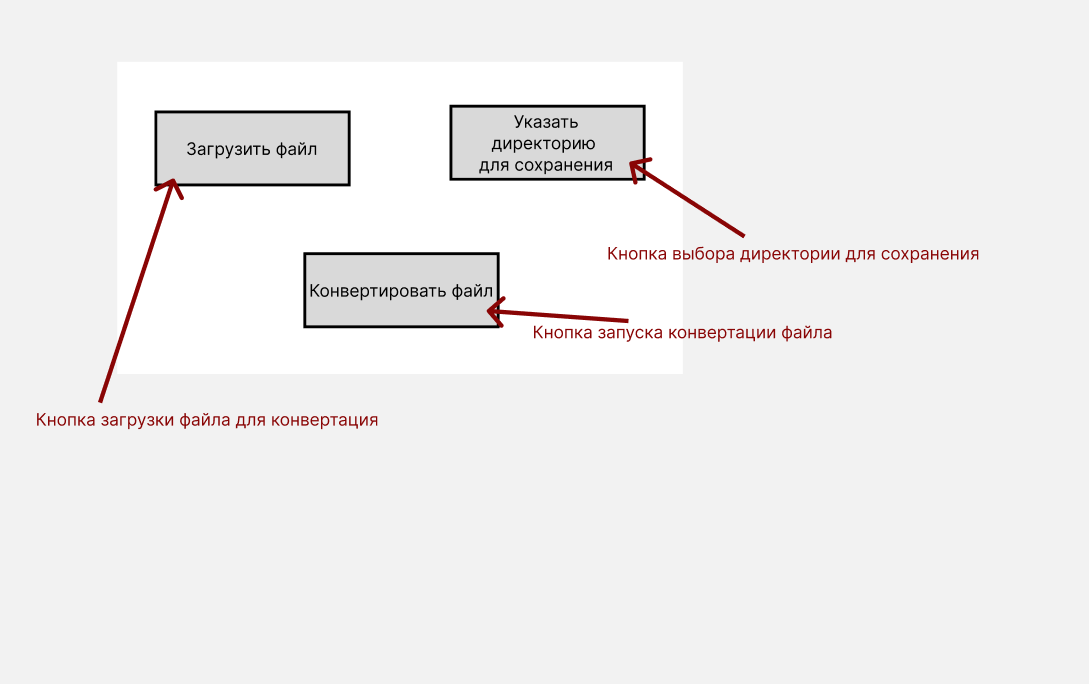


Рисунок 10 – Макет стартового окна

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

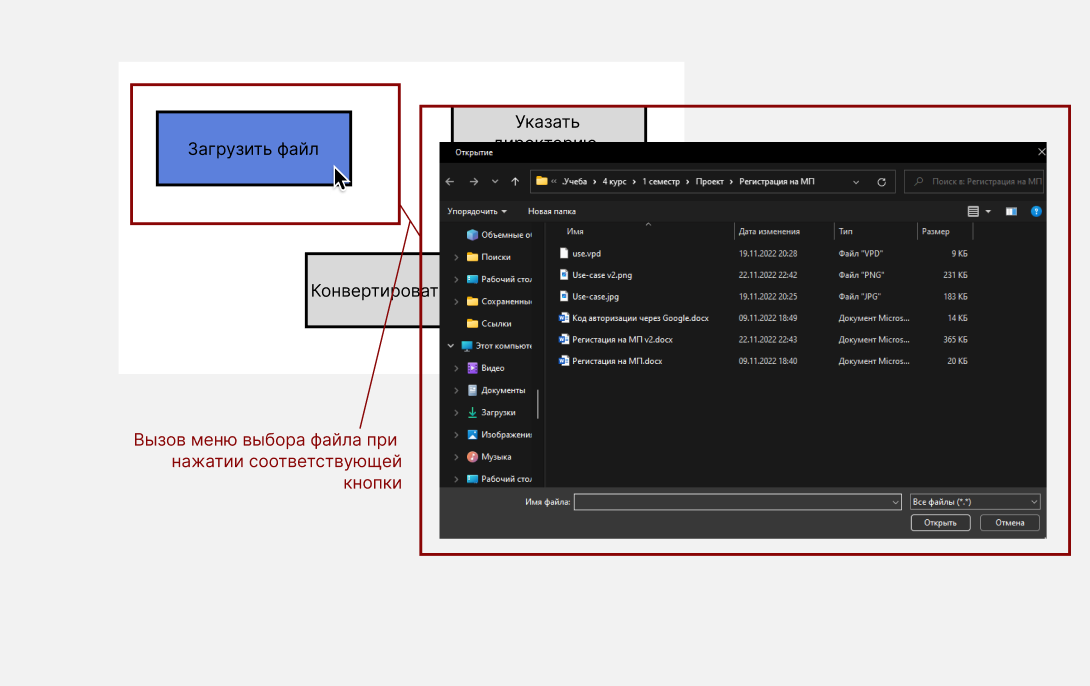


Рисунок 11 – Взаимодействие с кнопками выбора

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

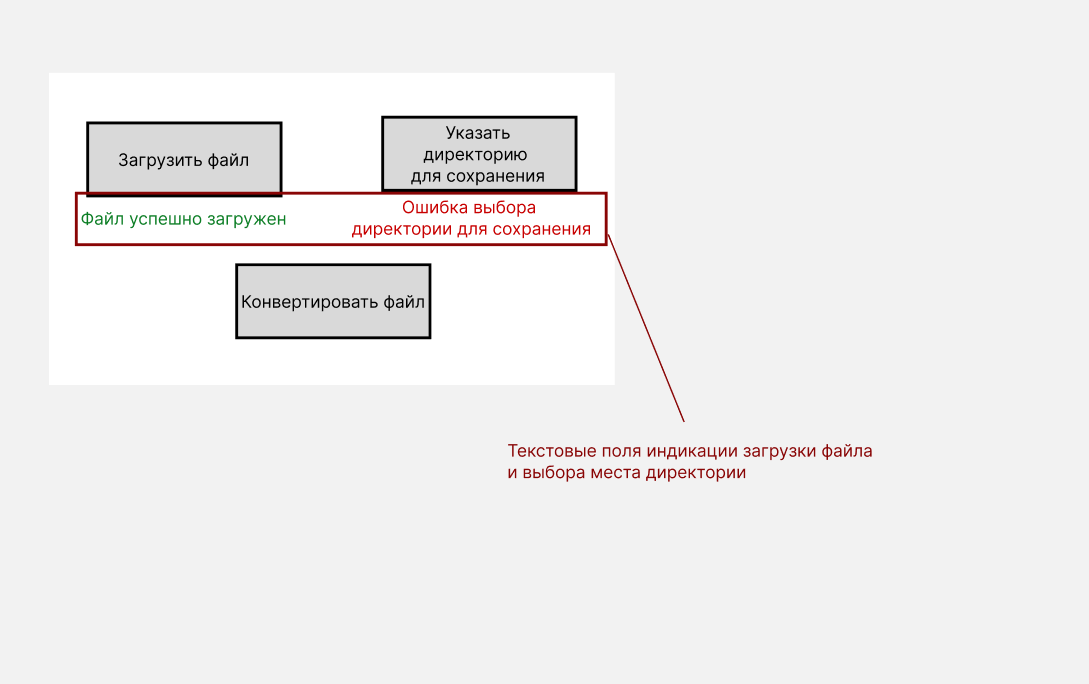


Рисунок 12 – Отображение полей состояния

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

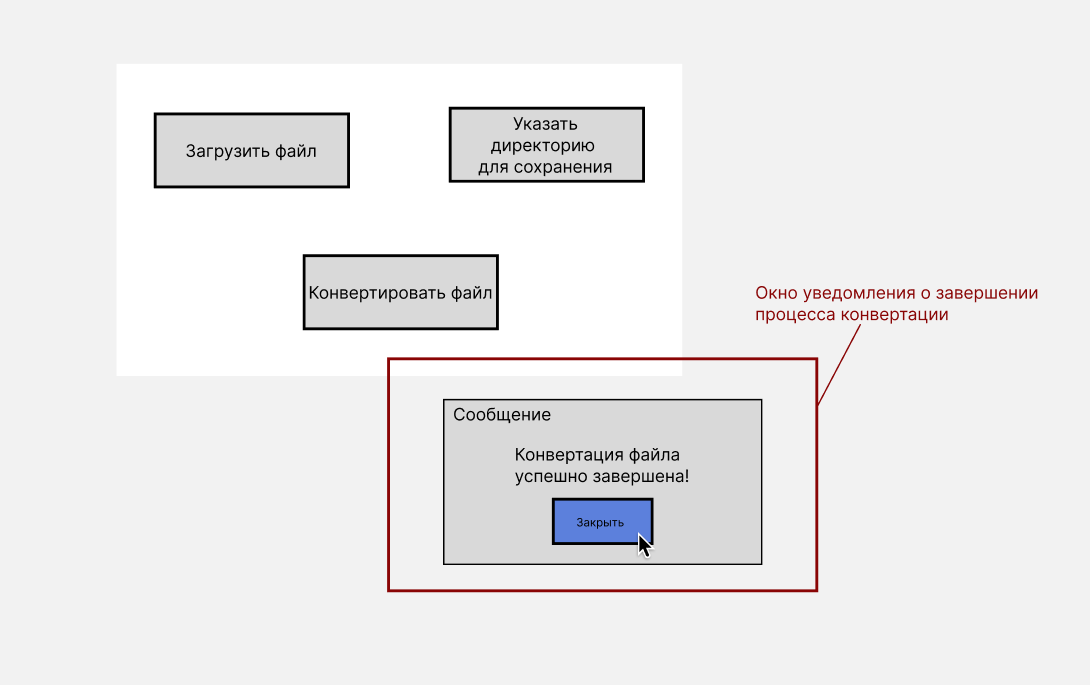


Рисунок 13 – Визуализация завершения процесса

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

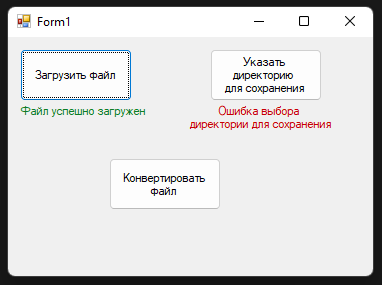


Рисунок 14 – Графический интерфейс в среде разработки

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. | Полученный файл с данными. | Конвертированный файл с расширением ekb. |
|  | Проверка работы компонента с файлом нестандартной конфигурации. | Файл нестандартной конфигурации. | Окно с уведомлением об ошибке конвертации. |
|  | Попытка конвертировать файл без указания директории для сохранения. | Указан путь к файлу с данными. | 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. | Пустой файл с данными. | Окно с уведомлением об ошибке конвертации. |
|  | Повторная конвертация загруженного файла без изменения директории. | Полученный файл с данными. | Перезаписанный файл с расширением ekb. |
|  | Проверка работы компонента с расширенным файлом. | Расширенный файл с данными. | Конвертированный файл с расширением ekb. |

Результаты тестирования представлены на рисунках 15-32.

