**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Научный руководитель,  Старший преподаватель департамента  программной инженерии факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Шершаков  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия», канд. техн. наук, профессор департамента программной инженерии факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **ПРОГРАММА ПОСТРОЕНИЯ МОДЕЛИ ПРОЦЕССА АЛГОРИТМОМ**  **FUZZY MINER**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.01-01 81 01-1-ЛУ** | |
|  | Исполнитель  студент группы БПИ182  \_\_\_\_\_\_\_\_\_\_\_\_\_ / И. С. Егоров /  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |
|  | |

**Москва 2019**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | УТВЕРЖДЕН  RU.17701729.04.01-01 81 01-1-ЛУ |  |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **ПРОГРАММА ПОСТРОЕНИЯ МОДЕЛИ ПРОЦЕССА АЛГОРИТМОМ**  **FUZZY MINER**  **Пояснительная записка**  **RU.17701729.04.01-01 81 01-1**  **Листов 26** | | | |

**Москва 2019**

**АННОТАЦИЯ**

В данном программном документе приведена пояснительная записка к «Программе построения модели процесса алгоритмом Fuzzy miner» («Program for building process models by using Fuzzy miner algorithm»), предназначенной для построения модели процесса в виде ориентированного графа на базе журналов событий алгоритмом “Fuzzy miner”.

В данном программном документе, в разделе «Введение», указано наименование

программы и документы, на основании которых ведется разработка.

В разделе «Назначение и область применения» указано функциональное и

эксплуатационное назначение программы и краткая характеристика области применения

программы.

В разделе «Технические характеристики», содержатся следующие подразделы:

* постановка задачи на разработку программы;
* описание алгоритма и функционирования программы;
* описание и обоснование выбора метода организации входных и выходных данных;
* описание и обоснование выбора состава технических и программных средств.

В разделе «Технико-экономические показатели» указана предполагаемая

потребность и экономические преимущества разработки по сравнению с отечественными и

зарубежными образцами или аналогами.

Перед прочтением данного документа рекомендуется ознакомиться с терминологией, приведенной в списке используемой литературы настоящей Пояснительной записке.

Настоящий документ разработан в соответствии со следующими требованиями:

1. ГОСТ 19.101-77 Виды программ и программных документов [1];
2. ГОСТ 19.102-77 Стадии разработки [2];
3. ГОСТ 19.103-77 Обозначения программ и программных документов [3];
4. ГОСТ 19.104-78 Основные надписи [4];
5. ГОСТ 19.105-78 Общие требования к программным документам [5];
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [7].

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc21408436)

[1. ВВЕДЕНИЕ 4](#_Toc21408437)

[**1.1.** **Наименование программы** 4](#_Toc21408438)

[**1.2.** **Основания для разработки** 4](#_Toc21408439)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 5](#_Toc21408440)

[**2.1.** **Функциональное назначение** 5](#_Toc21408441)

[**2.2.** **Эксплуатационное назначение** 5](#_Toc21408442)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 6](#_Toc21408443)

[**3.1.** **Постановка задачи** 6](#_Toc21408444)

[**3.2.** **Описание алгоритма и функционирования программы** 6](#_Toc21408445)

[**3.3.** **Описание и обоснование выбора метода организации входных и выходных данных** 8](#_Toc21408446)

[**3.3.1. Описание метода организации входных и выходных данных** 8](#_Toc21408447)

[**3.3.2. Обоснование выбора метода организации входных и выходных данных** 9](#_Toc21408448)

[**3.4.** **Описание и обоснование выбора состава технических и программных средств** 9](#_Toc21408449)

[**3.4.1. Описание состава технических и программных средств** 9](#_Toc21408450)

[**3.4.2. Обоснование выбора состава технических и программных средств** 9](#_Toc21408451)

[4. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 10](#_Toc21408452)

[**4.1.** **Предполагаемая потребность** 10](#_Toc21408453)

[**4.2.** **Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами** 10](#_Toc21408454)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 11](#_Toc21408455)

[ПРИЛОЖЕНИЕ 1 12](#_Toc21408456)

[ПРИЛОЖЕНИЕ 2 13](#_Toc21408457)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 26](#_Toc21408458)

1. ВВЕДЕНИЕ
   1. **Наименование программы**

Наименование программы – «Программа построения модели процесса алгоритмом Fuzzy miner».

