

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

АЛГОРИТМЫ И ПРОГРАММНЫЕ СРЕДСТВА
ВЫЯВЛЕНИЯ ШАБЛОНОВ СОСТАВНЫХ СОБЫТИЙ НА
ОСНОВЕ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ЖУРНАЛОВ
СОБЫТИЙ

Студент

10.06.24 Печаткин Константин Владимирович
(подпись, дата) (ФИО полностью)

Руководитель

10.06.2024 С.В. Шибанов
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

10.06.2024 А.С. Бычков
(подпись, дата) (фамилия, инициалы)

Рецензент

10.06.2024 А.Ю. Прохоренко
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.24 № 11)

Заведующий кафедрой

М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

14.06.2024 № 21)

Секретарь ГЭК

Е.И. Гурин
(подпись) (фамилия, инициалы)

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 13 » 09 2022 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Печаткин Константин Владимирович гр. 22ВВИм2 факультета ВТ

2. Тема работы Алгоритмы и программные средства выявления шаблонов составных событий на основе интеллектуального анализа журналов событий

Тема утверждена приказом ПГУ № 763/0 от " 13 " 09 2022 г.

3. Руководитель магистерской диссертации 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 и моделирования шаблонов составных событий на основе интеллектуального анализа журнала событий.

Средства выполнения: Операционная система Windows

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

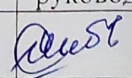
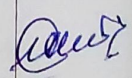
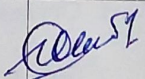
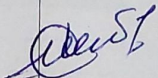
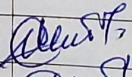
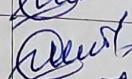
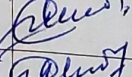
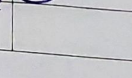
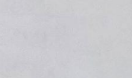
Язык программирования: C#

Технологии разработки: ООП, UML

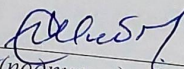
5. Перечень подлежащих разработке вопросов

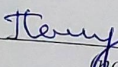
- 1) Анализ предметной области и постановка задачи на разработку
- 2) Сравнительный анализ алгоритмов и программных средств выявления вероятных последовательностей событий
- 3) Сравнительный анализ алгоритмов и программных средств выявления шаблонов составных событий
- 4) Разработка методики моделирования составных событий на основе алгоритмов секвенциального анализа и алгоритмов извлечения процессов
- 5) Проектирование программных средств
- 6) Разработка программных средств
- 7) Контроль качества программных средств
- 8) Экспериментальная оценка программных средств

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи на разработку	10%	13.09.2022	
2	Сравнительный анализ алгоритмов и программных средств выявления вероятных последовательностей событий	10%	26.09.2022	
3	Сравнительный анализ алгоритмов и программных средств выявления шаблонов составных событий	10%	10.12.2022	
4	Разработка методики моделирования составных событий на основе алгоритмов секвенциального анализа и алгоритмов извлечения процессов	15%	01.04.2023	
5	Проектирование программных средств	10%	10.09.2023	
6	Разработка программных средств	25%	10.02.2024	
7	Контроль качества программных средств	5%	01.03.2024	
8	Экспериментальная оценка программных средств	5%	10.04.2024	
9	Оформление пояснительной записки	5%	31.05.2024	

Дата выдачи задания " 13 " 09 20 22 г.

Руководитель магистерской диссертации 
(подпись, дата)

Задание к исполнению принял студент 
(подпись, дата)

Работу к защите допустить
Декан факультета  А.Р. Рязанов
13.06.2024 (подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 119 листов, 54 рисунка, 33 таблицы, 17 использованных источников и 3 приложения

СОБЫТИЕ, СОСТАВНОЕ СОБЫТИЕ, ПОСЛЕДОВАТЕЛЬНОСТЬ СОБЫТИЙ, СЕКВЕНЦИАЛЬНЫЙ АНАЛИЗ, ИЗВЛЕЧЕНИЕ ПРОЦЕССОВ, PROCESS MINING, ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ, ЖУРНАЛ СОБЫТИЙ

Объектом исследования: процессы анализа журналов в системе обработки событий.

Цель работы: исследование модельных представлений составных событий, методик и программных средств выявления, моделирования и оценки достоверности составных событий на основе данных журналов событий.

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

Задание: разработать методики, алгоритмы и программные средства для выявления и моделирования составных событий на основе интеллектуального анализа журнала событий.

Результат работы: разработаны алгоритмы и методики выявления и моделирования составных событий с использованием алгоритмов секвенциального анализа и алгоритмов извлечения процессов, которые реализованы в виде plugin-библиотек.

ПГУ1.090401.43.001 ПЗ

Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Печаткин К.В.	<i>Печаткин К.В.</i>	09.06.21	Алгоритмы и программные средства выявления шаблонов составных событий на основе интеллектуального анализа журналов событий Пояснительная записка	Лист	Лист
Провер.		Шибанов С.В.	<i>Шибанов С.В.</i>	10.06.21			Листов
Реценз.		<i>Митрохин М.А.</i>	<i>Митрохин М.А.</i>	10.06.21		3	119
Н. Контр.		Бычков А.С.	<i>Бычков А.С.</i>	10.06.21		Группа 22ВВИм2	
Утверд.		Митрохин М.А.	<i>Митрохин М.А.</i>	10.06.21			

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	10
ВВЕДЕНИЕ	11
1 Выявление шаблонов составных событий на основе интеллектуального анализа журналов событий	14
1.1 Актуальность задачи выявления шаблонов составных событий.....	14
1.1.1 Классификация событий	14
1.1.2 Классификация задач обработки событий	16
1.1.3 Выявление закономерностей возникновения событий в виде шаблонов составных событий	17
1.2 Обоснование выбора модельного представления шаблонов составных событий	19
1.2.1 Обзор модельных представлений шаблонов составных событий	20
1.2.2 Сравнительный анализ модельных представлений составного события	22
1.2.3 Модельные представления шаблонов составных событий на основе раскрашенных сетей Петри.....	23
1.3 Сравнительный анализ методов и алгоритмов выявления шаблонов составных событий	26
1.3.1 Классификация методов выявления составных событий	26
1.3.2 Обзор алгоритмов выявления шаблонов составных событий	27
1.3.3 Результаты сравнительного анализа алгоритмов выявления шаблонов составных событий	32
1.4 Сравнительный анализ программных средств выявления составных событий	33
1.4.1 Классификация программных средств для выявления составных событий	33

1.4.2 Обзор программных средств выявления составных событий.....	34
1.4.3 Результаты сравнительного анализа программных средств выявления составных событий.....	35
1.5 Выводы.....	35
2 Методика и алгоритмы выявления шаблонов составных событий	36
2.1 Методика выявления шаблонов составных событий.....	36
2.1.1 Обобщенное представление структуры журнала событий	36
2.1.2 Основные параметры процесса выявления шаблонов составных событий.....	37
2.1.3 Основные процессы моделирования шаблонов составных событий.....	38
2.2 Предварительная обработка журнала событий для формирования исходного набора событий.....	40
2.3 Алгоритмы выявления шаблонов последовательных событий	41
2.3.1 Алгоритм GSP	41
2.3.2 Алгоритм PrefixSpan.....	42
2.3.3 Алгоритм SPAM.....	44
2.4 Алгоритмы выявления шаблонов составных событий	45
2.4.1 Альфа-алгоритм	46
2.4.2 Эвристический алгоритм.....	47
2.4.3 Индуктивный алгоритм.....	49
2.5 Выводы.....	51
3 Программные средства выявления шаблонов составных событий	52
3.1 Назначение и функциональные возможности программных средств....	52
3.2 Модульная архитектура и схема функционирования программных средств.....	55

3.3 Библиотека классов для построения шаблонов последовательных событий	57
3.4 Библиотека классов для построения шаблонов составных событий.....	59
3.5 Библиотека классов для предварительной обработки журнала событий	61
3.6 Визуализация модельного представления составного события	62
3.7 Выводы.....	64
4 Контроль качества и экспериментальная оценка программных средств выявления составных событий	65
4.1 Процесс контроля качества и экспериментальная оценка программных средств	65
4.1.1 Выбор модели тестирования.....	65
4.1.2 Выбор метрик кода	67
4.2 Тестирование программных средств	67
4.2.1 Классы эквивалентности	68
4.2.2 Таблица решений	69
4.3 Оценка качества программного кода	71
4.4 Экспериментальная оценка программных средств	75
4.4.1 Планирование эксперимента	75
4.4.2 Проведение эксперимента.....	78
4.4.3 Анализ экспериментальных данных	82
4.5 Выводы.....	86
ЗАКЛЮЧЕНИЕ	87
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	89
ПРИЛОЖЕНИЕ А Результаты тестирования.....	92
ПРИЛОЖЕНИЕ Б Исходный код сигнатур классов plugin-библиотек.....	96

ПРИЛОЖЕНИЕ В Презентация	99
--------------------------------	----

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей магистерской диссертации применяются следующие сокращения и обозначения

БД – база данных

ИАД – интеллектуальный анализ данных (Data Mining)

КЭ – классы эквивалентности

ПС – программные средства

СА – секвенциальный анализ

ИП – извлечение процессов (Process Mining)

СУБД – система управления базами данных

UML – унифицированный язык моделирования (Unified Modeling Language)

ВВЕДЕНИЕ

Актуальность темы работы. С развитием информационных технологий объем доступной нам информации стремительно растет, и эта тенденция не замедляется. Поэтому необходимо постоянно улучшать средства обработки информации, в основном программные, которые способны эффективно обрабатывать огромные объемы данных за минимальное время. Развитие технологий обработки данных, объектно-ориентированного программирования и систем реального времени привело к появлению различных систем обработки событий, которые могут эффективно работать с этим объемом информации.

Разработка эффективных алгоритмов выявления составных событий имеет большое значение для различных областей применения. Например, в медицине анализ составных событий может помочь в обнаружении более сложных заболеваний, которые могут быть пропущены при анализе отдельных симптомов. В финансовой сфере анализ составных событий может помочь в обнаружении тенденций на рынке, связанных с взаимодействием различных финансовых инструментов. Кроме того, алгоритмы выявления составных событий позволят не только обнаруживать их, но и проводить более глубокий анализ. Например, программные средства могут автоматически выделять ключевые характеристики составных событий, проводить анализ влияния различных факторов на их происхождение и развитие, и т.д. Объектом исследования являются процессы анализа журналов в системах обработки событий. Предметом исследования являются модельные представления составных событий, методика, алгоритмы и программные средства выявления шаблонов составных событий на основе интеллектуального анализа журналов событий.

Цель и задачи работы. Целью работы является исследование модельных представлений составных событий, методик и программных средств выявления, моделирования и оценки достоверности составных событий на основе данных журналов событий.

Для достижения поставленной цели планируется решить ряд важных задач:

- формализация и модельное представление сложных составных событий,
- разработка методики выявления, моделирования и оценки достоверности шаблонов составных событий,
- разработка алгоритмов выявления, моделирования и оценки достоверности шаблонов составных событий,
- разработка программных средств выявления, моделирования и оценки достоверности шаблонов составных событий.

Научная новизна, оригинальность идей в работе. Разработана методика выявления, моделирования и оценки достоверности шаблонов составных событий, включающая в себя использование алгоритмов секвенциального анализа и модифицированных алгоритмов извлечения процессов.

Практическая значимость:

- разработано модельное представление сложных составных событий на основе раскрашенных сетей Петри,
- разработаны плагин-библиотеки для предварительной обработки и анализа журнала событий,
- разработаны программные средства для выявления и анализа сложных составных событий.

Методы проведенных исследований. Методы исследований основаны на теоретических положениях анализа данных, объектно-ориентированного программирования, алгебры событий, основанной на использовании операций алгебры логики и временных соотношений.

Основные положения, выносимые на защиту:

- алгоритмы и методика выявления и анализа шаблонов сложных составных событий, основанная на использовании алгоритмов секвенциального анализа и алгоритмов извлечения процессов,

– архитектура и реализация программных средств для выявления и анализа сложных составных событий,

Апробация работы. Основные положения и результаты работы докладывались на следующих конференциях:

– IX Международная научно-практическая конференция (школы-семинара) молодых ученых «Прикладная математика и информатика: современные исследования в области естественных и технических наук» (Тольятти, 2023), результат – диплом участника,

– X Всероссийская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы» (Пенза, 2023), результат – диплом участника,

– XV Международная научно-техническая конференция с элементами научной школы и конкурс научно-исследовательских работ для студентов, аспирантов и молодых ученых «Методы, средства и технологии получения и обработки измерительной информации («Шляндинские чтения – 2023»))» (Пенза, 2023), результат – диплом III степени,

– Всероссийская научно-практическая конференция молодых ученых «Наука, инновации и технологии: от идей к внедрению» (Комсомольск-на-Амуре, 2023), результат – диплом участника,

– XI Всероссийская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы» (Пенза, 2024), результат – диплом участника.

Структура и объем диссертации. Диссертация состоит из введения, 4 глав, заключения, списка использованных источников, включающего 3 источника, и 3 приложений. Материал диссертации изложен на 119 листах, включает 47 рисунков, 33 таблицы и 28 страниц приложений.

1 Выявление шаблонов составных событий на основе интеллектуального анализа журналов событий

В настоящее время подход к обработке событий претерпел значительные изменения. Развитие сетевой торговли, рост автоматизации и непрерывная интеграция привели к появлению информационных сред, которые регистрируют события внешнего мира. Сейчас организации используют различные методы обработки событий в режиме реального времени, что позволяет им не только быстро реагировать, но и принимать более эффективные решения.

1.1 Актуальность задачи выявления шаблонов составных событий

Анализ составных событий на сегодняшний день является актуальным и применяется во многих областях, таких как анализ поведения вредоносного программного обеспечения для его обнаружения, анализ действий нарушителей в информационных системах, анализ предпочтений пользователей на основе их поведения в системе, а также анализ журналов событий для выявления и устранения ошибок в работе системы.

1.1.1 Классификация событий

В общем случае события в различных системах обработки событий можно классифицировать по нескольким признакам, основными из которых являются уровень сложности, уровень соответствия шаблонам, уровень предсказуемости, уровень формализации и пр.

Классификация событий рассматриваются в ряде работ различных авторов. В работах [1, 2] события классифицируются на простые и сложные. В работе [2] более подробно приводится классификация сложных событий. Они могут агрегированными и составными.

Варианты общей классификации событий, обобщающей рассмотренные классификации, приводятся также в [3].

Обобщая имеющиеся классификации, можно констатировать, что по уровню сложности события могут быть классифицированы на элементарные события и сложные события (таблица 1). Элементарные события поступают

непосредственно от источников событий (датчиков, устройств, хранилищ данных, приложений и пр.), обнаруживаются по факту поступления данных о наступлении события, и представляют собой кортеж данных, описывающих контекст возникновения события.

Таблица 1 – Классификация событий по уровню сложности

Класс события	Описание
Элементарное	Событие, поступающее непосредственно от источника и которое невозможно разложить на более простые события
Сложное	Событие, образующееся комбинацией нескольких элементарных или сложных событий

Сложные события образуются комбинацией других элементарных и сложных событий, составляющих сложное событие. Сложные события обнаруживаются дополнительными специальными средствами на основе заданных алгоритмов и, в свою очередь, подразделяются на два класса – агрегированные события и составные события. Сложные агрегированные события обнаруживаются с помощью непрерывных запросов, агрегирующих в результирующем событии данные из составляющих событий. Сложные составные события описываются с помощью операций алгебры событий над составляющими событиями и обнаруживаются специальными программами на основе алгоритмов, работающими на соответствующих графовых, автоматных и сетевых моделях.

По уровню соответствия шаблонам события можно классифицировать на стандартные и нестандартные (таблица 2). События, отнесенные к классу стандартных событий, имеют четко определенную структуру и экземпляр события данного класса содержит все необходимые данные контекста событий.

Нестандартные события имеют отклонения от стандартной структуры, причинами которых могут быть сбои в работе сети, шумы, ошибки приложения и пр. Нестандартные события требуют специальной обработки при обнаружении. Наличие нестандартных событий в журналах событий должны быть учтены в алгоритмах анализа журналов событий.

Таблица 2 – Классификация событий по уровню соответствия шаблонам

Класс события	Описание
Стандартное	Событие, которое имеет стандартную структуру, соответствующую заранее заданному шаблону
Нестандартное	Событие, которое имеет отклонения от стандартной структуры, заранее заданной шаблоном по причине возможных сбоев в сети, «шумов» при передаче данных и ошибок приложений

По степени предсказуемости события классифицируются на ожидаемые и неожиданные (таблица 3). Появление неожиданного события в системах обработки событий может свидетельствовать как о сбоях в работе оборудования и приложений, так и о наличии неучтенных закономерностей в поведении источников событий.

Таблица 3 – Классификация событий по уровню предсказуемости

Класс события	Описание
Ожидаемое	Событие, которое ожидает один или несколько потоков обработки событий в системе
Неожиданное	Событие, которое не ожидает ни один из потоков обработки событий в системе

По уровню формализации события классифицируются на зарегистрированные и незарегистрированные в системе обработки событий. Появление незарегистрированного события может свидетельствовать, как о сбоях в работе аппаратно-программных средств комплекса обработки событий, так и о несогласованности описаний источников событий (метаданные, описывающие источник событий не согласуются с реальными данными от источника) и о динамических свойствах объектов наблюдения (появление нового объекта-источника событий, изменение свойств объекта наблюдения и пр.).

1.1.2 Классификация задач обработки событий

Обработка событий может включать в себя следующие виды задач:

- конструирование событий,
- обнаружение событий,
- выявление закономерностей в виде шаблонов составных событий.

Первый класс задач связан позволяет определять структуру событий на основе имеющихся данных. Включает в себя задачи по формированию и определению новых типов событий, их атрибутов и связей.

Задачи обнаружения событий направлены на выявление и идентификацию возникновения определенных событий в потоке данных. Это включает в себя поиск событий, которые соответствуют определенным шаблонам или условиям.

Последний класс задач связан с выявлением составных событий в потоке данных и построением шаблонов для них. Задачи этого типа включают в себя поиск типичных моделей поведения, часто встречающихся комбинаций событий и причинно-следственных связей между событиями.

Во время поиска типичных моделей поведения мы стремимся выявить общие сценарии или паттерны, которые часто встречаются в данных. Например, это может быть определенная последовательность событий, которая указывает на определенное действие или состояние.

При поиске часто встречающихся комбинаций мы ищем те комбинации событий, которые часто происходят вместе. Например, если событие «А» часто предшествует событию «В», мы можем выделить эту связь как типичную комбинацию.

При поиске причинно-следственных связей мы стараемся определить, какие события вызывают другие. Это позволит понять, какие события могут быть результатом других событий.

1.1.3 Выявление закономерностей возникновения событий в виде шаблонов составных событий

Как было сказано ранее, задача выявления закономерностей возникновения событий представляет собой поиск часто встречающихся комбинаций и различных взаимосвязей между событиями с целью выявления и моделирования составных событий с использованием журнала событий.

Рассмотрим, какие типы шаблонов могут использоваться для моделирования составных событий. Для этого могут использоваться:

- шаблоны последовательных событий,
- шаблоны составных событий.

При этом шаблоны последовательных событий могут быть классифицированы на последовательные и параллельные.

Последовательные шаблоны событий можно представить в следующем виде:

$$e_1 \rightarrow e_2 \rightarrow \dots \rightarrow e_n, t(e_i) < t(e_j), i \leq j,$$

где e_i – отдельный экземпляр события;

t – время возникновения экземпляра события.

Для событий в последовательном шаблоне определён строгий порядок по времени их возникновения, то есть, для событий e_1 и e_2 будет определен последовательный порядок тогда и только тогда, когда время возникновения события $t(e_2)$ строго больше времени возникновения события $t(e_1)$.

Рассмотрим параллельный шаблон. Его можно определить следующим образом:

$$e_1 \rightarrow e_2 \rightarrow \dots \rightarrow e_n, t(e_i) \leq t(e_j), i \leq j,$$

где e_i – отдельный экземпляр события, или множество экземпляров событий,

t – время возникновения экземпляра события.

Для событий в параллельном шаблоне определён частичный порядок по времени их возникновения, то есть время возникновения события $t(e_2)$ больше или равно времени возникновения события $t(e_1)$.

Шаблоны составных событий строятся с помощью операций алгебры событий [4] и временных соотношений (рисунок 1).

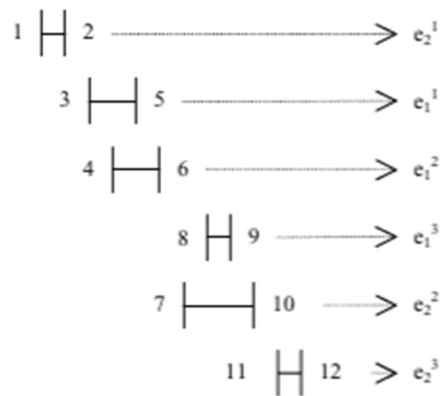


Рисунок 1 – Временные соотношения между событиями

Пример составного события, определяемого с помощью дизъюнкции, имеет вид:

$$(E_1 \vee E_2)(t) = E_1(t) \vee E_2(t),$$

где t – время возникновения составного события.

Этот шаблон можно интерпретировать следующим образом. Составное событие наступает в том случае, когда в указанный момент времени t происходит либо событие E_1 , либо событие E_2 .

1.2 Обоснование выбора модельного представления шаблонов составных событий

Составные события можно описывать, используя различные операции, объединенные в алгебру событий. Операции для моделирования составных событий представлены на рисунке 2.



Рисунок 2 – Операции моделирования составных событий

Данные операции очень похожи на те, которые используются в алгебре логики, но учитывают еще при этом и временной контекст событий. На рисунке выше представлена только часть операций, которые наиболее часто используются при моделировании составных событий. Они могут изменяться и модифицироваться в зависимости от систем и используемых ими инструментов [5].

1.2.1 Обзор модельных представлений шаблонов составных событий

В зависимости от типа задач обработки составных событий можно применять различные математические модели для представления событий. В данной работе особое внимание будет уделено графам, конечным автоматам и сетям Петри. Рассмотрим их более подробно.

Графовая модель определяется следующим образом:

$$G = (L, V),$$

где L – множество компонентов составного события (простых событий);

V – множество операций, определяемых алгеброй событий.

При этом операции между компонентами события записываются в следующем виде:

$$O = (Op, E_1, E_2),$$

где Op – тип операции (AND, OR и т.д.);

E_1 и E_2 – простые события, между которыми определена данная операция.

В модельном представлении составного события в виде графа листьями являются компоненты составного события (простые события) а узлы-родители определяют операцию (AND, OR и т.д.) над событиями, указанных в дочерних узлах-листьях. Данная модель достаточно удобна для интерпретации и визуализации составного события, однако, при росте взаимосвязей между

событиями-компонентами строить модель составного события становится невозможным [3].

Рассмотрим следующее модельное представление в виде конечного автомата. Эту модель можно представить следующим множеством:

$$M = (V, Q, q_0, F, \delta),$$

где V – множество всех событий, которые могут произойти;

Q – множество внутренних состояний автомата;

q_0 – начальное состояние (событие-инициатор);

F – множество конечных состояний (событий-терминаторов);

δ – функция переходов (операция алгебры логики).

Моделирование составного события с помощью данной модели включает в себя несколько этапов. Сначала на вход автомата поступает событие-инициатор, после которого автомат переходит в состояние приема событий-компонентов. Моделирование составного события завершается после поступления на вход одного из множества событий-терминаторов. После этого моделирование составного события можно считать завершенным. Конечный автомат в отличие от графа обладает формальным языком, позволяющим строить более точные модели составного события, однако ее также, как и граф, становится трудно строить и поддерживать при росте взаимосвязей между компонентами составного события [6].

Рассмотрим Сеть Петри в качестве модельного представления. Данная модель представляет собой следующее множество:

$$PN = (P, T, F),$$

где P – конечное множество мест (событий, которые могут произойти);

T – конечное множество переходов (операции алгебры логики);

F – множество дуг.

В данной модели позициями (местами) определяются компоненты составного события, а переходами – операции алгебры событий между компонентами [7].

1.2.2 Сравнительный анализ модельных представлений составного события

Более подробно достоинства и ограничения каждого из представленных модельных представлений приведены в таблице 4.

Таблица 4 – Характеристика модельных представлений составных событий

Модельное представление	Достоинства	Ограничения
Граф (Дерево)	– удобство интерпретации и визуализации, – гибкость представления.	– низкая эффективность при работе с событиями, имеющими сложный контекст.
Расширенный конечный автомат	– наличие формального языка, – возможность модификации и расширения.	– сложность визуализации при росте масштаба модели, – сложность разработки и поддержки модели.
Сеть Петри	– наличие формального языка, – возможность моделирования расширенных классов составных событий, включая параллельные процессы, циклы или неопределенные условия, – возможность моделирования агрегированных событий.	– необходима доработка модели для представления сложных составных событий.

Исходя из результатов, приведенных в таблице выше, можно сделать вывод, что наиболее подходящей моделью будет сеть Петри для моделирования, поскольку она не имеет наиболее значимых ограничений, в отличие от графа или конечного автомата.

Если рассматривать граф, то он является очень удобным инструментом для моделирования событий, однако граф не обладает формальным языком в связи с чем модели, построенные с помощью графа, будут обладать низкой точностью и не учитывать дополнительные детали и контекст события.

В нашем случае необходимо модельное представление, которое позволит с наибольшей точностью представить составное событие. Такой характеристикой обладают представления, основанные на конечных автоматах или сетях Петри. Данные математические аппараты обладают

формальным языком и позволяют наиболее точно описывать составные события. Однако в отличие от конечных автоматов сети Петри относительно легко визуализировать при росте масштаба модели и легче разрабатывать и поддерживать, чем модель, основанную на конечном автомате. Однако и у сетей Петри есть ограничения.

Классические сети Петри не могут учитывать информацию о контексте события и времени его возникновения. Поэтому сеть Петри необходимо доработать и добавить в нее возможность сохранения и учета информации о времени и контекста каждого компонента составного события. В результате этого получится раскрашенная сеть Петри, в которой информацию о контексте события будут хранить так называемые «цвета» маркеров – определенные числовые значения, позволяющие определить порядок и время возникновения каждого компонента составного события.

Таким образом, для моделирования составных событий в рамках данной работы будет использоваться раскрашенная сеть Петри.

