

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программные средства выявления вероятных
последовательностей событий на основе
интеллектуального анализа журнала событий**

Студент _____ Коршунова Ксения Сергеевна
(подпись, дата) (ФИО полностью)

Руководитель _____ Шибанов С.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Коршунова Ксения Сергеевна гр. 19ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программные средства выявления вероятных последовательностей
событий на основе интеллектуального анализа журнала
событий

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы _____ к.т.н., доцент кафедры «МО и ПЭВМ» Шибанов С.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение: программные средства предназначены для выявления вероятных
последовательностей событий на основе интеллектуального анализа журнала
событий

Средства выполнения: Операционная система Windows

Средства разработки: Visual Studio 2022, Framework 4.8, .NET

Язык программирования: C#

Технологии разработки: ООП, UML

5. Перечень подлежащих разработке вопросов

- 1) Анализ предметной области и постановка задачи на разработку
- 2) Сравнительный анализ алгоритмов и программных средств выявления вероятных последовательностей событий
- 3) Проектирование программных средств
- 4) Разработка программных средств
- 5) Контроль качества программных средств

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 100 листов, 35 рисунка, 29 таблиц, 15 использованных источников и 4 приложения.

ПОСЛЕДОВАТЕЛЬНОСТИ СОБЫТИЙ, СЕКВЕНЦИАЛЬНЫЙ АНАЛИЗ, ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ, ЖУРНАЛ СОБЫТИЙ.

Объектом исследования: программные средства для выявления вероятных последовательностей событий на основе интеллектуального анализа журналов событий.

Цель работы: реализовать программные средства для выявления вероятных последовательностей событий.

Технологии разработки: ООП, C#, Visual Studio, .Net Framework.

Задание: спроектировать и реализовать интерфейс пользовательского приложения, предварительную обработку журнала событий и алгоритмы секвенциального анализа.

Результат работы: спроектированы и разработаны программные средства для выявления вероятных последовательностей событий.

					ПГУ 09.03.02–8БР191.13ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата	Программные средства выявления вероятных последовательностей событий на основе интеллектуального анализа журнала событий Пояснительная записка			Лист	Лист	Листов	
Разраб.		Коршунова К.С.									
Провер.		Шибанов С.В.							4	100	
Реценз.								Группа 19ВИ1			
Н. Контр.		Такташкин Д.В.									
Утверд.		Козлов А.Ю.									

СОДЕРЖАНИЕ

Введение.....	7
1 Анализ предметной области и постановка задачи на разработку.....	9
1.1 Анализ предметной области разработки	9
1.1.1 Задача выявления вероятных последовательностей событий.....	9
1.1.2 Структура журнала событий.....	9
1.1.3 Общий сценарий программных средств.....	10
1.2 Постановка задачи на разработку программных средств.....	11
1.2.1 Цели разработки и назначение программных средств.....	11
1.2.2 Классы пользователей программных средств.....	12
1.2.3 Перечень автоматизируемых функций	12
1.3 Планирование и функционально-стоимостной анализ процесса разработки программных средств	13
2 Сравнительный анализ алгоритмов и программных средств выявления вероятных последовательностей событий.....	17
2.1 Обоснование выбора метода интеллектуального анализа для выявления вероятных последовательностей событий.....	17
2.2 Сравнительный анализ алгоритмов секвенциального анализа	18
2.3 Сравнительный анализ программных средств секвенциального анализа .	23
3 Проектирование программных средств.....	25
3.1 Функциональные требования программных средств.....	25
3.2 Архитектура программных средств	28
3.3 Обоснование выбора технологий и средств разработки.....	30
3.4 Требования к аппаратно-программному обеспечению.....	31
3.4.1 Требования к аппаратно-техническому обеспечению.....	31
3.4.2 Требования к системному программному обеспечению	31
3.4.3 Требования к прикладному программному обеспечению.....	31
4 Разработка программных средств	33
4.1 Разработка пользовательского интерфейса.....	33

	6
4.2 Разработка программных средств доступа к журналу событий	39
4.3 Разработка программных средств управления проектами	40
4.4 Разработка программных средств формирования вероятных последовательностей событий	43
4.4.1 Структура плагин библиотеки секвенциального анализа	43
4.4.2 Реализация алгоритма GSP	45
4.4.3 Реализация алгоритма PrefixSpan	46
4.4.4 Реализация алгоритма SPAM	47
5 Контроль качества программных средств	49
5.1 Планирование процесса контроля качества программных средств	49
5.1.1 Мероприятия по контролю качества программных средств	49
5.1.2 Обоснование выбора модели тестирования	49
5.1.3 Обоснование выбора метрик программного кода	51
5.2 Тестирование программных средств	52
5.2.1 Классы эквивалентности	52
5.2.2 Таблица решений	53
5.3 Оценка качества программного кода	55
Заключение	57
Список использованных источников	58
Приложение А. Глоссарий	60
Приложение Б. Сценарий вариантов использования	62
Приложение В. Исходный код программных средств	64
Приложение Г. Результаты тестирования	95

Введение

Практически все данные создаются и обрабатываются как непрерывные потоки событий. Накапливаются большие объемы данных, которые необходимо проанализировать для того, чтобы на их основе можно было сделать выводы. Для анализа таких размеров данных необходимо использовать Data Mining.

В настоящее время интеллектуальный анализ данных позволяет выявлять вероятные последовательности событий, которые поступают от наблюдаемых систем и фиксируются в журналах. Данное направление является актуальной задачей для исследования, так как может быть применено во всех сферах жизнедеятельности человека. Возможность нахождения вероятных цепочек событий позволяет определять причины возникновения и прогнозировать появление нормальных и аварийных ситуаций, которые находятся под наблюдением.

Данный программный продукт должен удовлетворять следующим критериям:

- позволять эффективно выявлять закономерности возникновения событий;
- иметь возможность загрузки журнала событий как из базы данных, так и из файла;
- иметь возможность работы с журналом событий;
- сохранение выявленных последовательностей в файл;
- иметь возможность работы с проектом;
- иметь возможность редактирования параметров секвенциального анализа.

Задачей выпускной квалификационной работы является анализ требований, проектирование и реализация системы, выполняющей поставленную задачу, и контроль качества данного приложения. Также необходимо провести сравнительный анализ алгоритмов и программных

средств секвенциального анализа. В качестве результата работы будет представлена спроектированная модель программного средства, которая позволяет продемонстрировать функционал, однако не является конечной версией продукта.

1 Анализ предметной области и постановка задачи на разработку

1.1 Анализ предметной области разработки

1.1.1 Задача выявления вероятных последовательностей событий

Выявление вероятных последовательностей событий, обработанных в системах и зафиксированных в журналах событий, является актуальной задачей в настоящее время. Данная задача может позволить проанализировать причины возникновения и прогнозировать появление, как нештатных, так и штатных ситуаций. События разделяются на 2 вида: простые и составные. К простым событиям относят любой факт, зафиксированный в системе (например, сообщение об ошибке). Сложное событие, является комбинацией простых (или других сложных) событий, рекурсивно связанных между собой операциями алгебры событий [1].

Первоначально для анализа и поиска вероятных цепочек событий применялись ассоциативные правила. Однако ассоциативные правила учитывают только факты совместного появления событий и не учитывают временной аспект.

В дальнейшем для выявления закономерностей событий стали применять секвенциальный анализ. В настоящее время данный метод является наиболее эффективным способом нахождения вероятных последовательностей событий. Данные алгоритмы предназначены для формирования вероятных цепочек событий, связанных во времени.

1.1.2 Структура журнала событий

Журнал событий – это документ, в котором хранятся записи о событиях. Данные журнала событий являются информацией для проведения секвенциального анализа.

В зависимости от целей и требований структура журнала событий может быть различна. Основными элементами структуры являются:

- 1) информация о дате и времени, когда произошло событие;
- 2) идентификатор события – это уникальный ключ, который используется для идентификации события;
- 3) наименование и описание события.

Журнал событий может быть использован для анализа работы различных систем, а также для обнаружения проблем.

1.1.3 Общий сценарий программных средств

Диаграмма деятельности используется для более детального отображения потока действий в системе. Для отображения сценария была разработана данная диаграмма, которая представлена на рисунке 1.



Рисунок 1 – Общий сценарий программных средств

В начале работы создается проект, далее необходимо загрузить журнал событий для дальнейшего проведения секвенциального анализа. В процессе загрузки присутствует возможность обработки данных журнала. Для того чтобы приступить к анализу, загруженных данных, необходимо создать решение. Далее аналитик выбирает алгоритм для выявления вероятных последовательностей событий. После устанавливает необходимые параметры для анализа. В результате должен быть выведен найденный набор вероятных цепочек событий, который можно сохранить в файл. В случае необходимости аналитик может изменить алгоритм и его параметры.

1.2 Постановка задачи на разработку программных средств

1.2.1 Цели разработки и назначение программных средств

Целью разработки является комплекс приложений, направленных на поиск вероятных последовательностей событий, выявление шаблонов составных событий и их анализ. В рамках данной преддипломной практики рассматривается задача создания программного средства (ПС), позволяющего выявлять вероятные последовательности событий на основе журнала событий, с помощью алгоритмов секвенциального анализа. Данное ПС должно обладать следующими возможностями:

- возможность загрузки журнала событий из базы данных (БД) и файла;
- возможность проведения секвенциального анализа несколькими алгоритмами;
- возможность изменения параметров алгоритмов анализа;
- возможность просмотра и сохранения выявленных последовательностей событий.

А также иметь простой и удобный интерфейс, позволяющий получить и проанализировать результат работы алгоритма.

1.2.2 Классы пользователей программных средств

Основным классом пользователей является аналитик. В его задачи входит:

- сбор данных;
- обработка данных – подготовка к анализу, устранение повторов, группировка и упорядочивание, собранной информации;
- анализ данных;
- подведение итогов анализа;
- визуализация, проведенного анализа.

В рамках построения данного ПС аналитик выявляет вероятные последовательности событий, анализирует их и сохраняет получившиеся результаты.

1.2.3 Перечень автоматизируемых функций

Данное программное средство должно выполнять автоматизированные функции работы с журналом событий, секвенциального анализа и с проектом.

К функциям работы с журналом относятся:

- предварительная обработка и фильтрация журнала событий;
- выгрузка журнала событий из базы данных/файла.

К функциям секвенциального анализа относятся:

- определение параметров секвенциального анализа журнала событий;
- выбор алгоритма секвенциального анализа;
- просмотр выявленных цепочек событий;
- сохранение выявленных цепочек событий в файл.

К функциям работы с проектом относятся:

- создание проекта;
- сохранение проекта;
- открытие проекта.

Также должна быть предоставлена возможность редактирования параметров и выбор другого алгоритма при необходимости.

1.3 Планирование и функционально-стоимостной анализ процесса разработки программных средств

Функционально-стоимостной анализ (ФСА) – это метод, позволяющий оценить стоимость продукта (услуги). Проведение ФСА предоставляет возможность получить оценку бюджета благодаря правильному распределению средств, которые направлены на производство продукта.

Преимуществами ФСА является:

- более точное понимание стоимости продукции, позволяющее устанавливать верные цены на продукт, правильно вкладывать средства в исследовательскую работу и автоматизацию необходимых процессов;
- большая ясность в отношении выполняемых функций системы;
- позволяет выявить и сократить объем операций, не добавляющих полезности продукции.

Для анализа была рассмотрена средняя заработная плата разработчика в 2023 году. По данным зарплата программиста в месяц в Пензе составляет около 50 тыс. рублей. Продолжительность рабочей недели составляет не более 40 часов, т. е. в месяц количество часов может быть 170. В результате подсчетов было выявлено, что за час работы разработчик получает 294 рублей в час.

Была разработана контекстная диаграмма бизнес-процесса в нотации IDEF0 в программном средстве Ramus. На рисунке 2 представлена диаграмма бизнес-процесса «Выпускная квалификационная работа».

IDEF0 – это нотация графического моделирования, используемая для изображения функциональной модели, чтобы отобразить структуру и ее функции.

Контекстная диаграмма IDEF0 – диаграмма, на которой объект моделирования представлен одним блоком. Данная диаграмма содержит блок и стрелки, отображающие связи моделируемого объекта. Функциональные блоки

на диаграммах изображаются прямоугольниками, означающими функции или задачи.

Каждая сторона блока имеет определенное значение. Левая сторона блока предназначена для входов, верхняя – для управления, правая – для выходов, нижняя – для механизмов [2].

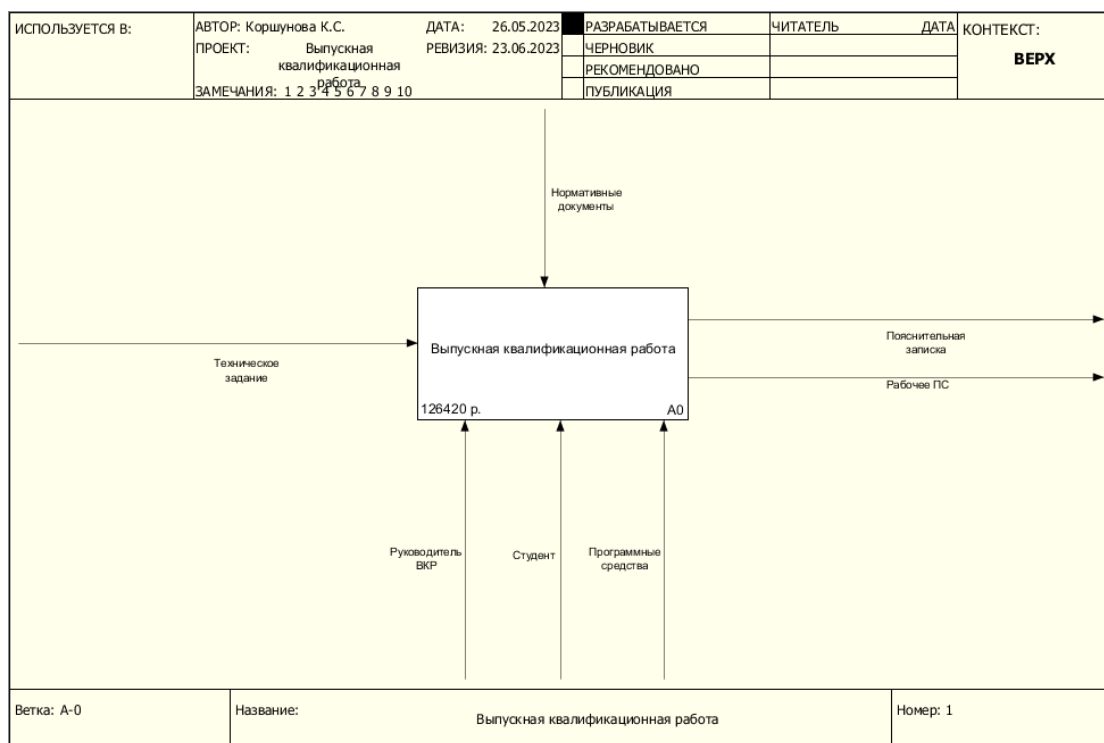


Рисунок 2 – Контекстная диаграмма процесса «Выпускная квалификационная работа»

После построения контекстной диаграммы строится диаграмма декомпозиции. На данной диаграмме отображаются функции моделируемой системы, которые должны быть реализованы. На рисунке 3 представлен первый уровень декомпозиции для поставленной задачи. На диаграмме изображена функциональная декомпозиция бизнес-процесса «Выпускная квалификационная работа» на пять процессов: «Анализ предметной области и постановка задачи на разработку», «Сравнительный анализ алгоритмов и программных средств выявления вероятных последовательностей событий», «Проектирование программных средств», «Разработка программных средств», «Контроль качества программных средств».

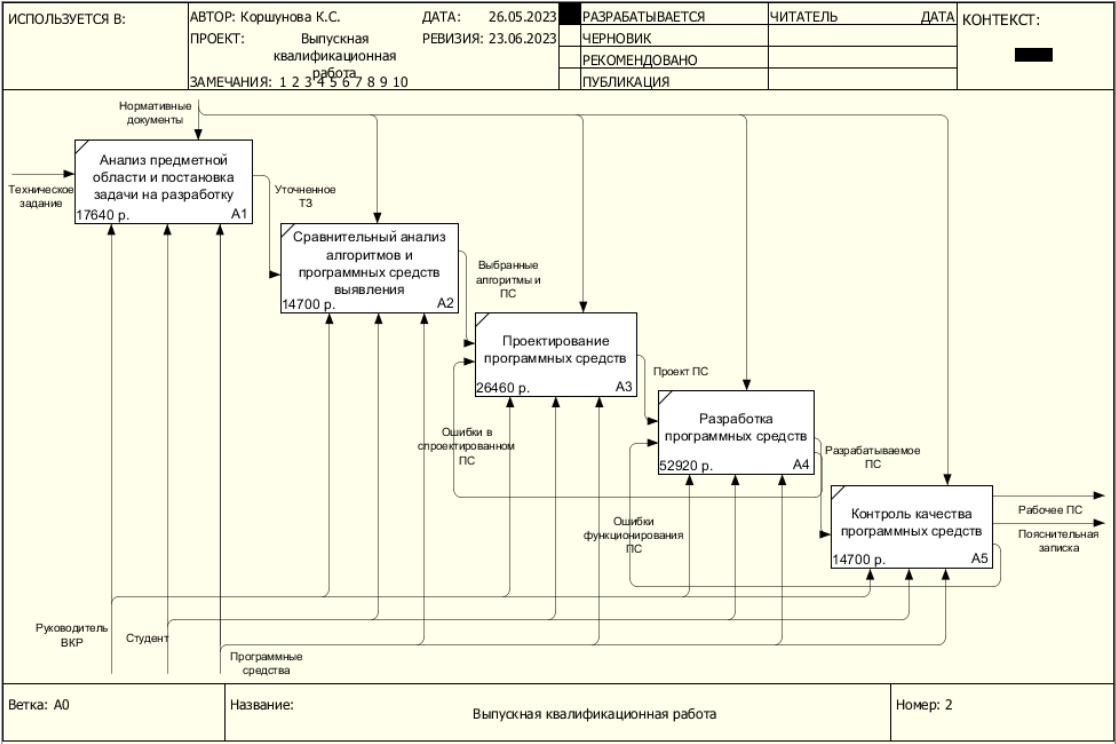


Рисунок 3 – Декомпозиция процессов

Был проведен анализ затрат времени на разработку программного средства и создана диаграмма Ганта, показывающая реализацию запланированной работы с течением времени. Данная диаграмма представлена на рисунке 4.

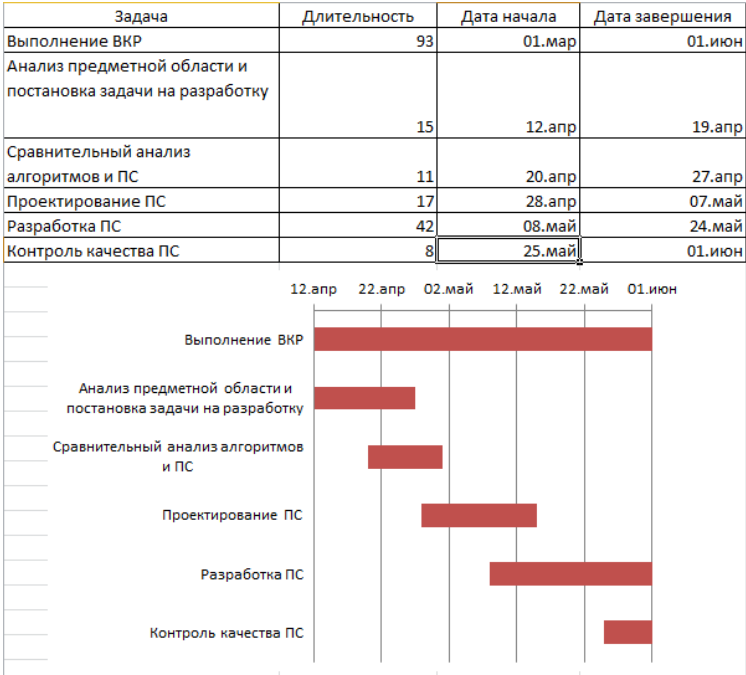


Рисунок 4 – Диаграмма Ганта

Также проведены подсчеты затрат на реализацию данной задачи. Итоги расчетов представлены в таблице 1.