Наименование программы на английском языке – «Program for building process models by using Fuzzy miner algorithm».

Краткое наименование программы – Fuzzy miner.

* 1. **Основания для разработки**

Разработка выполняется в рамках темы курсовой работы в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия» Национального исследовательского университета «Высшая школа экономики», факультет компьютерных наук, департамент программной инженерии.

Разработка ведется на основании приказа декана ФКН И. В. Аржанцева № 2.3-02/1012-02 от 10.12.2018 г. «Об утверждении тем, руководителей курсовых работ студентов образовательной программы «Программная инженерия» факультета компьютерных наук».

1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ
   1. **Функциональное назначение**

Функциональным назначением программы является построение модели процесса в виде ориентированного графа на базе журналов событий c использованием алгоритма Fuzzy miner.

* 1. **Эксплуатационное назначение**

Программа может быть использована аналитиками в области информационных технологий, знакомых с областью Process mining, заинтересованных в изучении или усовершенствовании процессов на базе их моделей, построенных по журналам событий. Программа является Desktop-приложением и не требует подключения к Интернету для использования.

.

1. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ
   1. **Постановка задачи**

Программа должна поддерживать следующие функции:

1. Чтение журнала событий из файла в формате .log (см. пункт 3.3.1);
2. Считывание весов важности метрик (см. пункт 3.2);
3. Обработка журнала событий;
4. Построение Fuzzy-модели;
5. Визуализация Fuzzy-модели при помощи пакета GraphViz (см. Приложение 3 Технического задания);
6. Возможность в процессе работы подстраивать параметры фильтрации рёбер и вершин, а также параметр решения конфликтов в визуализации Fuzzy-модели;
7. Возможность сохранить в формате .gif полученную визуализацию.
   1. **Описание алгоритма и функционирования программы**

Fuzzy miner является одним из основных алгоритмов в области Process mining.Fuzzy miner (авторы Wil van der Aalst и Christian W. Gunther) - инструмент для извлечения нечёткой модели процесса из журнала событий. Такой тип модели не показывает некоторые стороны процесса, однако может перестраиваться online в зависимости от параметров, введённых пользователем. Данный подход позволяет сконцентрировать внимание на изучении поведения модели, подходящего под запросы пользователя. Алгоритм рассчитывает специальные метрики, далее при помощи введённых пользователем весов важности метрик (в диапазоне от 0 до 1) считает важность всех вершин и рёбер в графе, а затем даёт ему возможность ввести значения, по которым нужно убрать лишнее из модели процесса (под лишним предполагаются рёбра и вершины, важность которых меньше заданной пользователем), тем самым подстраивая модель именно под запросы пользователя. Рассмотрим каждую метрику отдельно:

1. Binary frequency – обозначает частоту вызовов ребер, связывающих вершины (ей соответствует binFrequency – важность метрики с точки зрения пользователя);
2. Unary frequency - обозначает частоту обращения к той или иной вершине (ей соответствует unFrequency – важность метрики с точки зрения пользователя);
3. Routing significance - обозначает отношение входящих в вершину связей к исходящим (ей соответствует routingSig – важность метрики с точки зрения пользователя);
4. Distance significance – обозначает разность значимостей двух вершин, соединённых ребром (ей соответствует distanceSig – важность метрики с точки зрения пользователя);
5. Name significance - обозначает схожесть имён двух вершин, соединённых ребром (ей соответствует nameSig – важность метрики с точки зрения пользователя).

Также пользователю предоставляется возможность ввести значения ещё трёх параметров в диапазоне от 0 до 1 включительно:

1. Edge cutoff – параметр, задающий планку для удаления рёбер (все рёбра, важность которых по итогу меньше этого значения, должны быть удалены из модели, но гарантируется, что при удалении ребра количество компонент связностей в графе не возрастёт, то есть, если важность ребра меньше параметра фильтрации, но его удаление приведёт к увеличению компонент связности графа, то оно удалено не будет);
2. Node cutoff – параметр, задающий планку для удаления вершин (все вершины, важность которых меньше данного значения будут удалены из модели);
3. Ratio cutoff – параметр, задающий значение максимальной разности между значениями значимости двух рёбер между вершинами, находящимися в конфликте (образующих цикл из двух вершин). Если разность между значениями не превышает данного значения, значит этот конфликт важен пользователю и разрешать его не нужно. В ином случае нужно удалить то ребро, чья значимость меньше.