1.2.3 Модельные представления шаблонов составных событий на основе раскрашенных сетей Петри

Сети Петри являются одним из мощных математических аппаратов, позволяющих моделировать работу различного уровня систем и процессов, в том числе и составных событий. Как было выявлено ранее в результате сравнения модельных представлений составного события, что наиболее подходящим является раскрашенная сеть Петри. Рассмотрим данную модель более подробно.

Данную модель можно представить в следующем виде:

$$PN_c = (P, T, C, \hat{R}^-, \hat{R}^+, \hat{\mu}_0),$$

где $C = \{c_1, c_2, \dots, c_d\}$ – цвет сети;

$R^- \in R^{(d*m)*n}$ – матрица инцидентности дуг, входящих в переходы.

$$R^- = \begin{bmatrix} R_1^- \\ R_2^- \\ \vdots \\ R_d^- \end{bmatrix},$$

где R_k^- – матрица инцидентности входящих дуг цвета $C_k, k = 1, \dots, d$;

$R^- \in R^{(d*m)*n}$ – матрица инцидентности дуг, выходящих из переходов.

$$R^+ = \begin{bmatrix} R_1^+ \\ R_2^+ \\ \vdots \\ R_d^+ \end{bmatrix},$$

где R_k^+ – матрица инцидентности входящих дуг цвета C_k ;

$\hat{\mu}_0 \in R^{d*n}$ – матрица начальной маркировки.

Модель составного события, представленная в виде раскрашенной сети Петри, включает в себя следующие основные элементы: место, переход и маркер. Места в сети Петри используются для описания конкретного события в системе, в то время как маркер указывает на действие, произошедшее в контексте заданного события. Переходы необходимы для определения условий возникновения того или иного события, с которым они связаны.

Места, используемые в раскрашенной сети Петри, можно разделить на примитивные ($P_{\text{прим}}$), виртуальные ($P_{\text{вирт}}$), составные ($P_{\text{сост}}$) и позиции копирования ($P_{\text{копир}}$). Для моделирования простых событий можно использовать только примитивные и составные. Если необходимо сформировать составное событие, то для этого необходимо будет использовать виртуальные места. Места копирования используются для моделирования циклического события, которое может происходить более одного раза. На рисунках 3 и 4 представлены разновидности мест, используемых в раскрашенной сети Петри.

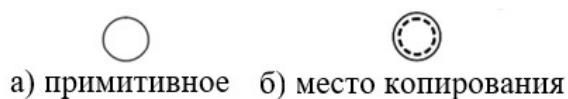


Рисунок 3 – Классификация мест

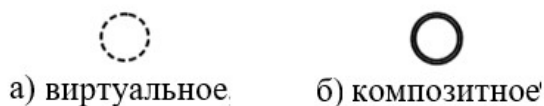


Рисунок 4 – Классификация мест

Переходы в раскрашенной сети Петри можно разделить на составные ($T_{\text{сост}}$), переходы-правила ($T_{\text{прав}}$), переходы-копирования ($T_{\text{копир}}$) и временные переходы ($T_{\text{врем}}$). Составные переходы предназначены для моделирования составных событий из простых, а переходы копирования – для моделирования циклического события, запускающего более одного раза. На рисунках 5 и 6 представлены примеры переходов, используемых в раскрашенной сети Петри для моделирования составного события.

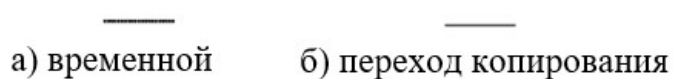


Рисунок 5 – Классификация переходов

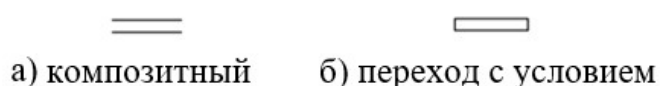


Рисунок 6 – Классификация переходов

Дуги в сети Петри можно разделить на нормальную и ингибиторную (замедляющую). Дуги используются для соединения мест и переходов в раскрашенной сети Петри. Причем, дуги замедления используются для моделирования отрицания составных событий. На рисунке 7 можно увидеть пример дуг, используемых в раскрашенной сети Петри.

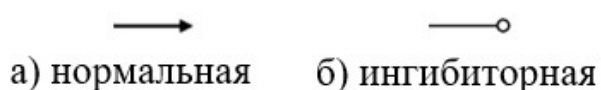


Рисунок 7 – Классификация дуг

Данный набор компонентов обеспечивает возможность описать различные шаблоны составных событий, учитывая особенности различных предметных областей.

1.3 Сравнительный анализ методов и алгоритмов выявления шаблонов составных событий

В данном разделе будет проведен анализ основных методов и алгоритмов, используемых для выявления шаблонов составных событий в различных контекстах. Будут рассмотрены их принципы работы, области применения, преимущества и недостатки, а также проведен сравнительный анализ эффективности их использования в различных ситуациях.

1.3.1 Классификация методов выявления составных событий

Для выявления и анализа составных событий существует достаточно большое количество методов и алгоритмов. Среди них можно отметить ассоциативные правила, алгоритмы секвенциального анализа и алгоритмы извлечения процессов.

Ассоциативные правила являются достаточно мощным инструментом для выявления последовательностей [8].

В общем виде их можно записать так:

$$if < cond > then < result > ,$$

где $<cond>$ – условие, при котором срабатывает правило;

$<result>$ – результат, возвращаемый ассоциативным правилом при наступлении условия $<cond>$.

Однако следует учесть тот факт, что данный подход к выявлению составных событий не совсем подходит, поскольку при анализе событий необходимо учитывать временной контекст, что ассоциативные правила сделать не могут. В таком случае необходимо использовать другие подходы, позволяющие учитывать время возникновения событий. Среди таких подходов наиболее подходящими являются секвенциальный анализ и

извлечение процессов (Process Mining). Но при применении данных подходов стоит обратить внимание на то, какие виды событий каждый из них выявляет лучше всего.

Секвенциальный анализ показывает наибольшую эффективность при выявлении последовательных событий, у которых строго определен порядок их возникновения. Но в реальной ситуации события могут происходить не только последовательно, но и одновременно (параллельно) или циклически. Для выявления таких особенностей можно использовать алгоритмы извлечения процессов. Однако при выявлении последовательных событий они могут показывать меньшую эффективность по сравнению с алгоритмами секвенциального анализа.

В таком случае можно использовать подход, включающий в себя как применение алгоритмов секвенциального анализа, так и алгоритмов извлечения процессов. Алгоритмы секвенциального анализа будут использоваться для выявления последовательных компонентов составного события, а остальную структуру уже будут формировать алгоритмы Process Mining. Подробнее этот подход будет описан в разделе 2.

1.3.2 Обзор алгоритмов выявления шаблонов составных событий

В настоящей работе для выявления шаблонов составных событий будут использоваться две группы алгоритмов: алгоритмы секвенциального анализа и алгоритмы извлечения процессов (Process Mining). Первая группа алгоритмов предназначена для выявления вероятных последовательностей событий, а вторая группа алгоритмов – для моделирования полной структуры составного события с учетом выявленных ранее вероятных последовательностей событий.

Для начала рассмотрим первую группу алгоритмов. Алгоритмы секвенциального анализа включают в себя такие подгруппы алгоритмов, как алгоритмы семейства Apriori, алгоритмы поиска в ширину, алгоритмы поиска в глубину, алгоритмы, предназначенные для поиска замкнутых последовательностей и инкрементные алгоритмы.

Алгоритмы семейства Apriori используют механизм формирования шаблонов кандидатов и последующего их сопоставления с записями в журнале событий. Алгоритмы данной группы относительно просты в реализации, однако за счет постоянного просмотра журнала событий могут оказывать значительную нагрузку на систему, особенно алгоритм AprioriAll, который ищет шаблоны фиксированной длины. Последующие модификации алгоритма, такие как AprioriSome и DynamicSome и GSP позволяют снизить нагрузку на базу данных путем варьирования длины искомого шаблона в зависимости от поддержки последовательности, выявленной на предыдущем шаге [2].

Алгоритмы поиска в глубину представляют исходные данные в виде вертикальной структуры, в которой вершиной является единичная последовательность, и производят поиск всех возможных сочетаний с использованием заданной единичной последовательности. Среди данного семейства алгоритмов можно отметить алгоритмы SPAM, SPADE и PrefixSpan. Алгоритм SPADE для поиска последовательностей использует структуру в виде id-list. Алгоритм SPAM для своей работы использует битовую карту (Bitmap) вертикальное представление растрового изображения. За счет того, что битовая карта обеспечивает значительный процент сжатия, последовательности занимают гораздо меньше места в памяти, и, следовательно, и повышается эффективность подсчета поддержки.

Алгоритмы поиска замкнутых последовательностей выполняют поиска тех последовательностей, которые не могут быть расширены без потери своей замкнутости. Замкнутые последовательности могут иметь вид $\langle D, A, F, C, D \rangle$. Среди данного семейства алгоритмов можно выделить алгоритм BIDE. Данный алгоритм для поиска применяет схему проверки замыкания, называемой двунаправленным расширением (BI-Directional Extension). Данный подход может быть менее эффективным по сравнению с другими, поскольку не все часто встречающиеся последовательности в журнале могут быть замкнутыми, однако, алгоритмы данного семейства позволяют искать

более конкретные и уникальные последовательности, которые не могут быть обнаружены стандартными подходами.

Инкрементные алгоритмы позволяют находить последовательности событий в условиях постоянно расширяемого и изменяемого журнала событий. Они хранят в памяти выявленные ранее последовательности событий и производят поиск новых, по мере добавления данных в журнал. В данной группе алгоритмов можно выделить IncSpan, MILE и ISM [2].

Было проведено сравнение для наиболее известных алгоритмов секвенциального анализа. Сравнение проводилось по двум критериям:

- эффективность,
- ресурсоемкость.

Первый из них показывает, сколько за единицу времени алгоритм может извлечь последовательностей из журнала, а второй – за интенсивность использования ресурсов процессора и оперативной памяти. Сравнение проводилось на системе со следующей конфигурацией: процессор Intel Core i7-11700K, оперативная память 32 ГБ, операционная система Windows 10.

Показатели критериев нормированы от 0 до 1. Результаты сравнения приведены в таблице 5.

Таблица 5 – Сравнение алгоритмов секвенциального анализа

Алгоритм	Эффективность	Ресурсоемкость
AprioriAll	0,8	0,95
GSP	0,85	0,9
SPAM	0,9	0,8
SPADE	0,85	0,85
PrefixSpan	0,9	0,7
BIDE	0,95	0,75

По результатам, представленным в таблице выше, можно сделать вывод, что наименьшее соотношение между эффективностью и ресурсоемкостью обладает алгоритм AprioriAll, так как он постоянно просматривает базу данных при сопоставлении с шаблоном-кандидатом, что значительно сказывается на ресурсоемкости данного алгоритма. Лучшие результаты показывает алгоритм SPAM за счет того, что он использует битовую

вертикальную структуру данных для поиска и развертку растрового изображения, за счет чего наблюдается низкая ресурсоемкость данного алгоритма и значительная эффективность при поиске последовательностей.

Рассмотрим вторую группу алгоритмов. Первоначально данный тип алгоритмов предназначался для построения и анализа моделей бизнес-процессов. Однако, если доработать данные алгоритмы, то с их помощью появляется возможность моделировать составные события. Существует достаточно большое количество алгоритмов извлечения процессов, начиная с самых простых, наподобие альфа-алгоритма (Alpha miner) и заканчивая алгоритмами, использующими кластеризацию. Для рассмотрения было выбрано четыре наиболее известных алгоритма извлечения процессов: альфа-алгоритм, эвристический, генетический и индуктивный. Их характеристика приведена в таблице 6.

Таблица 6 – Характеристика алгоритмов извлечения процессов

Алгоритм	Достоинства	Ограничения
Эвристический алгоритм	<ul style="list-style-type: none"> – учитывает предыдущий опыт, – возможность гибкой настройки, – эффективен при неполных данных в журнале. 	<ul style="list-style-type: none"> – возможны ошибки при неправильных начальных условиях, – низкая точность, – сложно задать начальные условия поиска.
Альфа-алгоритм	<ul style="list-style-type: none"> – прост в реализации, – эффективен при небольших объемах данных. 	<ul style="list-style-type: none"> – неэффективен при анализе больших объемов данных, – низкая точность.
Генетический алгоритм	<ul style="list-style-type: none"> – способен работать с событиями, имеющими сложные взаимосвязи, – имеет возможность работы с большими объемами данных, – возможность гибкой настройки, – параллельная обработка данных в журнале. 	<ul style="list-style-type: none"> – высокая ресурсоемкость, – сложность интерпретации результатов, – заикливание в локальных оптимумах пространства поиска.
Индуктивный алгоритм	<ul style="list-style-type: none"> – работа со сложными взаимосвязями в событиях, – работа с большими объемами данных в журнале. 	<ul style="list-style-type: none"> – эффективен только при больших объемах данных в журнале, – сложность интерпретации результатов.

Альфа-алгоритм прост в реализации, позволяет выявлять составные события при небольших объемах данных в журнале событий. Однако при росте количества записей в журнале его эффективность и точность снижаются. К тому же альфа-алгоритм не способен работать со сложными взаимосвязями в событиях [9].

Эвристический алгоритм, в отличие от альфа-алгоритма, позволяет учитывать предыдущий опыт поиска составных событий. Он позволяет гибко задать начальные условия и работать при неполных данных в журнале. Однако у данного алгоритма все равно низкая точность в сравнении с другими алгоритмами. Кроме того, пользователь может испытывать трудности при определении начальных условий поиска [9].

Генетический и индуктивный алгоритмы позволяют получить наиболее точную модель составного события. Они имеют гибкие настройки начальных условий поиска и позволяют работать со сложными взаимосвязями в событиях. При этом генетический алгоритм обрабатывает данные в журнале параллельно, что сокращает время поиска составных событий. Однако эти алгоритмы требуют высокие вычислительные мощности для работы [10].

Для алгоритмов извлечения процессов проводилось сравнение по эффективности и ресурсоемкости с аналогичной конфигурацией. Показатели критериев нормированы от 0 до 1. Результаты сравнения приведены в таблице 7.

Таблица 7 – Сравнение алгоритмов извлечения процессов

Алгоритм	Эффективность	Ресурсоемкость
Альфа-алгоритм	0,6	0,7
Эвристический	0,75	0,8
Генетический	0,85	0,9
Индуктивный	0,8	0,9

Из результатов выше следует, что по соотношению эффективности и ресурсоемкости хуже себя показал альфа-алгоритм, потому что он не способен моделировать сложные взаимосвязи между компонентами (скрытые переходы, циклы и т.д.), но при этом не требователен к ресурсам по сравнению с другими алгоритмами. Генетический и индуктивный алгоритмы показывают

наибольшую эффективность, поскольку могут работать со сложными взаимосвязями в событиях, но показатели ресурсоемкости высоки. Оптимальные показатели у эвристического алгоритма. Он строит более точные модели, чем альфа-алгоритм, но при этом ему ненужно много ресурсов для работы.

1.3.3 Результаты сравнительного анализа алгоритмов выявления шаблонов составных событий

В результате сравнения алгоритмов можно сделать следующие выводы. Альфа-алгоритм совсем не подходит для моделирования составных событий. У него низкая точность, и он неспособен выявлять сложные взаимосвязи в событиях, а также неэффективен при больших объемах данных в журнале. Лучше всего использовать доработанную версию алгоритма, известную как α^{++} , которая уже может выявлять сложные взаимосвязи в событиях.

Эвристический алгоритм показывает себя лучше за счет использования предыдущего опыта поиска и гибкой настройки начальных параметров поиска.

Генетический и индуктивный алгоритмы очень ресурсоемки, но данное ограничение можно оптимизировать. Например, можно использовать эвристический алгоритм для поиска начальных последовательностей событий, затем отбирать из них лучшие и использовать в качестве начальных последовательностей для генетического алгоритма.

Индуктивный алгоритм теряет свою эффективность при работе с небольшим набором данных в журнале.

Поэтому для моделирования шаблонов составных событий будет использоваться доработанная версия альфа-алгоритма α^{++} , а также совокупность эвристического и генетического алгоритмов.

Из алгоритмов секвенциального анализа для дальнейшей работы AprioriAll не подходит, поскольку он наименее эффективен из всех представленных. В дальнейшей работе будут использованы такие алгоритмы, как GSP, SPADE, BIDE, SPAM и PrefixSpan.

1.4 Сравнительный анализ программных средств выявления составных событий

В данном разделе будут проанализированы основные программные средства, используемые для выявления составных событий в различных предметных областях. Будут рассмотрены их особенности, возможности, преимущества и недостатки, а также проведен сравнительный анализ их эффективности и применимости в различных сценариях использования.

1.4.1 Классификация программных средств для выявления составных событий

Существует большое количество программных средств для выявления составных событий. Они могут быть классифицированы по методу анализа (методы, использующие вероятностное распределение, методы машинного обучения [11] и т.д.), по функциональности или по типу и формату анализируемых данных.

Программные средства, которые используют вероятностное распределение, вычисляют вероятность возникновения того или иного типа события, а затем выполняют предсказание шаблона событий в соответствии с вычисленным распределением. Второй тип программных средств использует обучаемые модели для выявления составного события.

Если рассматривать классификацию программных средств по формату анализируемых данных, то можно выделить средства, работающие:

- с файлами (csv, xml, xes и другие),
- с базами данных.

Программные средства, работающие с файлами, производят построчное считывание и парсинг данных о событиях из файла. В качестве исходных данных могут использоваться файлы с расширением csv, xml, xes и т.д. При этом программные средства работают быстрее с файлами, имеющими более примитивную структуру (в csv-файле достаточно разделить атрибуты события по разделителю, в то время как в xml-файле необходимо производить поиск, парсинг атрибутов каждого события и строить дерево).

Также есть программные средства, работающие с базой данных. Они подключаются к базе данных и выбирают оттуда данные о событиях. Производительность таких программных средств напрямую зависит от типа базы и организации данных в ней.

1.4.2 Обзор программных средств выявления составных событий

На сегодняшний день для решения задач Process Mining широко используются следующие инструменты:

- ARIS Process Performance Manager,
- ProM,
- Disco,
- Celonis.

ARIS Process Performance Manager является модулем системы ARIS, предназначенной для управления бизнес-процессами. ARIS Process Performance Manager обеспечивает контроллинг бизнес-процессов, позволяя компаниям измерять и улучшать производительность своих процессов.

ProM представляет собой мощный инструмент для анализа данных процессов. Он обладает широким функционалом, который позволяет работать с различными типами моделей (например, сети Петри, BPMN, и т. д.) и использовать различные форматы входных данных. ProM часто используется как инструмент в академических и научных исследованиях.

Disco – коммерческий инструмент для анализа процессов. Он отличается простотой и удобством использования, что делает его популярным среди широкого круга пользователей. В основном Disco использует нечеткие модели, что позволяет анализировать данные процессов с высокой степенью гибкости и адаптивности.

Celonis представляет собой одно из первых полнофункциональных приложений Process Mining на корпоративном уровне. Оно было разработано в рамках партнерства с компанией SAP и интегрируется с ее продуктами. Celonis позволяет компаниям проводить анализ своих бизнес-процессов, выявлять узкие места и оптимизировать их для повышения эффективности и

производительности. Приложение доступно как в десктопной, так и в облачной версиях, обеспечивая удобство использования и доступность для пользователей.

1.4.3 Результаты сравнительного анализа программных средств выявления составных событий

После проведения сравнительного анализа стало очевидно, что инструмент ProM представляет собой наиболее функциональное решение, доступное бесплатно. В отличие от него, все платные аналоги оказались ограниченными в своем функционале. Каждый из них предлагал лишь один, но тщательно проработанный алгоритм извлечения модели процессов, в то время как ProM обеспечивает широкий набор функций для работы с различными типами моделей и форматами входных данных.

1.5 Выводы

В данном разделе рассматривается актуальность проблемы выявления шаблонов составных событий, описываются основные области их применения и проводится сравнительный анализ алгоритмов поиска вероятных последовательностей событий и алгоритмов извлечения процессов. Более того, рассматриваются проблемы и ограничения, с которыми сталкиваются существующие инструменты для выявления и моделирования шаблонов составных событий. Рассматривается классификация основных операций для построения шаблонов составных событий. В процессе формирования подхода к моделированию шаблонов составных событий было принято решение использовать алгоритмы секвенциального анализа для поиска начальных последовательностей, а затем строить полную модель составного события с помощью алгоритмов извлечения процессов.

2 Методика и алгоритмы выявления шаблонов составных событий

В данном разделе будут рассматриваться основные методики и алгоритмы для выявления шаблонов составных событий. Особое внимание будет уделено общему процессу выявления шаблонов событий.

2.1 Методика выявления шаблонов составных событий

Для выявления шаблонов составных событий предлагается использовать подход, в котором сначала будут формироваться последовательные шаблоны, а затем, строиться общая структура составного события. Более подробно этот процесс будет рассмотрен далее.

2.1.1 Обобщенное представление структуры журнала событий

Для начала рассмотрим журнал событий. Он представляет собой некоторый набор записей, содержащих информацию о том или ином событии. По своей структуре запись может включать в себя название, идентификатор события, а также временную метку для определения момента происхождения события. Пример журнала событий представлен в таблице 8.

Таблица 8 – Фрагмент журнала событий

Case id	Activity	Timestamp	Resource
1201	Создание заказа	10.12.2022 10:01:02 AM	Измайлов Никифор Игоревич
1202	Завершение заказа	11.12.2022 11:05:01 AM	Измайлов Никифор Игоревич

Колонка `case_id` используется для хранения идентификатора события. Каждому произошедшему событию присваивается свой уникальный идентификатор.

Колонка `activity` представляет собой список действий, которые выполняются при совершении события с идентификатором `case_id`.

Колонка `timestamp` хранит временную метку, когда было зафиксировано событие с идентификатором `case_id`.

Колонка `resource` используется для хранения действующих лиц произошедшего события.

Стоит отметить, что представленная структура является одним из вариантов журнала событий, наиболее распространенных в Process Mining. В

реальности структура журнала может варьироваться в зависимости от типа системы и решаемых ей задач.

2.1.2 Основные параметры процесса выявления шаблонов составных событий

Процесс выявления шаблонов составных событий многоэтапный и включает в себя множество параметров. Подробнее рассмотрим параметры каждого этапа.

На этапе выборки данных из журнала учитываются типы произошедших событий, временной интервал, а также ресурс или инициатор события.

При выполнении секвенциального анализа задается минимальная поддержка искомой последовательности, таксономия, временное окно, а также контекст произошедшего события. Минимальная поддержка определяет пороговое значение, необходимое для того, чтобы последовательность рассматривалась как значимая. Этот порог может быть выражен как абсолютное число, указывающее на минимальное количество встреч данной последовательности в журнале событий. Например, если минимальная поддержка установлена на 5, то последовательность должна встречаться в журнале не менее 5 раз, чтобы считаться значимой. Кроме того, минимальная поддержка может быть выражена как относительная величина, вычисляемая относительно общего числа записей в журнале. Например, если минимальная поддержка установлена на уровне 0,1 (10% от общего числа записей), то последовательность должна встречаться как минимум в 10% записей журнала, чтобы считаться значимой. Таксономия определяет параметр, по которому необходимо проводить группировку последовательностей в журнале событий. В качестве таксономии можно использовать идентификатор инициатора события (ресурса), идентификатор заявки и т.д. Временное окно позволяет задавать интервал, в пределах которого будет определяться последовательность событий. Для определения временного окна используется время начала и окончания транзакции.

На этапе моделирования шаблона составного события используются такие параметры, как событие-инициатор, событие-терминатор и само событие. Событие-инициатор используется для начала процесса моделирования составного события. А событие-терминатор будет сигнализировать о завершении процесса моделирования.

Для каждой рассматриваемой задачи набор параметров может быть разный и определяться конкретным экспертом.

2.1.3 Основные процессы моделирования шаблонов составных событий

Как было описано ранее в разделе 1, алгоритмы секвенциального анализа лучше всего выявляют последовательности событий, в то время как алгоритмы извлечения процессов позволяют выявлять более сложные структуры составного события. Поэтому можно использовать эффективный подход для моделирования шаблонов составных событий, который включает в себя следующие этапы:

- выборка и предварительная обработка данных из журнала событий,
- выявление вероятных последовательностей событий с использованием алгоритмов секвенциального анализа,
- моделирование шаблонов составных событий с применением модифицированных алгоритмов извлечения процессов,
- анализ и модификация полученной модели составного события.

На рисунке 8 приведены этапы моделирования шаблонов составных событий.

За выполнение каждого из представленных этапов будет отвечать отдельная подсистема. Выборку и предобработку данных будет обеспечивать подсистема журнализации, за формирование последовательностей событий будет отвечать подсистема секвенциального анализа, а за моделирование шаблонов составных событий – подсистема моделирования составных событий.