Таблица 1 – Затраты на разработку проекта

Этап	Время	Тариф	Затраты
Анализ предметной области и постановка задачи на разработку	60 ч	294 рублей в час	17640 руб
Сравнительный анализ алгоритмов и ПС выявления вероятных последовательностей событий	50 ч	294 рублей в час	14700 руб
Проектирование ПС	90 ч	294 рублей в час	26460 руб
Разработка ПС	180 ч	294 рублей в час	52920 руб
Контроль качества ПС	50 ч	294 рублей в час	14700 руб
		Итого:	126420 руб

В результате подсчетов было выявлено, что для реализации программного средства выявления вероятных последовательности событий потрачено 126420 рубля.

2 Сравнительный анализ алгоритмов и программных средств выявления вероятных последовательностей событий

2.1 Обоснование выбора метода интеллектуального анализа для выявления вероятных последовательностей событий

Интеллектуальный анализ данных (ИАД) позволяет выявить закономерности в больших наборах данных. Существует множество различных методов ИАД, позволяющих обработать данные и визуализировать полученные результаты обработки. Для выявления вероятных последовательностей событий были рассмотрены следующие методы: поиск ассоциативных правил, секвенциальный анализ (СА).

Изначально для поиска возможных последовательностей событий и их анализа применялся метод ассоциативных правил. Алгоритмы поиска ассоциативных правил позволяют находить все наборы элементов, удовлетворяющих значениям минимальной поддержки и минимальной достоверности правил. Однако данный метод не позволяет учитывать временной аспект, что не позволяет более точно выявить все закономерности событий.

Секвенциальный анализ был сформулирован в XX веке. Данный метод позволяет выявлять часто встречающиеся последовательности объектов с учетом их временного интервала. Работа алгоритмов СА схожа с поиском ассоциативных правил. Отличием алгоритмов является набор входных параметров. На вход в секвенциальном анализе поступают последовательности данных.

Последовательность событий представляет собой упорядоченное множество объектов. Поддержка последовательности позволяет выявить частые последовательности. Если поддержка больше заданной минимальной поддержки, последовательность считается частой. Задачей секвенциального анализа является поиск всех частых последовательностей.

Основными характеристиками секвенциального анализа являются поддержка и временное окно.

Поддержка – это количество последовательностей, содержащих данную последовательность, относительно общего числа цепочек.

Временное окно – это временной интервал между событиями в последовательностях.

Секвенциальный анализ включает множество различных алгоритмов, позволяющих выявить закономерности возникновения событий. Данные алгоритмы будут рассмотрены в разделе №2.2.

Для реализации поиска вероятных последовательностей событий был выбран секвенциальный анализ, так как он является более эффективным методом для решения поставленной задачи.

2.2 Сравнительный анализ алгоритмов секвенциального анализа

Секвенциальный анализ содержит большое количество различных алгоритмов, позволяющих выявлять часто встречающиеся последовательности. Некоторые из них приведены в таблице 2.

Таблица 2 – Алгоритмы секвенциального анализа

Название алгоритма	Год создания	Авторы
AprioriAll	1995	Р. Агравалом , Р.Срикнатом (R. Agrawal, R. Srikant.)
GSP	1996	Р. Агравалом , Р.Срикнатом
SPADE	2001	М. Заки (M.J. Zaki)
SPAM	2002	Дж.Эйрес, Дж.Фланник, Дж.Герке и Т. Ю. (J.Ayres, J.Flannick, J.Gehrke and T. Yiu.)
FreeSpan	2001	Дж. Пей, Дж. Хан, Б. Мортазави-Асл, Дж. Ванг, Х. Пинто, К. Чен, У. Даял, М. Хсу. (J.Han , J.Pei, B.Mortazavi-Asl, Q. Chen, U.Dayal, M.-C. Hsu.)
PrefixSpan	2001	Дж. Пей, Дж. Хан, Б. Мортазави-Асл, Дж. Ванг, Х. Пинто, К. Чен, У. Даял, М. Хсу.

Алгоритмы выявления вероятных последовательностей событий можно разделить на пять групп: алгоритмы семейства Apriori, алгоритмы поиска в

ширину, алгоритмы поиска в глубину, алгоритмы поиска замкнутых последовательностей, инкрементные алгоритмы.

Существует большое количество различных видов алгоритма Apriori. AprioriAll является первым алгоритмом на основе Apriori.

Работа алгоритма разделена на несколько фаз:

- 1) фаза сортировки;
- 2) фаза отбора кандидатов;
- 3) фаза трансформации;
- 4) фаза генерации;
- 5) фаза максимизации.

На первой фазе происходит группирование записей по уникальному ключу, а после по времени внутри сформированных групп. На следующем шаге реализуется поиск одноэлементных последовательностей, удовлетворяющих значению минимальной поддержки. После выполняется преобразование транзакции в набор одноэлементных последовательностей, содержащихся в ней. Уже из полученных цепочек фаза генерации строит более длинные шаблоны последовательностей. И на последнем шаге происходит поиск последовательностей, которые не входят в более длинные последовательности.

Недостатком алгоритма AprioriAll является большое количество обращений к базе данных.

Алгоритм GSP является модификацией алгоритма AprioriAll. Данный алгоритм позволяет учитывать временные ограничения между событиями.

Работа алгоритма GSP состоит из нескольких этапов: генерация кандидатов и подсчет поддержки кандидатов. Генерация кандидатов включает в себя две фазы: объединение, упрощение.

На первом проходе по исходному набору данных алгоритм подсчитывает поддержку для каждого элемента. На следующих проходах из полученных часто встречающихся последовательностей будут формироваться кандидаты. Кандидатами называются последовательности, имеющие на один элемент больше, чем те из которых был сформирован. Далее рассчитывается поддержка

для каждого найденного кандидата, для того чтобы выявить какие являются частыми. Работа алгоритма завершается, когда при последующих проходах невозможно сформировать новых кандидатов, либо не будут найдены частые последовательности [3].

Алгоритм GSP является более эффективным, чем алгоритм AprioriAll. Недостатками данного алгоритма считается большое количество обращений к базе и генерируемых кандидатов.

SPADE относится к группе алгоритмов, использующих метод поиска в глубину. SPADE использует вертикальное форматирование.

Рассмотрим работу алгоритма SPADE. На первом сканировании базы происходит выявление последовательностей, состоящих из одного элемента. Далее выполняется преобразование вертикального представления в горизонтальное, после чего подсчитывается количество последовательностей для всех пар элементов. На следующих шагах формируются цепочки с помощью id-list. Благодаря методу подсчета поддержки, основанному на структуре id-list, этот алгоритм работает эффективнее GSP. Также данный метод позволяет сократить число сканирований базы данных, это предоставляет возможность значительно уменьшить время работы. Алгоритм прекращает работу, когда новые частые последовательности будут не найдены [4].

Еще одним алгоритмом, входящим в группу, которая использует метод в глубину, является SPAM. Данный алгоритм использует вертикальное представление данных. Также он имеет эффективный метод подсчета поддержки. Для растрового представления SPAM использует много памяти, поэтому оно очень эффективно для тех баз данных, которые имеют длинные последовательные шаблоны. Кроме того, важной особенностью этого алгоритма считается вывод новых наборов часто используемых элементов в режиме онлайн и поэтапно [4].

По сравнению с алгоритмами SPADE и PrefixSpan работает эффективнее на больших наборах данных, но занимает больше места.

Целью алгоритма FreeSpan считается сокращение генерации подпоследовательностей.

Алгоритм FreeSpan делит данные и набор часто используемых шаблонов, сокращая каждое тестирование соответствующим меньшим набором данных. Сканирование исходной базы данных выполняется три раза, независимо от максимальной длины последовательности [4].

Идея алгоритма PrefixSpan заключается в том, чтобы проверять только префиксные подпоследовательности, и соответствующие им постфиксные подпоследовательности проецировать в формируемый набор данных [4].

PrefixSpan работает значительно быстрее, а также использует меньше памяти, чем алгоритмы GSP и FreeSpan.

Для сравнения алгоритмов секвенциального анализа были выделены следующие критерии:

- 1) сканирование базы данных – функция, включающая в себя сканирование исходной базы данных, для определения, являются ли созданные последовательности-кандидаты часто встречающимися;
- 2) сокращение последовательности-кандидата – функция, которая предоставляет возможность некоторым алгоритмам использовать структуру данных, которая позволяет сокращать последовательности-кандидаты на ранних этапах интеллектуального анализа;
- 3) разделение пространства поиска – функция, позволяющая разбивать сгенерированное пространство поиска на большие последовательности-кандидаты, чтобы улучшить эффективность управления памятью;
- 4) метод поиска в глубину – подход, который позволяет проводить поиск таким образом, что на пути должны быть изучены все элементы узла, прежде чем переходить к следующему;
- 5) метод поиска в ширину – подход, который позволяет проводить поиск по уровням, чтобы найти полный набор шаблонов;
- 6) ограничение регулярного выражения – ограничение является антимонотонным, основанным на росте, если оно включает следующее

свойство: последовательность должна быть достижима путем роста из любого компонента, который соответствует части регулярного выражения, если он удовлетворяет ограничению;

7) поиск сверху вниз;

8) поиск снизу вверх;

9) вертикальная проекция базы данных – функция, при которой идет обращение к базе данных последовательностей только один или два раза, чтобы получить вертикальное расположение базы данных [4];

10) время выполнения (в секундах);

11) использование памяти.

Результаты сравнения алгоритмов секвенциального анализа представлены в таблице 3.

Таблица 3 – Результирующая таблица сравнения алгоритмов по критериям

Критерии	Алгоритмы секвенциального анализа					
	Apriori All	GSP	SPADE	FreeSpan	PrefixSpan	SPAM
Сканирование базы данных	✓	✓				
Сокращение последовательностей и-кандидата		✓	✓		✓	
Разделение пространства поиска	✓					
Метод поиска в глубину			✓	✓	✓	✓
Метод поиска в ширину		✓				
Ограничение регулярного выражения				✓	✓	
Поиск сверху вниз				✓	✓	
Поиск снизу вверх		✓	✓			✓
Вертикальная проекция базы данных			✓			✓
Время выполнения (в секундах)	-	2126	-	-	798	674
Использование памяти	-	687	-	-	320	1052

Все рассмотренные алгоритмы секвенциального анализа имеют свои преимущества и недостатки. Наиболее эффективным алгоритмом работы с большим набором данных является SPAM, его основным недостатком считается большое использование памяти.

2.3 Сравнительный анализ программных средств секвенциального анализа

Существуют различные программные средства для проведения СА. Некоторые из них приведены в таблице 4.

Таблица 4 – Программные средства секвенциального анализа

Название ПС	Год создания	Авторы/компании разработки
SPMF	2008	Филипп Фурнье-Вигер (Philippe Fournier-Viger) и др.
Xelopes	2011	Компания Prudsys в тесном сотрудничестве со специалистами российской фирмы ZSoft
RapidMiner	2004	Компания RapidMiner

SPMF – это библиотека интеллектуального анализа данных с открытым исходным кодом, реализованная на языке Java.

В данном инструментальном средстве представлены реализации 254 алгоритмов интеллектуального анализа данных для:

- наборов элементов;
- ассоциативных правил;
- последовательных шаблонов;
- последовательного анализа правил;
- выявления последовательности;
- временных рядов;
- кластеризации и классификации.

В SPMF исходный код каждого алгоритма может быть легко интегрирован в другое программное обеспечение Java. Кроме того, можно вызвать SPMF из других языков, таких как Python, R и C# [5].

Библиотека Xelopes используется для проведения интеллектуального анализа данных.

Архитектура данного программного средства выполнена в соответствии со стандартом Model Driven Architecture, которое дает возможность реализации библиотеки на трёх языках: Java, C++ и C#.

Рассмотрим достоинства библиотеки Xelopes:

- библиотека находится в открытом доступе;
- разработчики имеют возможность добавлять свои алгоритмы;
- не зависит от исходных данных;
- соответствует основным стандартам интеллектуального анализа данных [6].

RapidMiner – это программное средство с открытым исходным кодом, который позволяет обрабатывать данные. Особенность RapidMiner заключается в том, что аналитик может выполнить свою работу без написания кода.

Преимуществами RapidMiner являются:

- многофункциональный, простой и удобный графический интерфейс;
- использует встроенные инструменты для обработки и фильтрации данных;
- расширяемость;
- предоставляет возможность пользователям построить отчеты [7].

Каждое из выше описанных инструментов имеет свои преимущества и недостатки. Одни программные средства обладают более широким набором функций, а другие более просты и удобны в использовании.

3 Проектирование программных средств

3.1 Функциональные требования программных средств

Для проектирования требований к программному средству и описанию отдельных компонентов будут использованы диаграммы в нотации Unified Modeling Language (UML).

Унифицированный язык моделирования – это система обозначений, применяемая для объектно-ориентированного анализа и проектирования программных средств. UML используется для конструирования, визуализации и документирования различных систем.

UML состоит из трех видов строительных блоков:

- 1) диаграммы, применяемые для графического представления набора элементов;
- 2) сущности – основные элементы модели;
- 3) связи, использующиеся для соединения сущностей между собой.

Благодаря уникальной графической нотации элементов UML, позволяет отобразить важные особенности элементов проектируемых систем.

В состав унифицированного языка UML входит 13 различных видов диаграмм [8].

На диаграмме вариантов использования отображается взаимодействие между прецедентами, обозначающими функции системы, и актерами, которые представляют людей или системы. Данная диаграмма используется для описания функциональности программного средства.

Функциональные требования разрабатываемого ПС отображены на диаграмме вариантов использования, которая представлена на рисунке 5.

Для удобства на диаграмме использованы цветовые обозначения:

- бизнес-процессы показаны серым цветом;
- бизнес-операции – оранжевым цветом;
- операции с данными/БД – голубым цветом.

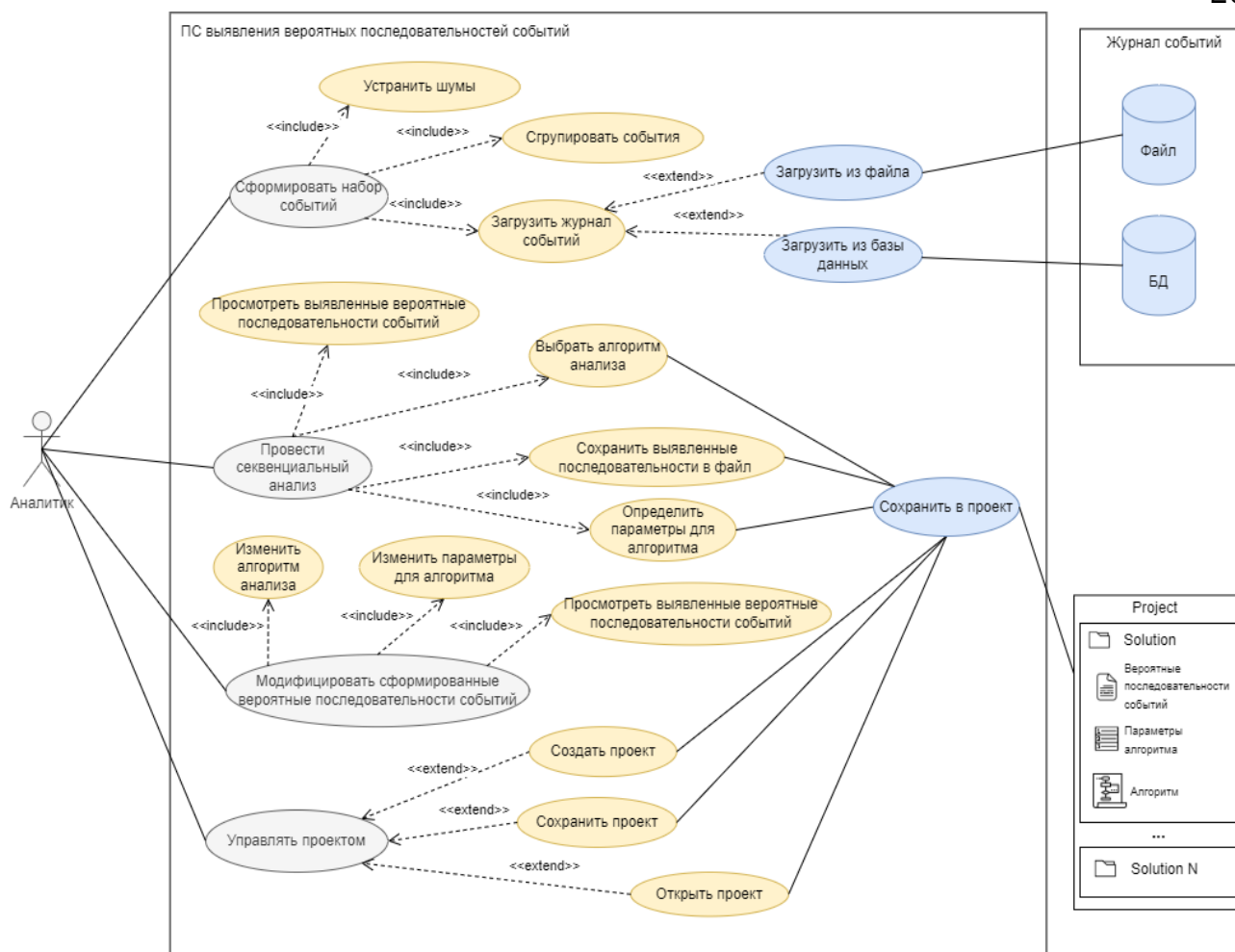


Рисунок 5 – Функциональные возможности программного средства (диаграмма вариантов использования)

Разрабатываемое ПС предназначено для выявления вероятных последовательностей событий. Оно должно соответствовать следующим требованиям:

- предварительная обработка и фильтрация журнала событий;
- выгрузка журнала событий из базы данных/файла;
- проведение секвенциального анализа (выбор алгоритма, определение параметров анализа);
- просмотр выявленных вероятных последовательностей событий;
- сохранение выявленных цепочек событий в файл;
- управление проектом.

Аналитик в программе совершает четыре бизнес-процесса: сформировывает набор данных, проводит секвенциальный анализ, управляет проектом, а также модифицирует сформированные вероятные последовательности событий.

К работе с журналом относятся такие операции, как устранение шумов, группировка событий и загрузка журнала событий. Загрузить журнал аналитик может как из базы данных, так и из файла.

К управлению проектом относятся операции создания, сохранения и открытия проекта.

Секвенциальный анализ включает операции выбора алгоритма, определение параметров, просмотр и сохранение выявленных вероятных последовательностей событий. С помощью бизнес-процесса, связанного с модификацией, аналитик может выбрать другой алгоритм и изменить его параметры.

В таблице 5 описывается сценарий прецедента «Провести секвенциальный анализ».

Таблица 5 – Сценарий прецедента «Провести секвенциальный анализ»

Название варианта использования	Провести секвенциальный анализ
Актеры	Аналитик
Краткое описание	Поиск вероятных последовательностей событий
Ссылки на другие варианты использования	Выбрать алгоритм анализа, определить параметры для алгоритма, посмотреть выявленные вероятные последовательности событий, сохранить выявленные последовательности в файл
Предусловие	Сформирован набор событий
Типичный ход событий	1. Аналитик выбирает алгоритм для проведения анализа. 2. Аналитик определяет параметры. 3. Аналитик получает и просматривает выявленные последовательности событий. 4. Аналитик сохраняет цепочки в файл.
Постусловие	Анализ выявленных последовательностей

Сценарии остальных вариантов использования представлены в приложении Б.

3.2 Архитектура программных средств

Для описания архитектуры проектируемого программного средства была разработана диаграмма развертывания, которая представлена на рисунке 6.