Перейдём непосредственно к подходам реализации.

Весь алгоритм можно разбить на пять последовательных шагов:

1. Считывание журнала событий и перевод его в ориентированный граф;
2. Подсчёт всех метрик;
3. Получение на базе введённых пользователем важностей метрик значимости вершин и рёбер графа;
4. Разрешение конфликтов между вершинами;
5. Фильтрация рёбер;
6. Фильтрация вершин.

Рассмотрим каждый шаг отдельно.

На первом шаге мы считываем все трассы из журнала событий. Каждое событие становится вершиной, а два события, идущих друг за другом, будут считаться связанными ребром. Таким образом, мы переходим от журнала событий к ориентированному графу, который хранится при помощи списка смежности (Из трассы A;B;C; получаем граф A->B->C, где -> - ориентированное ребро).

Подсчёт всех метрик – на этом шаге алгоритм подсчитывает каждую метрику для каждой вершины или ребра. Условно все метрики можно разделить на два типа: унарные – метрики, рассматривающие одну вершину, и бинарные – метрики, рассматривающие пару вершин. К унарным относятся Unary Frequency и Routing Significance, к бинарным – Binary Frequency, Distance significance, Name Significance. Рассмотрим их подсчёт:

1. Unary frequency считается для каждой вершины как частота вызова данной вершины в журнале событий, нормализованная относительно максимума данной метрики по всем вершинам;
2. Binary frequency считается для каждого ребра как частота вызова данного ребра в журнале событий, нормализованная относительно максимума данной метрики по всем рёбрам в графе;
3. Name significance считается с использованием подхода “динамического программирования”. При помощи динамики получается количество действий, необходимых для преобразования одного имени в другое. Далее нормализуем это значение относительно максимально возможной разницы и вычитаем из единицы.
4. Routing significance считается по следующей формуле: для вершины А, где b\_f(a, b) – значение Binary frequency для ребра a->b, n\_s(a, b) - значение Name Significanse для ребра(a, b), N – множество вершин, в которые есть ребро из вершины А, M – множество вершин, в которых есть ребро, идущее в А. Далее значение Routing Significance для каждой вершины нормализуется относительно максимального значения Routing significance в графе.
5. Distance significance считается следующим образом: пусть , где b\_f(a, b) – значение Binary frequency для ребра a->b, n\_s(a, b) - значение Name Significanse для ребра(a, b), , где u\_f(x) – значение Unary frequency для вершины x, r\_s(x) - значение Routing significanse для вершины x. Далее все значения agg\_un и agg\_bin для вершин и рёбер нормализуются относительно максимумов этих значений в графе. Тогда Distance significance для ребра a, b = .Далее значение Distance significance для каждой вершины нормализуется относительно максимального значения Distance significance в графе.

На следующем шаге насчитывается итоговая значимость для каждой вершины и каждого ребра следующим образом: итоговая важность вершины sig(A) = routingSig \* Routing Significance + + unFrequency \* Unary Frequency. Далее итоговые важности для всех вершин нормализуются относительно максимальной итоговой важности вершины в графе; итоговая важность ребра

sig(A, B) = binFrequency \* Binary Frequency + nameSig \* Name Significance + distanceSig \* Distance significance. Далее итоговые важности для всех рёбер нормализуются относительно максимальной итоговой важности ребра в графе.

После подсчёта этих величин можно переходить к следующему шагу: разрешению конфликтов между вершинами. Для этого введём силы связи между двумя вершинами, соединённых ребром (rel(A, B) - сила связи ребра, выходящего из вершины А и входящего в вершину B). Эта величина считается по формуле: . Теперь пусть у нас есть две конфликтующие вершины: A и B. Тогда, если модуль разности значений rel(A,B) и rel(B,A) превосходит значение Ratio cutoff, введённое пользователем, то ребро с меньшим значением rel удаляется из графа. Таким образом, пройдясь по всем конфликтам в графе мы сможем разрешить каждый из них.