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

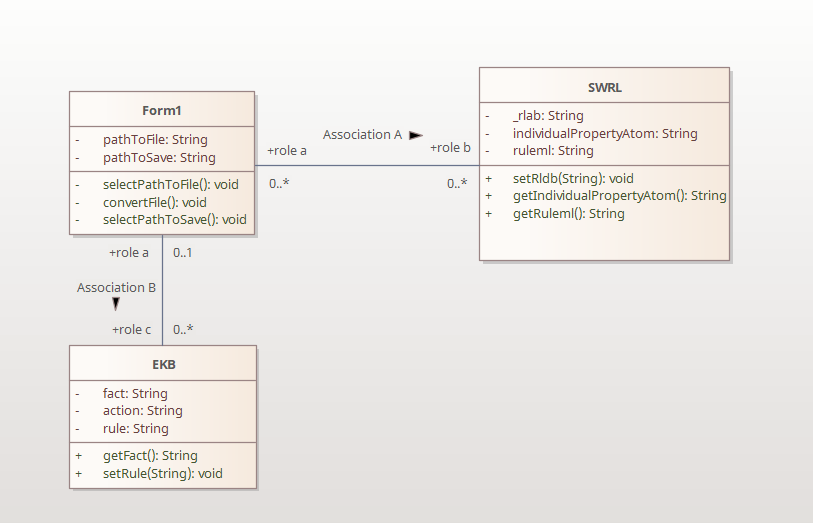


Рисунок 15 – Исходная диаграмма классов

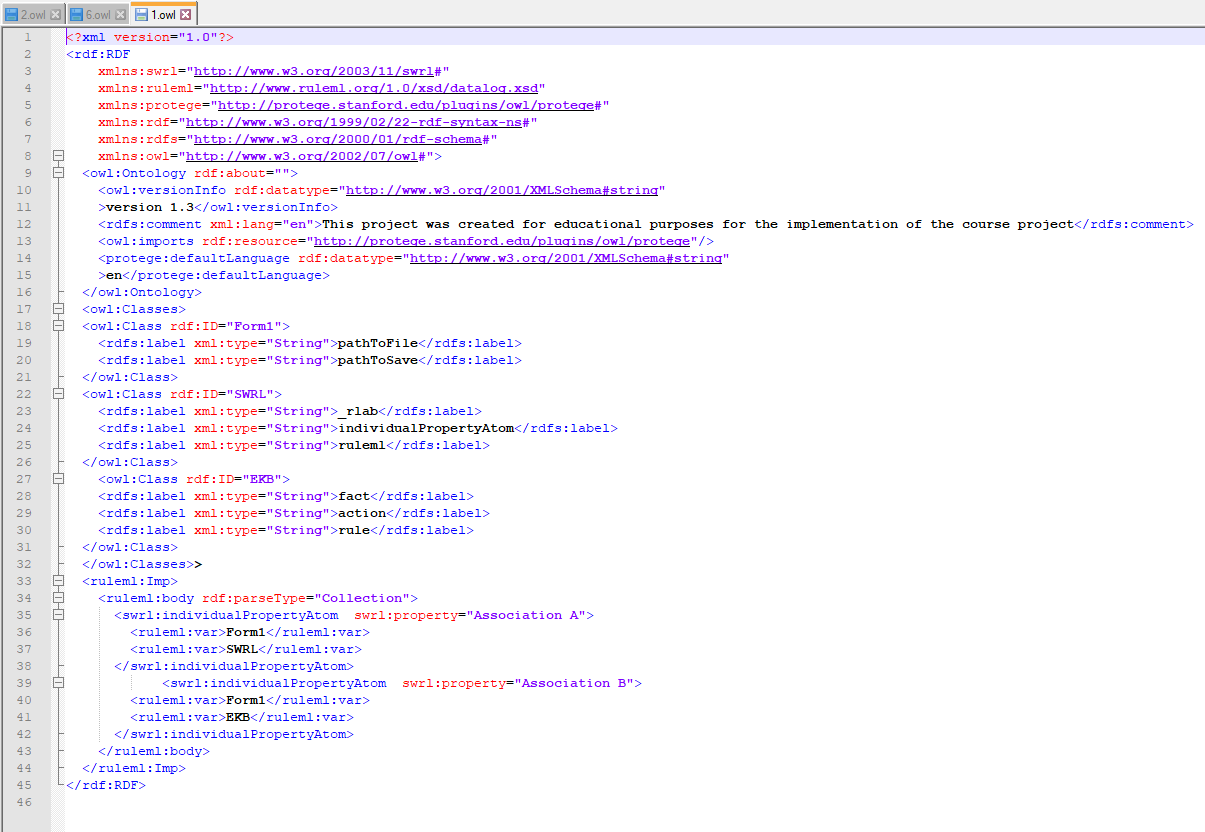


Рисунок 16 – Файл для конвертации

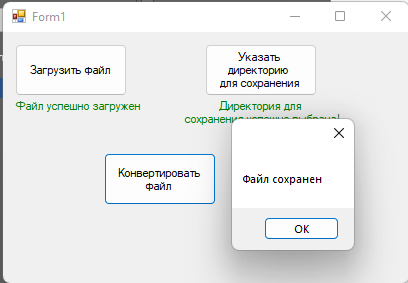


Рисунок 17 – Уведомление об успешной конвертации

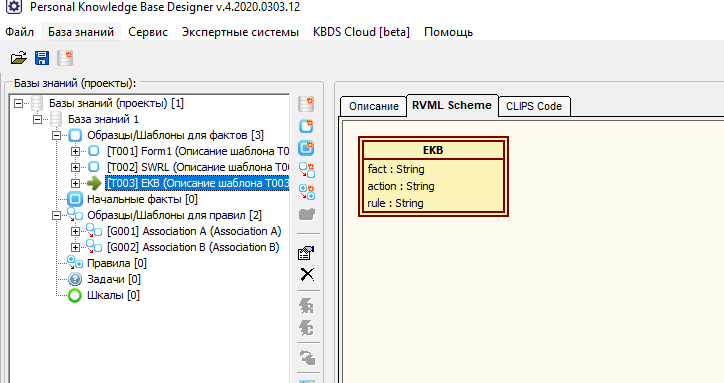


Рисунок 18 – Результат импортирования

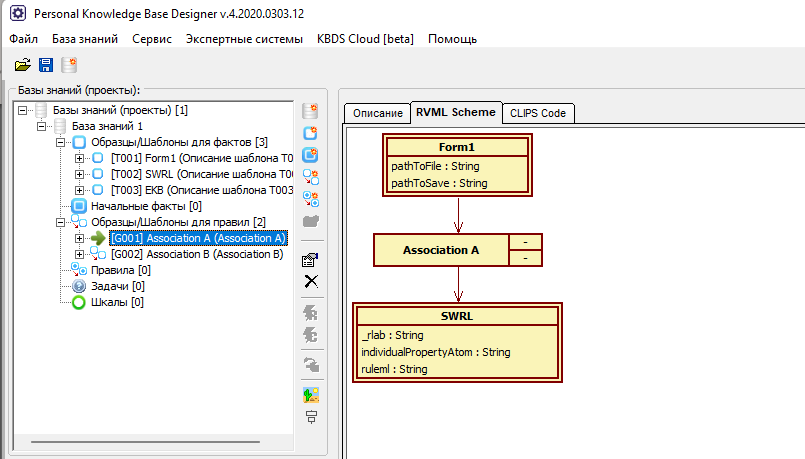


Рисунок 19 – Результат импортирования

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