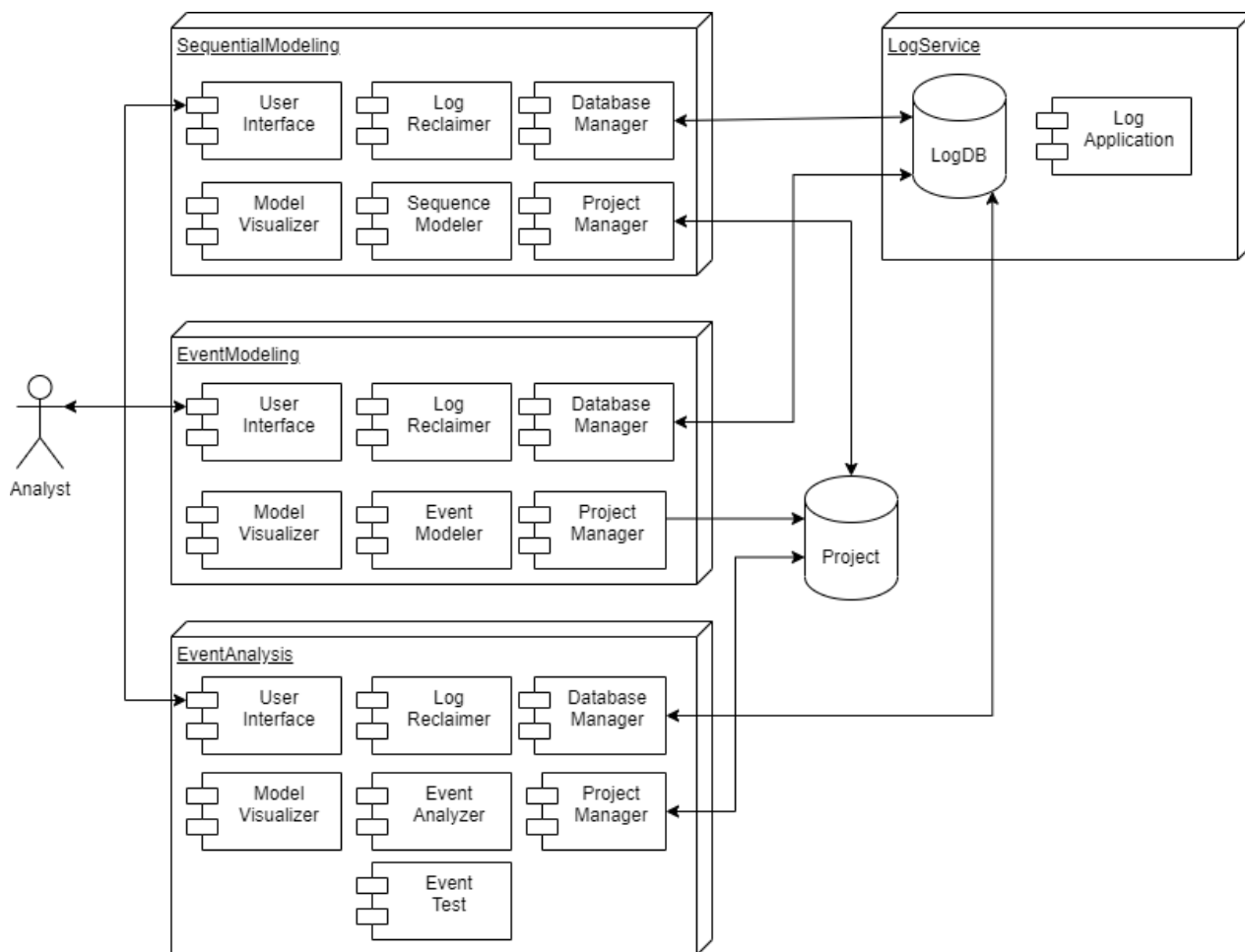


Рисунок 6 – Архитектура ПС (диаграмма развертывания)

Диаграмма развертывания используется с целью отобразить, как программное обеспечение взаимодействует с аппаратным обеспечением.

Данная диаграмма включает три основных элемента:

- 1) узел – это вычислительный ресурс, который имеет возможность выполнять программы и обрабатывать информацию;
- 2) компонент – это программный элемент;
- 3) артефакт, описывающий структуру, которая используется в разработке программного средства [9].

Данная система представляет собой три приложения, каждое из которых выполняет свои задачи. Приложение EventModeling предназначено для выявления вероятных шаблонов составных событий, а приложение EventAnalysis отвечает за анализ шаблонов событий. В рамках данного дипломного проекта разрабатывается приложение SequentialModeling, которое предназначено для поиска вероятных последовательностей событий, с помощью алгоритмов секвенциального анализа. В этом приложении три основные подсистемы: доступа к данным, работы с проектом, секвенциального анализа.

На данной диаграмме представлена архитектура, разрабатываемого программного средства, элементы которого рассмотрены в таблице 6.

Таблица 6 – Элементы диаграммы развертывания

Название элемента	Описание
Приложение для выявления вероятных последовательностей событий(SequentialModeling)	
Analyst	Аналитик, взаимодействующий с пользовательским интерфейсом приложения SequentialModeling
SequentialModeling	Приложение, выполняющее выявление вероятных последовательностей событий
User Interface	Компонент, отвечающий за пользовательский интерфейс приложения
Log Reclaimer	Компонент обработки и фильтрации журнала событий
Model Visualizer	Компонент
Sequence Modeler	Компонент проведения секвенциального анализа для выявления вероятных последовательностей событий
Database Manager	Компонент работы с базой данных
Project Manager	Компонент работы с проектом
Project	Проект
LogService	Узел, отвечающий за загрузку журнала событий
LogDB	База данных журнала событий
LogApplication	Компонент, отвечающий за хранение журнала событий в файле

Аналитик взаимодействует с системой через пользовательский интерфейс. У него присутствует возможность проведения секвенциального анализа, работы с данными журнала событий и проектом.

В результате работы данного приложения аналитик должен получить возможные вероятные последовательности событий.

3.3 Обоснование выбора технологий и средств разработки

На основе функциональных и технических требований были определены следующие технологии и средства разработки программного средства: Visual Studio и SQL Server.

Для реализации приложения, позволяющего выявлять вероятные последовательности событий, была выбрана среда разработки Microsoft Visual Studio. Данная среда позволяет создавать кроссплатформенные проекты на различных языках программирования, также присутствует возможность использования платформы .NET.

Разработчик имеет доступ к большому количеству разнообразных функций, которые могут быть использованы в любой версии операционной системы Windows, а также мобильных и интернет-приложениях.

В качестве базы данных для использования в проекте была выбрана SQL Server. Главными преимуществами SQL Server являются:

- высокая производительность и надежные механизмы транзакций;
- облегчает развертывание, объединение и распространение объемных данных;
- позволяет соединять упорядоченные и неупорядоченные данные;
- имеет возможность поддержки непрерывной памяти.

Также рассматривалась MongoDB, однако Mongo является документно-ориентированной базой данных, а важным аспектом данного проекта является систематичность хранения данных.

В качестве языка программирования был выбран C#. Он является Си-подобным языком близким к C++ и Java. Он имеет следующие преимущества:

- строго типизированный язык;
- объектно-ориентированный язык программирования;
- функциональность;
- гибкость;
- масштабируемость;

– мощный инструментарий.

Данный язык имеет обработку исключений, а также в нем присутствует сборщик мусора. С# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Благодаря объектно-ориентированному подходу есть возможность построения крупных приложений [10].

3.4 Требования к аппаратно-программному обеспечению

3.4.1 Требования к аппаратно-техническому обеспечению

Аппаратное обеспечение – это совокупность электронных и механических устройств, входящих в состав системы. Аппаратное обеспечение включает: процессор, оперативную память и др. Для разработки приложения выявления вероятных последовательностей событий необходимы следующие требования: Microsoft Visual Studio 2022, PostgreSQL, Microsoft Framework .NET v.4.8 или выше.

Также обязательными минимальными требованиями к компьютеру являются 64-разрядный процессор, 4 ГБ оперативной памяти и 20 ГБ свободного места на жестком диске.

3.4.2 Требования к системному программному обеспечению

Системное программное обеспечение – это набор программ, позволяющих управлять компонентами вычислительной системы. Для стабильной работы приложения необходимо выполнение следующих системных требований: Microsoft Windows 10 или более поздние версии, Npgsql Driver для PostgreSQL.

3.4.3 Требования к прикладному программному обеспечению

Прикладное программное обеспечение – это набор программ, позволяющих выполнять определенные задачи, а также направленных на взаимодействие с пользователем.

Требования к прикладному программному обеспечению определяются средой разработки и системой управления базами данных (СУБД). В качестве среды разработки был выбран Microsoft Visual Studio 2022, а СУБД – PostgreSQL. Необходимые требования к установке и работе представлены в таблице 7.

Таблица 7 – Требования к прикладному программному обеспечению

Параметры	Программные продукты	
	Visual Studio 2022	PostgreSQL
Оперативная система	Windows 10 или более поздние версии	Windows 10
Процессор	64-разрядный	64-разрядный с тактовой частотой минимум 1ГГц
Оперативная память	Не менее 4 ГБ (рекомендуется 16 ГБ)	2 ГБ
Свободное место на жестком диске	20 – 50 ГБ	512МБ (рекомендуется больше)

В таблице приведены минимальные требования к работе. Для стабильной работы необходимо больше оперативной памяти и места на жестком диске.

4 Разработка программных средств

4.1 Разработка пользовательского интерфейса

Пользовательский интерфейс разработан с помощью Windows Form на языке C# в Visual Studio. Главная форма программного средства представлена на рисунке 7.

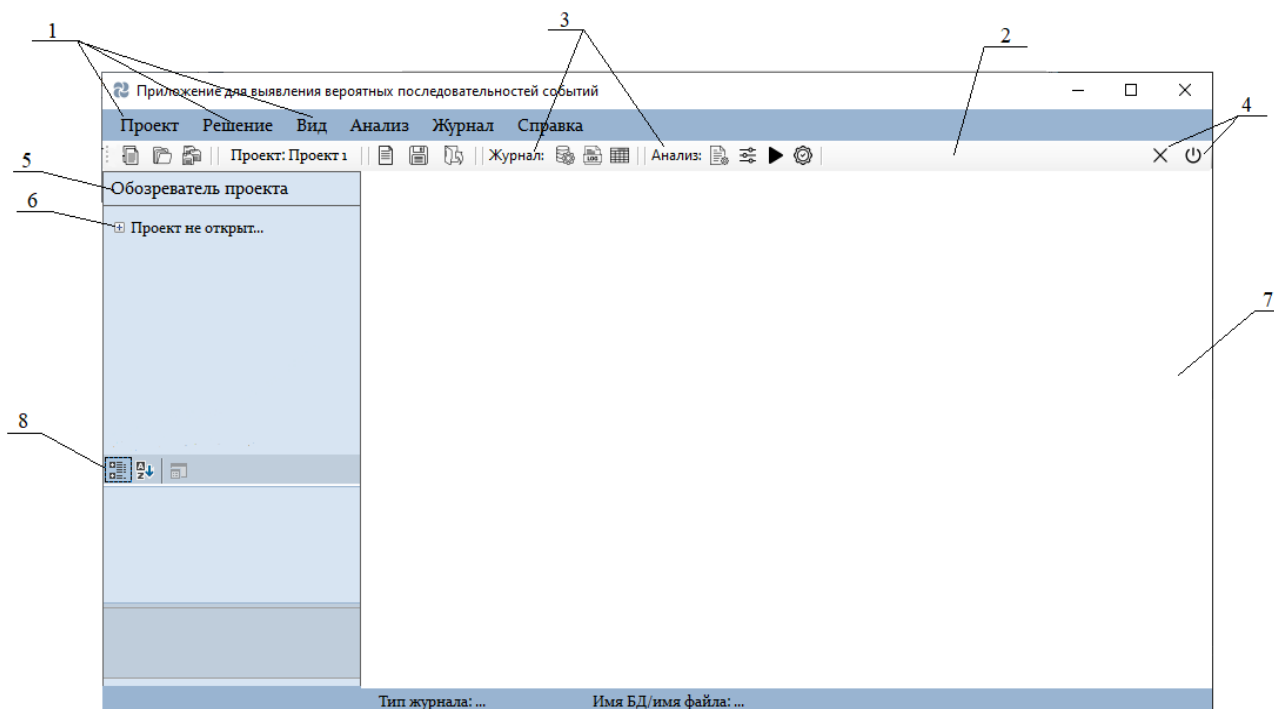


Рисунок 7 – Главная форма пользовательского интерфейса

Компоненты, указанные выносками на рисунке 7, описаны в таблице 8.

Таблица 8 – Описание компонентов главной формы интерфейса

№	Имя компонента	Тип компонента	Назначение
1	MenuStrip	Меню	Компонент для описания меню
2	ToolStrip	Панель инструментов	Компонент для отображения панели инструментов
3	ToolStripLabel	Надпись	Надпись на панели инструментов
4	ToolStripButton	Кнопка	Кнопка, расположенная на панели инструментов
5	Label	Надпись	Компонент для отображения текста на форме
6	TreeView	Визуальный элемент	Компонент для отображения данных в виде иерархии
7	Panel	Панель	Компонент, объединяющий элементы в группу
8	PropertyGrid	Панель свойств	Компонент, отображающий свойства объекта для их редактирования

На главной форме расположено меню, состоящее из нескольких разделов: проект, решение, вид, анализ, журнал и справка. Каждый раздел предоставляет определенные функции работы ПС. Описание пунктов меню представлено в таблице 9.

Таблица 9 – Меню интерфейса программного средства

№	Раздел	Функции раздела
1	Проект	Включает функции работы с проектом: создание, открытие, закрытие, сохранение, сохранение как, свойства и выход
2	Решение	Включает функции работы с решением: создание, открытие, сохранение и свойства
3	Вид	Включает функции, позволяющие скрыть/открыть обозреватель проекта, панели инструментов и состояния
4	Анализ	Включает функции работы с анализом: обработка набора данных, выбор алгоритма и определение его параметров, выполнение алгоритма и вывод результата работы
5	Журнал	Включает функции работы с журналом: подключение журнала событий и его прочтение
6	Справка	Включает документацию о разработке и работе программного средства

На главной форме интерфейса содержится панель инструментов, которая предоставляет возможность выполнения функций, содержащихся в меню. Также внизу находится панель состояния, которая отображает тип журнала и имя БД или файла, которые загружены в проект.

В начале работы пользователь создает проект или открывает существующий. Окно, предоставляющее возможность создания проекта, приведено на рисунке 8.

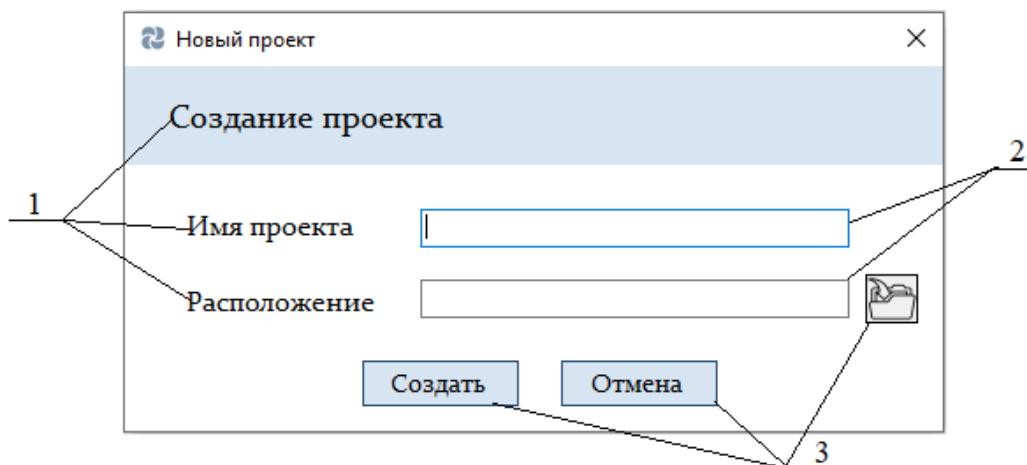


Рисунок 8 – Создание проекта

Компоненты, указанные выносками на рисунке 8, описаны в таблице 10.

Таблица 10 – Описание компонентов окна «Новый проект»

№	Имя компонента	Тип компонента	Назначение
1	Label	Надпись	Компонент для надписей
2	TextBox	Поле	Текстовое поле для ввода данных
3	Button	Кнопка	Компонент, позволяющий выполнить определенное действие при нажатии

Для проведения анализа необходимо подключить журнал. Окно для подключения журнала событий представлено на рисунке 9.

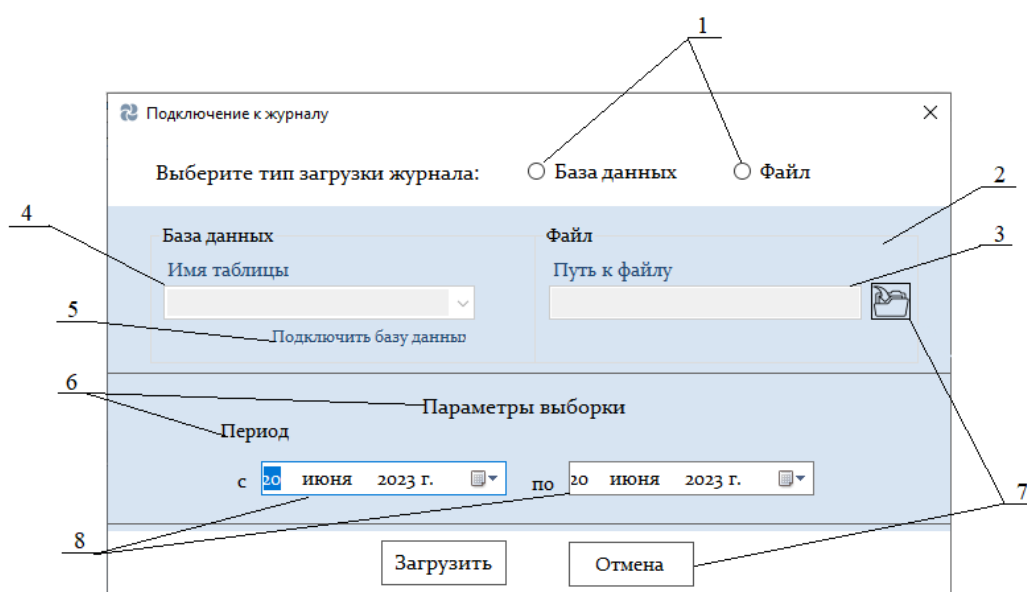


Рисунок 9 – Окно подключения журнала событий

Компоненты, указанные выносками на рисунке 9, описаны в таблице 11.

Таблица 11 – Описание компонентов окна подключения журнала

№	Имя компонента	Тип компонента	Назначение
1	RadioButton	Переключатель	Компонент, позволяющий выбрать один из двух и более свойств
2	GroupBox	Контейнер	Компонент для группировки элементов
3	TextBox	Текстовое поле	Компонент для ввода или редактирования текста
4	ComboBox	Выпадающий список	Компонент хранения элементов списка
5	LinkLabel	Ссылка	Компонент для вывода ссылок
6	Label	Надпись	Компонент для отображения текста на форме
7	Button	Кнопка	Компонент, позволяющий выполнить определенное действие при нажатии
8	DateTimePicker	Календарь	Компонент для выбора даты

Для начала работы с журналом событий из БД необходимо выполнить подключение. Окно «Подключение к базе данных» представлено на рисунке 10.

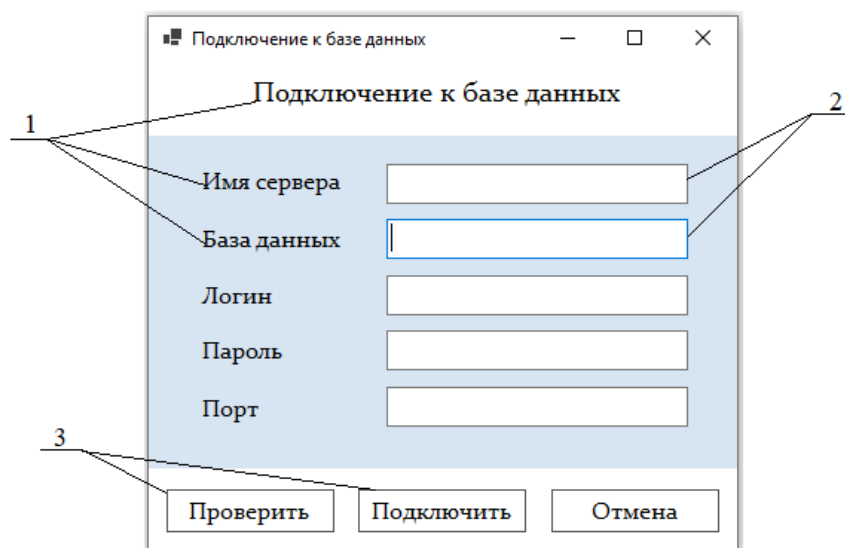


Рисунок 10 – Окно «Подключение к базе данных»

Компоненты, указанные выносками на рисунке 10, описаны в таблице 12.