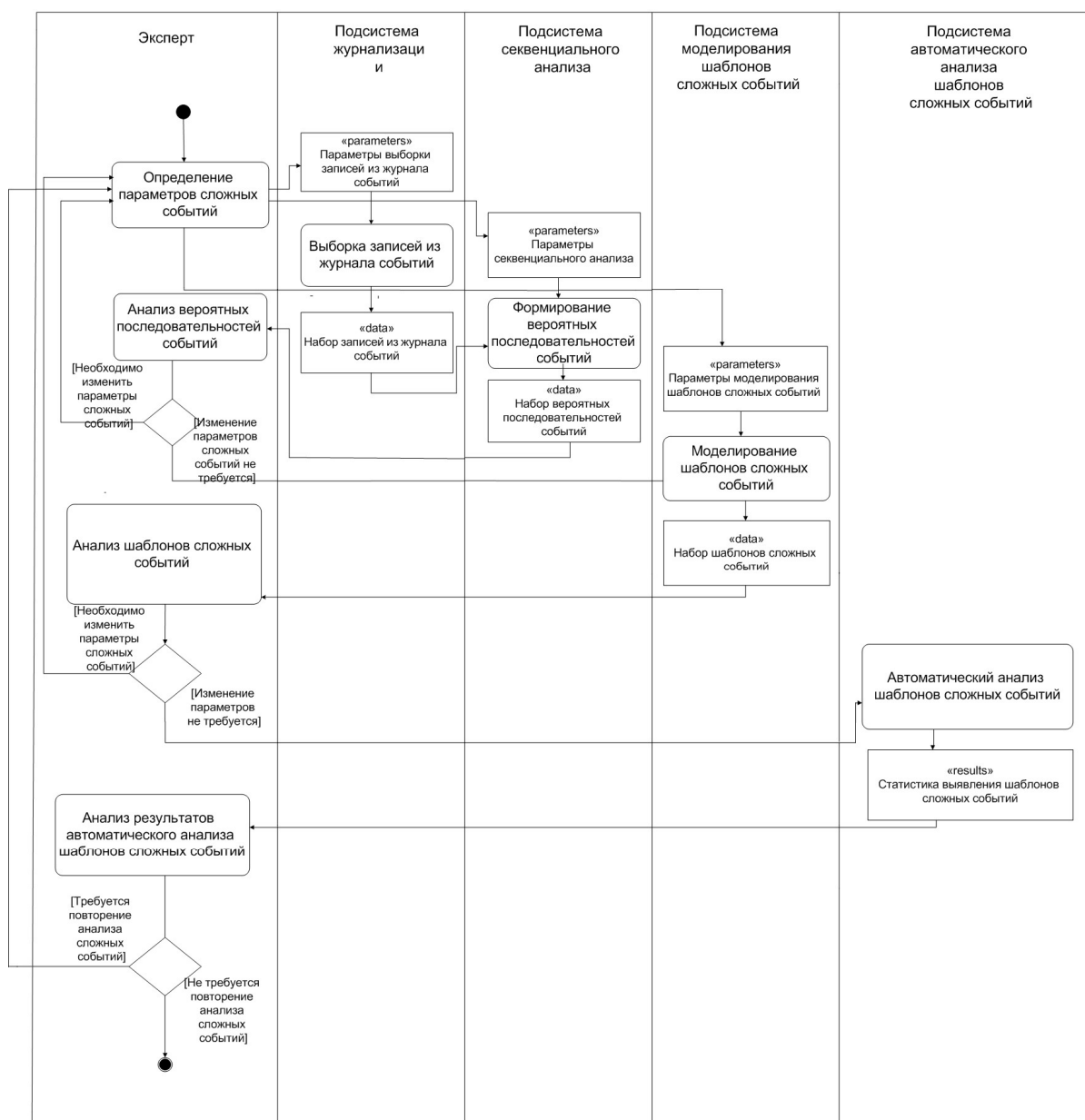


Рисунок 8 – Основные процессы выявления шаблонов составных событий

На каждом этапе возможна корректировка промежуточных результатов. На этапе выборки эксперт сможет просматривать выбранные записи и изменять параметры выборки, группировки и фильтрации. На этапе секвенциального анализа он сможет просматривать предварительно сформированные последовательности событий, изменять алгоритмы секвенциального анализа, задавать параметры поддержки последовательностей. А на этапе моделирования он сможет изменять алгоритм моделирования и просматривать предварительную модель составного события.

Такой подход позволит достичь наибольшей эффективности при выявлении и анализе составных событий.

2.2 Предварительная обработка журнала событий для формирования исходного набора событий

Этап предварительной обработки является основополагающим для анализа. От его результатов будут зависеть результаты последующих этапов, и при наличии шумов или неполных данных точность модели составного события может снижаться [12].

Сценарий предварительной обработки журнала приведен на рисунке 9.

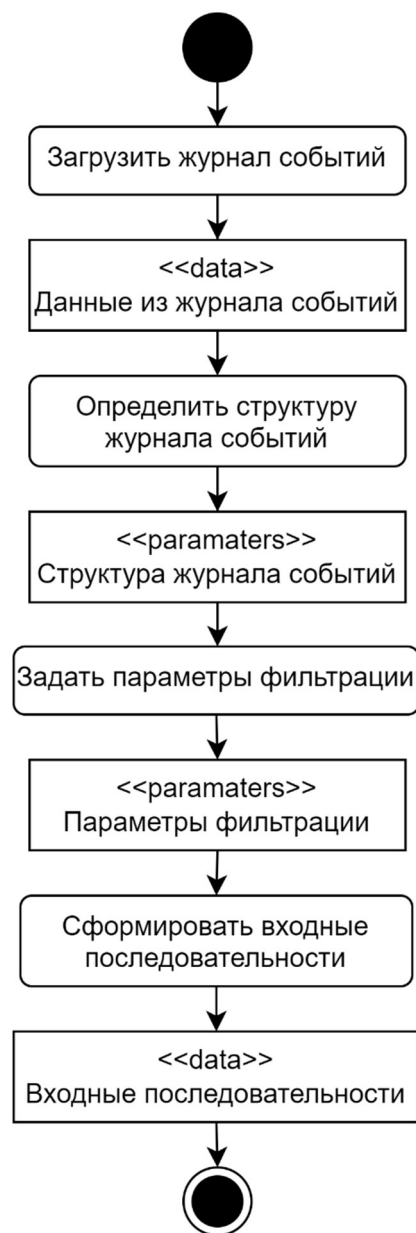


Рисунок 9 – Сценарий предварительной обработки журнала событий

Поскольку разные системы используют журнал событий с различной организацией данных, то сначала необходимо определить его структуру. Пользователь должен задать вручную основные атрибуты: атрибут группировки, атрибут идентификатора события и атрибут времени.

Атрибут группировки представляет собой столбец, по которому будут группироваться события в журнале. Из атрибута идентификатора события будут браться данные для формирования начальных последовательностей, а по атрибуту времени будут сортироваться события в порядке их возникновения от раннего к последнему.

Далее пользователь должен определить параметры фильтрации. Главный из них – временной диапазон. Он позволяет выбрать для анализа события, произошедшие за определенный период времени. Также пользователь может задать максимальное число выбираемых событий, чтобы не загружать весь журнал.

После предобработки формируются начальные последовательности для следующего этапа – секвенциального анализа.

2.3 Алгоритмы выявления шаблонов последовательных событий

Для выявления шаблонов последовательных событий будут использоваться алгоритмы секвенциального анализа. В рамках данной работы будут рассмотрены 3 алгоритма: GSP, PrefixSpan и SPAM.

2.3.1 Алгоритм GSP

Формирование вероятных последовательностей в алгоритме GSP проходит в несколько этапов. Для отображения сценария работы алгоритма разработана диаграмма деятельности, которая представлена на рисунке 10.

Алгоритм работает в режиме многократного прохождения. Сначала считываются выходные данные и для каждого элемента вычисляется его поддержка.

Затем генерируются кандидаты – последовательности, состоящие из элементов, на один больше, чем у предыдущих последовательностей.

В заключительном этапе для всех созданных кандидатов определяется их поддержка, и производится проверка на соответствие установленному минимальному уровню поддержки.

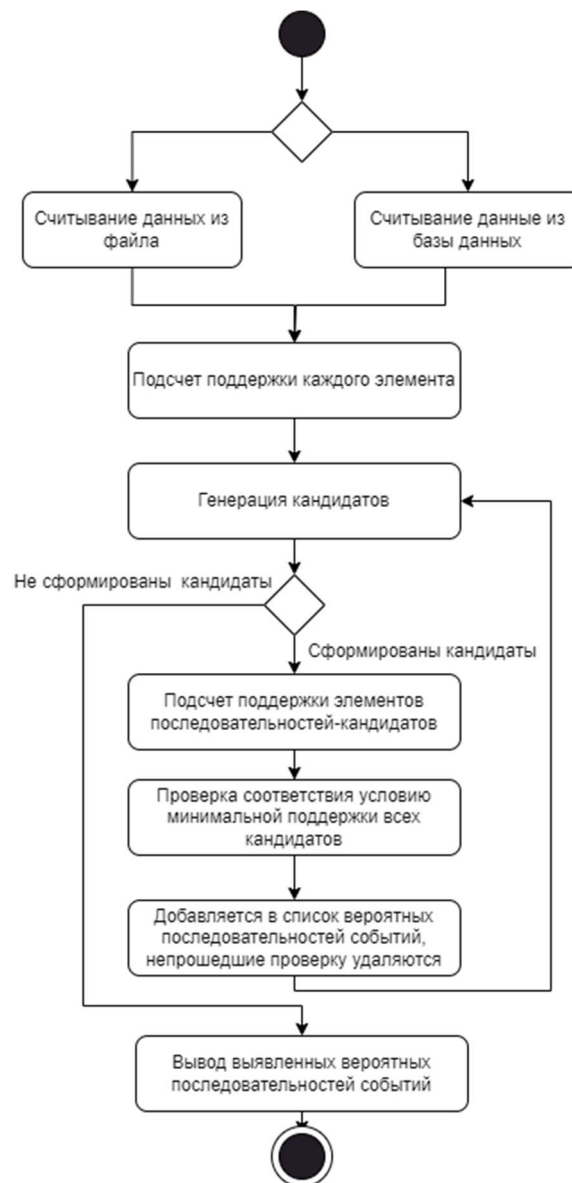


Рисунок 10 – Работа алгоритма GSP

Эти операции продолжаютсся до тех пор, пока не станет возможным сгенерировать дополнительные последовательности-кандидаты.

2.3.2 Алгоритм PrefixSpan

Принцип работы алгоритма PrefixSpan представлен в виде диаграммы деятельности на рисунке 11.

Алгоритм PrefixSpan работает, формируя возможные последовательности, анализируя префиксные цепочки и проектируя соответствующие им постфиксные подпоследовательности в базе данных.

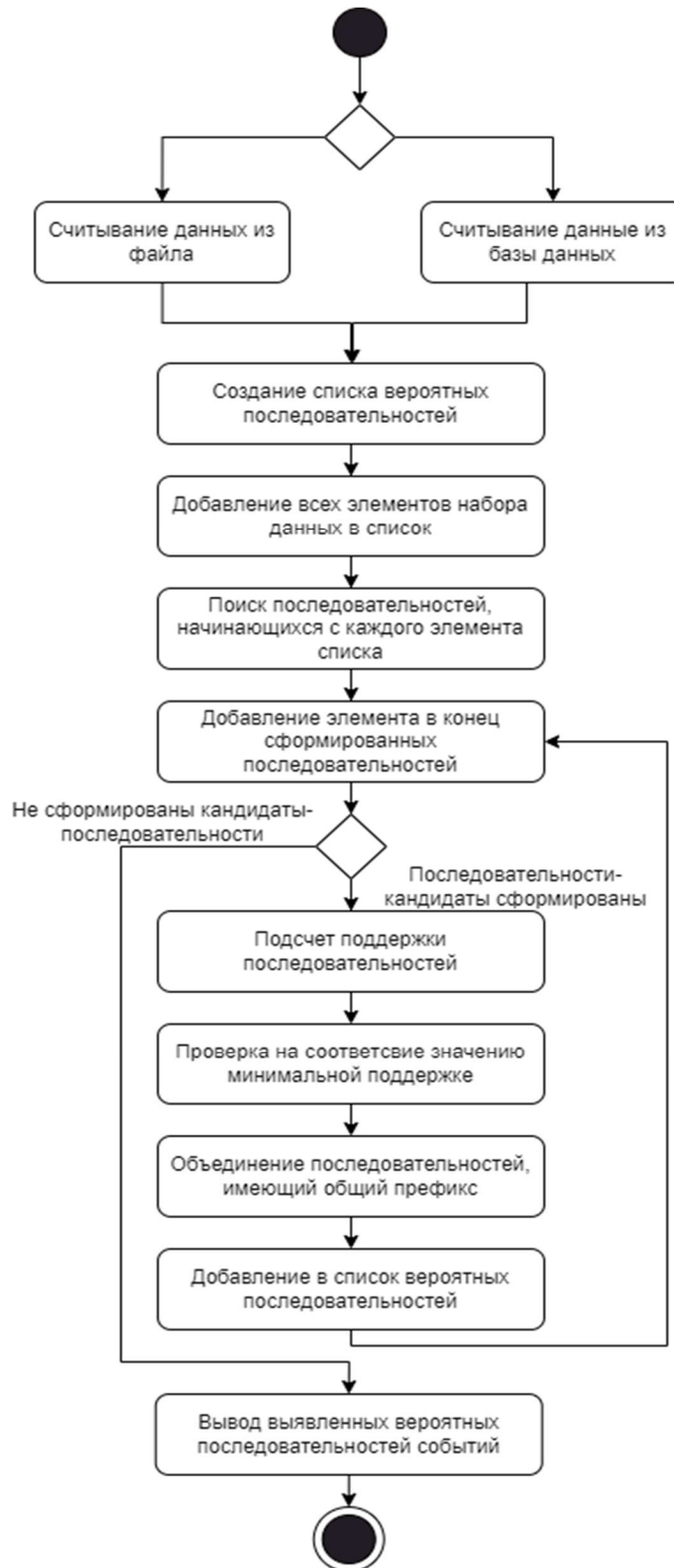


Рисунок 11 – Работа алгоритма PrefixSpan

Процесс начинается с загрузки данных и создания пустого списка последовательностей, в который постепенно включаются элементы из исходного набора. Затем алгоритм ищет все потенциальные последовательности, начинающиеся с каждого элемента списка.

На следующем этапе генерируются различные последовательности: каждая новая цепочка продолжается добавлением элемента к уже существующим последовательностям. Далее анализируется поддержка каждой сформированной последовательности и сравнивается с установленным порогом. Те последовательности, которые соответствуют критериям и имеют общую начальную часть, группируются и добавляются в список возможных решений. Процесс генерации продолжается до тех пор, пока не будут идентифицированы все доступные вероятные последовательности.

2.3.3 Алгоритм SPAM

Алгоритм начинает свою работу с анализа входных данных и далее производит подсчет поддержки для определения часто встречающихся элементов. Затем осуществляется создание последовательностей, которые начинаются с текущего префикса.

После вычисления поддержки для каждой из этих последовательностей производится проверка их соответствия минимально установленному порогу. В последующем этапе генерируются кандидаты, которые завершаются текущим префиксом, и также оцениваются их показатели поддержки в сравнении с минимальным требованием.

Завершающим этапом является объединение различных последовательностей в более длинные цепочки.

Работа алгоритма прекращается, когда становится невозможным выделить новые вероятные последовательности. Работа алгоритма SPAM представлена на рисунке 12.

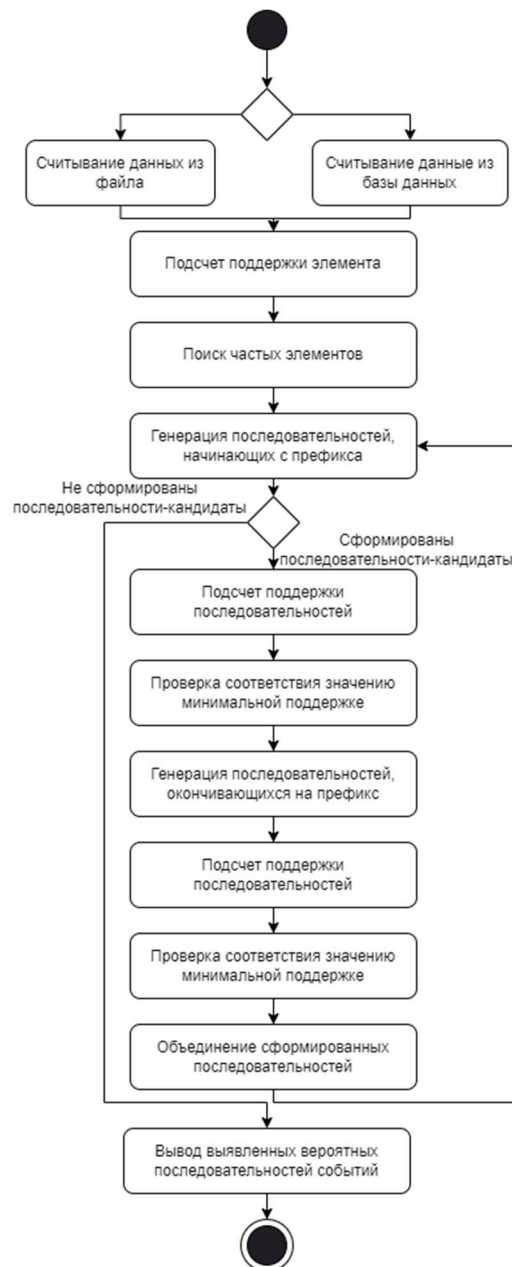


Рисунок 12 – Работа алгоритма SPAM

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

2.4 Алгоритмы выявления шаблонов составных событий

В процессе моделирования полного шаблона составного события будут использоваться алгоритмы извлечения процессов (Process Mining). В рамках данной работы будут рассмотрены 3 алгоритма: альфа-алгоритм, эвристический и индуктивный.

2.4.1 Альфа-алгоритм

Работа альфа-алгоритма заключается в поочередном переборе всех событий одного случая и определения типа связи всех его переходов. Описание работы алгоритма представлено в блок-схеме на рисунке 13.

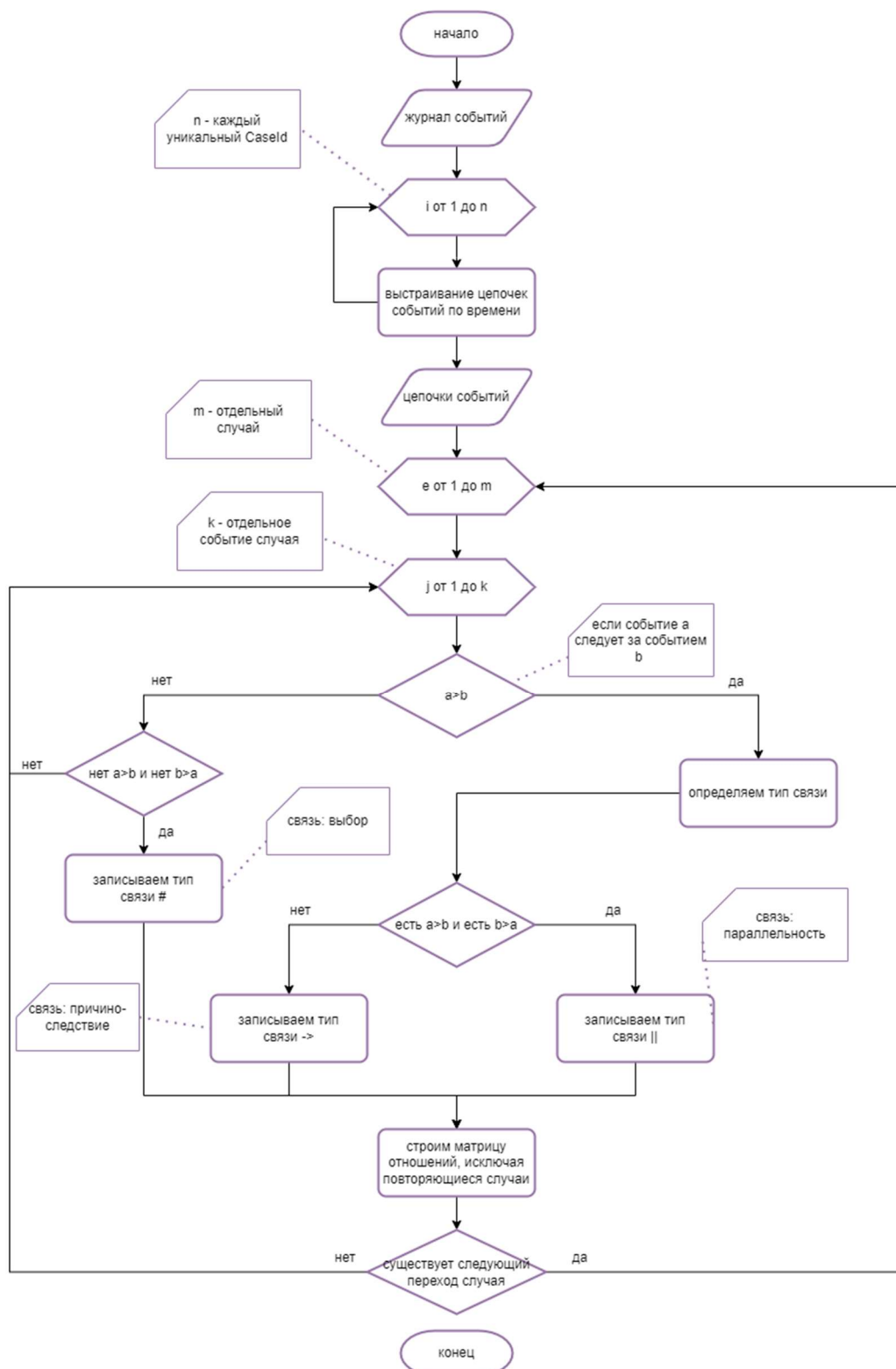


Рисунок 13 – Работа альфа-алгоритма

При получении журнала событий он проходит предварительную обработку для выявления последовательностей событий. На основе этих последовательностей определяются типы связей между различными событиями. Последовательно анализируются пары событий (a и b), где проверяется наличие между ними логической связи. Если таковой не обнаруживается, анализируется обратная связь (b и a), и при её наличии устанавливается тип связи как «выбор». В случае, если между событиями существует однонаправленная связь и отсутствует обратная, определяется тип связи как «причина-следствие».

Если обнаруживается двунаправленная связь между событиями (a и b, а также b и a), это указывает на наличие параллельной связи между ними.

На каждом этапе анализа формируется и корректируется матрица связей, при этом исключаются дублирующие переходы. По завершении обработки всех событий и связей получается окончательная матрица, на основе которой возможно создание сети Петри.

2.4.2 Эвристический алгоритм

Эвристический алгоритм, принимает на вход журнал событий и установленные параметры порога зависимости. Порог зависимости используется для отсеивания шумовых данных и его значение может колебаться в диапазоне от 0 до 1. Если значение порога приближается к 1, то в итоговой матрице будет меньше событий.

Эвристический алгоритм схож с методом альфа-правил для определения типов связей между событиями. В его структуре предусмотрены дополнительные внутренние циклы и проверки, направленные на оценку вероятности наличия связи между различными событиями.

Алгоритм описан в деталях с использованием блок-схемы, изображенной на рисунке 14. Первоочередная задача алгоритма заключается в формировании матрицы связей, одновременно вычисляя для каждого потенциального перехода уровень уверенности в его существовании. Переходы, значение уверенности для которых не превышает установленного

порога зависимости, исключаются из рассмотрения и классифицируются как шумовые.

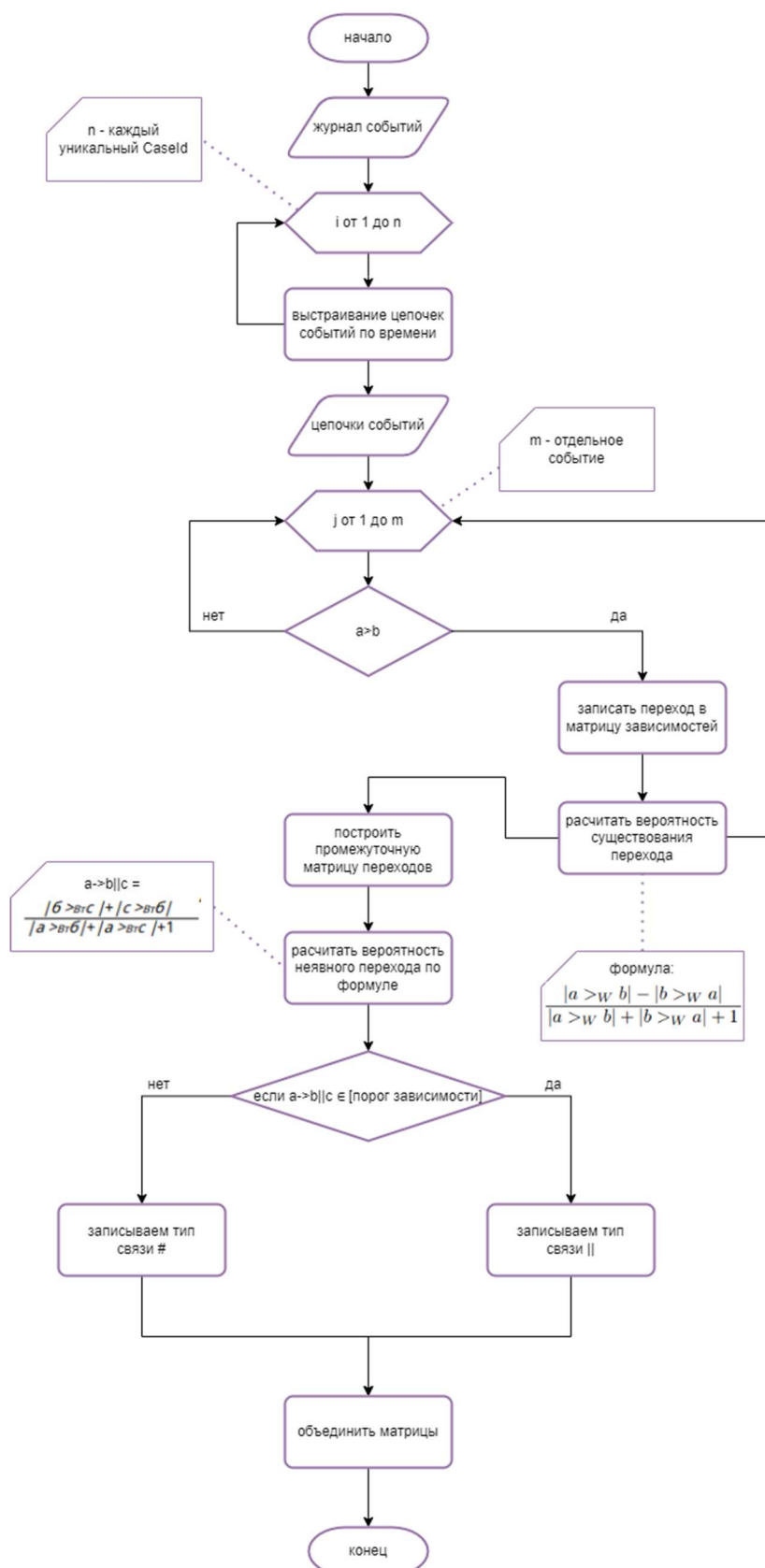


Рисунок 14 – Работа эвристического алгоритма

Отдельное условие появляется для неявных переходов. После построения матрицы, очищенной от шума, проверяются и подчитываются вероятности для переходов с неявным значением. Если значение вероятности больше установленного порога, то переход считается неявным, в противном случае переход считается с типом исключительного выбора, после чего получается результирующая матрица.

2.4.3 Индуктивный алгоритм

Индуктивный алгоритм способен обрабатывать четыре основных типа связей между событиями: «причина-следствие», «параллельность», «исключительный выбор» и «цикл».