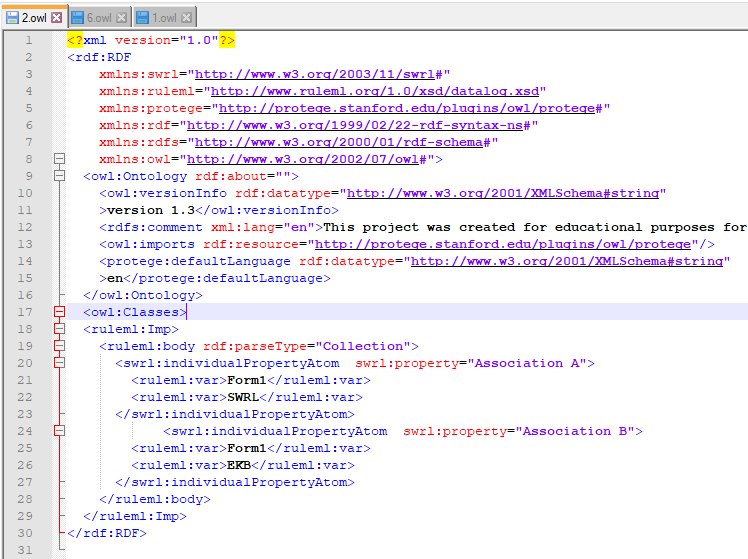


Рисунок 20 – Файл нестандартной конфигурации

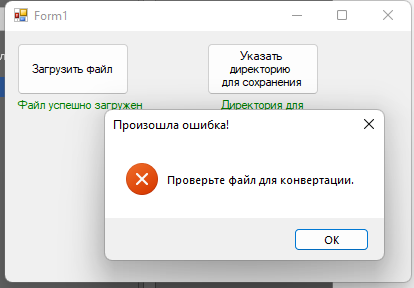


Рисунок 21 – Результат работы приложения

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

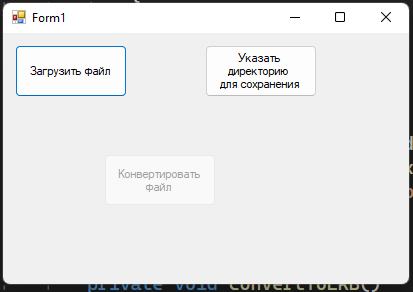


Рисунок 22 – Результат выполнения теста

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

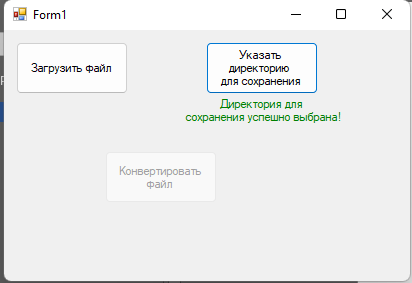


Рисунок 23 – Результат выполнения теста

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