Таблица 12 – Описание компонентов окна «Подключение к базе данных»

№	Имя компонента	Тип компонента	Назначение
1	Label	Надпись	Компонент для надписей
2	TextBox	Поле	Текстовое поле для ввода данных
3	Button	Кнопка	Компонент, позволяющий выполнить определенное действие при нажатии

После подключения журнала следует создать или открыть решение. Окно «Новое решение» представлено на рисунке 11.

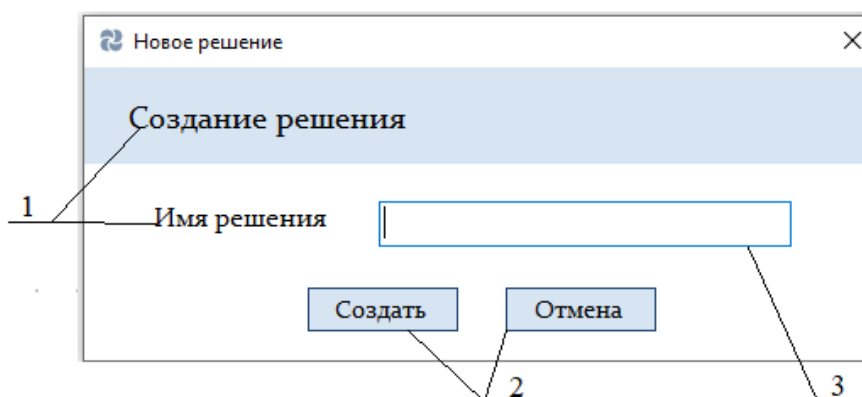


Рисунок 11 – Добавление решения в проект

Компоненты, указанные выносками на рисунке 11, описаны в таблице 13.

Таблица 13 – Описание компонентов окна «Новое решение»

№	Имя компонента	Тип компонента	Назначение
1	Label	Надпись	Компонент для надписей
2	Button	Кнопка	Компонент, позволяющий выполнить определенное действие при нажатии
3	TextBox	Поле	Текстовое поле для ввода данных

В разделе «Анализ» содержится работа с алгоритмами выявления вероятных последовательностей событий. Окно с настройками параметров алгоритмов представлено на рисунке 12.

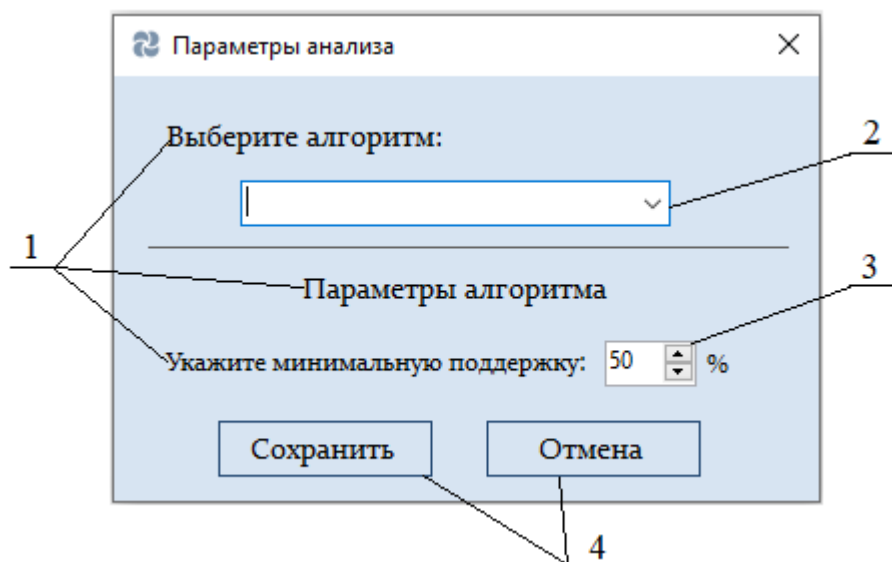


Рисунок 12 – Окно работы с алгоритмами

Компоненты, указанные выносками на рисунке 12, описаны в таблице 14.

Таблица 14 – Описание компонентов окна «Новое решение»

№	Имя компонента	Тип компонента	Назначение
1	Label	Надпись	Компонент для надписей
2	ComboBox	Выпадающий список	Компонент хранения элементов списка
3	NumericUpDown	Числовое поле	Компонент для выбора числа из диапазона
4	Button	Кнопка	Компонент, позволяющий выполнить определенное действие при нажатии

Раздел «Справка» содержит сведения о программе и документацию для работы. Информационное окно представлено на рисунке 13.

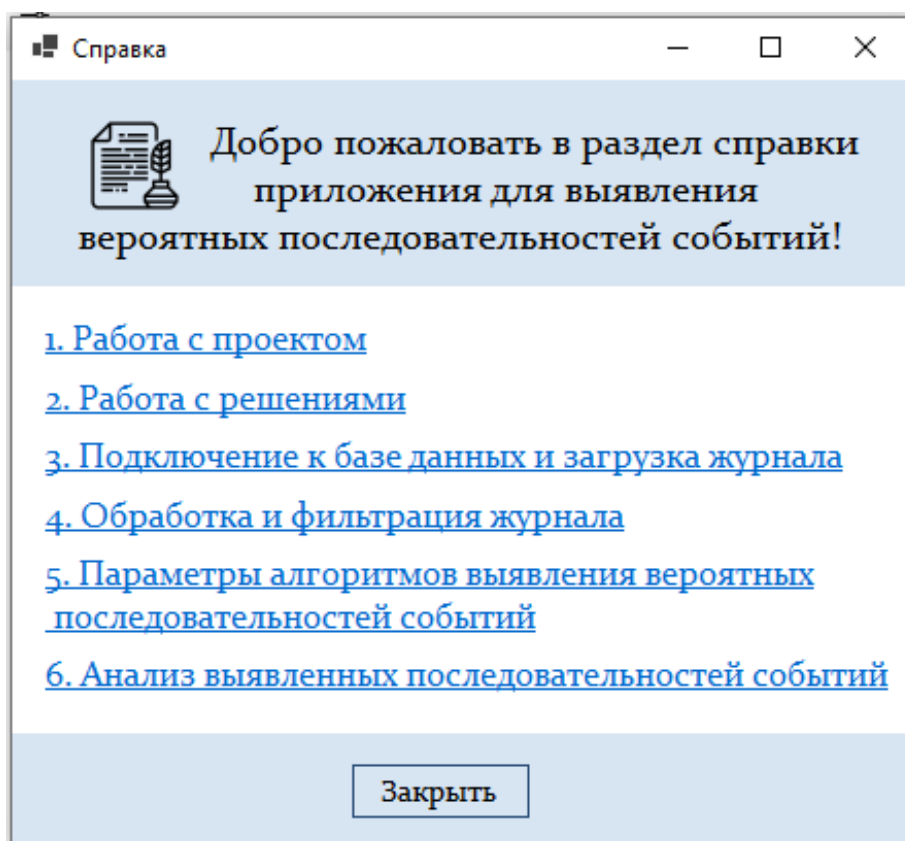


Рисунок 13 – Документация программы

Данные о разработке программного средства представлены на рисунке 14.

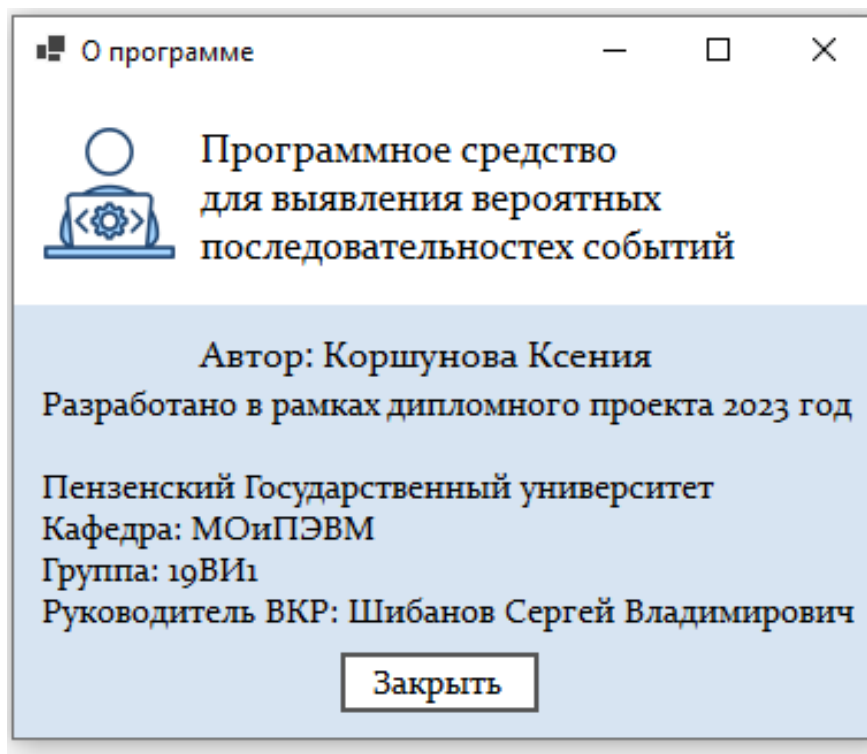


Рисунок 14 – Данные о программе

Справочная информация будет обновляться с учетом усовершенствования программного средства. Исходный код реализации пользовательского интерфейса представлен в листинге В.4 – В.9 в приложении В.

4.2 Разработка программных средств доступа к журналу событий

Для разработки программных средств доступа к журналу событий использована динамическая библиотека, предоставляющая готовые компоненты и функциональность для работы с базой данных.

Динамическая библиотека – это скомпилированный набор функций, загружаемый в программу во время выполнения. Одним из преимуществ динамической библиотеки считается то, что ее можно использовать в нескольких приложениях, что позволяет экономить ресурсы. Также функциональность динамической библиотеки можно обновить без перекомпиляции всей программы [12].

Для работы программных средств доступа к журналу событий используется динамическая библиотека eventloglb.dll. Она предназначена для с журналом событий. Данная библиотека предоставляет функции: выгрузка и сохранение журнала событий, предварительная обработка и фильтрация данных. Спецификация функций данной библиотеки приведена в таблице 15.

Таблица 15 – Спецификация функций библиотеки eventloglb.dll

Функция	Описание
load_db_event_log(db_pool, filter_params, event_log): int	Функция предназначена для загрузки и фильтрации журнала событий из БД в соответствии с условием, которое задано в filter_params. Выборка журнала записывается в event_log. При успешном выполнении возвращается 0, иначе – код ошибки
load_file_event_log(filename, filter_params, event_log): int	Функция используется для загрузки и фильтрации журнала событий из filename в соответствии с условием, которое задано в filter_params. Выборка журнала событий записывается в event_log. При успешном выполнении возвращается 0, иначе – код ошибки

Продолжение таблицы 15

Функция	Описание
preprocess_event_log(in_event_log, out_event_log): int	Функция предназначена для предобработки журнала событий in_event_log. Предобработка включает устранение шумов, фильтрация и другие. Результат обработки записывается в out_event_log. При успешном выполнении возвращается 0, иначе – код ошибки
export_event_log(filename, event_log): int	Функция используется для выгрузки журнала event_log в filename. При успешном выполнении возвращается 0, иначе – код ошибки

Для работы библиотеки необходим драйвер базы данных, позволяющий взаимодействовать с базой из программного средства. Драйверы предоставляют API (Application Programming Interface), который позволяет приложениям отправлять запросы к базе данных и получать результаты [11].

4.3 Разработка программных средств управления проектами

Для ПС была разработана структура проекта, которая представлена на рисунке 15.

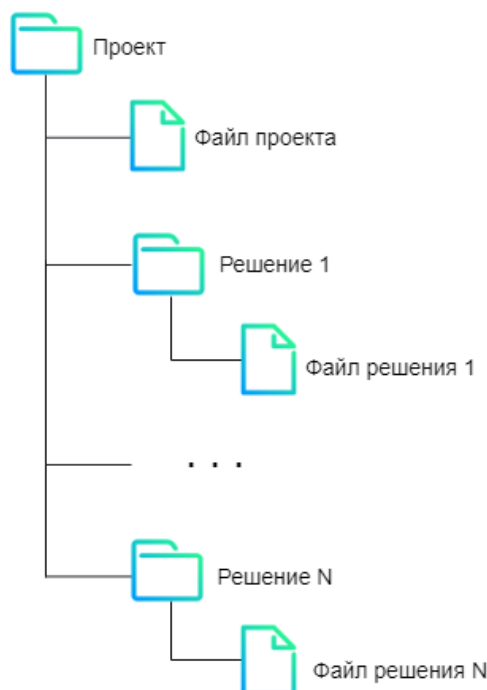


Рисунок 15 – Структура проекта

Папка проекта содержит файл с данными проекта и все решения. В файле проекта содержатся сведения о журнале событий. Проект может включать

несколько решений. В папке решения содержится файл, в котором хранятся данные об анализе, параметры алгоритма и результаты выявленных последовательностей.

Для описания ПС управления проектами была разработана диаграмма классов (рисунок 16).

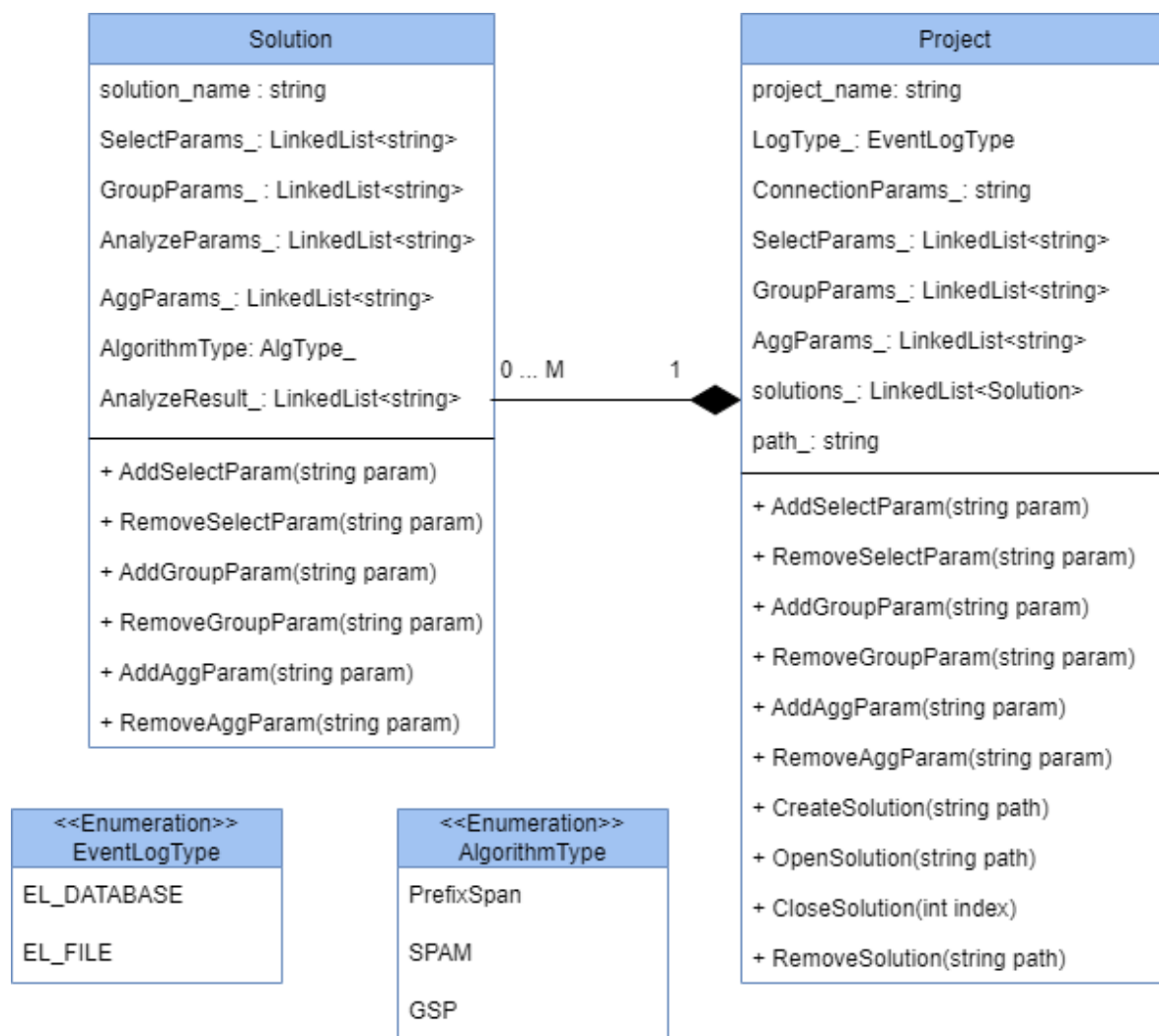


Рисунок 16 – Программные средства управления проектами

На диаграмме отображены два класса: Project, Solution. Описание полей классов представлено в таблице 16.

Таблица 16 – Описание классов ПС управления проектами

Поле	Тип данных	Описание
Класс Project		
project_name	string	Имя проекта
LogType_	EventLogType	Перечисление методов загрузки журнала событий

Продолжение таблицы 16

Поле	Тип данных	Описание
ConnectionParams_	string	Параметры подключения журнала событий
SelectParams_	LinkedList<string>	Список параметров выбора данных журнала
GroupParams_	LinkedList<string>	Список параметров группировки данных журнала
AggParams_	LinkedList<string>	Список параметров агрегации данных журнала
solutions_	LinkedList<Solution>	Список решений проекта
path_	string	Расположение проекта
Класс Solution		
solution_name	string	Имя решения
SelectParams_	LinkedList<string>	Список параметров выбора набора данных
GroupParams_	LinkedList<string>	Список параметров группировки набора данных
AnalyzeParams_	LinkedList<string>	Список параметров алгоритмов
AggParams_	LinkedList<string>	Список параметров агрегации набора данных
AlgorithmType	AlgType_	Перечисление видов алгоритмов
AnalyzeResult_	LinkedList<string>	Список результатов анализа

Также на диаграмме отображены два перечисления: EventLogType, AlgType. EventLogType включает способы загрузки журнала событий: из файла (EL_FILE), из базы данных (EL_DATABASE). AlgType включает виды алгоритмов, позволяющих выявить вероятные последовательности событий: GSP, SPAM и PrefixSpan.

Для разработки программных средств управления проектами использована библиотека projlb.dll. Спецификация функций данной библиотеки приведена в таблице 17.

Таблица 17 – Спецификация функций библиотеки projlb.dll.

Функция	Описание
open_project(path, project): int	Функция загрузки проекта, который расположен в пути path и записи в project объект, содержащий данные о проекте. При успешном выполнении возвращается 0, иначе – код ошибки
save_project(path, project): int	Функция сохранения project в расположении path. При успешном выполнении возвращается 0, иначе – код ошибки
open_solution(path, solution): int	Функция предназначена для загрузки решения, который расположен по path, и записи в решение solution объекта, включающий данные по этому решению. При успешном выполнении возвращается 0, иначе – код ошибки
save_solution(path, solution): int	Функция сохранения solution в расположении path. При успешном выполнении возвращается 0, иначе – код ошибки

Исходный код реализации библиотеки projlb.dll приведен в листинге В.1, В.2 приложения В.

4.4 Разработка программных средств формирования вероятных последовательностей событий

4.4.1 Структура плагинов библиотеки секвенциального анализа

Программные средства выявления вероятных последовательностей событий реализуют следующие алгоритмы: GSP, SPAM и PrefixSpan. Данные программные средства разработаны в виде плагина динамической библиотеки, которая может быть загружена во время выполнения программы.

Плагин – это независимо компилируемый программный модуль, который динамически подключается к программе. Их используют для расширения функциональных возможностей других программных приложений. Преимуществами плагинов являются:

- 1) возможность разделения функциональности на модули;
- 2) возможность интеграции со сторонними сервисами;
- 3) возможность расширения пользовательского интерфейса;
- 4) возможность поддержки новых форматов данных.

Была реализована библиотека seqmineib.dll в виде класса, позволяющая формировать вероятные последовательности событий. Сигнатура класса приведена в листинге 1.