В процессе работы алгоритма создается граф прямого следования. Как только граф сформирован, алгоритм проводит разрезы этого графа для выявления различных типов связей. Приоритетным этапом является поиск разрезов с исключительным выбором ($\#$), затем следуют разрезы "причина-следствие" ($\rightarrow$), параллельные разрезы ($\parallel$) и в конце разрезы, связанные с повторяющимися циклами ($\cup$).

Разрез с исключительным выбором выделяется тем, что разделяет граф на подмножества без возможных переходов между ними. Разрез последовательности разделяет граф так, чтобы каждое событие из одного множества предшествовало каждому событию из следующего. Параллельный разрез создает связи между действиями разных множеств. Разрез повторяющегося цикла определяет граф, разделяя его на события, которые являются частью цикла, и те, которые повторяются.

Подробное представление работы алгоритма и его этапов доступно на рисунке 15.

При определении наилучшего момента для проведения разреза, алгоритм в первую очередь ориентируется на поиск нетривиального разреза с исключительным выбором. Термин «нетривиальный» в данном контексте указывает на то, что разрез успешно разделяет граф как минимум на две отдельные группы. Если алгоритм обнаруживает такое нетривиальное

разделение, он применяет максимальное разделение исключительного выбора. Максимальный разрез означает, что не существует других разрезов, которые могут разделить граф на большее количество групп или наборов.

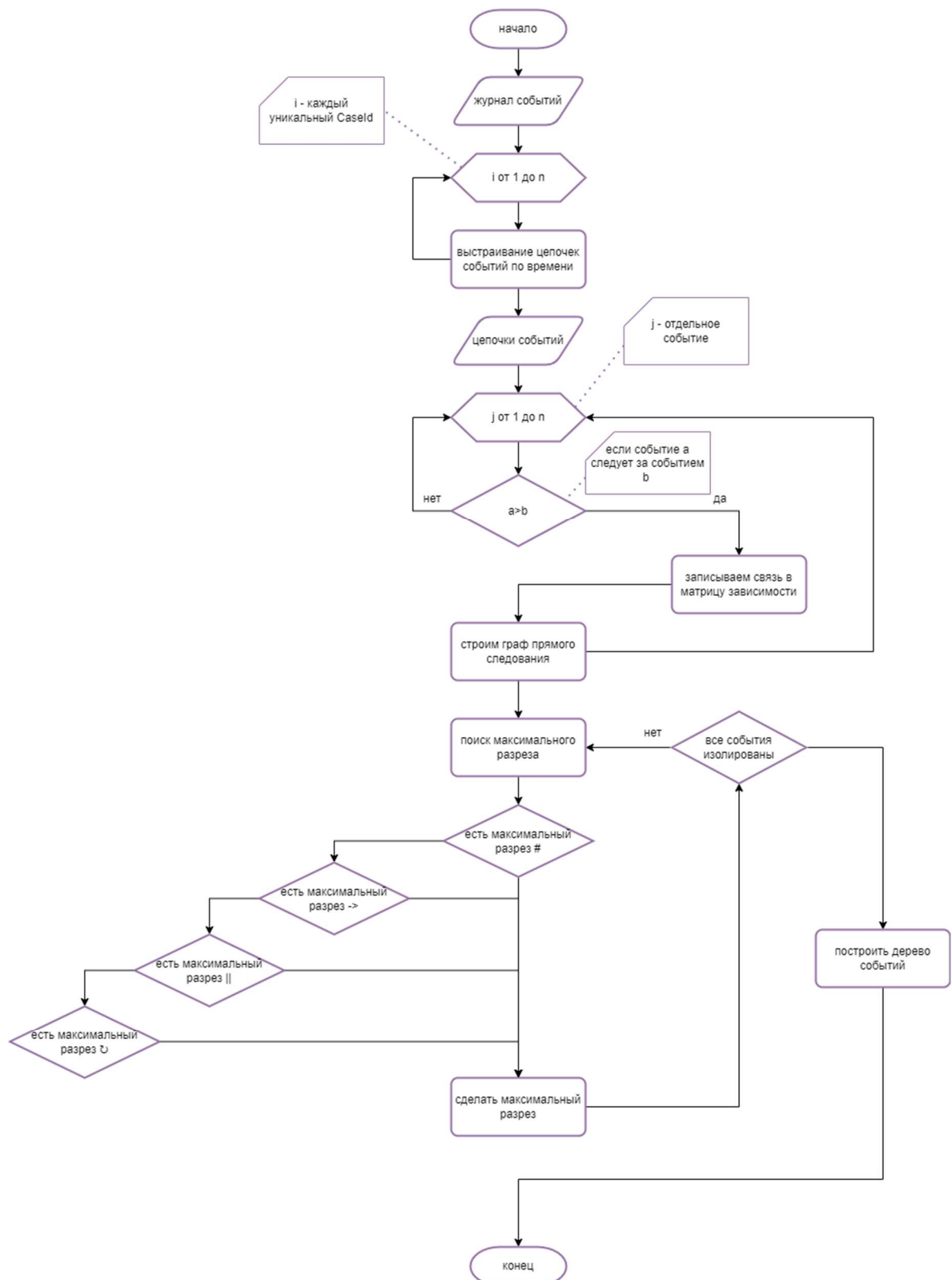


Рисунок 15 – Работа индуктивного алгоритма

Если нет возможности провести нетривиальные разрезы с исключительным выбором, алгоритм переключается на поиск нетривиального разреза типа "причина-следствие", применяя затем максимальное разрезание. На каждом этапе алгоритм продвигается вперед, сокращая связи, пока каждое отдельное событие не становится изолированным. После достижения этого этапа алгоритм формирует дерево событий, используя извлеченную информацию и учитывая типы связей между соответствующими событиями.

2.5 Выводы

В данном разделе был рассмотрен наиболее эффективный подход к выявлению и моделированию шаблона составного события. Была рассмотрена обобщенная структура журнала событий, а также этапы моделирования составного события. Более подробно были рассмотрены алгоритмы секвенциального анализа и моделирования составного события. Для каждого из них была приведена схема работы в виде диаграммы деятельности.

3 Программные средства выявления шаблонов составных событий

В данном разделе будет подробно рассматриваться реализация программных средств для обработки и анализа журнала событий, их функциональные возможности и архитектура. Также будет рассмотрена реализация плагин-библиотек, используемых программными средствами.

3.1 Назначение и функциональные возможности программных средств

Программные средства обработки и анализа журнала событий предназначены для формирования и анализа модели составного события. Для этого используются методы секвенциального анализа и извлечения процессов. Сам процесс анализа осуществляется в виде отдельно создаваемых проектов и решений.

Для проекта задается начальная выборка из журнала событий на этапе предобработки. Анализ и моделирование уже осуществляется в решениях. В них пользователь-аналитик может выбрать необходимый алгоритм и уровень поддержки. Промежуточные результаты сохраняются в решении.

Журнал событий может выгружаться как из файла, так и из базы данных. На рисунке 16 приведены функциональные возможности программных средств [13].

Аналитик в программе совершает шесть бизнес-процессов: сформировывает набор данных, проводит секвенциальный анализ, моделирует шаблоны событий, 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.

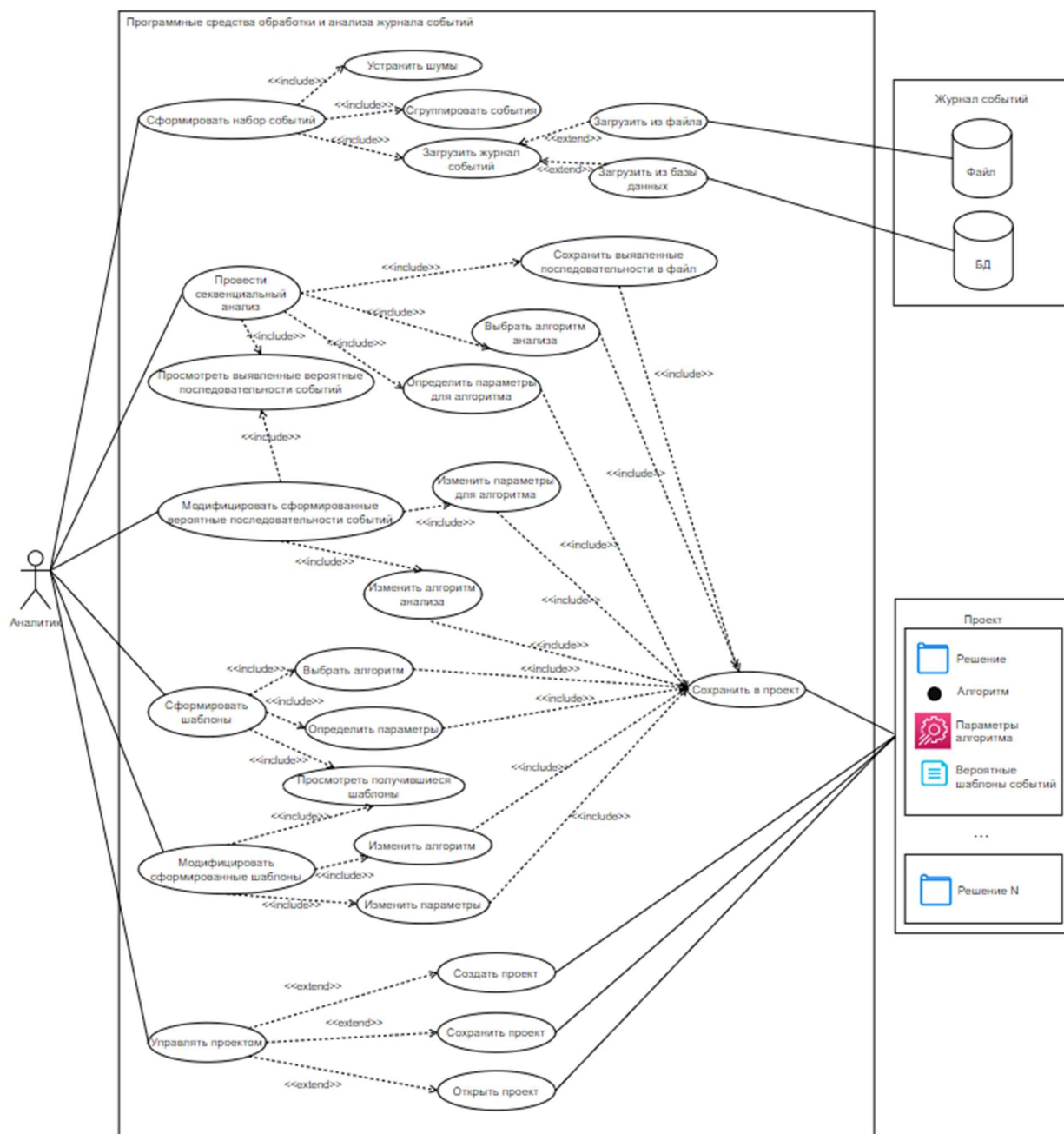


Рисунок 16 – Функциональные возможности программных средств обработки и анализа журнала событий

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 9 описывается сценарий прецедента «Провести секвенциальный анализ».

Таблица 9 – Сценарий прецедента «Провести секвенциальный анализ»

Название 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

В таблице 10 описывается сценарий прецедента «Сформировать шаблоны».

Таблица 10 – Сценарий прецедента «Сформировать шаблоны»

Название использования	Сформировать шаблоны
Актеры	Аналитик
Краткое описание	Моделирование шаблонов событий
Ссылки на другие варианты использования	Выбрать алгоритм анализа, определить параметры для алгоритма, посмотреть выявленные сформированные шаблоны событий, сохранить модельные представления событий в файл
Предусловие	Сформирован набор событий; Определена вероятная последовательность событий
Типичный ход событий	1. Аналитик выбирает алгоритм для проведения анализа. 2. Аналитик определяет параметры. 3. Аналитик получает и просматривает смоделированные шаблоны событий. 4. Аналитик сохраняет модельные представления в файл.
Постусловие	Анализ модельных представлений

Моделирование шаблонов событий включает операции выбора алгоритма, определение параметров, просмотр и сохранение получившихся модельных представлений событий. С помощью бизнес-процесса, связанного

с модификацией, аналитик может выбрать другой алгоритм и изменить его параметры.

3.2 Модульная архитектура и схема функционирования программных средств

Программные средства были реализованы с использованием модульной архитектуры, где каждый компонент отвечает за определенную функциональную возможность. Архитектура приведена на рисунке 17.

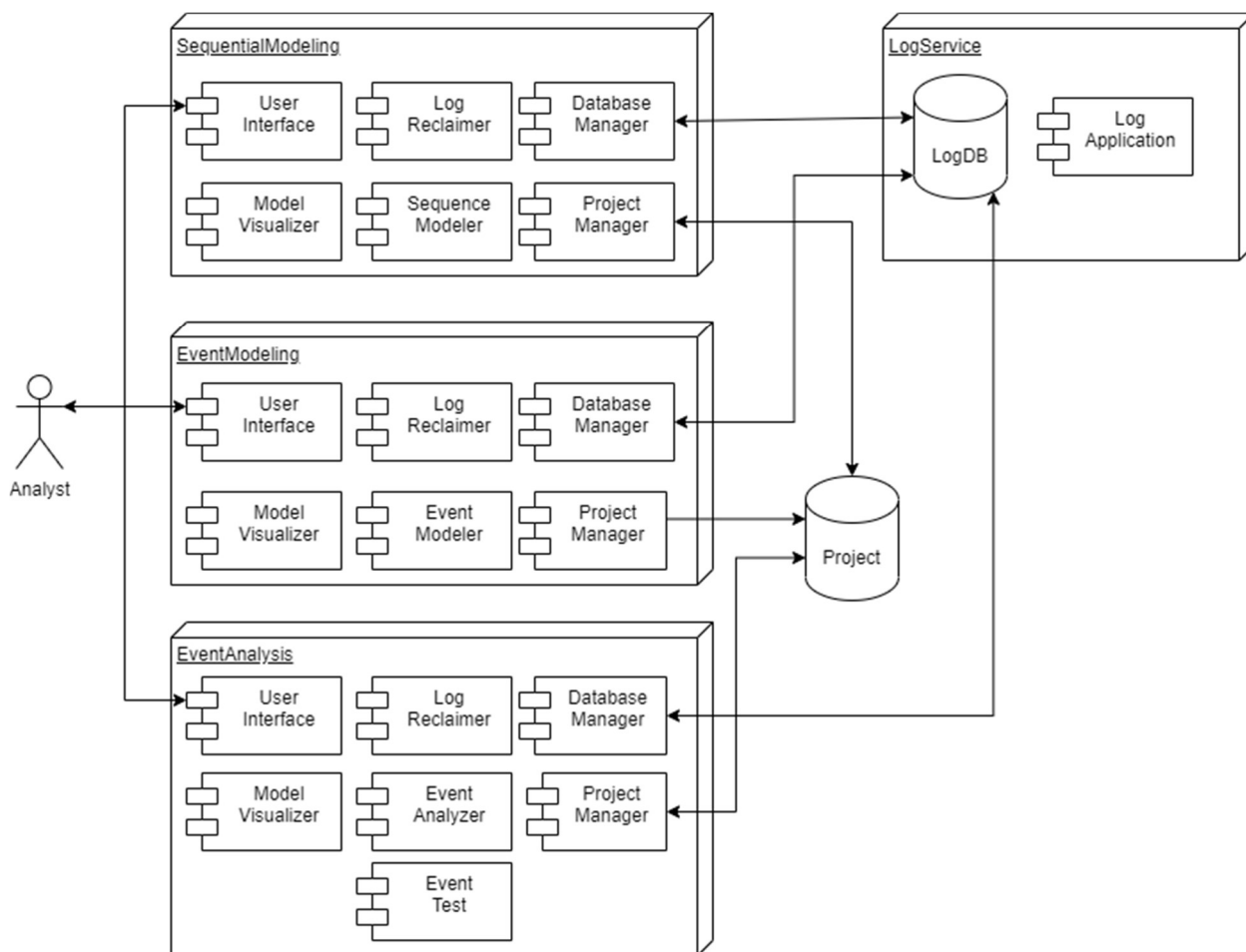


Рисунок 17 – Архитектура программных средств

На диаграмме выше представлена полная система, состоящая из трех подсистем. Подсистема SequentialModeling осуществляет поиск вероятных последовательностей событий, с помощью алгоритмов секвенциального анализа. Подсистема EventModeling предназначена для моделирования шаблонов составных событий, а подсистема EventAnalysis отвечает за интеллектуальный анализ шаблонов событий.

Элементы, представленной диаграммы развертывания рассмотрены в таблице 11.

Таблица 11 – Элементы диаграммы развертывания

Название элемента	Описание
Analyst	Аналитик, взаимодействующий с системой
EventModeling	Подсистема, выполняющая моделирование вероятных шаблонов составных событий
SequentialModeling	Подсистема, выполняющая поиск вероятных последовательностей составных событий
User Interface	Пользовательский интерфейс программы
Log Reclaimer	Компонент обработки и фильтрации журнала событий
Model Visualizer	Компонент, отвечающий за визуализацию результата
Event Modeler	Компонент для выявления вероятных шаблонов составных событий
Database Manager	Компонент работы с базой данных
Project Manager	Компонент работы с проектом
Project	Проект
LogService	Модуль, отвечающий за загрузку журнала событий
LogDB	База данных журнала событий
LogApplication	Компонент, отвечающий за хранение журнала событий в файле

Аналитик взаимодействует с подсистемами SequentialModeling, EventModeling и EventAnalysis через компонент User Interface. В нем он может указывать необходимые параметры для этапов выявления и моделирования шаблонов составных событий и просматривать промежуточные результаты.

Рассмотрим общий сценарий работы программных средств (рисунок 18). Перед началом работы создается проект, в который загружается журнал событий из файла или базы данных.

После того, как был создан проект, создается одно или несколько решений, в каждом из которых определяется алгоритм для анализа и уровень поддержки для поиска последовательностей событий.

Далее, в созданном решении выполняется процесс секвенциального анализа и моделирования вероятных шаблонов событий. Если необходимо изменить параметры анализа или алгоритм, то проводятся соответствующие изменения, которые также сохраняются в проекте.

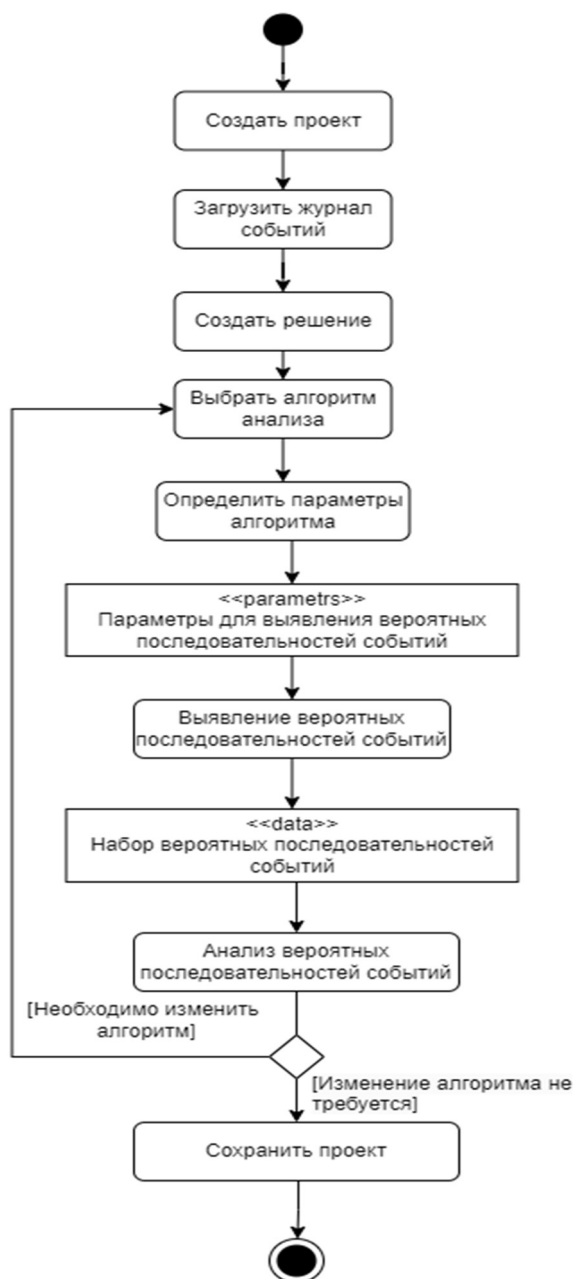


Рисунок 18 – Сценарий работы программных средств

После анализа можно просматривать получившиеся результаты в окне предпросмотра.

3.3 Библиотека классов для построения шаблонов последовательных событий

Для осуществления секвенциального анализа была разработана библиотека SeqMineLib [14], включающая в себя реализацию алгоритмов GSP, PrefixSpan и BIDE. На рисунке 19 приведена структура библиотеки SeqMineLib.

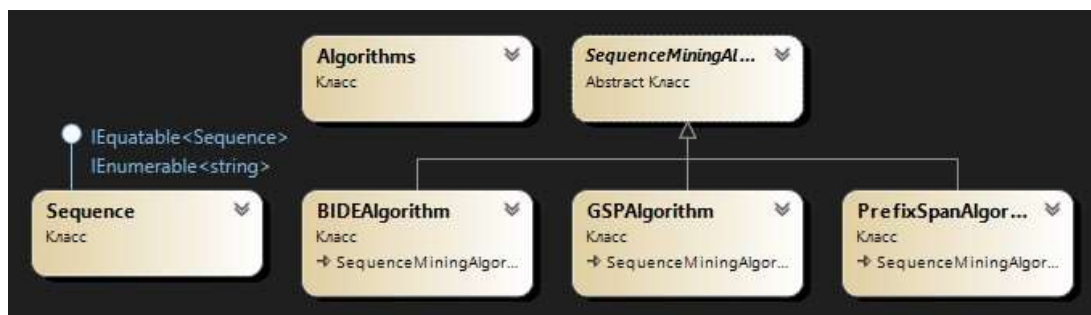


Рисунок 19 – Структура библиотеки SeqMineLib

Спецификация классов данной библиотеки приведена в таблице 12.

Таблица 12 – Описание классов библиотеки SeqMineLib

Класс	Описание
SequenceMiningAlgorithm	Абстрактный класс алгоритма секвенциального анализа. Содержит поля для входных данных и результатов анализа.
GSPAlgorithm	Класс, реализующий алгоритм GSP.
PrefixSpanAlgorithm	Класс, реализующий алгоритм PrefixSpan.
BIDEAlgorithm	Класс, реализующий алгоритм BIDE.
Algorithms	Класс, используемый для управления другими классами алгоритмов. Включает в себя список всех доступных алгоритмов секвенциального анализа и конструкторы для них.
Sequence	Класс, используемый для представления последовательностей событий.

В таблице 13 приведена спецификация полей класса SequenceMiningAlgorithm. Исходный код класса приведен в приложении Б.

Таблица 13 – Описание полей класса SequenceMiningAlgorithm

Название поля	Описание
inputData	Поле, используемое для хранения входных последовательностей событий в виде списка.
freqPatterns	Поле, используемое для хранения результатов секвенциального анализа – выявленных последовательностей событий в виде словаря, ключ в котором – выявленная последовательность, а значение – частота ее появления в журнале.

Методы класса SequenceMiningAlgorithm описаны в таблице 14.

Таблица 14 – Описание методов класса SequenceMiningAlgorithm

Название метода	Описание
Mine	Метод, выполняющий поиск вероятных последовательностей событий.
GetResult	Возвращает результат секвенциального анализа.

Рассмотрим класс Algorithms. Данный класс не имеет полей, а содержит только набор статичных методов, используемых для управления классами библиотеки.

Таблица 15 – Описание методов класса Algorithms

Название метода	Описание
GetAlgorithmNames	Возвращает список названий алгоритмов.
GetAlgorithms	Возвращает все классы (типы данных) алгоритмов.
CreateAlgorithm	Создает экземпляр выбранного типа алгоритма с помощью механизма рефлексии.

Метод CreateAlgorithm принимает на вход выбранный тип из списка алгоритмов и массив параметров – список транзакций (начальных последовательностей событий) и уровень поддержки, для которого необходимо найти последовательности.

3.4 Библиотека классов для построения шаблонов составных событий

Для построения шаблона составного события была разработана библиотека ProcMineLib [14], включающая в себя реализацию альфа-алгоритма, эвристического и индуктивного. На рисунке 20 приведена структура библиотеки ProcMineLib.

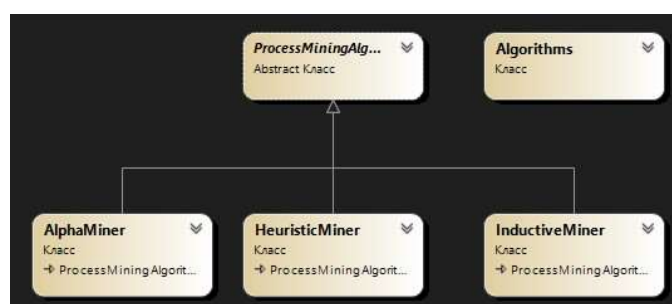


Рисунок 20 – Структура библиотеки ProcMineLib

Спецификация классов данной библиотеки приведена в таблице 16.

Таблица 16 – Описание классов библиотеки ProcMineLib

Класс	Описание
ProcessMiningAlgorithm	Абстрактный класс алгоритма извлечения процесса. Содержит поля для входных данных и результатов моделирования составного события.
AlphaMiner	Класс, реализующий альфа-алгоритм.
HeuristicMiner	Класс, реализующий эвристический алгоритм.

Продолжение таблицы 16

Класс	Описание
InductiveMiner	Класс, реализующий индуктивный алгоритм.
PetriNet	Класс, используемый для представления сети Петри. Содержит список мест, переходов и дуг.
Algorithms	Класс, используемый для управления другими классами алгоритмов. Включает в себя список всех доступных алгоритмов извлечения процессов и конструкторы для них.
Transition	Используется для представления переходов в сети Петри.
Place	Используется для представления мест в сети Петри.
Arc	Используется для представления дуг в сети Петри. Содержит два поля для обозначения начала и конца дуги.

В таблице 17 приведено описание полей класса ProcessMiner. Исходный код класса приведен в приложении Б.

Таблица 17 – Описание полей класса ProcessMiner

Название поля	Описание
Result	Поле, которое используется для хранения результатов моделирования составного события.
Sequences	Поле, в котором хранятся входные последовательности событий.

В таблице 18 представлено описание методов класса ProcessMiner.