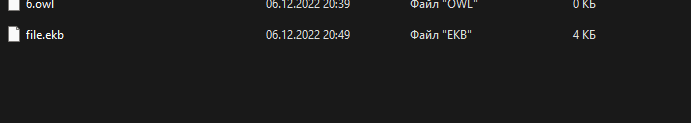


Рисунок 24 – Выбор нестандартного файла

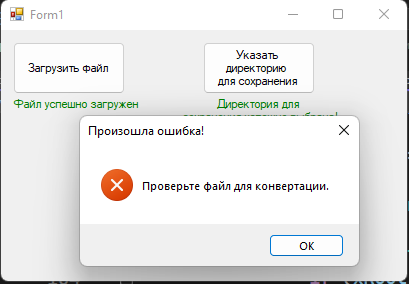


Рисунок 25 – Результат выполнения теста

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

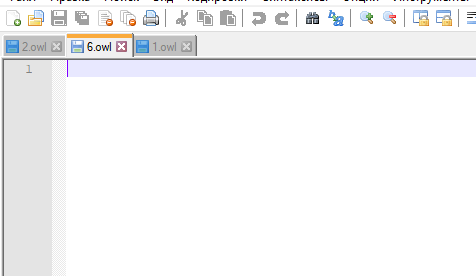


Рисунок 26 – Пустой файл

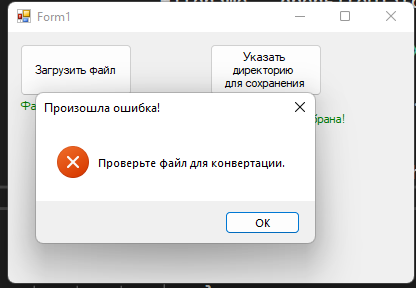


Рисунок 27 – Результат выполнения теста

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

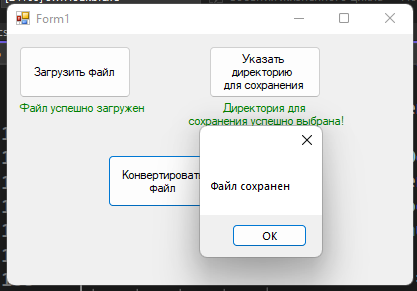


Рисунок 28 – Результат выполнения теста