Листинг 1 – Сигнатура класса SeqMineLib

```
public class SeqMineLib
{
    public struct AlgData{}
    public struct KValues{}
    public static AlgData[] getAlgList(){}
    private static int SeqGSP(List<string> eventLog, KValues
parameters, out List<string> result){}
    private static int SeqSPAM(List<string> eventLog, KValues
parameters, out List<string> result){}
    private static int SeqPrefixSpan(List<string> eventLog,
KValues parameters, out List<string> result){}
```

```

        public static int ExecAlg(string algType, List<string>
eventLog, KValues parameters, out List<string> result){}

    }

```

Спецификация структуры данной библиотеки приведена в таблице 18.

Таблица 18 – Спецификация структур библиотеки seqminelb.dll

Структура	Описание
AlgData	Используется для хранения данных об алгоритмах секвенциального анализа. Содержит следующие поля: id – идентификатор алгоритма; name – название алгоритма; tip – всплывающая подсказка.
KValues	Используется для хранения дополнительных параметров при запуске алгоритмов секвенциального анализа.

Спецификация функций библиотеки описана в таблице 19.

Таблица 19 – Спецификация функция библиотеки seqminelb.dll

Функция	Описание
getAlgList	Возвращает список доступных алгоритмов секвенциального анализа в библиотеке.
SeqGSP(eventLog, parameters, result): int	Реализует GSP-алгоритм. На вход принимает журнал событий в виде списка строк (List<string>), дополнительные параметры для запуска алгоритма parameters и записывает в список result результат работы алгоритма. Возвращает 0 в случае успешного завершения алгоритма, в противном случае, – код ошибки.
SeqSPAM(eventLog, parameters, result): int	Реализует SPAM-алгоритм. На вход принимает журнал событий в виде списка строк (List<string>), дополнительные параметры для запуска алгоритма parameters и записывает в список result результат работы алгоритма. Возвращает 0 в случае успешного завершения алгоритма, в противном случае, – код ошибки.
SeqPrefixSpan(eventLog, parameters, result): int	Реализует алгоритм PrefixSpan. На вход принимает журнал событий в виде списка строк (List<string>), дополнительные параметры для запуска алгоритма parameters и записывает в список result результат работы алгоритма. Возвращает 0 в случае успешного завершения алгоритма, в противном случае, – код ошибки.
ExecAlg(algType, parameters, eventLog, result): int	Осуществляет запуск алгоритма в соответствии с его идентификатором в algType и передает в него дополнительные параметры parameters и журнал событий eventLog. Результат работы алгоритма записывает в result. В случае успеха возвращает 0, в противном случае, – код ошибки.

Данная библиотека позволяет работать с алгоритмами секвенциального анализа, такими как: GSP, SPAM, PrefixSpan.

С помощью функции `getAlgList`, при открытии формы настройки параметров анализа будут отражены реализованные алгоритмы для поиска вероятных последовательностей событий. Функция `ExecAlg` запускает работу выбранного алгоритма. Исходный код реализации библиотеки `seqminelb.dll` приведен в листинге В.3 приложения В.

4.4.2 Реализация алгоритма GSP

Формирование вероятных последовательностей в алгоритме GSP проходит в несколько этапов. Для отображения сценария работы алгоритма разработана диаграмма деятельности, которая представлена на рисунке 17.

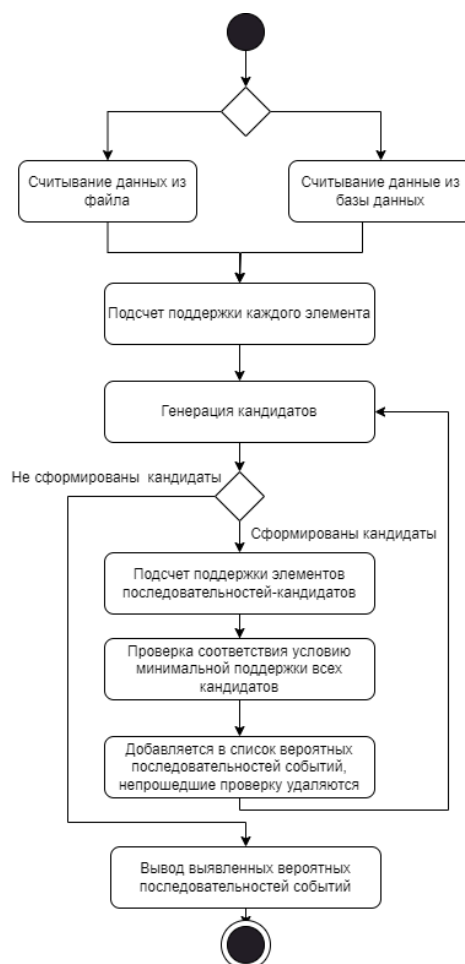


Рисунок 17 – Сценарий работы алгоритма GSP

Работа алгоритма совершается в несколько проходах. Вначале происходит считывание данных, после подсчитывается поддержка для каждого

элемента. Далее формируются кандидаты – последовательности, которые имеют на один элемент больше, чем те из которых был сформирован. В конце подсчитывается поддержка для всех найденных кандидатов, и происходит проверка на соответствие заданной минимальной поддержке. Эти этапы осуществляются до момента, пока невозможно сформировать новых последовательностей-кандидатов.

4.4.3 Реализация алгоритма PrefixSpan

Для отображения сценария работы алгоритма PrefixSpan разработана диаграмма деятельности, которая представлена на рисунке 18.

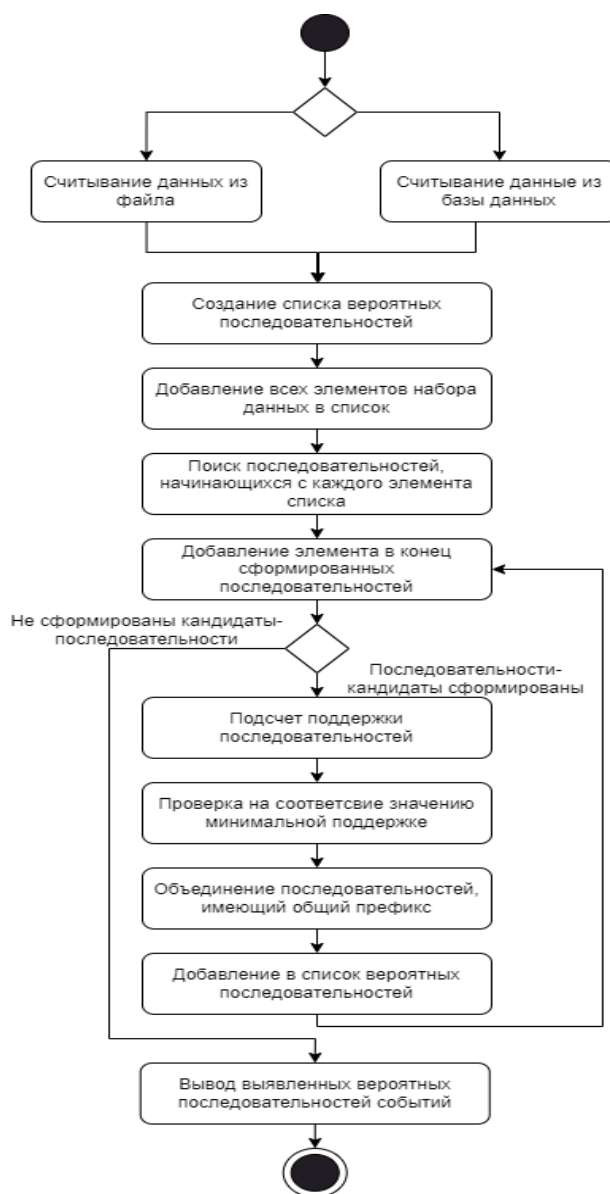


Рисунок 18 – Сценарий работы алгоритма PrefixSpan

Алгоритм PrefixSpan формирует вероятные последовательности, исследуя только префиксные последовательности, и проецирует только соответствующие им постфиксные подпоследовательности в проецируемые базы данных.

Вначале происходит считывание данных, после чего создается пустой список последовательностей, в который добавляются элементы набора данных. Далее происходит поиск последовательностей, которые начинаются на каждый элемент списка. На следующем этапе генерируются последовательности. К каждой сформированной последовательности в конец добавляется элемент. После чего происходит подсчет их поддержки и сравнение ее с минимально заданной поддержкой. Последовательности, которые удовлетворяют условию и имеют общий префикс, объединяются и добавляются в список вероятных последовательностей. Генерация последовательностей продолжается до тех пор, пока не будут выявлены все возможные вероятные последовательности.

4.4.4 Реализация алгоритма SPAM

Работа алгоритма начинается со считывания данных, далее выполняется подсчет поддержки и поиск частых элементов. После чего происходит генерация последовательностей, начинающихся на текущий префикс. Для всех найденных последовательностей подсчитывается поддержка. На следующем этапе осуществляется сравнения подсчитанных значений каждой последовательности с минимально заданной поддержкой. Аналогичным образом происходит генерация кандидатов, оканчивающихся на текущий префикс, подсчет и сравнение поддержки с минимально заданной. Далее происходит объединение последовательностей в более длинные последовательности. Работа алгоритма завершается, когда невозможно выявить новые вероятные последовательности.

Работа алгоритма SPAM представлена на рисунке 19.

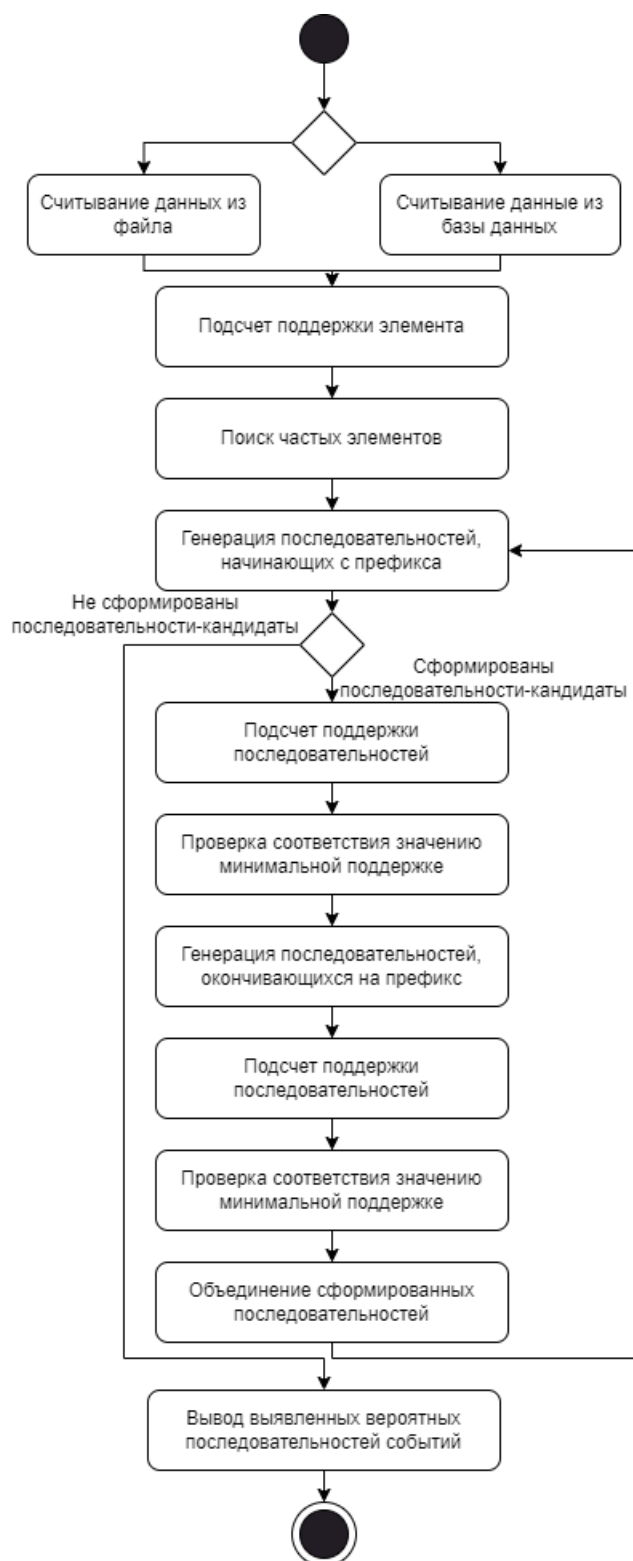


Рисунок 19 – Сценарий работы алгоритма SPAM

Идея алгоритм SPAM заключается в сокращении генерации кандидатом, благодаря этому он эффективно выявляет вероятные последовательности событий в больших наборах данных.

5 Контроль качества программных средств

5.1 Планирование процесса контроля качества программных средств

5.1.1 Мероприятия по контролю качества программных средств

Контроль качества программных средств является последним этапом разработки, который направлен на выявление и устранение ошибок. Основная задача контроля качества заключается в проверке готового продукта с помощью тестирования и проверки соответствия требованиям заказчика. Если продукт не соответствует требованиям, он считается некачественным.

Мероприятия по контролю качества можно классифицировать следующим образом:

- тестирование программного обеспечения во время его работы;
- проверка соответствия стандартам правил построения кода программного средства;
- симуляция работы программного обеспечения;
- анализ архитектуры программного обеспечения [13].

Контроль качества позволяет получить актуальную информацию о состоянии системы в настоящее время.

В данной выпускной квалификационной работе проведено тестирование и подсчет метрик программного кода в качестве мероприятий по контролю качества.

5.1.2 Обоснование выбора модели тестирования

Тестирование позволяет проверить соответствие между ожидаемым и реальным результатом работы программы. Задачами тестирования являются:

- выявление нарушений работы программы;
- проверка корректности работы системы при известных предлагаемых обстоятельствах;

- проверка системы на соответствие заданным требованиям.

Виды тестирования сгруппированы по целям, исполнению кода, по разработке тестовых сценариев и другие. Все группы тестирования представлены на рисунке 20.

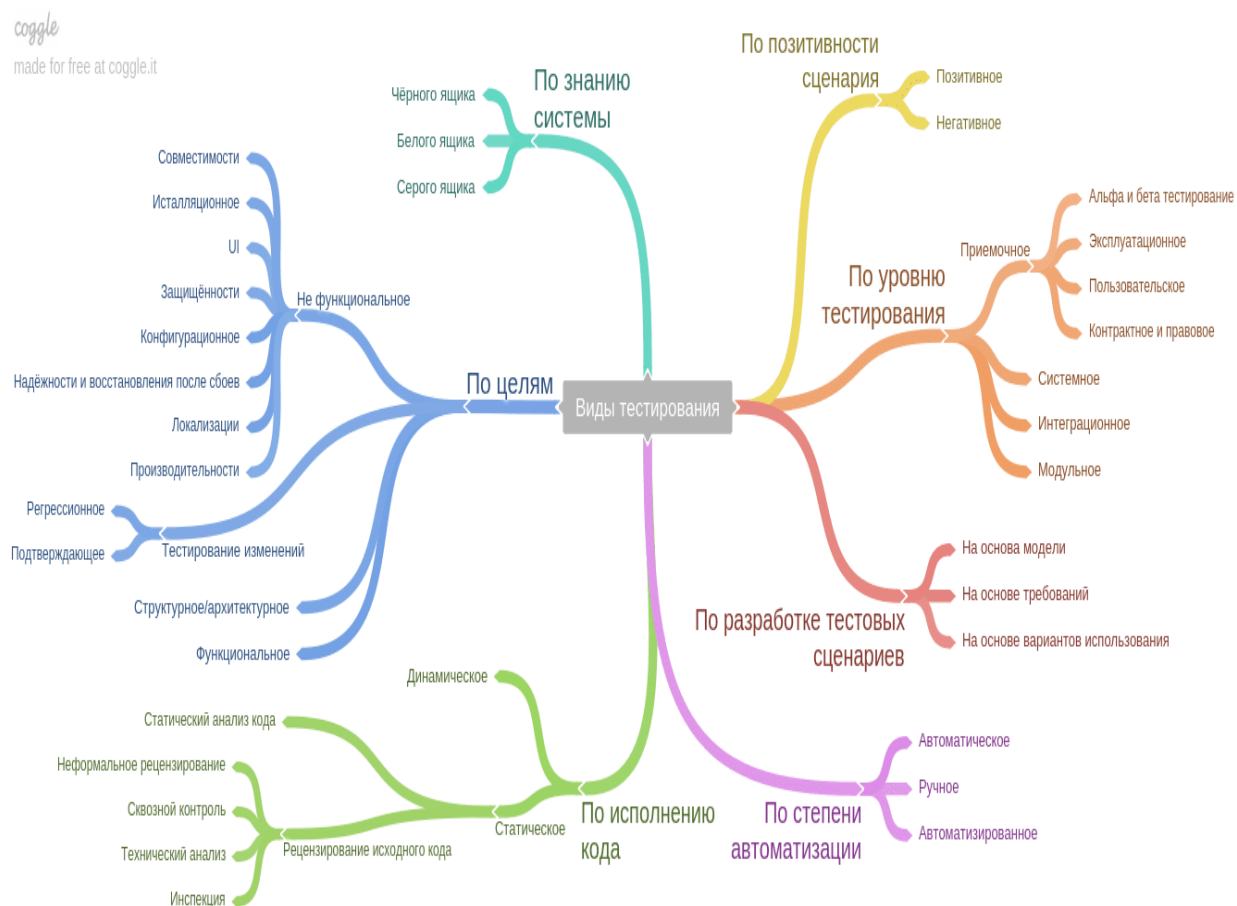


Рисунок 20 – Виды тестирования

К основным группам тестирования относятся: функциональные и нефункциональные.

Целью функционального тестирования является проверка функциональных требований приложения, т. е. способность корректно выполнять поставленные задачи.

Нефункциональное тестирование программного средства проверяет соответствие таким требованиям, как:

- удобство приложения для пользователя;

- масштабируемость;
- производительность работы системы;
- безопасность системы (защита данных) [14].

В рамках выпускной квалификационной работы было проведено функциональное тестирование.

5.1.3 Обоснование выбора метрик программного кода

Для контроля качества программного обеспечения необходимо использование метрик. Метрика программного обеспечения представляет собой численное значение, которое отражает определенное свойство программного обеспечения или его спецификаций. Существует множество различных метрик программ, которые можно разделить на несколько групп, таких как количественные метрики, метрики надежности, метрики сложности потока управления программы, метрики сложности потока управления данными, объектно-ориентированные метрики и гибридные метрики [15].

Некоторые метрики и их описание представлены в таблице 19.

Таблица 19 – Метрики программного кода

Название	Описание
Метрика Холстеда	Данная метрика позволяет получить оценки словаря, длины, объема программы и теоретического словаря программы.
Метрика МакКейба	Данная метрика используется для определения количества базисных путей в управляющем графе программы, а базисные пути – это пути, которые позволяют получить все возможные пути из входа в выходы графа.
Метрика Майерса	Данная метрика использует интервальную меру сложности, которая включает в себя цикломатическую меру и количество отдельных условий.
Метрика Харрисона	Данная метрика позволяет учитывать размер программы и уровень вложенности.
Метрика Чепина	Данная метрика делит все множество переменных на группы в зависимости от их применения в программе.

В данной работе для оценки качества программного кода была выбрана метрика Холстеда.

5.2 Тестирование программных средств

5.2.1 Классы эквивалентности

Для добавления решения в проект необходимо ввести имя решения. Поле для ввода имени является обязательным для заполнения. Для корректного формата ввода названия решения составлены классы эквивалентности (КЭ), представленные в таблице 20.

Таблица 20 – Классы эквивалентности

	Правильные КЭ	Неправильные КЭ
Длина поля	1-30 (1)	Null (2)
		>30 (3)
Структура	Имя начинается с буквы/цифры (4)	Имя начинается с символа (5)
Содержание	Буквы (6) Цифры(7) Символы (- _) (8)	Остальные символы (9)

Длина поля для ввода имени решения изображена на рисунке 21.



Рисунок 21 – Графическое представление длины поля

Содержание поля ввода имени решения представлено на рисунке 22.



Рисунок 22 – Графическое представление содержания поля

Тест-кейсы, составленные для КЭ, приведены в таблице 21.

В ходе выполнения тест-кейсов ошибок не обнаружено, следовательно ПС доступа к журналу работает корректно. Результаты тестов представлены на рисунках Г.6 – Г.9 приложения Г.

Для проверки корректной работы анализа для выявления вероятных последовательностей событий была составлена таблица решений, которая приведена в таблице 24.