Таблица 18 – Описание методов класса ProcessMiner

Название метода	Описание
BuildProcessModel	Метод, осуществляющий построение шаблона составного события.
GetResult	Возвращает результат моделирования составного события.

В таблице 19 представлено описание полей класса PetriNet.

Таблица 19 – Описание полей класса PetriNet

Название поля	Описание
transitions	Список переходов.
places	Список мест.
arcs	Список дуг.

В таблице 20 представлено описание методов класса PetriNet.

Таблица 20 – Описание методов класса PetriNet

Название метода	Описание
AddTransition	Добавляет переход в сеть Петри.
RemoveTransition	Удаляет переход из сети Петри.
AddPlace	Добавляет место в сеть Петри.
RemovePlace	Удаляет место из сети Петри.
AddArc	Добавляет дугу в сеть Петри.
RemoveArc	Удаляет дугу из сети Петри.

Класс Algorithms, аналогично библиотеке SeqMineLib, содержит статические методы для управления алгоритмами. Он также позволяет получить список всех алгоритмов и создать экземпляр класса конкретного алгоритма извлечения процессов.

3.5 Библиотека классов для предварительной обработки журнала событий

Для предварительной обработки журнала событий и формирования входных последовательностей была разработана библиотека EventLogLib [14]. На рисунке 21 приведена структура библиотеки EventLogLib.

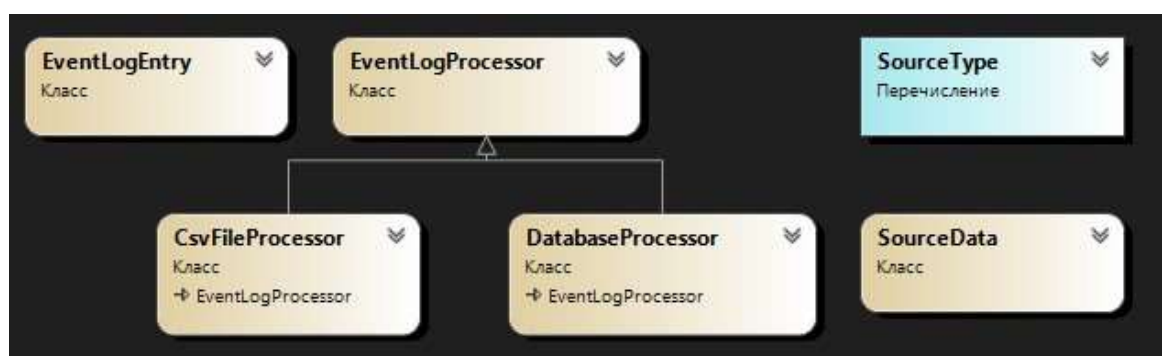


Рисунок 21 – Структура библиотеки EventLogLib

Описание классов приведено в таблице 21.

Таблица 21 – Описание классов библиотеки EventLogLib

Название класса	Описание
EventLogProcessor	Абстрактный класс обработки журнала событий. Содержит поле для временного хранения выбранных данных из источника, поле SourceData для хранения данных о самом источнике, а также метод для формирования входных последовательностей.
CsvFileProcessor	Класс, используемый для работы с журналами событий в виде csv-файлов.
DatabaseProcessor	Класс, используемый для доступа к журналу событий, который хранится в базе данных.
SourceData	Класс, предназначенный для хранения данных об источнике журнала событий (тип журнала, параметры выборки и т.д.).
EventLogEntry	Класс, используемый для обобщенного представления строки журнала событий.

Описание полей класса EventLogProcessor приведено в таблице 22. Исходный код класса приведен в приложении Б.

Таблица 22 – Описание полей класса EventLogProcessor

Название поля	Описание
srcData	Хранит информацию об источнике журнала событий (тип, параметры подключения к журналу, параметры выборки и т.д.).
limit	Максимальное количество выбираемых строк из журнала событий.

Описание методов представлено в таблице 23.

Таблица 23 – Описание методов класса EventLogProcessor

Название метода	Описание
SelectEventLog	Производит выборку данных из журнала событий.
GetActivitySequences	Формирует входные последовательности для анализа.

Для работы с журналом событий на данный момент реализовано два класса, – CsvFileProcessor и DatabaseProcessor, которые позволяют работать с журналом в виде csv-файла и базы данных, соответственно. В будущем, возможно расширять типы поддерживаемых журналов событий и добавлять новые классы, которые могут работать с другими типами файлов.

3.6 Визуализация модельного представления составного события

После построения модели составного события необходимо решить задачу о ее визуализации. В качестве компонента отображения можно использовать холст (Canvas), который будет содержать элементы модели в виде фигур. Так как для визуализации модели используется сеть Петри, то необходимо отобразить ее составные компоненты, а именно, места, переходы и дуги. Для отображения мест можно эллипс, а для переходов – прямоугольник. Дуги отображать можно в виде линий.

Если с типами используемых компонентов для визуализации особых проблем нет, то с их позиционированием могут возникнуть некоторые трудности. Необходимо наиболее равномерно расположить эти элементы в окне предпросмотра, чтобы ее удобно было просматривать. Для этого существует множество подходов и алгоритмов. Наиболее простой из них – использование сетки. Для этого окно предпросмотра делится на области

равной ширины и высоты, в каждой из которых будут располагаться элементы сети Петри. Однако, в данном случае некоторые элементы все равно могут пересекать друг друга, если, например, название у элемента очень длинное, или дуги могут проходить поверх других расположенных элементов. В таком случае можно использовать другие методы, наиболее подходящие для визуализации сетей. Один из них – метод силового поля. Данный метод основан на представлении элементов сети в виде физической системы, в которой элементы – частицы, имеющие определенный заряд. Для вершин (мест и переходов) рассчитывается сила отталкивания по следующей формуле:

$$F_{\text{отталк}}(u, v) = k * \frac{p(u)-p(v)}{|p(u)-p(v)|^2},$$

где k – коэффициент силы отталкивания;

$p(u)$ и $p(v)$ – вершины сети (места и переходы).

Для ребер (дуг) вычисляется сила притяжения по формуле:

$$F_{\text{притяж}}(u, v) = d * (|p(u) - p(v)| - L) * \frac{p(v) - p(u)}{|p(v) - p(u)|},$$

где d – коэффициент упругости;

L – оптимальная длина ребра (дуги).

После определения сил отталкивания и притяжения можно определить расстояние, на котором будут располагаться элементы сети Петри друг от друга. В случае пересечения элементов сети можно менять коэффициенты силы отталкивания, упругости, а также заряд у каждого элемента сети.

Наиболее наглядно, как метод силового поля позволяет распределить элементы сети Петри на экране, можно увидеть на рисунке 22. Данный

результат был получен при оптимальной длине дуги, равной 13 мм и значении коэффициента упругости, равном 3.

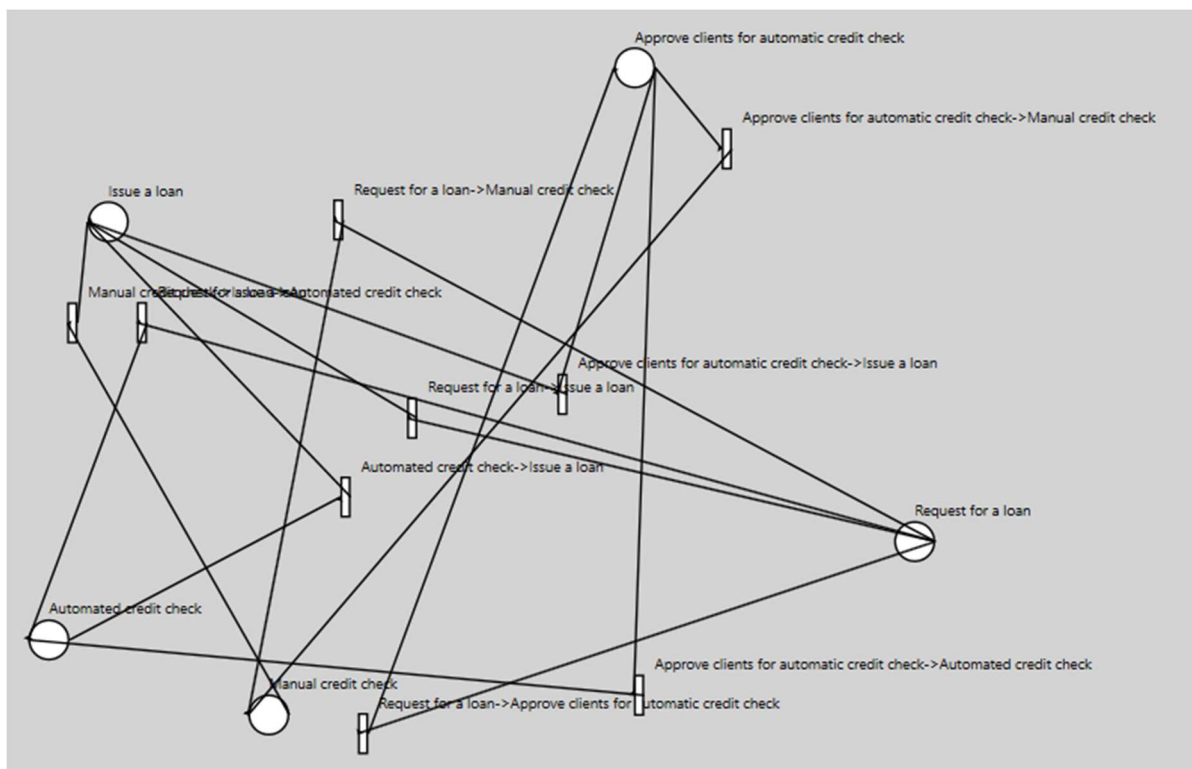


Рисунок 22 – Визуализация сети Петри методом силового поля

При визуализации составного события с небольшим количеством компонентов данные значения будут оптимальны, однако при росте числа компонентов в модели составного события потребуется изменить данные значения в большую сторону, чтобы добиться максимального разброса элементов друг от друга.

3.7 Выводы

В данной главе были рассмотрены основные аспекты, касающиеся программной реализации процесса выявления и моделирования составных событий. Были рассмотрены функциональные возможности программных средств и их архитектура. Было уделено внимание спецификации плагинов-библиотек, используемых данными программными средствами. Были рассмотрены классы библиотек секвенциального анализа (SeqMineLib), моделирования шаблона составного события (ProcMineLib) и предобработки

журнала событий (EventLogLib). Были рассмотрены принципы визуализации полученной модели составного события в виде сети Петри.

4 Контроль качества и экспериментальная оценка программных средств выявления составных событий

В данном разделе будет подробно рассмотрен процесс контроля качества разработанных программных средств. Будет рассмотрен процесс тестирования их функциональных возможностей. Для исходного кода программных средств будет проведен расчет и анализ метрик.

4.1 Процесс контроля качества и экспериментальная оценка программных средств

Контроль качества программных средств является неотъемлемым этапом в процессе разработки, направленным на выявление и устранение ошибок. Его главная цель заключается в тестировании готового продукта и сопоставлении его с функциональными требованиями. Если продукт не соответствует этим требованиям, он считается некачественным. Процесс контроля качества может включать в себя следующие этапы: тестирование работы программного обеспечения, проверка соответствия стандартам разработки кода, моделирование работы программного обеспечения и анализ его архитектуры [15].

Контроль качества предоставляет актуальную информацию о текущем состоянии системы. В рамках данной работы проведено функциональное тестирование и анализ метрик программного кода в качестве основных этапов контроля качества.

4.1.1 Выбор модели тестирования

Тестирование играет важную роль в проверке соответствия ожидаемых и реальных результатов работы программы. Его цели включают выявление ошибок в работе программы, проверку правильности функционирования системы при различных условиях и сопоставление системы с установленными требованиями. Виды тестирования классифицируются по различным критериям, включая цели, выполнение кода и разработку тестовых сценариев.

Основные группы включают функциональное и нефункциональное тестирование. Функциональное тестирование направлено на проверку соответствия функциональным требованиям приложения, в то время как нефункциональное тестирование оценивает такие аспекты, как удобство использования, масштабируемость, производительность и безопасность системы.

В рамках данной исследовательской работы было проведено функциональное тестирование. Все группы тестирования представлены на рисунке 23 [16].

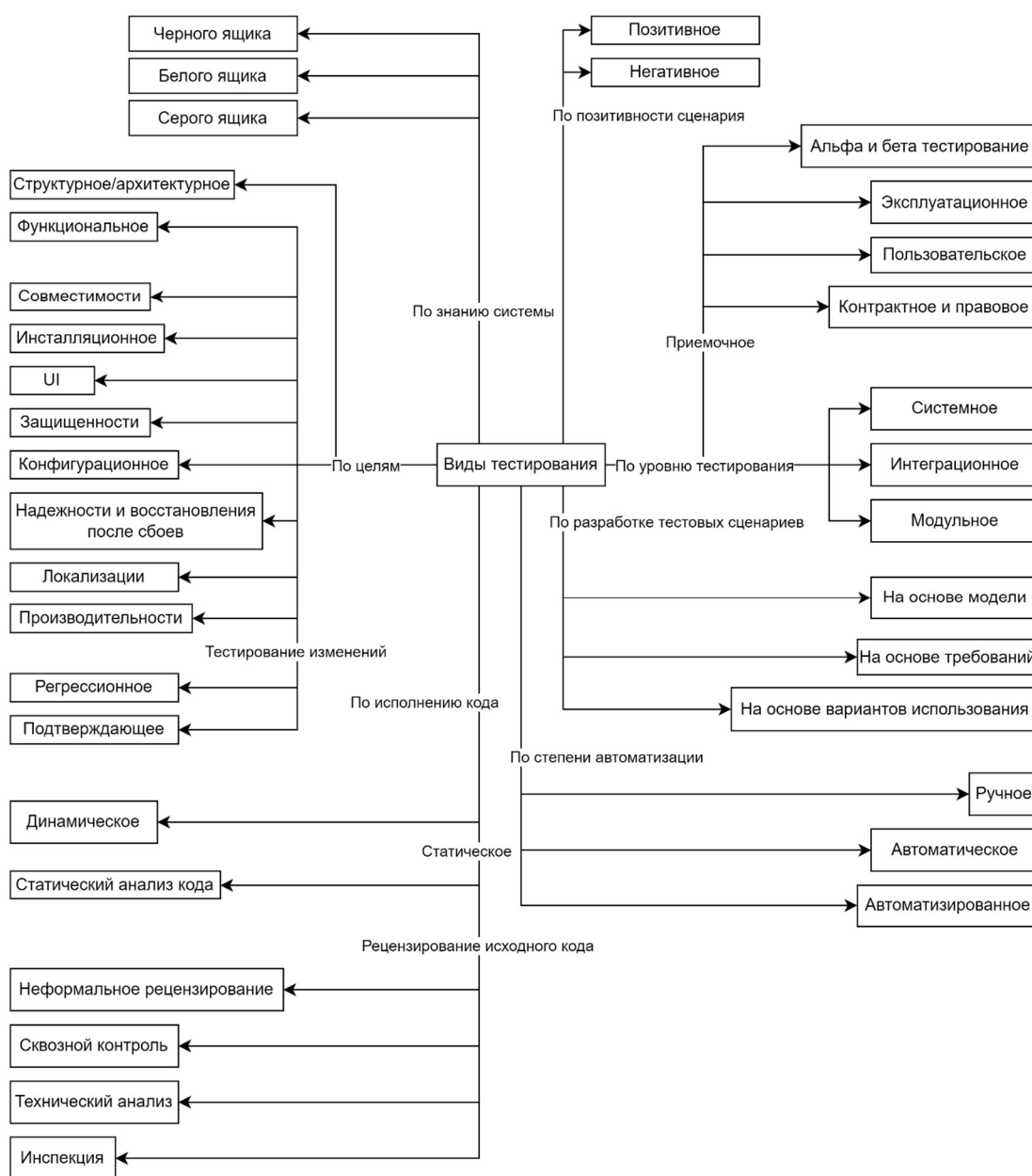


Рисунок 23 – Виды тестирования

Контроль качества программного обеспечения представляет собой многоэтапный процесс, включающий в себя не только тестирование, но и анализ архитектуры, метрик кода и других аспектов разработки. Важно учитывать все эти факторы для обеспечения высокого уровня качества конечного продукта.

4.1.2 Выбор метрик кода

Для оценки качества исходного кода программных средств используются метрики. Метрика программного обеспечения представляет собой численное значение, отражающее определенное свойство программы или её характеристики. Существует множество различных видов метрик, которые могут быть разделены на несколько категорий, включая количественные, надежности, сложности потока управления, сложности потока данных, объектно-ориентированные и гибридные метрики. Описание некоторых из них представлено в таблице 24 [17].

Таблица 24 – Метрики кода

Название	Описание
Метрика Холстеда	Позволяет получить оценки словаря, длины и объема программы, а также теоретического словаря.
Метрика МакКейба	Позволяет определить количество базисных путей в управляющем графе программы, а базисные пути – это пути, которые позволяют получить все возможные пути из входа в выходы графа.
Метрика Майерса	Предоставляет интервальную меру сложности, которая включает в себя цикломатическую меру и количество отдельных условий.
Метрика Харрисона	Учитывает размер программы и уровень вложенности.
Метрика Чепина	Делит все множество переменных на группы в зависимости от их применения в программе.

В рамках данной исследовательской работы для оценки качества кода была выбрана метрика Холстеда.

4.2 Тестирование программных средств

Чтобы провести функциональное тестирование, необходимо составить тест-кейсы, которые учитывают все возможные условия использования программных средств. Условия могут включать в себя тип и диапазон входных данных, а также корректность поведения программы при выполнении определенной функции. Для учета первого можно использовать классы

эквивалентности (КЭ), а для оценки корректности выполнения функции можно использовать таблицу решений. Далее будет рассмотрен процесс разработки и выполнения тест-кейсов с использованием данных средств.

4.2.1 Классы эквивалентности

Классы эквивалентности позволяют разработать тест-кейсы для проверки реакции программы на входные данные, их тип и диапазон. Рассмотрим принцип разработки тест-кейсов на примере процесса создания нового проекта. Чтобы создать новый проект, требуется указать его название. Поле для ввода названия обязательно для заполнения. С учетом требований корректного формата ввода названия проекта выделим соответствующие классы эквивалентности (таблица 25).

Таблица 25 – 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)	Остальные символы (9)
	Цифры (7)	
	Символы (-) (8)	

На рисунке 24 можно увидеть допустимое количество символов, которое можно указать в имени проекта.



Рисунок 24 – Допустимая длина имени проекта

Имя проекта можно включать в себя такой тип входных данных, как буквы и цифры. Символы в имени проекта использоваться не могут. На рисунке 25 представлено множество допустимых знаков в имени проекта.



Не буквы, не цифры, не символы

Рисунок 25 – Допустимое множество знаков в имени проекта

Таблица 27 – Таблица решений для проверки доступа к журналу событий

	1	2	3	4
Причины				
Чтение журнала из файла	1	1	0	0
Чтение журнала из БД	0	0	1	1
Успешное подключение к БД	0	0	1	0
Следствия				
Сообщение «Подключитесь к БД»	0	0	0	1
Сообщение «Выберете файл»	0	1	0	0
Подключение создано	1	0	1	0

Теперь, используя данную таблицу, возможно составить тест-кейсы для проверки доступа к журналу событий. Данные тест-кейсы представлены в таблице 28.

Таблица 28 – Тест-кейсы для проверки доступа к журналу событий

№	Входные данные	Результат теста
1	Подключение журнала из файла Файл выбран	Подключение создано
2	Подключение журнала из файла Файл не выбран	Сообщение «Выберете файл»
3	Подключение журнала из БД Подключение к базе данных совершено	Подключение создано
4	Подключение журнала из БД Нет подключения к базе данных	Сообщение «Подключитесь к БД»

В ходе выполнения тест-кейсов ошибок не обнаружено, следовательно, при выполнении функции доступа к журналу программные средства работают корректно. Результаты тестов представлены на рисунках А.6–А.9 приложения А.

Аналогичным образом можно составить тест-кейсы для проверки функции выявления шаблонов событий. Выделим все действия пользователя, инициирующие выполнение данной функции и ответную реакцию программных средств. Составим таблицу решений (таблица 29).

Таблица 29 – Таблица решений для проверки выявления шаблонов событий

	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

Продолжение таблицы 29

	1	2	3	4
Причины				
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

Также составим тест-кейсы (таблица 30) на основе таблицы решений.

Таблица 30 – Тест-кейсы для проверки выявления шаблонов событий

Входные данные	Результат
Алгоритм не выбран	Сообщение «Выберите алгоритм»
Алгоритм выбран Определены корректные параметры анализа Выполнение анализа	Сохранение параметров алгоритма в решении и отображение сведений о проведенном анализе
Алгоритм выбран Определены некорректные параметры анализа	Сообщение «Параметры алгоритма некорректные»
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 данном случае программа также работает корректно, и функция выполняется в соответствии с ранее определенным сценарием. Результаты тестов представлены на рисунках А.10–А.13 приложения А.

4.3 Оценка качества программного кода

Для оценки качества и сложного исходного кода программных средств была выбрана метрика Холстеда. Рассмотрим расчет метрик на примере функции получения поддержки (листинг 1) в алгоритме секвенциального анализа GSP.

Листинг 1 – Код метода получения поддержки последовательности событий

```
public static void getSupport(List<Sequance> seq)
{
    foreach (Sequance sq in seq)
    {
```

```

foreach (Sequance 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 31.

Таблица 31 – Количество операторов и операндов в функции получения поддержки последовательности

Операторы	Число вхождений	Операнды	Число вхождений
if	2	seq	2
=	3	sq	1
++	1	start	2
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 метода по следующей формуле:

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

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

Таким же образом были подсчитаны метрики кода для метода поиска отношений между событиями алгоритма моделирования составного события.

Код представлен в листинге 2.

Листинг 2 – Код метода поиска отношений между событиями

```
public RelationshipMatrix SearchRelationships()  
{  
    var result = new RelationshipMatrix();  
    var actRelationships =  
this.ActivitiesAndRelationships();  
    var actList = actRelationships.Keys.ToList();  
    foreach (var actX in actList)  
    {  
        foreach (var actY in actList)  
        {  
            var relType = RelationshipType.None;
```