**Тест №8 – Проверка работы компонента с расширенным файлом.**

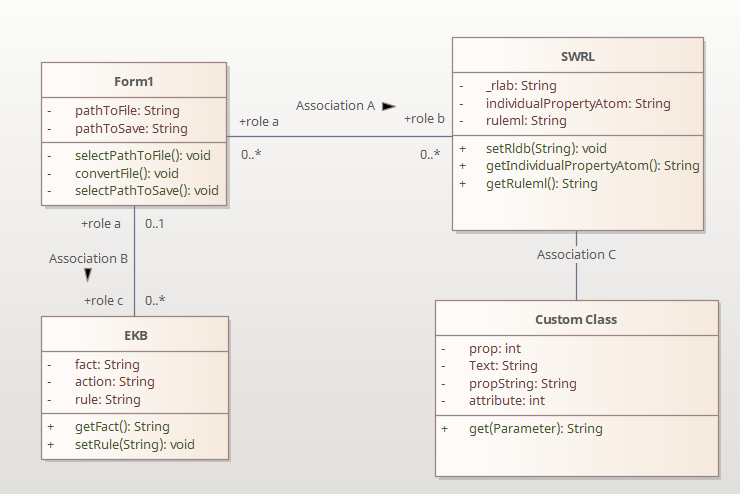


Рисунок 29 – Обновленная диаграмма

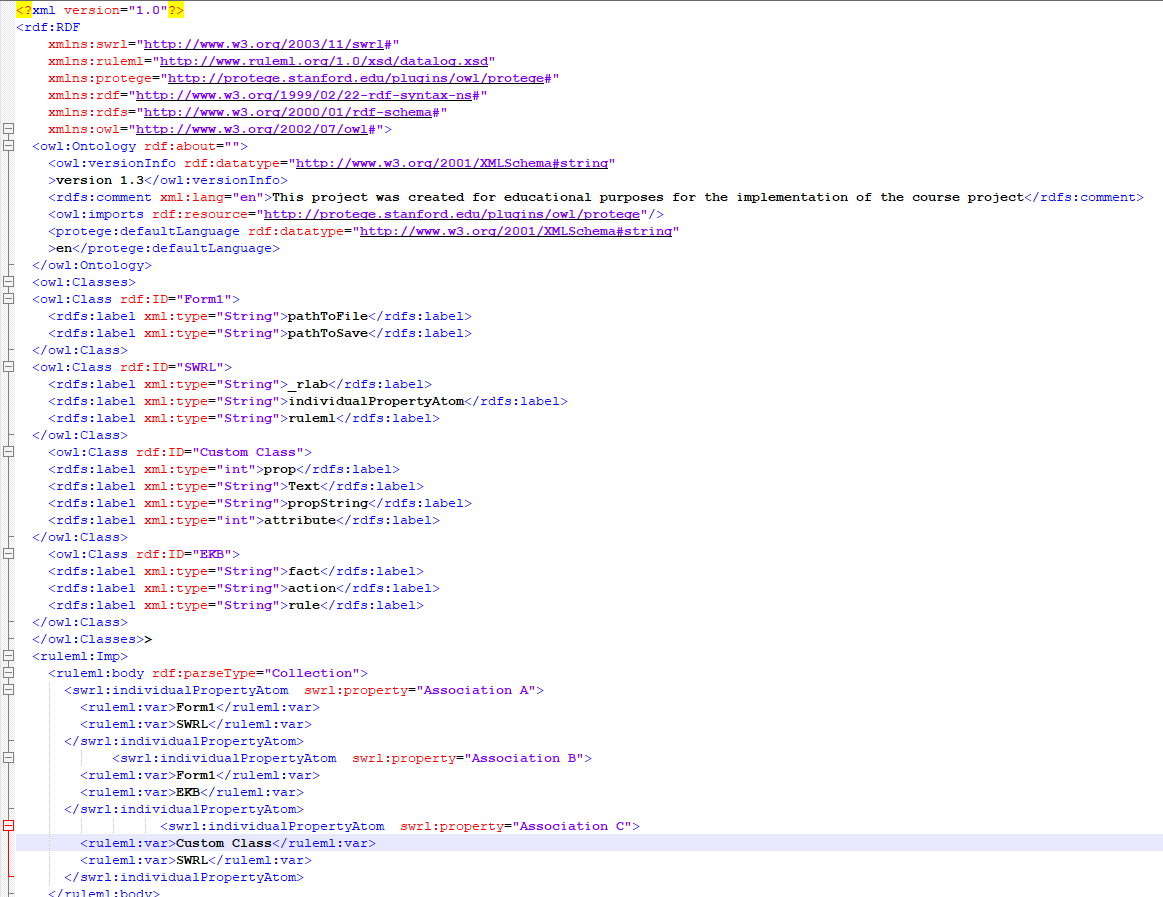


Рисунок 30 – Обновленный структурный файл

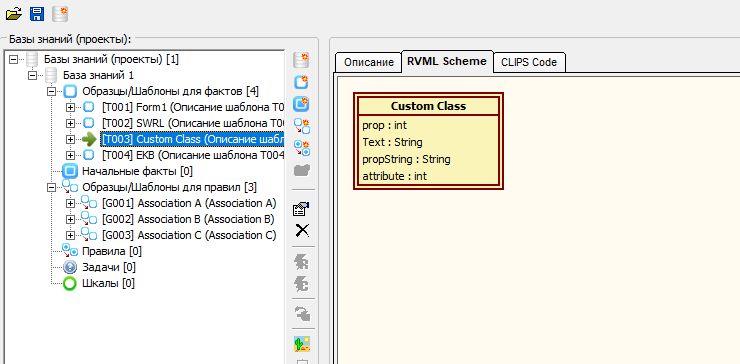


Рисунок 31 – Добавленный класс

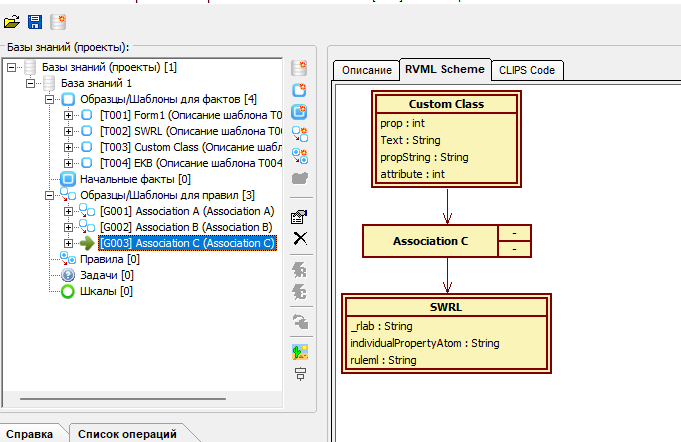


Рисунок 32 – Связь добавленного класса

Заключение

В ходе выполнения проекта было разработано программное обеспечение, обеспечивающее автоматизированное преобразование файлов SWRL в формат редактора баз знаний EKB.