Таблица 24 – Таблица решений для программных средств выявления шаблонов

	1	2	3	4
Причины				
Алгоритм выбран	0	1	1	1
Определены корректные параметры анализа	0	1	0	1
Определены некорректные параметры анализа	0	0	1	0
Выполнение анализа	0	1	0	1
Результаты анализа				
Следствия				
Сообщение «Выберите алгоритм»	1	0	0	0
Сообщение «Параметры алгоритма некорректные»	0	0	1	0
Параметры алгоритма сохранены в решение	0	1	0	1
Отображение сведений о проведенном анализе	0	1	0	1
Отображение результатов в рабочей области	0	0	0	1

Из таблицы решений были построены тест-кейсы, описание которых представлено в таблице 25.

Таблица 25 – Описание тест-кейсов для ПС выявления вероятных последовательностей

№	Входные данные	Результат
1	Алгоритм не выбран	Сообщение «Выберите алгоритм»
2	Алгоритм выбран Определены корректные параметры анализа Выполнение анализа	Сохранение параметров алгоритма в решении и отображение сведений о проведенном анализе
3	Алгоритм выбран Определены корректные параметры анализа Выполнение анализа Результаты анализа	Сохранение параметров алгоритма в решении, отображение сведений о проведенном анализе и просмотр результата работы алгоритма в рабочей области

В ходе выполнения тест-кейсов ошибок не обнаружено, следовательно ПС выявления вероятных последовательностей событий работает корректно. Результаты тестов представлены на рисунках Г.10 – Г.13 приложения Г.

5.3 Оценка качества программного кода

Для оценки качества программного кода выбрана метрика Холстеда. Расчет метрик был проведен для метода подсчета поддержки последовательностей, представленного на листинге 2.

Листинг 2 – Код метода получения поддержки последовательности

```
public static void getSupport(List<Sequence> seq)
{
    foreach (Sequence sq in seq)
    {
        foreach (Sequence ssq in Program.transact)
        {
            int start = 0;
            bool getAll = true;
            foreach (Comp cp in sq.seq)
            {
                if (!ssq.findElemSeq(cp, ref start))
                {
                    getAll = false;
                    break;
                }
            }
            if (getAll)
            {
                sq.support++;
            }
        }
    }
}
```

Для расчета метрики необходимо определить значение количества уникальных операторов и операндов, а также их общее число вхождений. Результаты представлены в таблице 26.

Таблица 26 – Количество операторов и операндов

Операторы	Число вхождений	Операнды	Число вхождений
if	2	seq	2
=	3	sq	1
++	1	start	2

Продолжение таблицы 26

Операторы	Число вхождений	Операнды	Число вхождений
!	1	getAll	3
foreach()	2	cp	2
;	5	sq.seq	1
in	2	Program.transact	1
{}	6	ssq	1
.	4	sq.support	1
ssq.findElemSeq	1	0	1
break	1	true	1
,	1	false	1
n1=13	N1=29	n2=13	N2=17

В результате было выявлено, что в методе содержится 13 оператор и операндов. Суммарное вхождение операторов составляет 29, а операндов – 17.

Была подсчитана длина программы по следующей формуле:

$$N = N_1 + N_2,$$

где N_1 – количество вхождений операторов;

N_2 – количество вхождений операндов.

Длина программы для данного метода составляет 46.

Был подсчитан словарь программы по следующей формуле:

$$n = n_1 + n_2,$$

где n_1 – количество уникальных операторов;

n_2 – количество уникальных операндов.

Словарь программы для данного метода составляет 26.

Был подсчитан объем программы по следующей формуле:

$$HV = N \log_2 n.$$

Объем программы для метода составляет 216,22.

Также была подсчитана сложность программы по следующей формуле:

$$HD = (n_1 \div 2) \times (N_2 \div n_2).$$

Сложность метода составляет 8,01, что говорит о простоте программы.

Заключение

В процессе выполнения выпускной квалификационной работы был выполнен анализ предметной области, в котором рассмотрена актуальность создания приложения для выявления вероятных последовательностей событий, и поставлена задача на разработку. Также был проведен сравнительный анализ алгоритмов и программных средств секвенциального анализа и функционально-стоимостной анализ процесса разработки.

В ходе проектирования были разработаны требования к аппаратно-программному обеспечению. Также определены требования к архитектуре и функционалу приложения.

В результате было разработано приложение для выявления вероятных последовательностей событий на основе интеллектуального анализа журнала событий, которое выполняет следующие функции:

- предварительная обработка журнала событий;
- загрузка журнала событий из базы данных/файла;
- выбор алгоритма анализа и определение его параметров;
- просмотр и сохранение выявленных последовательностей в файл.

В рамках контроля качества приложения было проведено тестирование и оценка программного кода, которое показало, что разработанная система работает стабильно и корректно.

Дальнейшие исследования будут включать в себя расширение и дополнение функциональных возможностей программных средств, а также пополнение реализации видов алгоритмов секвенциального анализа для поиска вероятных последовательностей событий.

Список использованных источников

1. Шибанов, С. В. Интегрированная модель активных правил / С. В. Шибанов, А. А. Скоробогатько, Э. В. Лысенко // Математическое и программное обеспечение систем в промышленной и социальной сферах. – 2011. – № 1. – С. 41 – 46.
2. ITteach.RU // Методология IDEF0 [Электронный ресурс] URL: <https://itteach.ru/bpwin/metodologiya-idef0> (дата обращения: 29.05.2023);
3. Pandia // Определение Data Mining [Электронный ресурс] URL: <https://pandia.ru/text/78/271/8297-3.php?ysclid=ley018nt1t322708633> (дата обращения: 10.05.2023).
4. Jawahar. S A COMPARATIVE STUDY OF SEQUENTIAL PATTERN MINING ALGORITHMS / Jawahar. S // International Journal of Application or Innovation in Engineering & Management (IJAIEEM). – 2015. – №2319 – С. 19 – 24.
5. SPMF An Open-Source Data Mining Library // Introduction [Электронный ресурс] URL: <https://www.philippe-fournier-viger.com/spmf/index.php?link=algorithms.php> (дата обращения: 08.05.2023).
6. Севостьянова, А. В. Обзор пакета Xelopes и его основных возможностей / Севостьянова А. В., Столяров А. И. // Теория и практика применения свободного программного обеспечения: материалы Российской молодежной конференции с элементами научной школы. – Москва : Изд-во ФГБОУ ВО «МГТУ», 2016. – С. 112 – 115.
7. Научные переводы // RapidMiner – платформа для статистической обработки данных [Электронный ресурс] URL: <https://xn--80aegcaa6cbngm5a6c1ci.xn--p1ai/rapidminer-platforma/> (дата обращения: 11.05.2023).
8. Мартин Фаулер, UML Основы / Мартин Фаулер. – СПб: Символ Плюс, 2004. – 192 с.

9. Lucidchart // Урок по диаграммам развертывания [Электронный ресурс] URL: <https://www.lucidchart.com/pages/ru/uml-deployment-diagram> (дата обращения: 18.05.2023).
10. Васильев, А. Н. Программирование на C# для начинающих. Основные сведения / А. Н. Васильев. – М. : Эксмо, 2018. – 592 с.
11. Е. П. Моргунов. PostgreSQL. Основы языка SQL (учебное пособие) / Моргунов Е. П. – Санкт-Петербург: Издательство БХВ-Петербург, 2018. – 336 с.
12. RadioProg // Статические и динамические библиотеки [Электронный ресурс] URL: <https://radioprogram.ru/post/1084> (дата обращения: 27.05.2023).
13. Интуит // Качество ПО и методы его контроля [Электронный ресурс] URL: https://intuit.ru/studies/professional_skill_improvements/1530/courses/64/lecture/1874?page=3 (дата обращения: 31.05.2023).
14. QA evolution // Виды тестирования ПО [Электронный ресурс] URL: <https://qaevolution.ru/testirovanie-po/vidy-testirovaniya-po/> (дата обращения: 31.05.2023).
15. Смирнов, А. В. Методы оценки и управления качеством программного обеспечения / А. В. Смирнов // Информатика, вычислительная техника и управление. – 2019. – №2. – С. 20 – 25.

ПРИЛОЖЕНИЕ А
ГЛОССАРИЙ
(справочное)

База данных (БД) – это совокупность данных, хранящихся в связанных таблицах.

Интеллектуальный анализ данных (ИАД, Data Mining) – это способ исследования информации и обнаружение ранее не выявленных закономерностей в наборе данных.

Классы эквивалентности (КЭ) – это метод тестирования, позволяющий сократить количество тестовых сценариев, не уменьшая при этом покрытие тестами.

Программные средства (ПС) – это объект, который состоит из набора программ, процедур и правил, в том числе данные, относящиеся к функционированию системы.

Секвенциальный анализ (СА) – это разновидность интеллектуального анализа данных.

Система управления базами данных (СУБД) – это комплекс программных средств, позволяющих создавать базы данных и управлять данными.

Функционально-стоимостной анализ (ФСА) – это метод, позволяющий оценить стоимость продукта (услуги).

Application Programming Interface (API) – это набор способов и правил, позволяющих программам взаимодействовать между собой и обмениваться данными.

IDEF0 – это нотация графического моделирования, используемая для изображения функциональной модели, чтобы отобразить структуру и ее функции.

Structured Query Language (SQL) – декларативный язык программирования, который используется для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Unified Modeling Language (UML) – это система обозначений, применяемая для объектно-ориентированного анализа и проектирования программных средств.

ПРИЛОЖЕНИЕ Б
СЦЕНАРИЙ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ
(рекомендуемое)

Таблица Б.1 – Сценарий прецедента «Сформировать набор событий»

Название варианта использования	Сформировать набор событий
Актеры	Аналитик
Краткое описание	Подготовка данных для анализа
Ссылки на другие варианты использования	Устранить шумы, сгруппировать события, загрузить журнал событий
Предусловие	Создан/открыт проект
Типичный ход событий	1. Аналитик загружает журнал событий из базы данных/файла. 2. Аналитик проводит обработку и фильтрацию журнала событий.
Постусловие	Анализ данных

Таблица Б.2 – Сценарий прецедента «Управлять проектом»

Название варианта использования	Управлять проектом
Актеры	Аналитик
Краткое описание	Управление проектом
Ссылки на другие варианты использования	Создать проект, открыть проект, сохранить проект
Предусловие	Запуск программного средства
Типичный ход событий	1. Аналитик создает/открывает проект. 2. Аналитик после выполненной работы сохраняет проект.
Постусловие	—

Таблица Б.3 – Сценарий прецедента «Модифицировать сформированные вероятные последовательности событий»

Название варианта использования	Модифицировать сформированные вероятные последовательности событий
Актеры	Аналитик
Краткое описание	Модифицировать сформированные вероятные последовательности событий
Ссылки на другие варианты использования	Изменить алгоритм анализа, изменить параметры для алгоритма, просмотреть выявленные вероятные последовательности событий
Предусловие	Вероятные последовательности событий
Типичный ход событий	1. Аналитик выбирает алгоритм для проведения анализа. 2. Аналитик определяет необходимые параметры для анализа. 3. Аналитик получает и просматривает выявленные вероятные последовательности событий.
Постусловие	Анализ выявленных последовательностей событий

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД ПРОГРАММНЫХ СРЕДСТВ
(рекомендуемое)

Листинг В.1 – Class Project библиотеки projlb.dll

```

using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.Linq;

namespace ELPClassLibrary
{
    enum EventLogType
    {
        EL_DATABASE,
        EL_FILE
    }
    [DefaultPropertyAttribute("Name")]
    public class Project
    {
        private string project_name;
        private EventLogType LogType_;
        private string ConnectionParams_;
        private LinkedList<string> SelectParams_;
        private LinkedList<string> GroupParams_;
        private LinkedList<string> AggParams_;
        private LinkedList<Solution> solutions_;
        private string path_;

        [CategoryAttribute("Свойства"), DescriptionAttribute(" ")]
        public string ConnectionParams { get => ConnectionParams_;
set => ConnectionParams_ = value; }
        [CategoryAttribute("Свойства"), DescriptionAttribute(" ")]
        public LinkedList<string> SelectParams { get =>
SelectParams_; set => SelectParams_ = value; }
        [CategoryAttribute("Свойства"), DescriptionAttribute(" ")]
        public LinkedList<string> GroupParams { get =>
GroupParams_; set => GroupParams_ = value; }
        [CategoryAttribute("Свойства"), DescriptionAttribute(" ")]
        public LinkedList<string> AggParams { get => AggParams_;
set => AggParams_ = value; }
        [CategoryAttribute("Свойства"), DescriptionAttribute("Тип
журнала")]
        internal EventLogType LogType { get => LogType_; set =>
LogType_ = value; }
        [CategoryAttribute("Свойства"),
DescriptionAttribute("Решения")]
        internal LinkedList<Solution> Solutions { get =>
solutions_; set => solutions_ = value; }
        [CategoryAttribute("Свойства"),
DescriptionAttribute("Расположение проекта")]
        public string Path { get => path_; set => path_ = value; }
        [CategoryAttribute("Свойства"), DescriptionAttribute("Имя
проекта")]

```

```

        public string Project_name { get => project_name; set =>
project_name = value; }

        public void CreateProject(string nameProject, string path)
        {
            project_name = nameProject;
            path_ = path;
        }
        public void AddSelectParam(string param)
        {
            if (SelectParams_ == null)
                SelectParams_ = new LinkedList<string>();
            SelectParams_.AddLast(param);
        }
        public void RemoveSelectParam(string param)
        {
            if (SelectParams_ != null)
                SelectParams_.Remove(param);
        }
        public void AddGroupParam(string param)
        {
            if (GroupParams_ == null)
                GroupParams_ = new LinkedList<string>();

            SelectParams_.AddLast(param);

        }
    }

```

Листинг В.2 – Class Solution библиотеки projlb.dll

```

using System.Collections.Generic;

namespace ELPCClassLibrary
{
    enum AnalyzeType
    {
        EVENT_SEQUENCE,
        EVENT_TEMPLATE
    }
    enum AlgorithmType
    {
        GSP,
        SPAM,
        PrefixSpan
    }
    public class Solution
    {
        private string solution_name;
        private AnalyzeType analyzeType_;
        private LinkedList<string> SelectParams_;
        private LinkedList<string> GroupParams_;
    }
}

```

```

private LinkedList<string> AggParams_;
private AlgorithmType AlgType_;
private LinkedList<string> AnalyzeParams_;
private LinkedList<string> AnalyzeResult_;

    public LinkedList<string> SelectParams { get =>
SelectParams_; set => SelectParams_ = value; }
    public LinkedList<string> GroupParams { get =>
GroupParams_; set => GroupParams_ = value; }
    public LinkedList<string> AggParams { get => AggParams_;
set => AggParams_ = value; }
    public LinkedList<string> AnalyzeParams { get =>
AnalyzeParams_; set => AnalyzeParams_ = value; }
    public LinkedList<string> AnalyzeResult { get =>
AnalyzeResult_; set => AnalyzeResult_ = value; }
    internal AlgorithmType AlgType { get => AlgType_; set =>
AlgType_ = value; }
    public string Solution_name { get => solution_name; set =>
solution_name = value; }
    internal AnalyzeType AnalyzeType_ { get => analyzeType_;
set => analyzeType_ = value; }

    public void AddSelectParam(string param)
    {
        if (SelectParams_ == null)
            SelectParams_ = new LinkedList<string>();
        SelectParams_.AddLast(param);
    }
    public void RemoveSelectParam(string param)
    {
        if (SelectParams_ != null)
            SelectParams_.Remove(param);
    }
    public void AddGroupParam(string param)
    {
        if (GroupParams_ == null)
            GroupParams_ = new LinkedList<string>();

        SelectParams_.AddLast(param);
    }

    public void RemoveGroupParam(string param)
    {
        if (GroupParams_ != null)
            GroupParams_.Remove(param);
    }
    public void AddAggParam(string param)
    {
        if (AggParams_ == null)
            AggParams_ = new LinkedList<string>();

```

```

        AggParams_.AddLast(param);
    }
    public void RemoveAggParam(string param)
    {
        if (AggParams_ != null)
            AggParams_.Remove(param);
    }
}

```

Листинг В.3 – Код библиотеки projlb.dll

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;

namespace seqminelb
{
    public class SeqMineLib
    {
        public struct AlgData
        {
            public int id;
            public string name;
            public string tip;
        }

        public struct KValues
        {
            public Dictionary<string, double> data;
        }

        public static AlgData[] getAlgList()
        {
            return new AlgData[] {
                new AlgData { id = 0, name = "GSP", tip = "GSP-
алгоритм"},
                new AlgData { id = 1, name = "SPAM", tip = "SPAM-
алгоритм" },
                new AlgData { id = 2, name = "PREFIX_SPAN", tip =
"Алгоритм PrefixSpan" }
            };
        }

        private static int SeqGSP(List<string> eventLog, KValues
parameters, out List<string> result)
        {
            public static List<Item> countItem = new List<Item>();
            public static List<Item> frequent = new List<Item>();

```

```

        public static List<Sequence> transact = new
List<Sequence>();
        public static List<Sequence> sequences = new
List<Sequence>();
        public static List<Sequence> freqSequences = new
List<Sequence>();
        public static List<Sequence> candid = new
List<Sequence>();
        public static List<Sequence> frequentCandid = new
List<Sequence>();
        public static List<Sequence> interCandid = new
List<Sequence>();
        public static bool keepOnStage3 = true;
        public static bool FirstOfStage3 = true;
        public static int minSup;
        Stopwatch stopwatch = new Stopwatch();
        stopwatch.Start();

        List<string> sequenceList = new List<string>();
        minSup = (int) (minSup * sequenceList.Count/100);

static string[] DataSplitter(string data)
{
    int counter = 0;
    Dictionary<string, string> comps = new
Dictionary<string, string>();
    while (true)
    {
        int startComp = indexOf(data, '{');
        int endComp = indexOf(data, '}');
        if (startComp != -1)
        {
            string cypher = "GSP:" + counter.ToString();
            string tmp = data.Substring(startComp, endComp
- startComp + 1);
            string tmp2 = data.Substring(startComp + 1,
endComp - startComp - 1);
            data = data.Replace(tmp, cypher);
            comps.Add(cypher, tmp2);
            counter++;
        }
        else
        {
            break;
        }
    }

    string[] spliter = data.Split(',');
    for (int i = 0; i < spliter.Length; i++)
    {
        if (comps.ContainsKey(spliter[i]))
        {

```

```

        spliter[i] = comps[spliter[i]];
    }
}

return spliter;
}

static int indexOf(string data, char c)
{
    int index = -1;
    for (int i = 0; i < data.Length; i++)
    {
        if (data[i] == c)
        {
            index = i;
            break;
        }
    }
    return index;
}

}

public static class Stage1
{
    public static void StartStage()
    {
        //Рассчитывается поддержка события

        for (int i = 0; i < Program.transact.Count; i++)
        {
            initCounterBool(Program.countItem);
            Sequence sq = Program.transact.ElementAt(i);
            // Comp
            for (int j = 0; j < sq.seq.Count; j++)
            {
                Comp cmp = sq.seq.ElementAt(j);
                // Item
                for (int x = 0; x < cmp.length; x++)
                {
                    int index = -1;
                    for (int z = 0; z <
Program.countItem.Count; z++)
                    {
                        if
(Program.countItem.ElementAt(z).name ==
cmp.item.ElementAt(x).name)
                        {
                            index = z;
                            break;
                        }
                    }
                }
            }
        }
    }
}

```

```

        if (index != -1)
        {
            if
(!Program.countItem.ElementAt(index).added)
            {

Program.countItem.ElementAt(index).counter++;

Program.countItem.ElementAt(index).added = true;
            }
        }
        else
        {
            Program.countItem.Add(new
Item(cmp.item.ElementAt(x).name));
        }
    }
}
//Получаем частые события и добавляем их в список
for (int i = 0; i < Program.countItem.Count; i++)
{
    if (Program.countItem.ElementAt(i).counter >=
Program.minSup)
    {
Program.frequent.Add(Program.countItem.ElementAt(i));
    }
}

void initCounterBool(List<Item> list)
{
    for (int j = 0; j < list.Count; j++)
    {
        list.ElementAt(j).added = false;
    }
}
}
public static class Stage2
{
    public static void StartStage()
    {
        for (int i = 0; i < Program.frequent.Count - 1; i++)
        {
Program.sequences.Add(getSequence(Program.frequent.ElementAt(i).na
me, Program.frequent.ElementAt(i).name));
            for (int j = i + 1; j < Program.frequent.Count;
j++)
            {

```