```

        if
        (actRelationships[actX].Next.ContainsKey(actY))
        {
            relType = RelationshipType.Follow;
            if
            (actRelationships[actY].Next.ContainsKey(actX))
                relType =
RelationshipType.ReciprocalFollow;
            else
                relType =
RelationshipType.Cause;
        }
        else if
        (!actRelationships[actY].Next.ContainsKey(actX))
            relType =
RelationshipType.NotFollow;
        result.Add(new Tuple<string,
string>(actX, actY), relType);
    }
    }
    return result;
}

```

Определим количество операторов и операндов в данном методе (таблица 32) и рассчитаем метрики.

Таблица 32 – Количество операторов и операндов в методе поиска отношений между событиями

Операторы	Число вхождений	Операнды	Число вхождений
=	8	result	3
Return	1	actRelationships	4
ToList	1	actList	3
Foreach	2	actX	5
If	2	actY	5
Else	1	relType	6
Else if	1		
new	1		
.	10		
{ }	3		
!	1		
in	2		
;	9		

Длина программы:

$$N = N1 + N2 = 42 + 26 = 68.$$

Словарь программы:

$$n = n1 + n2 = 13 + 6 = 19.$$

Объем программы:

$$HV = N \cdot \log_2(n) = 59 \cdot 4,17 = 293.112.$$

Сложность программы:

$$HD = \left(\frac{n1}{2}\right) \cdot \left(\frac{N2}{n2}\right) = \left(\frac{12}{2}\right) \cdot \left(\frac{26}{6}\right) = 26.$$

Сложность метода приблизительно равна 26, поэтому можно считать её средней сложности.

4.4 Экспериментальная оценка программных средств

Для того, чтобы оценить конкретные результаты работы программных средств в зависимости от количества и типа входных данных, необходимо провести экспериментальную оценку. Для ее организации необходимо определить тип и параметры входных данных и провести статистический анализ полученных результатов. В данной работе будет поведена экспериментальная оценка программных средства в зависимости от типа исходного журнала событий и параметров анализа.

4.4.1 Планирование эксперимента

Для эксперимента было выбрано два журнала – на 660 записей, другой – 10000 записей. Структура журналов будет отличаться, однако основные атрибуты у них будут присутствовать – атрибут группы, типа события и времени. В качестве варьируемых параметров будут использоваться тип алгоритма секвенциального анализа и уровень поддержки искомых последовательностей. Структура первого журнала приведена на рисунке 26.

Case	Event Type	Start Time
Case_1	Request for a loan	1.8.2015 21:11
Case_1	Approve clients for automatic credit check	2.8.2015 0:52
Case_1	Issue a loan	2.8.2015 23:04
Case_1	Automated credit check	2.8.2015 5:21
Case_10	Request for a loan	1.8.2015 22:33
Case_10	Approve clients for automatic credit check	2.8.2015 11:48
Case_10	Manual credit check	3.8.2015 2:15

Рисунок 26 – Структура журнала событий на 660 записей

Журнал содержит такие атрибуты, как идентификатор события (case), тип события (Event Type) и время возникновения (Start Time).

На рисунке 27 можно увидеть структуру журнала на 10000 записей.

	Case	Event Type	Start Time	Cost
	97395276100	Confirmed Delivery Date	09.08.2017 22:57:48	24
	97395276100	Delivery Completed	06.08.2017 18:13:19	27
	97395276100	Shipment Sent	04.08.2017 20:45:04	10
	97395276100	Sales Order Created	31.07.2017 09:26:26	25
	97395276100	Payment Received	25.08.2017 13:49:00	27
	97395276100	Picking Done	02.08.2017 20:47:38	16
	97395276100	Invoice Created	10.08.2017 07:21:05	10
	97395276099	Payment Received	05.04.2016 09:25:27	7
	97395276099	Invoice Created	21.03.2016 15:49:37	32

Рисунок 27 – Структура журнала событий на 10000 записей

Этот журнал содержит идентификатор события (case), тип события (Event Type), время возникновения (Start Time), а также дополнительный атрибут, отображающий стоимость заказа.

Для того, чтобы отобразить ход проведения эксперименты, был разработан сценарий (рисунок 28).

Сначала мы открываем разработанные программные средства и создаем новый проект. Затем мы указываем в окне создания проекта его название, источник журнала (в рамках эксперимента файл), и параметры выборки из журнала. После создания проекта выполняем предобработку журнала и выявляем входные последовательности для дальнейшего анализа. Далее создаем новое решение, в котором указываем параметры анализа (алгоритм, уровень поддержки) и выполняем анализ.

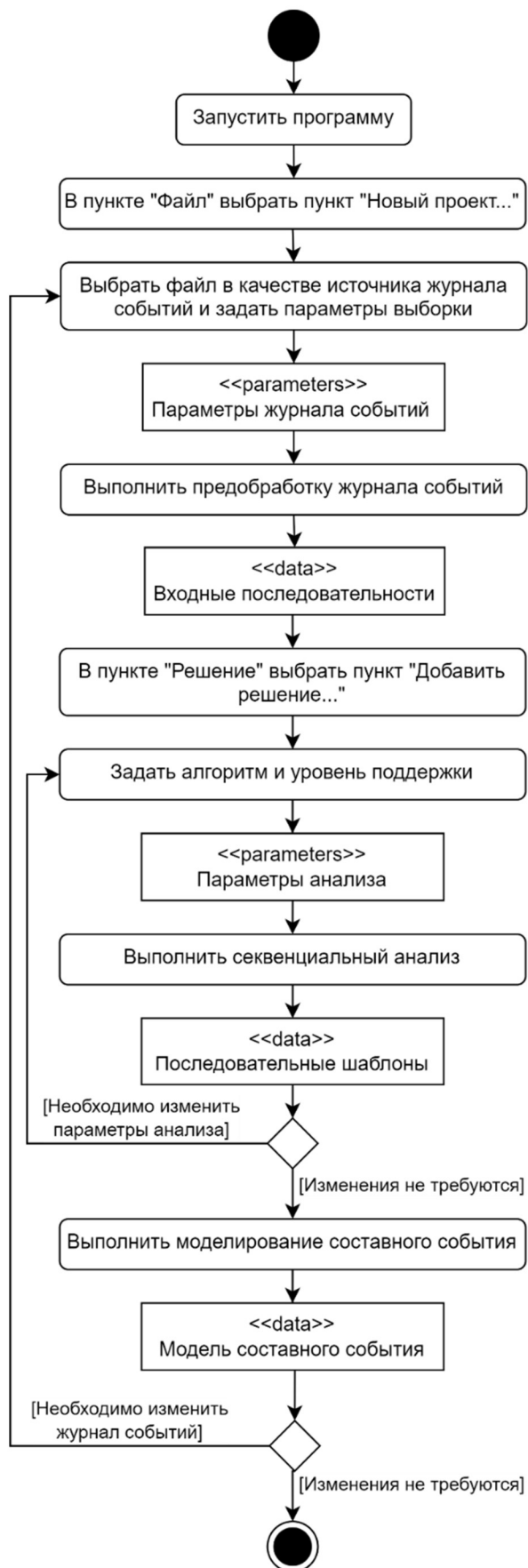


Рисунок 28 – Сценарий эксперимента

Мы получим первый промежуточный результат работы программных средств. Далее мы изменяем алгоритм и уровень поддержки, чтобы выявить зависимость выходной модели составного события от типа алгоритма и уровня поддержки искомых последовательностей. Те же самые шаги проводим и для журнала, содержащего 10000 записей. Для того, чтобы сократить задержки в процессе анализа журнала, поставим ограничение в 2000 записей на выборку из журнала событий. Этого будет достаточно, чтобы провести сравнение с моделью составного события из журнала с небольшим количеством записей.

4.4.2 Проведение эксперимента

При открытии программы мы увидим интерфейс главного окна (рисунок 29), на котором в левой части расположена область обозревателя проекта (Project Explorer), отображающего структуру проекта в виде дерева. В правой части расположена, консоль (Output), в которой выводится информация об отдельных этапах анализа, и область предпросмотра (Preview) для визуализации результатов анализа. В верхней части расположена панель инструментов и меню.

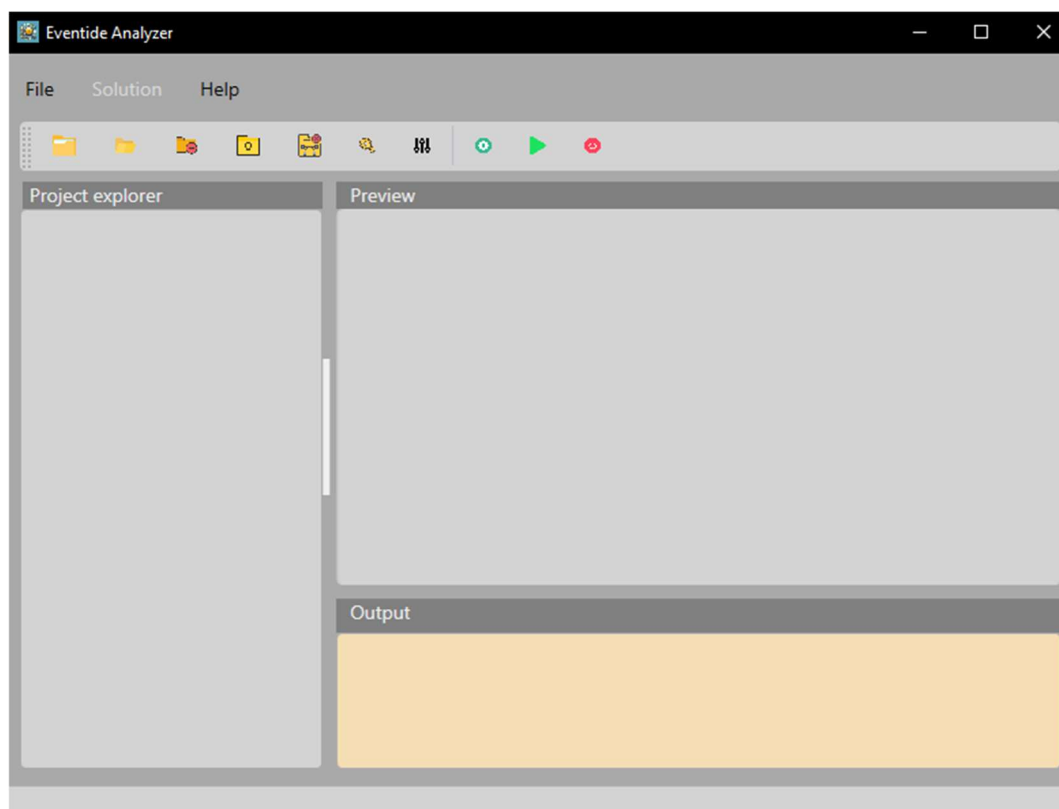


Рисунок 29 – Главное окно программы

В меню выбираем пункт «File» и в выпавшем списке (рисунок 30) выбираем создание нового проекта «New project...».

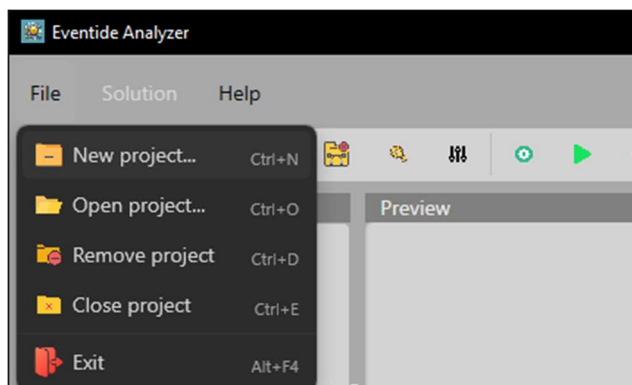


Рисунок 30 – Пункт меню «File»

В появившемся диалоговом окне указываем название проекта «Project1». В качестве источника журнала событий выбираем «File» и указываем к нему путь (рисунок 31).

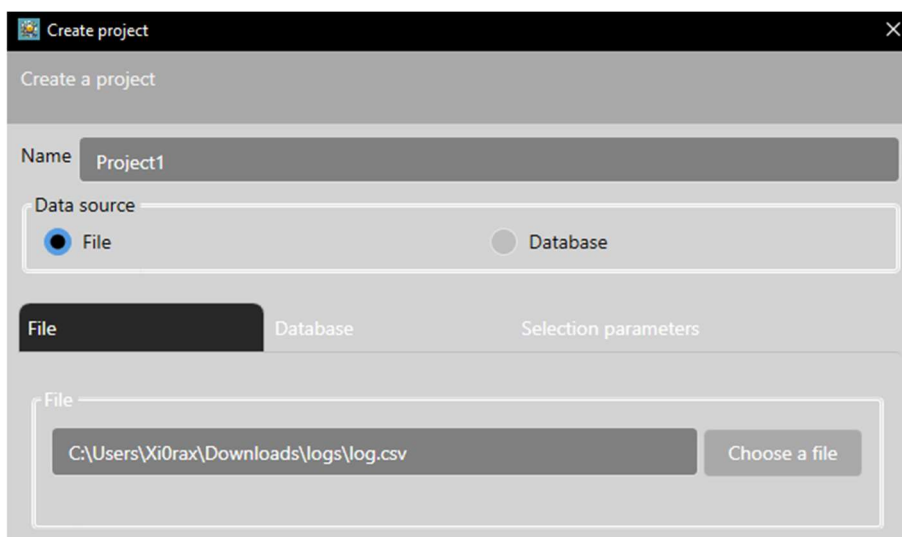


Рисунок 31 – Выбор источника журнала событий

Во вкладке «Selection parameters» (рисунок 32) определяем структуру журнала событий и параметры выборки. Указываем, какой столбец в нем будет использоваться в качестве атрибута группировки (Group column), какой в качестве идентификатора или типа события (Event column), а также атрибут времени (Time column). Также можно задать ограничения на количество выбираемых строк из журнала событий (limit) и временной диапазон произошедших событий (Range).

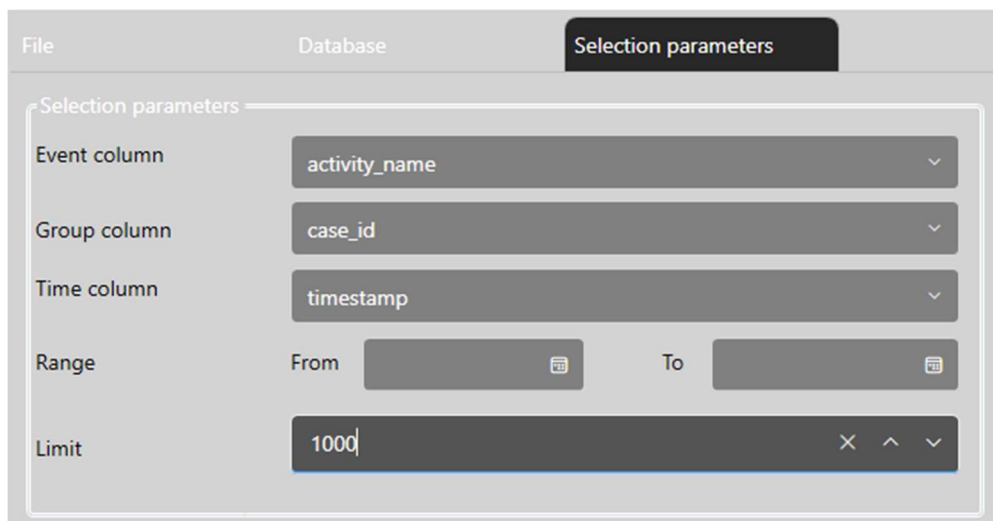


Рисунок 32 – Параметры выборки

После определение параметров фильтрации и структуры журнала событий можно выполнить его предварительную обработку. Нажимаем на кнопку «Preprocess event log» на панели инструментов (рисунок 33).

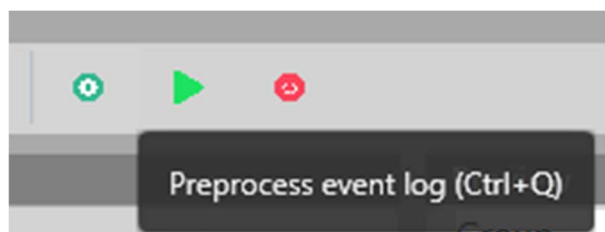


Рисунок 33 – Кнопка «Preprocess event log»

После этого мы получим входные последовательности (рисунок 34), которые можно использовать для секвенциального анализа и моделирования составного события.

Preview
Transaction
Request for a loan,Approve clients for automatic credit check,Issue a loan,Automated credit check
Request for a loan,Approve clients for automatic credit check,Manual credit check,Issue a loan
Request for a loan,Reject request for a loan
Request for a loan,Approve clients for automatic credit check,Manual credit check,Issue a loan
Request for a loan,Issue a loan
Request for a loan,Approve clients for automatic credit check,Manual credit check,Issue a loan

Рисунок 34 – Результаты предобработки журнала

Для проведения анализа необходимо создать решение. Для этого переходим в пункт меню «Solution» и в выпавшем списке выбираем «Add solution...» (рисунок 35).

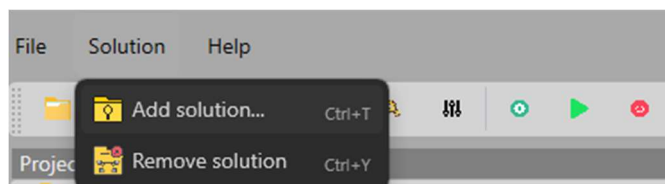


Рисунок 35 – Пункт меню «Решение»

В появившемся диалоговом окне (рисунок 36) указываем название решения, предпочитаемый алгоритм секвенциального анализа и уровень поддержки.

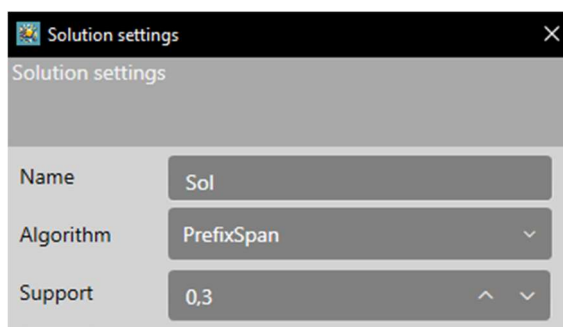


Рисунок 36 – Параметры анализа

После этого можно провести анализ. Для этого нажимаем кнопку «Run analysis» на панели инструментов (рисунок 37).

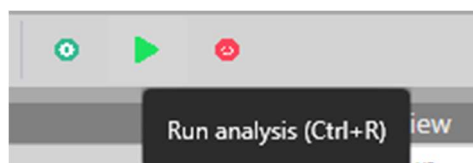


Рисунок 37 – Кнопка «Выполнить анализ»

После анализа будет получен список последовательностей с указанным уровнем поддержки (рисунок 38) и модель составного события (рисунок 39). Последовательности будут сохранены в файле sequences.json, а модель события – в файле pmodel.json.

Preview	
Sequence	Support
Request for a loan	26
Request for a loan->Approve clients for automatic credit check	18
Request for a loan->Approve clients for automatic credit check->Issue a loan	12
Request for a loan->Approve clients for automatic credit check->Manual credit check	9
Request for a loan->Approve clients for automatic credit check->Manual credit check->Issue a loan	8

Рисунок 38 – Результаты секвенциального анализа

Модель составного события представляет собой сеть Петри. Данную модель можно масштабировать и перемещать ее отдельные элементы.

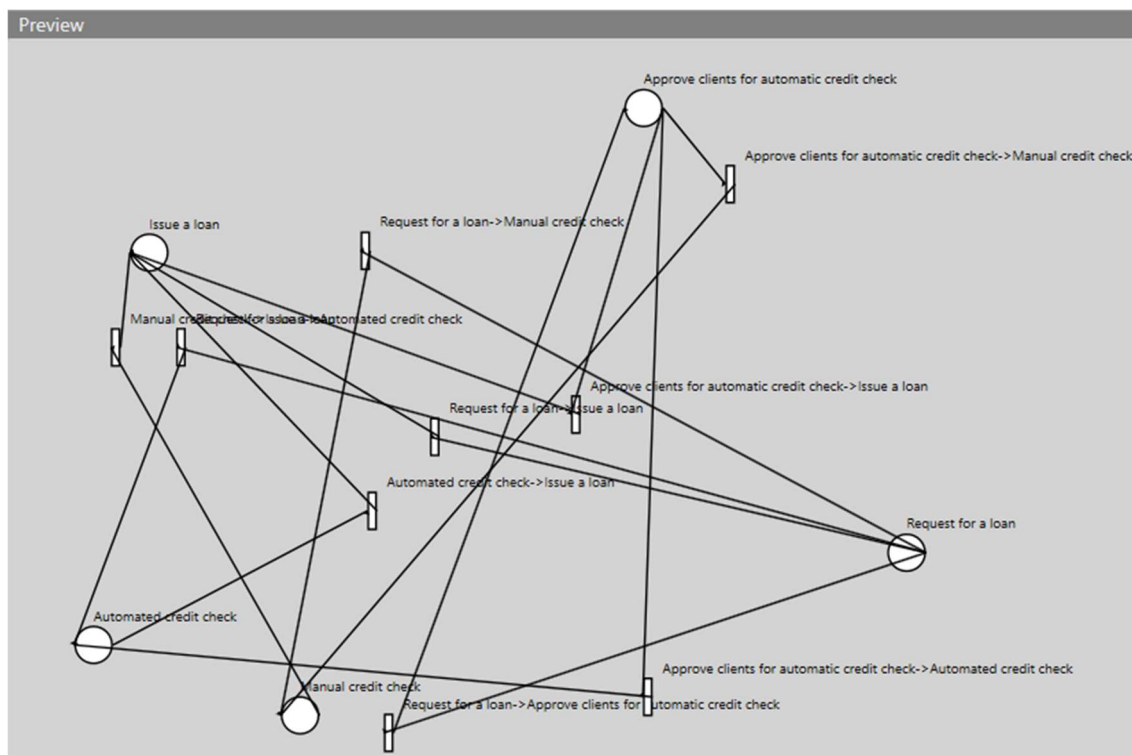


Рисунок 39 – Модель составного события

В данной модели можно увидеть только операции следования (SEQUENCE) между элементами составного события. Для того, чтобы проверить работу программных средств при моделировании других типов связей, используем журнал на 10000 записей.

4.4.3 Анализ экспериментальных данных

Для проведения следующего этапа эксперимента будет использован журнал событий, содержащий 10000 записей с ограничением выборки на 2000 записей. Эксперимент будет проведен для уровней поддержки, равных 0.2 и 0.4, соответственно, для каждого алгоритма секвенциального анализа (GSP, PrefixSpan и BIDE).

Рассмотрим результаты эксперимента. При использовании алгоритма GSP и уровне поддержки 0.2 программные средства позволяют найти значительное количество последовательных и параллельных компонентов

составного события (рисунок 40). Затраченное время на поиск и моделирование составляет 2500 мс.

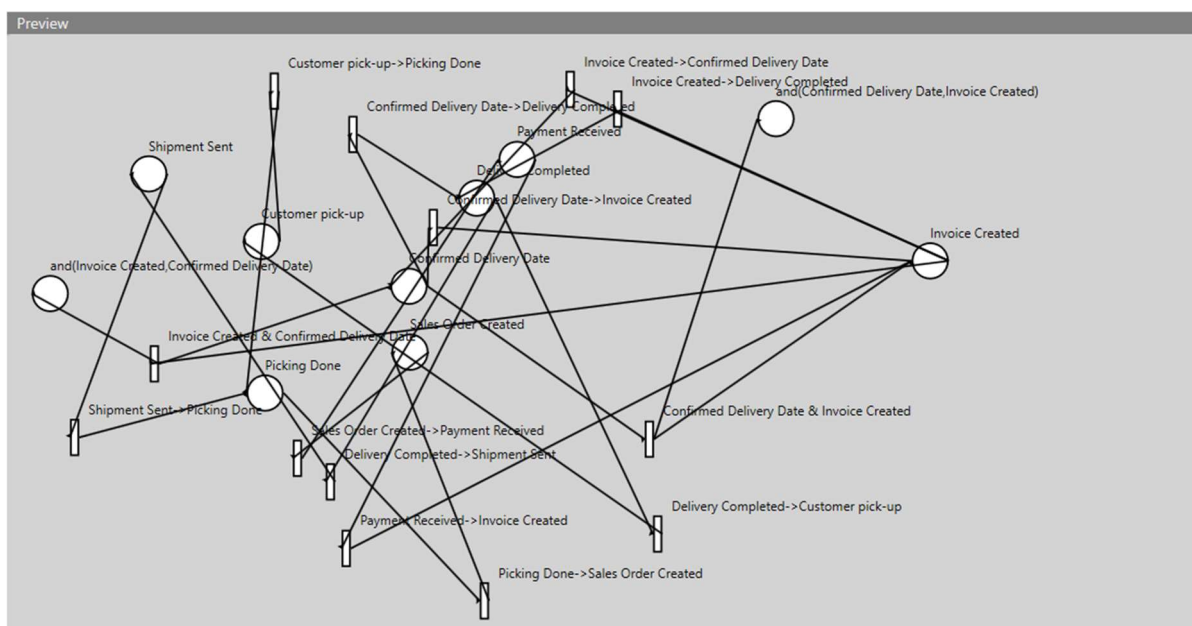


Рисунок 40 – Результаты эксперимента (Алгоритм GSP, уровень поддержки 0.2)

Однако при увеличении уровня поддержки до 0.4 программные средства не позволяют найти параллельные компоненты составного события (рисунок 41). Время на поиск и моделирование составило 1200 мс.

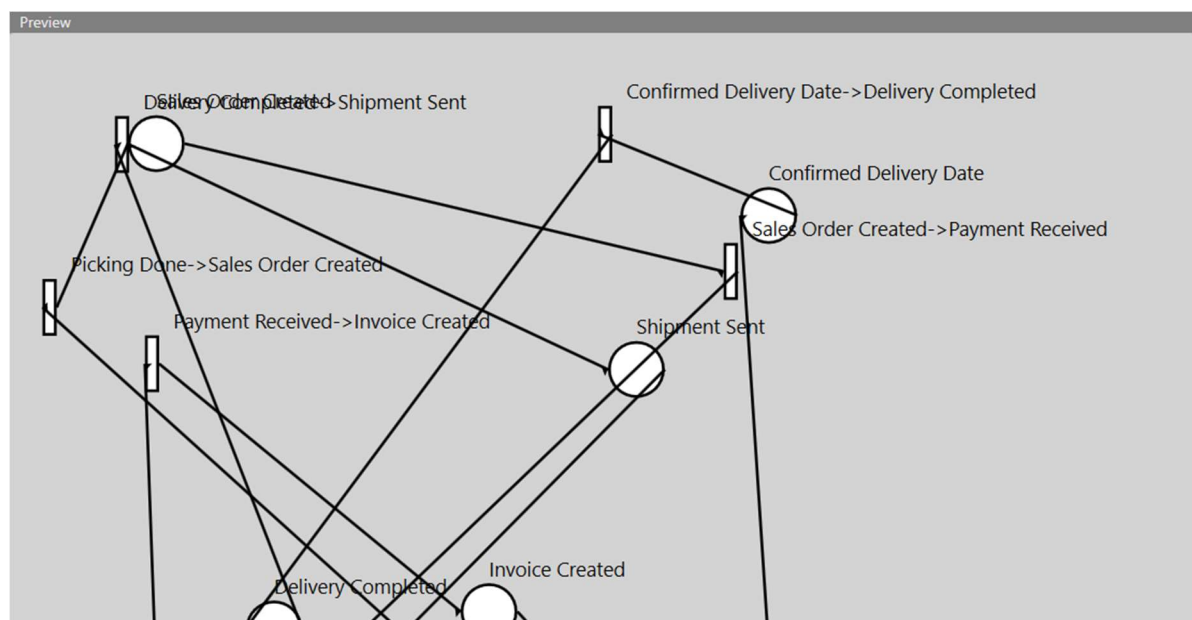


Рисунок 41 – Результаты эксперимента (Алгоритм GSP и уровень поддержки 0.4)

Проведем следующий этап эксперимента с использованием алгоритма PrefixSpan. При уровне поддержки 0.2 мы можем увидеть большое количество

последовательных и параллельных компонентов составного события (рисунок 42). Затраченное время составило 1500 мс.

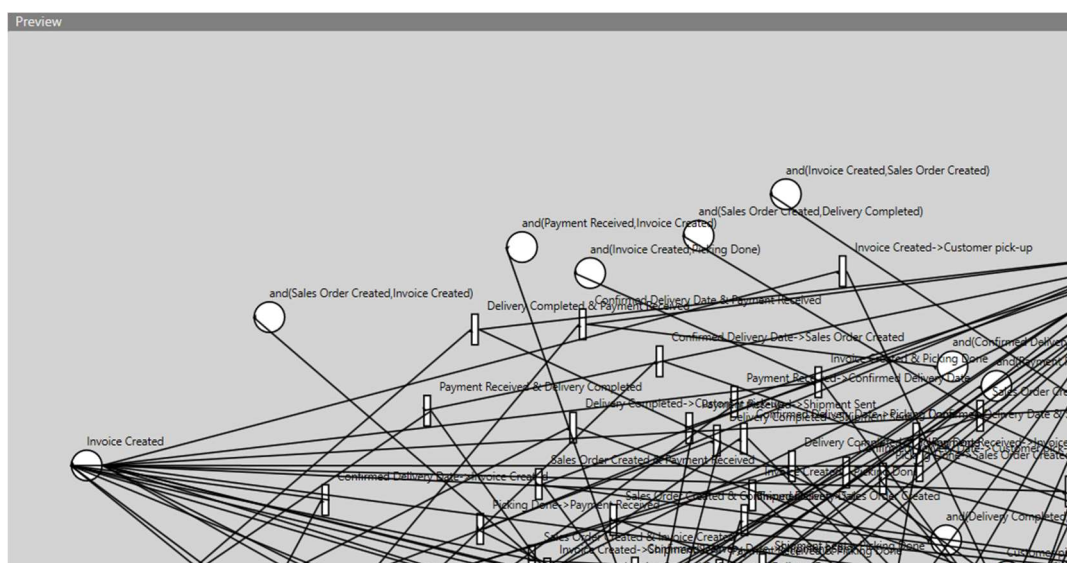


Рисунок 42 – Результаты эксперимента (Алгоритм PrefixSpan, уровень поддержки 0.2)

Изменим уровень поддержки до 0.4. При этом мы можем видеть, что программные средства все равно находят параллельные компоненты составного события (рисунок 43). Затраченное время составило 900 мс.

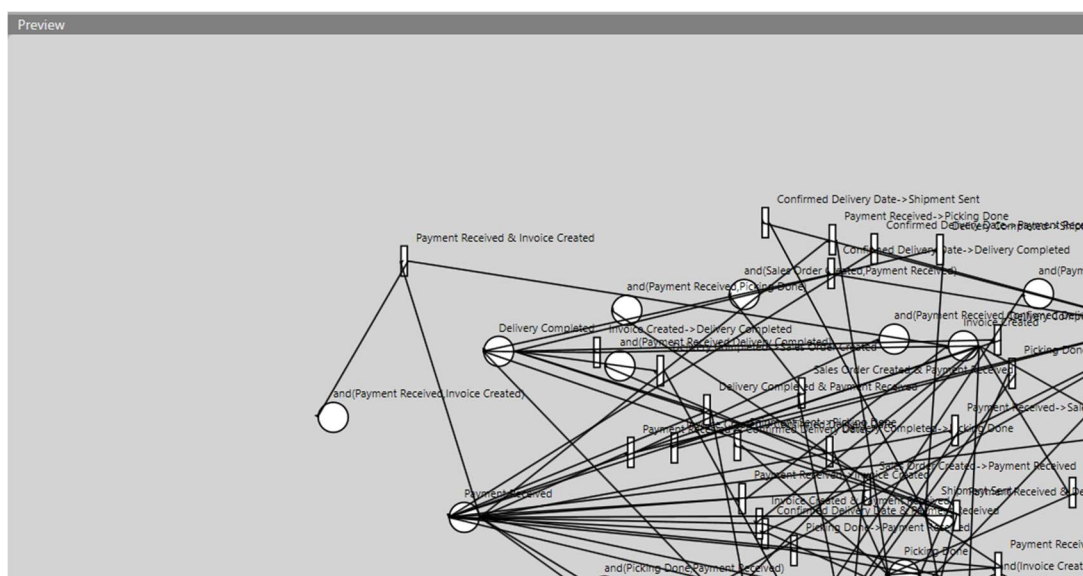


Рисунок 43 – Результаты эксперимента (Алгоритм PrefixSpan, уровень поддержки 0.4)

Аналогично проведем эксперимент с использованием алгоритма BIDE. При уровне поддержки 0.2 программные средства позволяют найти последовательные и параллельные компоненты составного

события (рисунок 44). Время на поиск и моделирование шаблона составного события составляет 1900 мс.

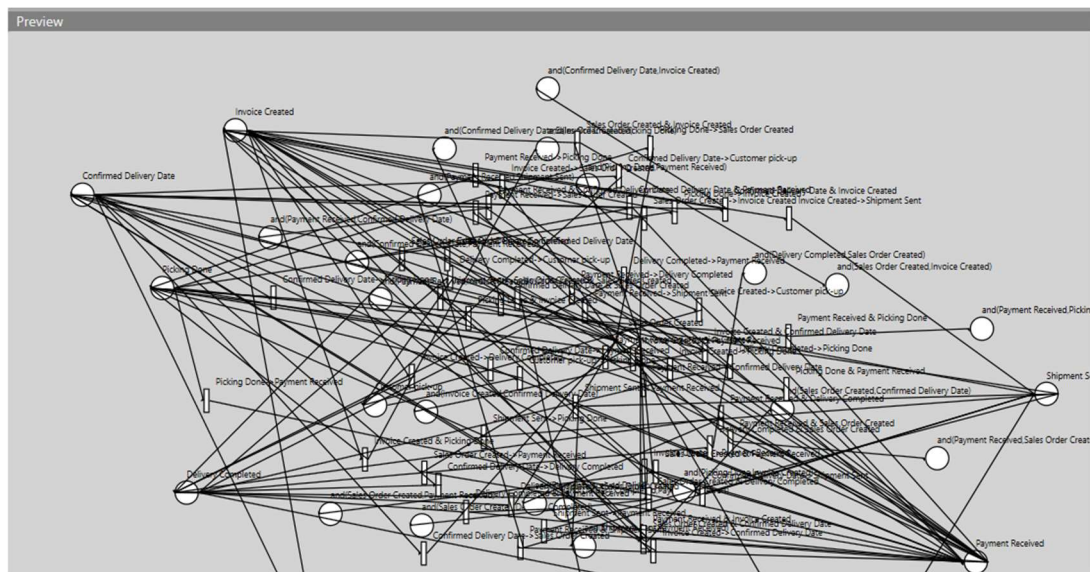


Рисунок 44 – Результаты эксперимента (Алгоритм BIDE, уровень поддержки 0.2)

Изменим уровень поддержки для алгоритма BIDE до 0.4 и увидим, что программные средства также могут найти параллельные компоненты составного события, при этом, с алгоритмом BIDE программные средства находят больше параллельных компонентов, чем с алгоритмом PrefixSpan при том же уровне поддержки (рисунок 45). Затраченное время составляет 1600 мс.

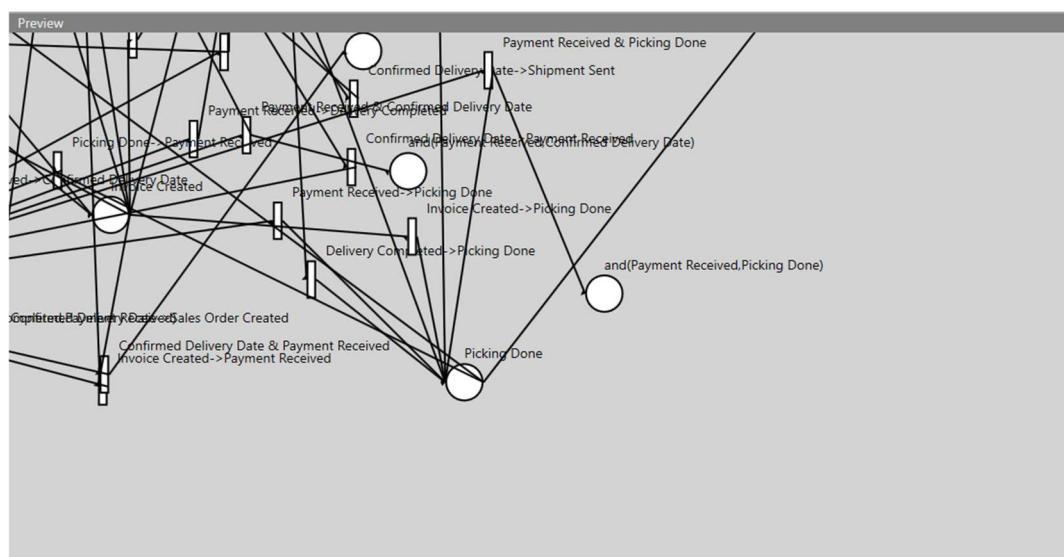


Рисунок 45 – Результаты эксперимента (Алгоритм BIDE, уровень поддержки 0.4)

Более подробно результаты эксперимента приведены в таблице 33.

Таблица 33 – Результаты эксперимента

Алгоритм	Уровень поддержки	Последовательные компоненты	Параллельные компоненты	Время
GSP	0.2	+	+	2500 мс
	0.4	+	-	1200 мс
PrefixSpan	0.2	+	+	1500 мс
	0.4	+	+	900 мс
BIDE	0.2	+	+	1900 мс
	0.4	+	+	1600 мс

Исходя из данных в таблице выше можно сказать, что моделирование составного события с использованием алгоритма PrefixSpan осуществляется лучше всего. Время для поиска и моделирования затрачивается гораздо меньше, поскольку алгоритму не нужно перебирать все последовательные шаблоны, а достаточно только изменять префикс и отбрасывать постфикс тех шаблонов, которые не соответствуют требуемому уровню поддержки. Для улучшения результатов можно уменьшить уровень поддержки поиска, чтобы получить дополнительные детали составного события. Хуже всего себя показали программные средства при использовании алгоритма GSP. Требуется значительное время для поиска шаблонов, а при увеличении уровня поддержки вовсе теряются параллельные компоненты составного события. Алгоритм BIDE требует значительное время для поиска шаблонов, но при этом позволяет получить больше деталей для модели составного события, чем алгоритм PrefixSpan.

4.5 Выводы

В данном разделе был рассмотрен процесс контроля качества программных средств выявления шаблонов событий. Процесс контроля качества включает в себя функциональное тестирование программных средств и оценку качества кода. Для функционального тестирования были разработаны тест-кейсы с использованием классов эквивалентности и таблицы решений. Для оценки качества исходного кода программных средств была рассчитана и проанализирована метрика Холстеда.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной работы был выполнен анализ предметной области, в котором рассмотрена актуальность разработки алгоритмов и программных средств для выявления и моделирования составных событий на основе интеллектуального анализа журнала событий, и поставлена задача на разработку программных средств. Был проведен обзор и сравнительный анализ модельных представлений составных событий, а также сравнение и анализ алгоритмов секвенциального анализа и алгоритмов извлечения процессов. Была разработана методика для выявления и моделирования составных событий, основанная как на использовании алгоритмов секвенциального анализа, так и на основе модифицированных алгоритмов извлечения процессов.

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

В результате были разработаны программные средства для выявления и моделирования составных событий на основе интеллектуального анализа журнала событий, которое выполняет следующие функции:

- предварительная обработка журнала событий,
- загрузка журнала событий из базы данных/файла,
- выбор алгоритма секвенциального анализа и определение его параметров,
- просмотр и сохранение выявленных последовательностей в файл,
- выбор алгоритма моделирования шаблона составного события и определение его параметров,
- просмотр и сохранение сформированной модели в файл.

Реализованы плагин-библиотеки для моделирования составных событий и предобработки журнала событий, которые могут быть использованы другими разработчиками в своем продукте.

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

В рамках экспериментальной оценки было установлено, чтобы программные средства выдают адекватный результат моделирования составного события и определены оптимальные условия для эффективного моделирования составных событий.

Дальнейшие исследования будут включать в себя расширение выбора алгоритмов секвенциального анализа для поиска вероятных последовательностей событий, а также алгоритмов моделирования шаблонов составных событий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Данилина, Е. А. модельное представление сложных событий / Е. А. Данилина, Д. А. Королева, С. В. Шибанов // Молодежь и наука: актуальные проблемы фундаментальных и прикладных исследований. : материалы II Всероссийской национальной научной конференции студентов, аспирантов и молодых ученых, Комсомольск-на-Амуре, 08–12 апреля 2019 года. Том Часть 2. – Комсомольский-на-Амуре государственный университет: Комсомольский-на-Амуре государственный университет, 2019. – С. 269-272.
2. Шибанов, С. В. Алгоритмы секвенциального анализа для построения вероятных последовательностей событий / С. В. Шибанов // Новые информационные технологии и системы: Сборник научных статей по материалам XVII Международной научно-технической, Пенза, 18–19 ноября 2020 года. – Пенза: Пензенский государственный университет, 2020. – С. 69-73.
3. Печаткин, К. В. Обзор модельных представлений составных событий / К. В. Печаткин, С. В. Шибанов // Информационные технологии в науке и образовании. Проблемы и перспективы: сборник статей по материалам X Всероссийской научно-практической конференции, Пенза, 15 марта 2023 года. – Пенза: Пензенский государственный университет, 2023. – С. 195-198.
4. Adaikkalavan R. Event Operators: Formalization, Algorithms, and Implementation Using Interval-Based Semantics. / R. Adaikkalavan. – Arlington: MS Thesis, 2002. – 82 p.
5. Шибанов, С. В. Расширенная модель сложных событий в системах активных правил / С. В. Шибанов, Д. А. Королева, Е. А. Данилина // Методы и средства обработки и хранения информации : Межвузовский сборник научных трудов / Под ред. Б.В. Кострова. – Рязань: Редакционно-издательский центр РГРТУ, 2019. – С. 186-195.
6. Engstrom, H., Berndtsson, M., Lings, B.: ACOOD Essentials. Technical report / H. Engstrom, M. Berndtsson, University of Skovde. – 1997.

7. Gatzui S., Dittrich K.R., "Events in an Active Object-Oriented Database System", Proceedings of the 1st International Workshop on Rules in Database Systems / S. Gatzui, K.R. Dittrich, Edinburgh, Scotland, 30 August – 1 September 1993. pp. 23-39.

8. Введение в анализ ассоциативных правил. [Электронный ресурс]. URL: <https://loginom.ru/blog/associative-rules> (дата обращения: 20.05.2024)

9. Печаткин, К. В. Processmining как инструмент выявления сложных событий / К. В. Печаткин, С. В. Шибанов // Прикладная математика и информатика: современные исследования в области естественных и технических наук: Сборник материалов IX Международной научно-практической конференции (школы-семинара) молодых ученых, Тольятти, 18–20 апреля 2023 года. – Тольятти: Тольяттинский государственный университет, 2023. – С. 314-319.

10. Барсегян, А. А. Анализ данных и процессов: учеб. пособие / А. А. Барсегян, С. И. Елизаров, М. Д. Тесс, И. И. Холод, М. С. Куприянов. – СПб.: BVX, 2018.

11. Печаткин, К. В. Применение машинного обучения для обработки журнала событий / К. В. Печаткин, С. В. Шибанов // Методы, средства и технологии получения и обработки измерительной информации («Шляндинские чтения – 2023»): Материалы XV Международной научно-технической конференции с элементами научной школы и конкурсом научно-исследовательских работ для студентов, аспирантов и молодых ученых, Пенза, 02–03 ноября 2023 года / Под редакцией Е.А. Печерской. – Пенза: Пензенский государственный университет, 2023. – С. 168-170.

12. Печаткин, К. В. Обзор алгоритмов фильтрации журналов событий / К. В. Печаткин, С. В. Шибанов // Наука, инновации и технологии: от идей к внедрению: Материалы Всероссийской научно-практической конференции молодых ученых, Комсомольск-на-Амуре, 16–17 ноября 2023 года. – Комсомольск-на-Амуре: Комсомольский-на-Амуре государственный университет, 2023. – С. 526-529.

13. Прошкина Е. Н. Язык моделирования информационных процессов UML 2.0 (учебное пособие) / Прошкина Е. Н, Балашова И. Ю. – Пенза, Издательство ПГУ, 2018. – 112 с.

14. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке С# / Р. Мартин, М. Мартин – пер. с англ. – СПб: Символ-Плюс, 2011. – 768 с.

15. 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: 20.05.2024).

16. QA evolution // Виды тестирования ПО [Электронный ресурс] URL: <https://qaevolution.ru/testirovanie-po/vidy-testirovaniya-po/> (дата обращения: 05.04.2024).

17. 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

(Обязательное)

Алгоритмы и программные средства выявления шаблонов составных
событий на основе интеллектуального анализа журнала событий

Результаты тестирования



Рисунок А.1 – Результаты тестирования 1

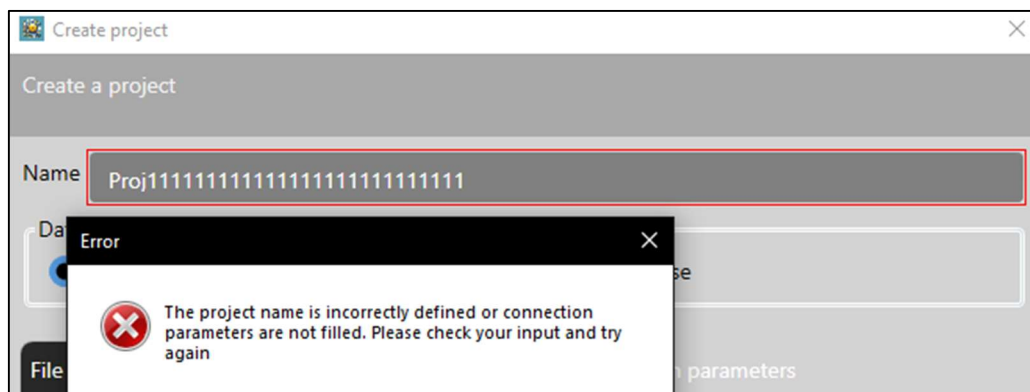


Рисунок А.2 – Результаты тестирования 2

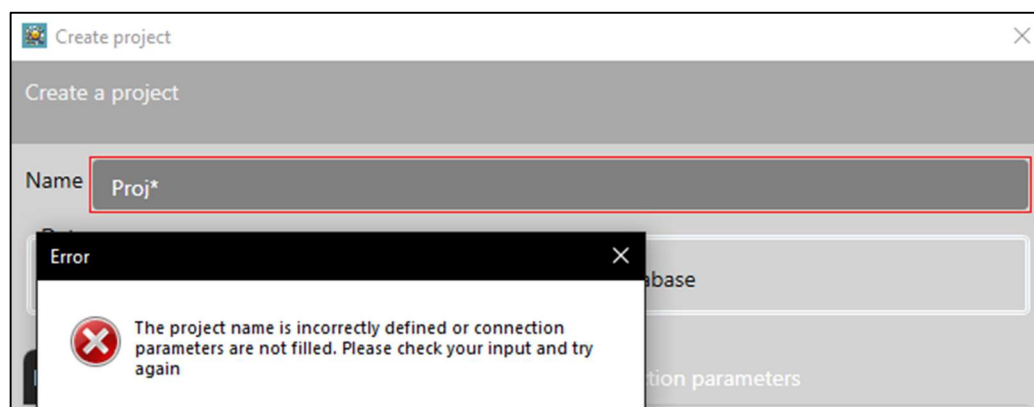


Рисунок А.3 – Результаты тестирования 3

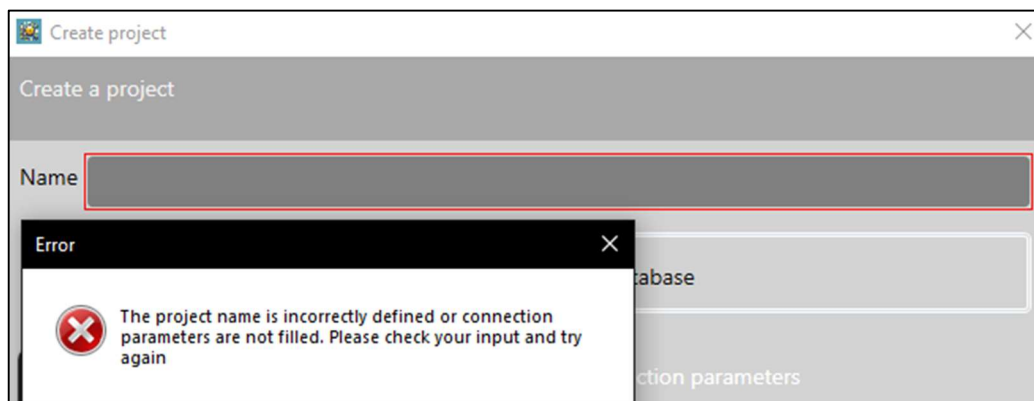


Рисунок А.4 – Результаты тестирования 4

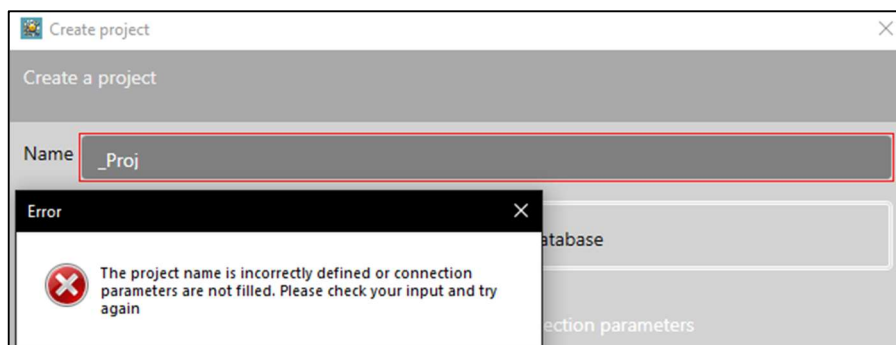


Рисунок А.5 – Результаты тестирования 5

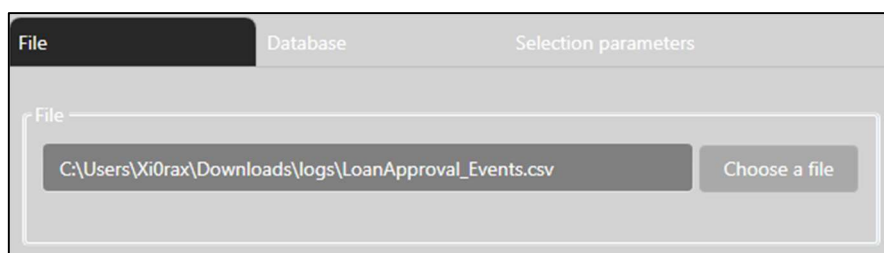


Рисунок А.6 – Результаты тестирования 6

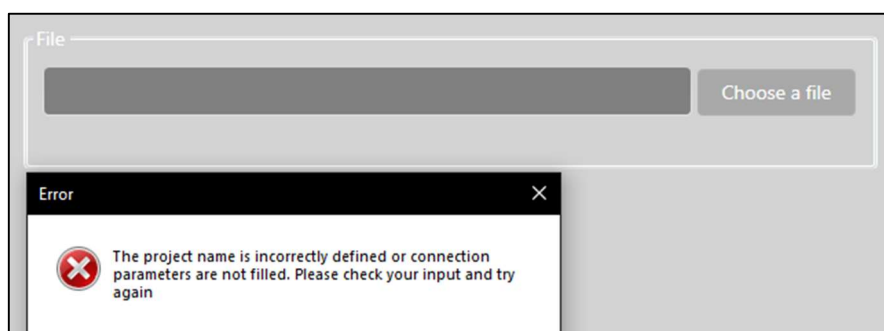


Рисунок А.7 – Результаты тестирования 7

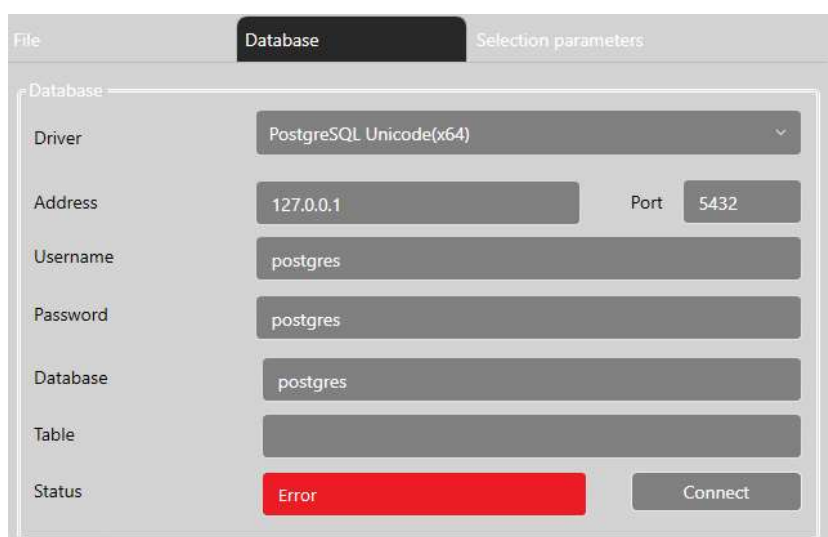


Рисунок А.8 – Результаты тестирования 8

The image shows a software window titled "Database" with a "Selection parameters" tab. It contains a form for configuring a database connection. The fields are as follows:

Field	Value
Driver	PostgreSQL Unicode(x64)
Address	127.0.0.1
Port	5432
Username	postgres
Password	postgres
Database	postgres
Table	log
Status	Active

A green "Active" status bar is visible at the bottom left, and a "Connect" button is at the bottom right.

Рисунок А.9 – Результаты тестирования 9

ПРИЛОЖЕНИЕ Б

(Обязательное)

Алгоритмы и программные средства выявления шаблонов составных
событий на основе интеллектуального анализа журнала событий

Исходный код сигнатур классов plugin-библиотек

Листинг Б.1 – Файл SequenceMiningAlgorithm.cs