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 'Диаграмма последовательностей - Copy'

Пользователь

Version 1.0 Phase 1.0 Proposed

worker created on 20.11.2022. Last modified 20.11.2022

| OUTGOING BEHAVIORAL RELATIONSHIPS |
| --- |
| Name: Запустить механизм выбора файла  Sequence from Пользователь to Программный компонент |
| Name: Запустить механизм выбора директории  Sequence from Пользователь to Программный компонент |
| Name: Выбрать необходимую директорию  Sequence from Пользователь to Программный компонент |
| Name: Запустить механизм конвертации файла  Sequence from Пользователь to Программный компонент |
| Name: Выбрать необходимый файл  Sequence from Пользователь to Программный компонент |

Приложение Б

Листинг программы

**Класс Form1**

public partial class Form1 : Form

{

string filename; //Путь к файлу

string pathToDid; //Путь к директории

string ekbText = "";

bool status = false;

//Листы объектов классов

List<SWRL> swrl;

List<Association> associations;

public Form1()

{

InitializeComponent();

openFileDialog1.Filter = "Файл owl (\*.owl)|\*.owl|All files(\*.\*)|\*.\*"; //Фильтр файлов

convertBtn.Enabled = false;

swrl = new List<SWRL>();

associations = new List<Association>();

}

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

{

FolderBrowserDialog FBD = new FolderBrowserDialog();

FBD.ShowNewFolderButton = false;

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

{

saveLbl.Text = "Директория для \r\nсохранения успешно выбрана!";

saveLbl.ForeColor = Color.Green;

pathToDid = FBD.SelectedPath;

}

else

{

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

pathToDid = null;

saveLbl.ForeColor = Color.Red;

}

checkToReady();

}

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

{

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

{

convertBtn.Enabled = true;

}

else

{

convertBtn.Enabled = false;

}

}

private void parseTheFile() //Метод парсинга файла

{

try

{

string attr = "";

string className = "";

string start = "";

string end = "";

string typeText = "";

string assocName = "";

List<(string, string)> Attribute;

filename = openFileDialog1.FileName;

XmlDocument xDoc = new XmlDocument();

xDoc.Load(filename);

XmlElement xRoot = xDoc.DocumentElement;

if (xRoot != null) //Проверка корня файла на пустоту

{

var child = xRoot.ChildNodes; //Начало парсинга

if (child.Item(1) == null)

{

status = false;

return;

}

var classes = child.Item(1).ChildNodes;

for (int i = 0; i < classes.Count; i ++) //Парсинг классов

{

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

var cl = classes[i];

className = cl.Attributes["rdf:ID"].Value;

var classChildren = cl.ChildNodes;

for(int j = 0; j < classChildren.Count; j++) //Парсинг переменных

{

var attrib = classChildren[j];

typeText = attrib.Attributes["xml:type"].Value;

attr = attrib.InnerText;

Attribute.Add((typeText, attr));

}

swrl.Add(new SWRL(className, Attribute));

}

var imp = child.Item(3).ChildNodes;

var assocChild = imp.Item(0).ChildNodes;

for(int i = 0; i < assocChild.Count; i++) //Парсинг связей классов

{

var assoc = assocChild[i];

assocName = assoc.Attributes["swrl:property"].Value;

var nodes = assoc.ChildNodes;

start = nodes.Item(0).InnerText;

end = nodes.Item(1).InnerText;

associations.Add(new Association(assocName, start, end));

}

Console.WriteLine("efefefef");

}

status = true;

}

catch (Exception ex)

{

status = false;

return;

}

}

private void saveToEkb() //Метод сохранения файла

{

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

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

MessageBox.Show("Файл сохранен");

}

private void convertToEKB() //Метод преобразования файла в формат EKB

{

//Заполнение шапки структуры

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 < swrl.Count) //Заполнение конструкций 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>{swrl[l].ClassName}</Name>" +

$"\r\n<ShortName>{swrl[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 < swrl[l].Attribute.Count) //Заполнение переменных класса

{

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

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

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

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

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

$"\r\n<DataType>{swrl[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;

}

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

{

associations.Clear();

swrl.Clear();

parseTheFile();

convertToEKB();

if (status)

{

saveToEkb();

}

else

{

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

}

}

}

**Класс SWRL**

internal class SWRL

{

public string ClassName { get; set; }

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

public SWRL(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;

}

}