```

Program.sequences.Add(getSequence(Program.frequent.ElementAt(i).name,
Program.frequent.ElementAt(j).name));

Program.sequences.Add(getSequence(Program.frequent.ElementAt(j).name,
Program.frequent.ElementAt(i).name));

Program.sequences.Add(getOtherSequence(Program.frequent.ElementAt(
i).name, Program.frequent.ElementAt(j).name));
    }
}

Program.sequences.Add(getSequence(Program.frequent.ElementAt(Program.
frequent.Count - 1).name,
Program.frequent.ElementAt(Program.frequent.Count - 1).name));

    // Получаем поддержку
    getSupport(Program.sequences);
    foreach (Sequence sq in Program.sequences)
    {
        if (sq.support >= Program.minSup)
        {
            int indx = -1;
            for (int i = 0; i <
Program.freqSequences.Count; i++)
            {
                if
(Program.freqSequences.ElementAt(i).compareTo(sq))
                {
Program.freqSequences.ElementAt(i).support++;
                    indx = i;
                    break;
                }
            }
            if (indx == -1)
            {
                Program.freqSequences.Add(sq);
            }
        }
    }
}

private static Sequence getSequence(string c1, string c2)
{
    Sequence sq = new Sequence();
    sq.createComp(c1);
    sq.createComp(c2);
    return sq;
}

private static Sequence getOtherSequence(string c1, string
c2)

```

```

{
    Sequence sq = new Sequence();
    sq.createComp(c1, c2);
    return sq;
}
public static void getSupport(List<Sequence> seq)
{
    foreach (Sequence sq in seq)
    {
        foreach (Sequence ssq in Program.transact)
        {
            int start = 0;
            bool getAll = true;
            foreach (Comp cp in sq.seq)
            {
                if (!ssq.findElemSeq(cp, ref start))
                {
                    getAll = false;
                    break;
                }
            }
            if (getAll)
            {
                sq.support++;
            }
        }
    }
}
public static class Stage3
{
    public static void StartStage()
    {
        if (Program.FirstOfStage3)
        {
            Program.FirstOfStage3 = false;
        }

        // Генерируется последовательности-кандидаты
        foreach (Sequence sq in Program.freqSequences)
        {
            string first = sq.withoutFirstItem();
            foreach (Sequence ssq in Program.freqSequences)
            {
                if (sq.ToString() != ssq.ToString())
                {
                    if (first == ssq.withoutLastItem())
                    {
                        Program.candid.Add(connectSequence(sq,
ssq));
                    }
                }
            }
        }
    }
}

```

```

    }
}

// Получаем поддержку кандидатов
Stage2.getSupport(Program.candid);

Program.frequentCandid.Clear();
foreach (Sequance sq in Program.candid)
{
    if (sq.support >= Program.minSup)
    {
        Program.frequentCandid.Add(sq);
    }
}

if (Program.frequentCandid.Count > 1)
{
    Program.candid.Clear();
    Program.freqSequences.Clear();
    Program.interCandid.Clear();
    foreach (Sequance sq in Program.frequentCandid)
    {
        Program.freqSequences.Add(sq);
        Program.interCandid.Add(sq);
    }
}
else
{
    Program.keepOnStage3 = false;
    if (Program.frequentCandid.Count != 1)
    {
        Program.frequentCandid = Program.interCandid;
    }
}

    stopwatch.Stop();
    result = new List<string>();
    return 0;
}

private static int SeqSPAM(List<string> eventLog, KValues
parameters, out List<string> result)
{
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    List<List<int>> sequences = new List<List<int>>
    int minSupport;

    List<List<int>> frequentPatterns =
SPAMAlgorithm(sequences, minSupport);

```

```

        static List<List<int>> SPAMAlgorithm(List<List<int>>
sequences, int minSupport)
        {
            List<List<int>> frequentPatterns = new
List<List<int>>();

            List<int> items = GetUniqueItems(sequences);

            Dictionary<int, int> itemSupportCounts =
GetItemSupportCounts(sequences, items);

            List<int> frequentItems =
GetFrequentItems(itemSupportCounts, minSupport);

            foreach (int item in frequentItems)
            {
                List<List<int>> projectedSequences =
GetProjectedSequences(sequences, item);

                List<List<int>> patterns =
SPAMAlgorithm(projectedSequences, minSupport);

                foreach (List<int> pattern in patterns)
                {
                    pattern.Insert(0, item);
                    frequentPatterns.Add(pattern);
                }
            }

            List<List<int>> frequentSingleItems =
GetFrequentSingleItems(itemSupportCounts, minSupport);

            foreach (List<int> itemset in frequentSingleItems)
            {
                frequentPatterns.Add(itemset);
            }
            return frequentPatterns;
        }

static List<int> GetUniqueItems(List<List<int>> sequences)
{
    List<int> items = new List<int>();

    foreach (List<int> sequence in sequences)
    {
        foreach (int item in sequence)
        {
            if (!items.Contains(item))
            {
                items.Add(item);
            }
        }
    }
}

```

```

    }
    return items;
}

static Dictionary<int, int>
GetItemSupportCounts(List<List<int>> sequences, List<int> items)
{
    Dictionary<int, int> itemSupportCounts = new
Dictionary<int, int>();

    foreach (List<int> sequence in sequences)
    {
        foreach (int item in sequence)
        {
            if (items.Contains(item))
            {
                if (itemSupportCounts.ContainsKey(item))
                {
                    itemSupportCounts[item]++;
                }
                else
                {
                    itemSupportCounts[item] = 1;
                }
            }
        }
    }
    return itemSupportCounts;
}

static List<int> GetFrequentItems(Dictionary<int, int>
itemSupportCounts, int minSupport)
{
    List<int> frequentItems = new List<int>();

    foreach (KeyValuePair<int, int> itemSupportCount in
itemSupportCounts)
    {
        if (itemSupportCount.Value >= minSupport)
        {
            frequentItems.Add(itemSupportCount.Key);
        }
    }
    return frequentItems;
}

static List<List<int>>
GetProjectedSequences(List<List<int>> sequences, int item)
{
    List<List<int>> projectedSequences = new
List<List<int>>();

```

```

        foreach (List<int> sequence in sequences)
        {
            int index = sequence.IndexOf(item);

            if (index != -1)
            {
                List<int> projectedSequence =
sequence.GetRange(index + 1, sequence.Count - index - 1);
                projectedSequences.Add(projectedSequence);
            }
        }
        return projectedSequences;
    }

    static List<List<int>>
GetFrequentSingleItems(Dictionary<int, int> itemSupportCounts, int
minSupport)
    {
        List<List<int>> frequentSingleItems = new
List<List<int>>();

        foreach (KeyValuePair<int, int> itemSupportCount in
itemSupportCounts)
        {
            if (itemSupportCount.Value >= minSupport)
            {
                frequentSingleItems.Add(new List<int>() {
itemSupportCount.Key });
            }
        }
        return frequentSingleItems;
    }

    stopwatch.Stop();
    result = new List<string>();
    return 0;
}

private static int SeqPrefixSpan(List<string> eventLog,
KValues parameters, out List<string> result)
{
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    private readonly SequenceDatabase _sequenceDatabase;
    private readonly Dictionary<uint, int> _iExtensions = new
Dictionary<uint, int>();
    private readonly Dictionary<uint, int> _sExtensions = new
Dictionary<uint, int>();
    private IList<Sequence> _lLengthSequentialPatterns;

    private void PrefixSpan()
    {
        _allSequentialPatterns.Clear();
    }
}

```

```

        ProjectedDatabase projectedDatabase =
        _sequenceDatabase.ConvertToProjectedDatabase();

        for (int i = 0; i < _lLengthSequentialPatterns.Count; ++i)
        {
            Sequence seqPattern = _lLengthSequentialPatterns[i];

            _allSequentialPatterns.Add(seqPattern);

            LogPattern(seqPattern);

            ProjectedDatabase freqItemProjectedDb =
            ConstructProjectedDatabase(seqPattern, projectedDatabase);

            if (freqItemProjectedDb == null ||
            !freqItemProjectedDb.IsEmpty) continue;

            PrefixSpan(freqItemProjectedDb);
        }

    private void PrefixSpan(ProjectedDatabase projectedDatabase)
    {
        if (Stopped) return;

        _sExtensions.Clear();
        _iExtensions.Clear();

        projectedDatabase.FindLocalFrequentItems(_minSupport,
        _sExtensions, _iExtensions);

        var newSequentialPatterns = new List<Sequence>();

        foreach (var pair in _sExtensions.OrderBy(e => e.Key))
        {
            newSequentialPatterns.Add(
                Sequence.CreateNewBySStep(projectedDatabase.Prefix,
                pair.Key, pair.Value));
        }
        foreach (var pair in _iExtensions.OrderBy(e => e.Key))
        {
            newSequentialPatterns.Add(
                Sequence.CreateNewByIStep(projectedDatabase.Prefix,
                pair.Key, pair.Value));
        }

        for (int i = 0; i < newSequentialPatterns.Count; ++i)
        {
            Sequence seqPattern = newSequentialPatterns[i];

            _allSequentialPatterns.Add(seqPattern);
        }
    }

```

```

        LogPattern(seqPattern);

        ProjectedDatabase seqPatternProjectedDatabase =
ConstructProjectedDatabase(seqPattern, projectedDatabase);

        if (seqPatternProjectedDatabase == null ||
!seqPatternProjectedDatabase.IsEmpty) continue;

        PrefixSpan(seqPatternProjectedDatabase);
    }
}

private ProjectedDatabase ConstructProjectedDatabase(Sequence
prefix, ProjectedDatabase sourceProjectedDatabase)
{
    var prefixLastItem = prefix.LastItem;
    var projectedDatabase = new ProjectedDatabase(prefix);

    int minSequencesInProjectedDb = sourceProjectedDatabase.Size
- (int) _minSupport;

    for (int sid = 0, lastSid = sourceProjectedDatabase.Size;
sid < lastSid; sid++)
    {
        int itemsetIndex;
        if ((itemsetIndex =
sourceProjectedDatabase[sid].Contains(prefixLastItem)) >= 0)
        {
            var pseudoSequence =
PseudoSequence.MakeProjection(sourceProjectedDatabase[sid],
prefix, itemsetIndex);

            if (pseudoSequence != null && pseudoSequence.IsEmpty)
            {
                projectedDatabase.Add(pseudoSequence);
                continue;
            }
        }

        if (--minSequencesInProjectedDb < 0) return null;
    }

    return projectedDatabase;
}

private IList<Sequence> Get1LengthSequentialPatterns()
{
    if (_miningSettings.OneLengthSequentialPatterns != null &&
_miningSettings.OneLengthSequentialPatterns.Count > 0)
    {
        return _miningSettings.OneLengthSequentialPatterns;
    }
}

```

```

        else
        {
            return
            _sequenceDatabase.FindOneLengthSequentialPattern(_minSupport);
        }
    }

    protected override void Mine()
    {
        _lLengthSequentialPatterns = GetlLengthSequentialPatterns();
        PrefixSpan();
    }
}

internal class ProjectedDatabase : List<PseudoSequence>
{
    private readonly Sequence _prefix;

    private IDictionary<uint, int> _iExtensions;

    private IDictionary<uint, int> _sExtensions;

    public ProjectedDatabase()
    {
        _prefix = new Sequence();
    }

    public ProjectedDatabase(Sequence prefix)
    {
        _prefix = prefix;
    }

    public int Size
    {
        get { return Count; }
    }

    public Sequence Prefix
    {
        get { return _prefix; }
    }

    public bool IsNotEmpty
    {
        get { return (Size > 0); }
    }

    public override string ToString()
    {
        return Prefix.ToString();
    }
}

```

```

private void FindLocalItems()
{
    _sExtensions = new Dictionary<uint, int>();
    _iExtensions = new Dictionary<uint, int>();

    var prefixLastItemset = Prefix[Prefix.Size - 1];

    var alreadyAddedSExt = new HashSet<uint>();
    var alreadyAddedIExt = new HashSet<uint>();

    for (int sid = 0, lastSid = Size; sid < lastSid; sid++)
    {
        alreadyAddedSExt.Clear();
        alreadyAddedIExt.Clear();

        for (int tid = 0, lastTid = this[sid].Size; tid <
lastTid; tid++)
        {
            var actualItemset = this[sid][tid];

            var isLastItemsetOfPrefixSubsetOfActualItemset =
false;
            var isActualItemsetPostfix = (tid == 0 &&
this[sid].IsLastItemIExtension);

            for (int itemId = 0, lastItemId =
actualItemset.Length; itemId < lastItemId; itemId++)
            {
                uint actualItem = actualItemset[itemId];

                if (isActualItemsetPostfix)
                {
                    if (!alreadyAddedIExt.Add(actualItem)) continue;

                    IncrementSupport(_iExtensions, actualItem);
                }
                else
                {
                    if (alreadyAddedSExt.Add(actualItem))
                    {
                        IncrementSupport(_sExtensions, actualItem);
                    }

                    if (!isLastItemsetOfPrefixSubsetOfActualItemset)
                    {
                        isLastItemsetOfPrefixSubsetOfActualItemset =
prefixLastItemset.IsSubsetOf(actualItemset, itemId - 1);
                    }

                    if (isLastItemsetOfPrefixSubsetOfActualItemset &&
alreadyAddedIExt.Add(actualItem))

```

```

        {
            IncrementSupport(_iExtensions, actualItem);
        }
    }
}

private static void IncrementSupport(IDictionary<uint, int>
extensionContainer, uint extension)
{
    int support;
    extensionContainer.TryGetValue(extension, out support);
    extensionContainer[extension] = ++support;
}

public void FindLocalFrequentItems(
    double minSupport,
    IDictionary<uint, int> sequenceExtensionsItems,
    IDictionary<uint, int> itemsetExtensionsItems)
{
    FindLocalItems();

    foreach (var pair in _sExtensions.Where(pair => pair.Value
    >= minSupport))
    {
        sequenceExtensionsItems.Add(pair.Key, pair.Value);
    }

    foreach (var pair in _iExtensions.Where(pair => pair.Value
    >= minSupport))
    {
        itemsetExtensionsItems.Add(pair.Key, pair.Value);
    }
}

internal class PseudoSequence
{
    private readonly uint[][] _items;
    private readonly uint[] _firstItemset;
    private readonly int _offset;
    private readonly Sequence _prefix;
    private readonly bool _isLastItemIExtension;
    internal IList<int> _firstInstances;

    public PseudoSequence(Sequence sequence)
    {
        _items = sequence._items;
        _firstItemset = _items[0];
        _firstInstances = new List<int>() { -1 };
    }
}

```

```

    }

    public PseudoSequence(Sequence prefix, PseudoSequence
sequence, int offset, bool isLastItemIExtension)
    {
        _offset = offset;
        _prefix = prefix;
        _items = sequence._items;
        _isLastItemIExtension = isLastItemIExtension;

        if (offset < _items.Length)
        {
            _firstItemset = _items[offset];
        }

        SetFirsInstanceOfPrefix(sequence);
    }

    public PseudoSequence(Sequence prefix, PseudoSequence
sequence, int offset, uint[] firstItemset, bool
isLastItemIExtension)
        : this(prefix, sequence, offset, isLastItemIExtension)
    {
        _firstItemset = firstItemset;
    }

    internal uint[][] RawItems
    {
        get { return _items; }
    }

    public int Size
    {
        get { return _items.Length - _offset; }
    }

    public bool IsNotEmpty
    {
        get { return (Size > 0); }
    }

    public uint[] this[int itemsetIndex]
    {
        get { return itemsetIndex == 0 ? _firstItemset :
_items[_offset + itemsetIndex]; }
    }

    public bool IsLastItemIExtension
    {
        get { return _isLastItemIExtension; }
    }

    public override string ToString()
    {

```

```

var builder = new StringBuilder();

for (int i = 0, j = Size; i < j; i++)
{
    uint[] itemset = this[i];

    if (i > 0)
        builder.Append(" ");

    if (itemset.Length > 1)
        builder.Append("(");

    for (int k = 0; k < itemset.Length; k++)
    {
        if (k > 0)
            builder.Append(" ");

        builder.Append(itemset[k]);
    }

    if (itemset.Length > 1)
        builder.Append(")");
}

builder.Append(">");

return builder.ToString();
}

public static PseudoSequence MakeProjection(PseudoSequence
sequence, Sequence prefix, int firstItemsetIndex = 0, bool
removeEmpty = true)
{
    var prefixLastItem = prefix.LastItem;
    var lastItemsetIndex = sequence.Size - 1;
    var prefixLastItemset = prefix[prefix.Size - 1];
    var isPrefixLastItemsetIExtensions =
prefix.IsLastItemIExtension;

    for (int tid = firstItemsetIndex, lastTid = sequence.Size;
tid < lastTid; tid++)
    {
        var isActualItemsetPostfix = (tid == 0 &&
sequence.IsLastItemIExtension);
        if (!isPrefixLastItemsetIExtensions &&
isActualItemsetPostfix) continue;

        var actualItemset = sequence[tid];
        var lastItemIndexInActualItemset = actualItemset.Length -
1;
        for (var itemId = 0; itemId < actualItemset.Length;
itemId++)

```

```

        {
            if (actualItemset[itemId] != prefixLastItem) continue;
            if (removeEmpty && tid == lastItemsetIndex && itemId ==
lastItemIndexInActualItemset) return null;

            prefixLastItemset.IsSubsetOf(actualItemset, itemId))
        {
            return itemId == lastItemIndexInActualItemset
            new PseudoSequence(prefix, sequence,
sequence._offset + tid + 1, false);
            new PseudoSequence(prefix, sequence,
sequence._offset + tid, sequence[tid].Slice(itemId + 1), true);
        }
    }
}
return null;
}

private void SetFirsInstanceOfPrefix(PseudoSequence sequence)
{
    _firstInstances = new List<int>(sequence._firstInstances);

    if (_prefix.IsLastItemIExtension)
    {
        _firstInstances[_firstInstances.Count - 1] =
_isLastItemIExtension ? _offset : _offset - 1;
    }
    else
    {
        _firstInstances.Add(_isLastItemIExtension ? _offset :
_offset - 1);
    }
}

stopwatch.Stop();
result = new List<string>();
return 0;
}

public static int ExecAlg(string algType, List<string>
eventLog, KValues parameters, out List<string> result)
{
    switch (algType)
    {
        case "GSP":
            return SeqGSP(eventLog, parameters, out
result);
        case "SPAM":
            return SeqSPAM(eventLog, parameters, out
result);
        case "PREFIX_SPAN":

```

```

        return SeqPrefixSpan(eventLog, parameters, out
result);
        default:
            result = new List<string>();
            return -1;
    }
}
}
}

```

Листинг В.4 – Class Form1

```

using ELPClassLibrary;
using System.ComponentModel;
using System.Diagnostics.Eventing.Reader;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            Program.f1 = this;
            InitializeComponent();
        }
        private void добавитьНовоеToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            NewSolution newForm = new NewSolution();
            newForm.Show();
        }

        private void toolStripButton1_Click(object sender,
EventArgs e)
        {
            NewProject newForm = new NewProject();
            newForm.Show();
        }
        private void
параметрыАлгоритмаToolStripMenuItem_Click_Click(object sender,
EventArgs e)
        {
            Algorithm newForm = new Algorithm();
            newForm.Show();
        }

        private void toolStripButton6_Click(object sender,
EventArgs e)
        {
            Algorithm newForm = new Algorithm();

```