```
using EventAnalyzeProjLib;
using System.Text.Json.Nodes;

namespace SeqMineLib
{
    public abstract class SequenceMiningAlgorithm(List<Sequence>
inputData, double minSupport, Logger logger = null)
    {
        protected Logger logger = logger;

        protected readonly double _minSupport = minSupport;

        protected List<Sequence> transactions = inputData;

        protected Dictionary<Sequence, int> freqPatterns = [];

        public abstract string Name { get; }

        public abstract void Mine();

        public Dictionary<Sequence, int> GetResult()
        {
            return freqPatterns;
        }

        public SequenceMiningAlgorithm() : this([], 0) {}

        public void Save(string filename)
        {
            JsonObject json = [];

            foreach (var item in freqPatterns)
            {
                json[item.Key.ToString()] = item.Value;
            }
            File.WriteAllText(filename, json.ToString());
        }
    }
}
```

Листинг Б.2 – Файл ProcessMiner.cs

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

namespace ProcMineLib
{
    public abstract class ProcessMiner
    {
        public string Name { get; set; }

        public abstract void Mine(string logData);
    }
}
```

```

    }
}

Листинг Б.3 – Файл EventLogProcessor.cs

namespace EventLogLib
{
    public class EventLogProcessor(SourceData src)
    {
        protected List<EventLogEntry> eventLog = [];

        protected List<string> attributes;

        protected readonly SourceData srcData = src;

        public List<string> Attributes { get => attributes; set =>
attributes = value; }

        public virtual IEnumerable<EventLogEntry> SelectEventLog()
        {
            return eventLog;
        }

        public List<List<string>> GetActivitySequences()
        {
            var sequences = eventLog
                .OrderBy(e => e.Timestamp)
                .GroupBy(e => new { e.GroupColumn, })
                .Select(group => group.Select(e =>
e.EventColumn).ToList())
                .Take(int.Parse(srcData.FilterParams["limit"]))
                .ToList();

            return sequences;
        }
    }
}

```

ПРИЛОЖЕНИЕ В

(Обязательное)

Алгоритмы и программные средства выявления шаблонов составных
событий на основе интеллектуального анализа журнала событий

Презентация

Алгоритмы и программные средства выявления шаблонов составных событий на основе интеллектуального анализа журнала событий

ВЫПОЛНИЛ: СТУДЕНТ ГРУППЫ 22ВВИМ ПЕЧАТКИН К.В.

РУКОВОДИТЕЛЬ: ДОЦЕНТ, К.Т.Н ШИБАНОВ С.В.

Актуальность исследования

- Развитие медицины, необходимость обнаружения более сложных заболеваний и анализа отдельных симптомов
- Развитие бизнеса, необходимость анализа поведения клиентов и выявление шаблонов потребления
- Необходимость анализа и мониторинга курсов валют в финансовой отрасли
- Необходимость выявления связей между пользователями в социальных сетях

Области применения



Объект и предмет исследования

- ▶ Объектом исследования являются процессы анализа журналов в системах обработки событий с целью выявления скрытых закономерностей.
- ▶ Предметом исследования модельные представления составных событий, методика, алгоритмы и программные средства выявления шаблонов составных событий на основе интеллектуального анализа журналов событий.



Цель и задачи

Цель:

исследование является разработка модельных представлений составных событий, методики, алгоритмов и программных средств выявления, моделирования и оценки достоверности шаблонов составных событий на основе данных журналов событий.

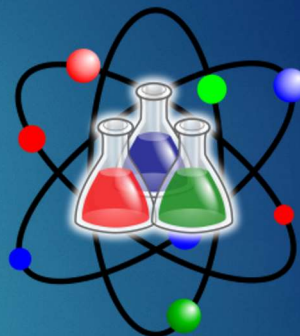


Задачи:

- формализация и модельное представление сложных составных событий;
- разработка методики выявления, моделирования и оценки достоверности шаблонов составных событий;
- разработка алгоритмов выявления, моделирования и оценки достоверности шаблонов составных событий;
- разработка программных средств выявления, моделирования и оценки достоверности шаблонов составных событий.

Научная новизна, оригинальность идей

Разработана методика выявления, моделирования и оценки достоверности шаблонов составных событий, включающая в себя использование алгоритмов секвенциального анализа и модифицированных алгоритмов извлечения процессов.



Практическая значимость

- разработано модельное представление сложных составных событий на основе раскрашенных сетей Петри
- разработаны плагины библиотеки для предварительной обработки и анализа журнала событий
- разработаны программные средства для выявления и анализа сложных составных событий.

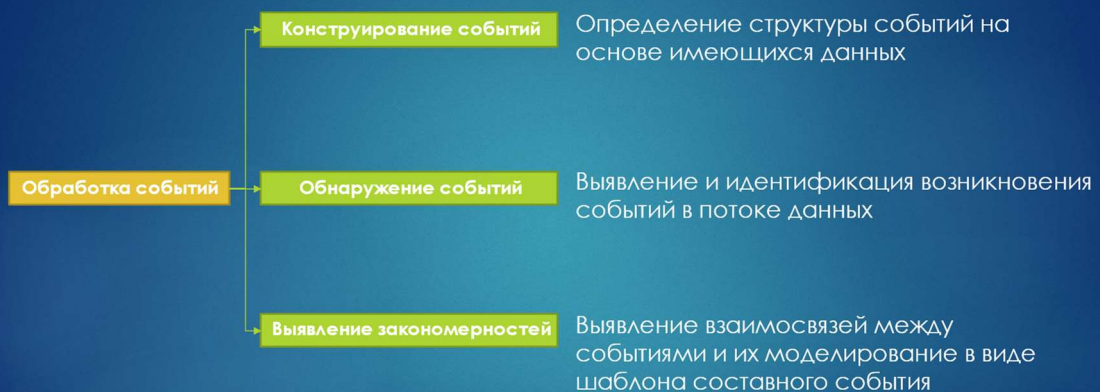


Событие и его виды

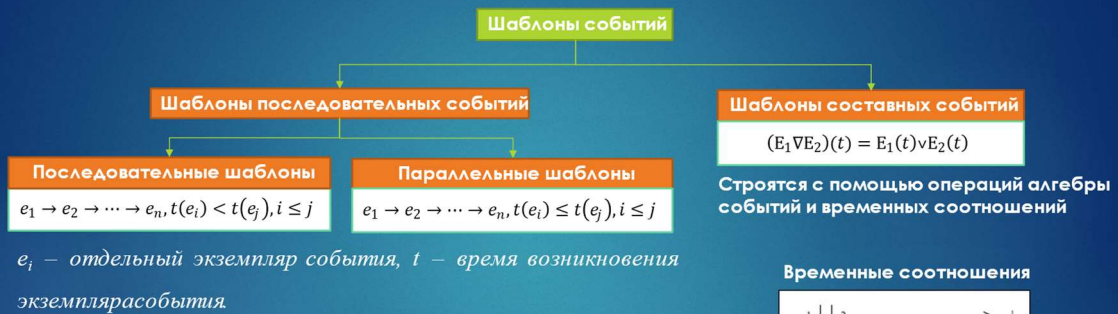
Событие – определенное действие в системе, которое может быть инициировано пользователем, устройством, или самой системой



Задачи обработки событий



Выявление закономерностей в виде шаблонов составных событий

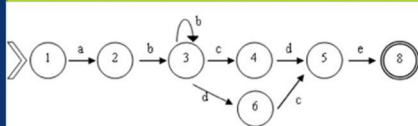


Алгебра событий

Операция	Описание операции	Представление
Дизъюнкция событий (OR)	Сложное событие возникает при появлении события E_1 или события E_2	$E_1 \nabla E_2$
Конъюнкция событий (AND)	Сложное событие возникает при появлении события E_1 и события E_2	$E_1 \Delta E_2$
Последовательность событий (SEQUENCE)	Сложное событие возникает при появлении события E_2 последовательно после события E_1	$E_1; E_2$
Одновременное событие	Сложное событие возникает при одновременном появлении событий E_1 и E_2	$E_1 = E_2$
Отрицание события	Сложное событие возникает при отсутствии появления события E_1 во временном интервале	$\neg(E_2)[E_1, E_3]$
Соединение событий	Сложное событие возникает, когда из n различных событий $E_1, E_2, \dots, E_n$ произошло m событий	$ANY(m, E_1, E_2, \dots, E_n)$
Апериодическое событие	Сложное событие возникает при появлении E_2 в пределах событий E_1 и E_3	$A(E_1; E_2; E_3)$ или $A^*(E_1; E_2; E_3)$
Периодическое событие	Сложное событие возникает при наличии постоянных временных шагов $\dagger$ между появлением событий E_1 и E_2	$P(E_1, \Pi[\text{параметры}], E_3)$ или $P^*(E_1, \Pi[\text{параметры}], E_3)$
Событие PLUS	Сложное событие возникает через время $\dagger$, после того, как произошло событие E_1	$Plus(E_1, n, \dagger)$

Модельные представления составных событий

Расширенный конечный автомат



$$M = (V, Q, q, F, \delta),$$

V – множество всех событий, которые могут произойти;

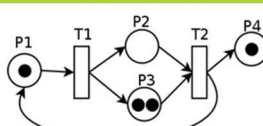
Q – множество внутренних состояний автомата;

q_0 – начальное состояние (событие-инициатор);

F – множество конечных состояний (событий терминаторов);

δ – функция переходов (операция алгебры логики).

Сеть Петри



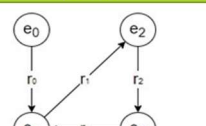
$$PN = (P, T, F),$$

P – конечное множество мест (событий, которые могут произойти);

T – конечное множество переходов (операции алгебры логики);

F – множество дуг.

Граф



$$G = (L, V),$$

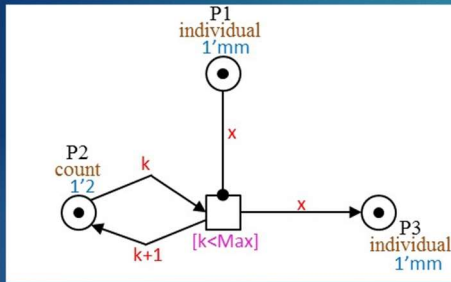
L – множество компонентов составного события (простых событий);

V – множество операций, определяемых алгеброй событий.

Модельные представления составных событий

Модельное представление	Достоинства	Ограничения
Граф (Дерево)	<ul style="list-style-type: none"> – удобство интерпретации и визуализации; – гибкость представления. 	<ul style="list-style-type: none"> – низкая эффективность при работе с событиями, имеющими сложный контекст.
Расширенный конечный автомат	<ul style="list-style-type: none"> – наличие формального языка; – возможность модификации и расширения. 	<ul style="list-style-type: none"> – сложность визуализации при росте масштаба модели; – сложность разработки и поддержки модели.
Сеть Петри	<ul style="list-style-type: none"> – наличие формального языка; – возможность моделирования расширенных классов составных событий, включая параллельные процессы, циклы или неопределенные условия; – возможность моделирования агрегированных событий. 	<ul style="list-style-type: none"> – необходима доработка модели для представления сложных составных событий.

Модельное представление на основе раскрашенной сети Петри



$$PN_c = (P, T, C, \hat{R}^-, \hat{R}^+, \hat{\mu}_0),$$

$C = \{c_1, c_2, \dots, c_d\}$ – цвет сети;

$R^- \in R^{(d+m) \times n}$ – матрица инцидентности дуг, входящих в переходы;

$$R^- = \begin{bmatrix} R_1^- \\ R_2^- \\ \vdots \\ R_d^- \end{bmatrix},$$

R_k^- – матрица инцидентности входящих дуг цвета $C_k, k = 1, \dots, d$;

$R^+ \in R^{(d+m) \times n}$ – матрица инцидентности дуг, выходящих из переходов;

$$R^+ = \begin{bmatrix} R_1^+ \\ R_2^+ \\ \vdots \\ R_d^+ \end{bmatrix},$$

где R_k^+ – матрица инцидентности входящих дуг цвета C_k ;

$\hat{\mu}_0 \in R^{d \times n}$ – матрица начальной маркировки.

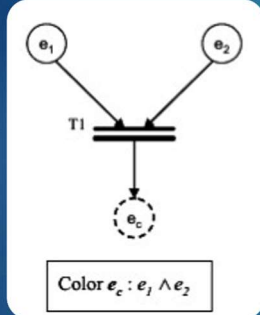
Модельное представление на основе раскрашенной сети Петри



Модельное представление на основе раскрашенной сети Петри

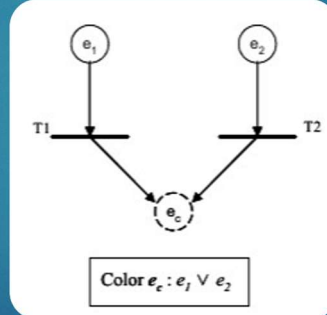
Примеры моделирования операций алгебры событий

Конъюнкция простых событий



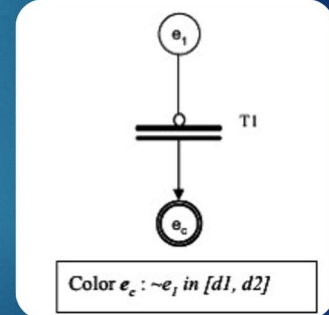
Color $e_c : e^1 \vee e^2$

Дизъюнкция простых событий



Color $e_c : e^1 \wedge e^2$

Отрицание событий



Color $e_c : \sim e^1 \text{ in } [q1, q2]$

Ассоциативные правила

Ассоциативные правила $A \rightarrow T$ имеют следующий вид:

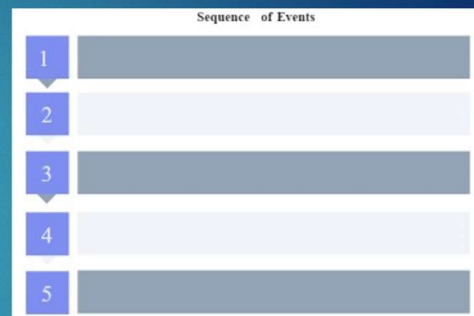
если $\langle \text{условие} \rangle$ то $\langle \text{результат} \rangle$,

где $\langle \text{условие} \rangle$ – набор объектов из множества A с которыми связаны объекты того же множества, включенные в $\langle \text{результат} \rangle$ данного правила.

Являются мощным инструментом для построения шаблонов событий, однако не учитывают временной аспект

Секвенциальный анализ

Секвенциальный анализ – это метод анализа данных, который используется для извлечения закономерностей и шаблонов из последовательностей событий или элементов.



Извлечение процессов (Process Mining)

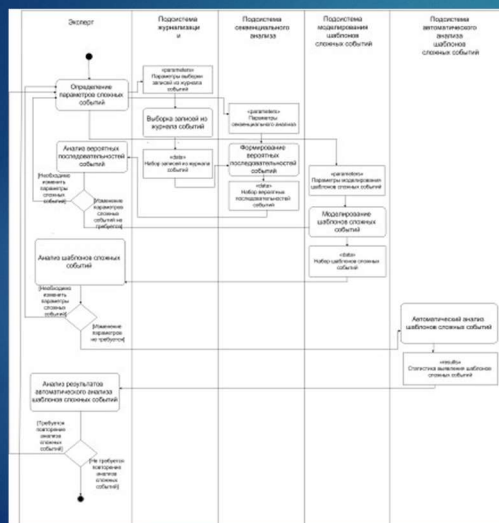
Process Mining – общее название ряда методов и подходов, предназначенных для анализа и усовершенствования процессов в информационных системах или бизнес-процессов на основании изучения журналов событий.



Достоинства и ограничения подходов

Секвенциальный анализ	Process Mining
<ul style="list-style-type: none"> ✓ Учет временного контекста ✓ Эффективное выявление и моделирование последовательностей – Не может моделировать сложные связи между событиями 	<ul style="list-style-type: none"> ✓ Учет временного контекста ✓ Может моделировать сложные связи ✓ Позволяет создавать модельные представления – Изначально создавался для моделирования процессов

Эффективный подход к выявлению и моделированию составных событий



Включает в себя следующие этапы:

- ✓ выборка и предварительная обработка данных из журнала событий;
- ✓ выявление вероятных последовательностей событий с использованием алгоритмов секвенциального анализа;
- ✓ моделирование шаблонов составных событий с применением модифицированных алгоритмов извлечения процессов;
- ✓ анализ и модификация полученной модели составного события.

Алгоритмы секвенциального анализа

Используются для формирования цепочек событий

Алгоритмы семейства

Apriori:

- Apriori
- AprioriAll
- GSP (Generalized Sequential Pattern)

Алгоритмы поиска

замкнутых последовательностей:

- CloSpan
- BIDE (Bidirectional Depth-First Enumeration)

Алгоритмы поиска в

глубину:

- SPADE (Sequential Pattern Discovery using Equivalence classes)
- PrefixSpan
- SPAM (Sequential Pattern Mining)

Инкрементные алгоритмы:

- SuffixTree
- FASTUP
- ISM (Incremental Sequence Mining)
- ISE (Incremental Sequential pattern mining)
- GSP+ (Generalized Sequential Pattern Plus)
- MFS+ (Mining Frequent Sequential patterns Plus)
- IncSp (Incremental Sequential Pattern)
- IncSpan (Incremental Span)
- IncSpan+ (Incremental Span Plus)
- MILE (Mining Incremental Large sequential patterns)

Алгоритмы секвенциального анализа

Алгоритм	Эффективность	Ресурсоемкость
AprioriAll	0,8	0,95
GSP	0,85	0,9
SPAM	0,9	0,8
SPADE	0,85	0,85
PrefixSpan	0,9	0,7
BIDE	0,95	0,75

Алгоритмы извлечения процессов

Используются для построения модельного представления на основе выявленных цепочек событий

- Альфа-алгоритм (Alpha-miner)
- Эвристический алгоритм (Heuristics miner)
- Генетический алгоритм (Genetic miner)
- Нечеткий алгоритм (Fuzzy miner)
- Вероятностный алгоритм
- Region miner
- Индуктивный алгоритм (Inductive miner)
- DWS-алгоритм (дизъюнктивная workflow-схема)

Алгоритмы извлечения процессов

Алгоритм	Эффективность	Ресурсоемкость
Альфа-алгоритм	0,6	0,7
Эвристический	0,75	0,8
Генетический	0,85	0,9
Индуктивный	0,8	0,9

Программные средства для выявления и моделирования составных событий

ARIS Process Performance Manager

Обеспечивает контроллинг бизнес-процессов, позволяя компаниям измерять и улучшать производительность своих процессов.

ProM

Обладает широким функционалом, который позволяет работать с различными типами входных данных. Часто используется как инструмент в академических и научных исследованиях.

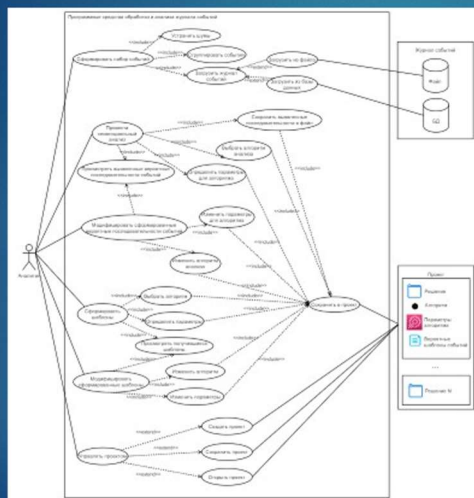
Celonis

Разработано в рамках партнерства с компанией SAP и интегрируется с ее продуктами.

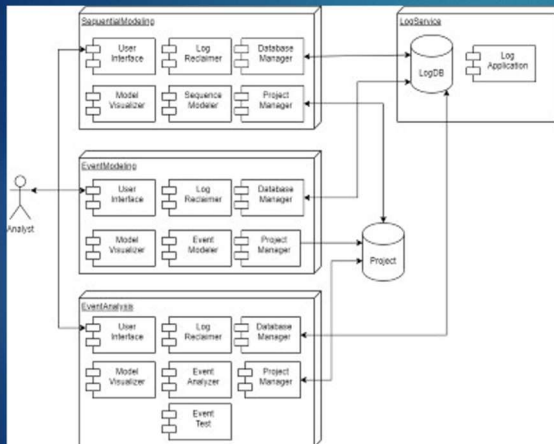
Disco

Коммерческий инструмент для анализа процессов использует нечеткие модели, что позволяет анализировать данные процессов с высокой степенью гибкости и адаптивности.

Программные средства для выявления и моделирования составных событий

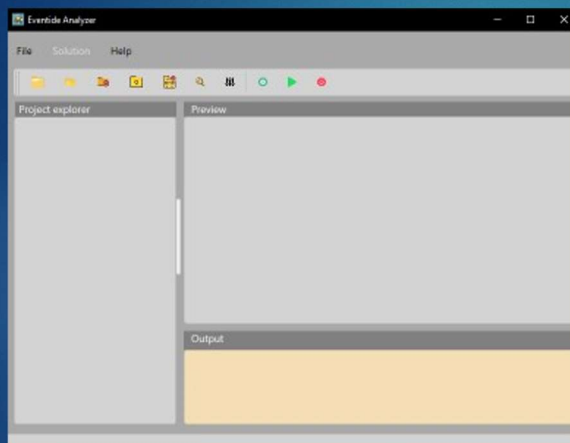


Программные средства для выявления и моделирования составных событий



Название элемента	Описание
Analyst	Аналитик, взаимодействующий с системой
EventModeling	Подсистема, выполняющая моделирование вероятных шаблонов составных событий
SequentialModeling	Подсистема, выполняющая поиск вероятных последовательностей составных событий
User Interface	Пользовательский интерфейс программы
Log Reclaimer	Компонент обработки и фильтрации журнала событий
Model Visualizer	Компонент, отвечающий за визуализацию результата
Event Modeler	Компонент для выявления вероятных шаблонов составных событий
Database Manager	Компонент работы с базой данных
Project Manager	Компонент работы с проектом
Project	Проект
LogService	Модуль, отвечающий за загрузку журнала событий
LogDB	База данных журнала событий
Log Application	Компонент, отвечающий за хранение журнала событий в файле

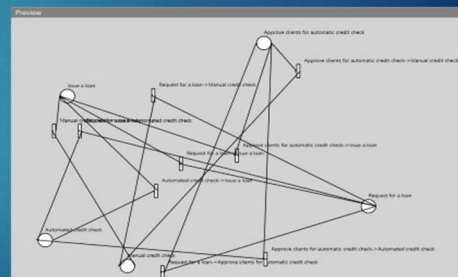
Программные средства для выявления и моделирования составных событий



Главное окно программы

Preview
Transaction
Request for a loan, Approve clients for automatic credit check, Issue a loan, Automated credit check
Request for a loan, Approve clients for automatic credit check, Manual credit check, Issue a loan
Request for a loan, Reject request for a loan
Request for a loan, Approve clients for automatic credit check, Manual credit check, Issue a loan
Request for a loan, Issue a loan
Request for a loan, Approve clients for automatic credit check, Manual credit check, Issue a loan

Пример сформированных цепочек событий



Пример модели составного события

Библиотека классов для построения шаблонов последовательных событий SeqMineLib



Класс	Описание
SequenceMiningAlgorithm	Абстрактный класс алгоритма секвенциального анализа. Содержит поля для входных данных и результатов анализа.
GSPAlgorithm	Класс, реализующий алгоритм GSP.
PrefixSpanAlgorithm	Класс, реализующий алгоритм PrefixSpan.
BIDEAlgorithm	Класс, реализующий алгоритм BIDE.
Algorithms	Класс, используемый для управления другими классами алгоритмов. Включает в себя список всех доступных алгоритмов секвенциального анализа и конструкторы для них.
Sequence	Класс, используемый для представления последовательностей событий.

Библиотека классов для построения шаблонов составных событий ProcMineLib



Класс	Описание
ProcessMiningAlgorithm	Абстрактный класс алгоритма извлечения процесса. Содержит поля для входных данных и результатов моделирования составного события.
AlphaMiner	Класс, реализующий альфа-алгоритм.
HeuristicMiner	Класс, реализующий эвристический алгоритм.
InductiveMiner	Класс, реализующий индуктивный алгоритм.
PetriNet	Класс, используемый для представления сети Петри. Содержит список мест, переходов и дуг.
Algorithms	Класс, используемый для управления другими классами алгоритмов. Включает в себя список всех доступных алгоритмов извлечения процессов и конструкторы для них.
Transition	Используется для представления переходов в сети Петри.
Place	Используется для представления мест в сети Петри.
Arc	Используется для представления дуг в сети Петри. Содержит два поля для обозначения начала и конца дуги.

Библиотека классов для предварительной обработки журнала событий EventLogLib



Название класса	Описание
EventLogProcessor	Абстрактный класс обработки журнала событий. Содержит поле для временного хранения выбранных данных из источника, поле SourceData для хранения данных о самом источнике, а также метод для формирования входных последовательностей
CsvFileProcessor	Класс, используемый для работы с журналами событий в виде csv-файлов.
DatabaseProcessor	Класс, используемый для доступа к журналу событий, который хранится в базе данных.
SourceData	Класс, предназначенный для хранения данных об источнике журнала событий (тип журнала, параметры выборки и т.д.).
EventLogEntry	Класс, используемый для обобщенного представления строки журнала событий.

Экспериментальная оценка

Алгоритм	Уровень поддержки	Последовательные компоненты	Параллельные компоненты	Время
GSP	0.2	+	+	2500 мс
	0.4	+	-	1200 мс
PrefixSpan	0.2	+	+	1500 мс
	0.4	+	+	900 мс
BIDE	0.2	+	+	1900 мс
	0.4	+	+	1600 мс

Направления дальнейших исследований

Расширение выбора алгоритмов секвенциального анализа для поиска вероятных последовательностей событий, а также алгоритмов моделирования шаблонов составных событий.



Публикации по теме исследования

- ▶ Печаткин, К. В. Применение машинного обучения для обработки журнала событий / К. В. Печаткин, С. В. Шибанов // Методы, средства и технологии получения и обработки измерительной информации ("Шлядинские чтения - 2023"): Материалы XV Международной научно-технической конференции с элементами научной школы и конкурсом научно-исследовательских работ для студентов, аспирантов и молодых ученых, Пенза, 02–03 ноября 2023 года / Под редакцией Е.А. Печерской. – Пенза: Пензенский государственный университет, 2023. – С. 168-170.
- ▶ Печаткин, К. В. Обзор модельных представлений составных событий / К. В. Печаткин, С. В. Шибанов // Информационные технологии в науке и образовании. Проблемы и перспективы: сборник статей по материалам X Всероссийской научно-практической конференции, Пенза, 15 марта 2023 года. – Пенза: Пензенский государственный университет, 2023. – С. 195-198.

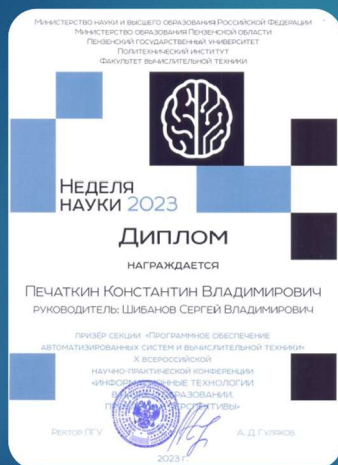
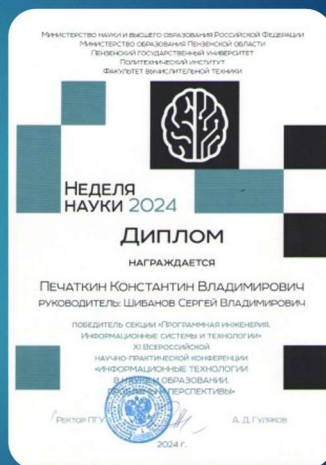
Публикации по теме исследования

- ▶ Печаткин, К. В. ProcessMining как инструмент выявления сложных событий / К. В. Печаткин, С. В. Шибанов // Прикладная математика и информатика: современные исследования в области естественных и технических наук: Сборник материалов IX Международной научно-практической конференции (школы-семинара) молодых ученых, Тольятти, 18–20 апреля 2023 года. – Тольятти: Тольяттинский государственный университет, 2023. – С. 314-319.
- ▶ Печаткин, К. В. Обзор алгоритмов фильтрации журналов событий / К. В. Печаткин, С. В. Шибанов // Наука, инновации и технологии: от идей к внедрению: Материалы Всероссийской научно-практической конференции молодых ученых, Комсомольск-на-Амуре, 16–17 ноября 2023 года. – Комсомольск-на-Амуре: Комсомольский-на-Амуре государственный университет, 2023. – С. 526-529.

Публикации по теме исследования

- ▶ Печаткин, К. В. Обзор программных реализаций алгоритмов секвенциального анализа / К. В. Печаткин, С. В. Шибанов // Информационные технологии в науке и образовании. Проблемы и перспективы: сборник статей по материалам XI Всероссийской научно-практической конференции, Пенза, 13 марта 2024 года. – Пенза: Пензенский государственный университет, 2024.

Апробация работы



Апробация работы



Апробация работы



Спасибо за
внимание!