```

        newForm.Show();
    }
private void просмотрСправкиToolStripMenuItem_Click(object sender,
EventArgs e)
{
    Spravka newForm = new Spravka();
    newForm.Show();
}

private void oПрограммеToolStripMenuItem_Click(object
sender, EventArgs e)
{
    OProgramm newForm = new OProgramm();
    newForm.Show();
}

private void создатьToolStripMenuItem_Click(object sender,
EventArgs e)
{
    NewProject newForm = new NewProject();
    newForm.Show();
}

private void toolStripButton5_Click(object sender,
EventArgs e)
{
    NewSolution newForm = new NewSolution();
    newForm.Show();
}

private void выходToolStripMenuItem_Click(object sender,
EventArgs e)
{
    Application.Exit();
}

private void toolStripButton9_Click(object sender,
EventArgs e)
{
    Application.Exit();
}

private void свойстваToolStripMenuItem_Click(object
sender, EventArgs e)
{
    PropertiesProject newForm = new PropertiesProject();
    newForm.Show();
}

private void свойстваToolStripMenuItem1_Click(object
sender, EventArgs e)
{

```

```

        PropertiesSolution newForm = new PropertiesSolution();
        newForm.Show();
    }

    private void подключениеКБДToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        DB newForm = new DB();
        newForm.Show();
    }

    private void загрузкаЖурналаToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        DownloadLog newForm = new DownloadLog();
        newForm.Show();
    }

    private void
загрузкаЖурналаИзФайлаToolStripMenuItem_Click(object sender,
EventArgs e)
    {
    }

    private void toolStripLabel1_Click(object sender,
EventArgs e)
    {
        DownloadLog newForm = new DownloadLog();
        newForm.Show();
    }

    private void treeView1_AfterSelect(object sender,
TreeViewEventArgs e)
    {
        string nameP = Program.project.Project_name;
        string nameS = Program.solution.Solution_name;
        TreeNode node = treeView1.SelectedNode;
        switch (node.Text)
        {

            case "Параметры алгоритма":
                Algorithm newForm = new Algorithm();
                newForm.TopLevel = false;
                this.panel4.Controls.Add(newForm);
                newForm.Dock = DockStyle.Fill;
                newForm.Show();
                break;
            case "Подключенная БД":
                DB formDB = new DB();
                formDB.TopLevel = false;
                this.panel4.Controls.Add(formDB);

```

```

        formDB.Dock = DockStyle.Fill;
        formDB.Show();
        break;
    case "Загруженный набор данных":
        DownloadLog formLog = new DownloadLog();
        formLog.TopLevel = false;
        this.panel4.Controls.Add(formLog);
        formLog.Dock = DockStyle.Fill;
        formLog.Show();
        break;
    }
    Program.projectTest.project_name = nameP;
    Program.projectTest.path_ = Program.project.Path;
    Program.projectTest.LogType_ = "Файл";
    propertyGrid1.SelectedObject = Program.project;

    }
    if (node.Text == nameS)
    {
        propertyGrid1.SelectedObject = Program.project;
    }

```

Листинг B.5 – Class NewProject : Form

```

using System.Collections.Generic;
using System.Linq;
public NewProject()
{
    InitializeComponent();
}
private void button2_Click(object sender, EventArgs e)
{
    label6.Visible = true;
    if (textBox1.Text.Length < 1)
    {
        label6.Text = "имя не должно быть пустым";
        label6.Location = new Point(262, 103);
    }
    else if (textBox1.Text.Length > 40)
    {
        label6.Text = "слишком длинное имя";
        label6.Location = new Point(289, 103);
    }
    else
    {
        Program.f1.label5.Visible = false;
        Program.f1.treeView1.Visible = true;
        Program.project.CreateProject(textBox1.Text,
        textBox2.Text);
        Program.f1.toolStripLabel1.Text = "Проект: " +
        Program.project.Project_name;
    }
}

```

```

        Program.fl.treeView1.Nodes[0].Text =
Program.project.Project_name;
        Close();
    }
    private void button3_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        var path_dialog = new FolderBrowserDialog();
        if (path_dialog.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
        {
            textBox2.Text = path_dialog.SelectedPath;
        }
    }

```

Листинг В.6 – Class DownloadLog : Form

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WinFormsApp1
{
    public partial class DownloadLog : Form
    {
        public DownloadLog()
        {
            Program.dll = this;
            InitializeComponent();
            radioButton1.CheckedChanged += new
EventHandler(radioButton1_CheckedChanged);
            radioButton2.CheckedChanged += new
EventHandler(radioButton2_CheckedChanged);
        }

        private void radioButton1_CheckedChanged(object sender,
EventArgs e)
        {
            RadioButton radioButton = sender as RadioButton;

```

```

        if (radioButton1.Checked)
        {
            groupBox2.Enabled = false;
            groupBox1.Enabled = true;
        }
        else if (radioButton2.Checked)
        {
            groupBox1.Enabled = false;
            groupBox2.Enabled = true;
        }
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (textBox1.Text.Length < 1 && groupBox2.Enabled ==
true)
        {
            label8.Visible = true;
            label8.Text = "выберите файл";
        }
        else if (linkLabel1.Text == "Подключить базу данных"
&& groupBox1.Enabled == true)
        {
            label9.Visible = true;
            label9.Text = "подключитесь к БД";
        }
        else if (comboBox1.Text.Length < 1 &&
groupBox1.Enabled == true)
        {
            label9.Visible = true;
            label9.Text = "выберите таблицу БД";
        }
        else {
            string pathFile = textBox1.Text;
            DataTable dt = new DataTable();
            string[] lines =
System.IO.File.ReadAllLines(pathFile);
            if (lines.Length > 0)
            {
                //first line to create header
                string firstLine = lines[0];
                string[] headerLabels = firstLine.Split(',');
                foreach (string headerWord in headerLabels)
                {
                    dt.Columns.Add(new
DataColumn(headerWord));
                }
                //For Data
                for (int i = 1; i < lines.Length; i++)
                {
                    string[] dataWords = lines[i].Split(',');
                    DataRow dr = dt.NewRow();

```

```

        int columnIndex = 0;
        foreach (string headerWord in
headerLabels)
        {
            dr[headerWord] =
dataWords[columnIndex++];
        }
        dt.Rows.Add(dr);
    }
}
if (dt.Rows.Count > 0)
{
    ReadLog rl = new ReadLog();
    rl.dataGridView1.DataSource = dt;
    rl.dataGridView1.DefaultCellStyle.Font = new
Font("Constantia", 10);

    rl.dataGridView1.ColumnHeadersDefaultCellStyle.Font = new
Font("Constantia", 10);
    rl.TopLevel = false;
    Program.f1.panel4.Controls.Add(rl);
    rl.Dock = DockStyle.Fill;
    rl.Show();
}
string type = radioButton2.Text;
if (radioButton1.Checked)
{
    type = radioButton1.Text;
}
Program.f1.label2.Text = "Тип журнала: " + type;
Close();
}
}
private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
    DB db = new DB();
    db.Show();
}
private void button2_Click(object sender, EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    if (ofd.ShowDialog() == DialogResult.OK)
    {
        textBox1.Text = ofd.FileName;
        string nameF = ofd.SafeFileName;
        Program.f1.label3.Text = "Имя БД/имя файла: " +
nameF;

        Program.f1.treeView1.Nodes[0].Nodes[0].Nodes[0].Text = nameF;
    }
}
}

```

Листинг В.7 – Class NewSolution : Form

```

using System.Collections.Generic;
using System.Linq;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormsApp1
{
    public partial class NewSolution : Form
    {
        public NewSolution()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            label6.Visible = true;
            if (textBox1.Text.Length < 1)
            {
                label6.Text = "имя не должно быть пустым";
                //label6.Location = new Point(262, 103);
            }
            else if (textBox1.Text.Length > 30)
            {
                label6.Text = "слишком длинное имя";
                //label6.Location = new Point(289, 103);
            }
            else
            {
                string n = textBox1.Text;
                Program.fl.treeView1.Nodes[0].Nodes.Add(n);
                Program.project.CreateSolution(n);
                Close();
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}

```

Листинг В.8 – Class OProgramm : Form

```

using System.Collections.Generic;

```

```

using System.Linq;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormsApp1
{
    public partial class OProgramm : Form
    {
        public OProgramm()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}

```

Листинг В.9 – Class PropertiesProject : Form

```

using System.Collections.Generic;
using System.Linq;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormsApp1
{
    public partial class PropertiesProject : Form
    {
        public PropertiesProject()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Close(); }}}

```

ПРИЛОЖЕНИЕ Г
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ
(рекомендуемое)



Новое решение

Создание решения

Имя решения

Создать Отмена

Рисунок Г.1 – Результаты тест-кейса 1

The screenshot shows a window titled "Новое решение" (New Solution). It has a light blue header bar with the title and a close button (X). Below the header, the main area has a light blue background with the text "Создание решения" (Creating solution). In the center, there is a label "Имя решения" (Solution name) followed by a text input field containing many 'n' characters. Below the input field, the text "слишком длинное имя" (name too long) is displayed in red. At the bottom, there are two buttons: "Создать" (Create) and "Отмена" (Cancel).

Рисунок Г.2 – Результаты тест-кейса 2



Новое решение

Создание решения

Имя решения

в имени присутствуют недопустимые символы

Создать Отмена

Рисунок Г.3 – Результаты тест-кейса 3



Новое решение

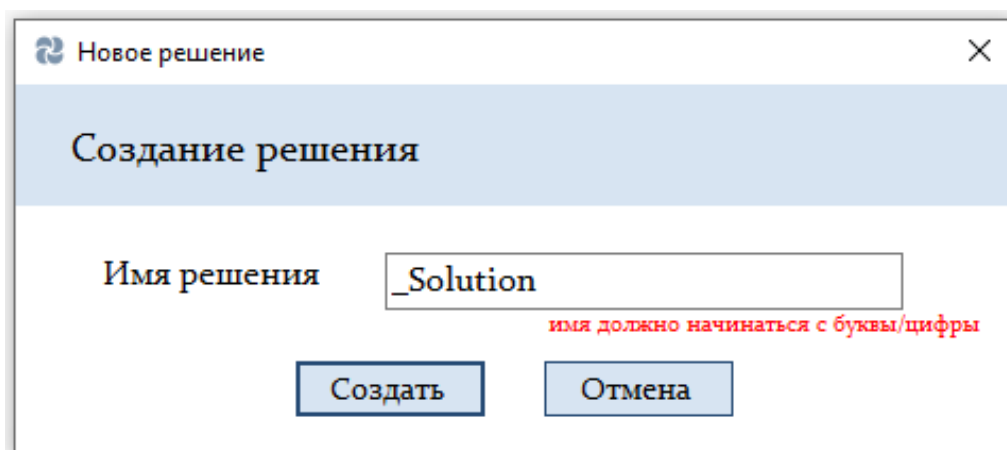
Создание решения

Имя решения

имя не должно быть пустым

Создать Отмена

Рисунок Г.4 – Результаты тест-кейса 4



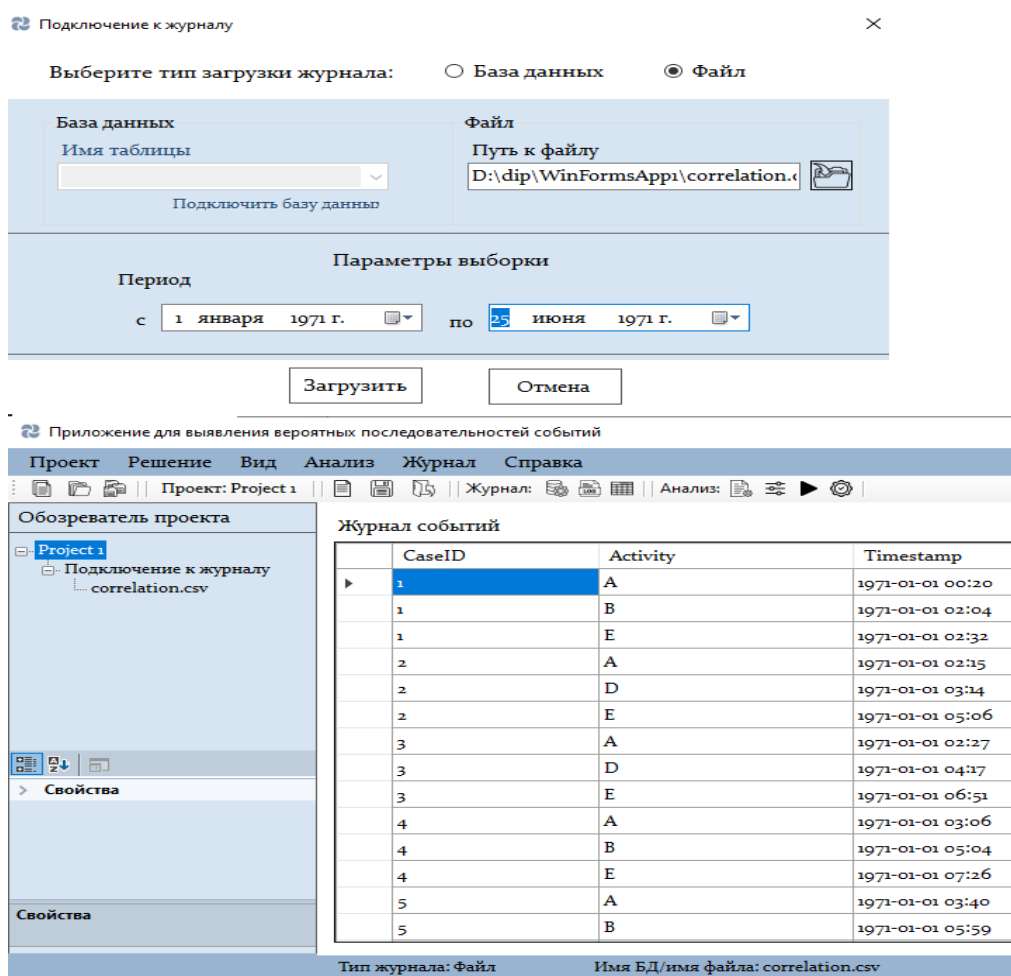
Новое решение

Создание решения

Имя решения

имя должно начинаться с буквы/цифры

Рисунок Г.5 – Результаты тест-кейса 5



Подключение к журналу

Выберите тип загрузки журнала: ☐ База данных ☒ Файл

База данных
Имя таблицы

Файл
Путь к файлу

Параметры выборки
Период
с по

Приложение для выявления вероятных последовательностей событий

Проект Решение Вид Анализ Журнал Справка

Проект: Project 1 | Журнал: Анализ:

Обозреватель проекта

- Project 1
 - Подключение к журналу correlation.csv

Журнал событий

CaseID	Activity	Timestamp
1	A	1971-01-01 00:20
1	B	1971-01-01 02:04
1	E	1971-01-01 02:32
2	A	1971-01-01 02:15
2	D	1971-01-01 03:14
2	E	1971-01-01 05:06
3	A	1971-01-01 02:27
3	D	1971-01-01 04:17
3	E	1971-01-01 06:51
4	A	1971-01-01 03:06
4	B	1971-01-01 05:04
4	E	1971-01-01 07:26
5	A	1971-01-01 03:40
5	B	1971-01-01 05:59

Тип журнала: Файл Имя БД/имя файла: correlation.csv

Рисунок Г.6 – Результаты тест-кейса 6

Подключение к журналу

Выберите тип загрузки журнала: ☐ База данных ☒ Файл

База данных

Имя таблицы

Подключить базу данных

Файл

Путь к файлу

выберите файл

Параметры выборки

Период

с 22 июня 2023 г. по 22 июня 2023 г.

Загрузить Отмена

Рисунок Г.7 – Результаты тест-кейса 7

Подключение к базе данных

Имя сервера localhost

База данных AnalysisLog

Логин postgres

Пароль *****

Порт 5422

Проверить Подключить Отмена

Подключение к журналу

Выберите тип загрузки журнала: ☒ База данных ☐ Файл

База данных

Имя таблицы

Test

Подключена база данных: AnalysisLog

Файл

Путь к файлу

Параметры выборки

Период

с 1 января 2015 г. по 11 февраля 2016 г.

Загрузить Отмена

Рисунок Г.8 – Результаты тест-кейса 8

Подключение к журналу

Выберите тип загрузки журнала: ☒ База данных ☐ Файл

База данных

Имя таблицы подключитесь к БД

[Подключить базу данных](#)

Файл

Путь к файлу

Параметры выборки

Период

с 22 июня 2023 г. по 22 июня 2023 г.

Загрузить Отмена

Рисунок Г.9— Результаты тест-кейса 9

Параметры алгоритма

Выберите алгоритм: выберите алгоритм

Параметры алгоритма

Укажите минимальную поддержку: 50 %

Сохранить Отмена

Рисунок Г.10 – Результаты тест-кейса 10

Параметры алгоритма

Выберите алгоритм:

PrefixSpan

Параметры алгоритма

Укажите минимальную поддержку: 50 %

Сохранить Отмена

Рисунок Г.11 – Результаты тест-кейса 11

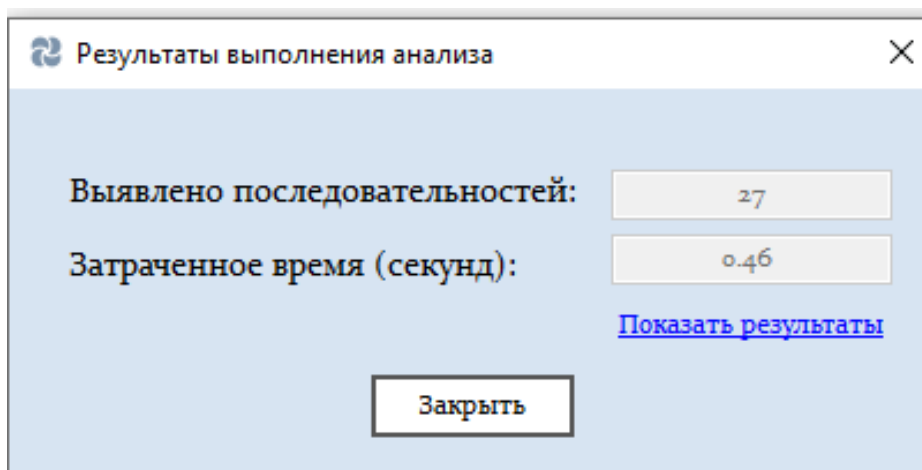


Рисунок Г.12 – Результаты тест-кейса 12

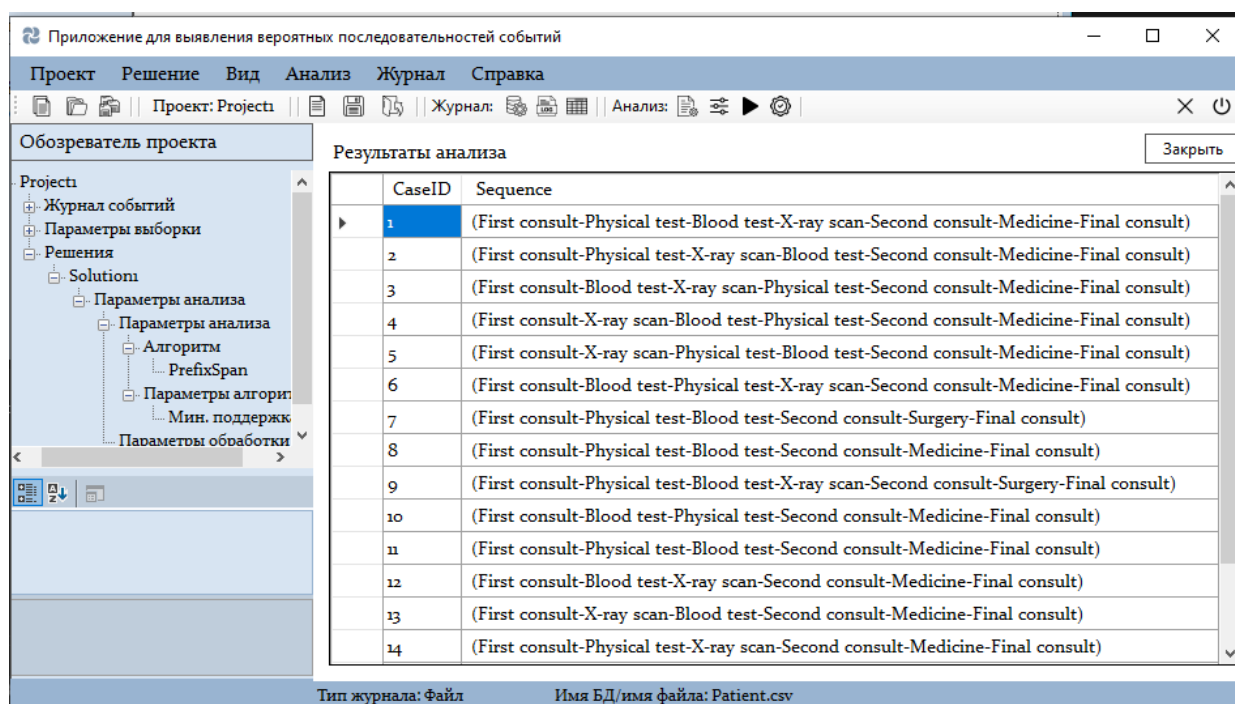


Рисунок Г.13 – Результаты тест-кейса 13