Фильтрация рёбер реализована при помощи алгоритма поиска в глубину на графе (depth-first search) и работает следующим образом. Запустим первоначальный обход графа и посчитаем количество вызовов dfs для полного обхода графа. Далее рассмотрим по очереди все рёбра в нашем графе. Если на очередном шаге рассмотрения ребра его важность меньше введённого пользователем параметра Edge cutoff, будем пытаться удалить его. Затем с помощью алгоритма поиска в глубину посчитаем количество вызовов dfs, для полного обхода графа. Если оно отличается от первоначального, значит удаление данного ребра приведёт к нарушению связности графа, а, значит, удалять ребро нельзя. Иначе, ребро можно удалить.

И, наконец, финальная стадия алгоритма – фильтрация вершин графа. Рассмотрим каждую вершину в графе. Если её важность меньше параметра Node cutoff, введённого пользователем, значит такую вершину нужно удалить. Это можно сделать следующим образом: все рёбра, исходящие из данной вершины, нужно будет пустить из вершин, из которых выходит ребро в удаляемую и запустить в ту, в которую выходило из удаляемой вершины.

После завершения всех шагов полученный граф будет называться Fuzzy-моделью, переведён из формата списка смежности в формат .dot и, при помощи пакета GraphViz, преобразован в визуализированную модель и выведен пользователю на экран.

* 1. **Описание и обоснование выбора метода организации входных и выходных данных**
     1. **Описание метода организации входных и выходных данных**

Входные данные представлены в виде журналов событий в формате .log. Данный формат должен иметь следующий вид: в строках последовательно должны идти трассы, которые представляют из себя упорядоченный набор имён вызываемых событий, состоящих из строчных и заглавных букв латинского алфавита или цифр. После каждого имени в трассе стоит символ “точка с запятой” (см. пример в Приложении 5 Технического задания).

Также на вход подаются значения важности метрик binFrequency, unFrequency, routingSig, distanceSig, nameSig (подробнее про каждую из них изложено в пункте 3.2) в диапазоне от 0 до 1 включительно и величины параметров фильтрации рёбер и вершин, решений конфликтов (подробнее про них написано в пункте 3.2) в диапазоне от 0 до 1 включительно.

Выходные данные представляют из себя визуализированную Fuzzy-модель процесса. По запросу пользователя получившуюся модель можно сохранить как изображение в формате .gif.

* + 1. **Обоснование выбора метода организации входных и выходных данных**

Модель в виде изображения является самым наглядным и понятным способом отображения Fuzzy-модели.

* 1. **Описание и обоснование выбора состава технических и программных средств**
     1. **Описание состава технических и программных средств**

Для надежной работы программы необходим следующий состав программных средств:

1. Операционная система Microsoft Windows 7 или более поздняя версия;
2. Установленный Microsoft .NET Framework 4.6 или более поздняя версия;
3. Установленный пакет GraphViz (см. Приложение 3 Технического задания).

Для надёжной и бесперебойной работы программы требуется следующий состав технических средств [12]:

1. персональный компьютер, оснащенный 32-разрядным (x86) или 64-разрядным (x64) процессором с тактовой частотой 1 ГГц или выше;
2. 200МБ оперативной памяти или больше;
3. не менее 250 МБ свободного места на жестком диске;
4. видеокарта и монитор, поддерживающие режим SuperVGA с разрешением не менее, чем 1024x768 точек;
5. мышь или совместимое указывающее устройство;
6. клавиатура;
7. CD-ROM привод, разъем USB или подключение к сети Интернет.
   * 1. **Обоснование выбора состава технических и программных средств**

Разработка данной программы велась под управлением OC Windows 7, и при ее создании использовались языки C# 7, C++ 14 и Microsoft .NET Framework 4.6.

GraphViz выбран в качестве пакета для построения визуализации графа в силу его больших возможностей, бесплатного распространения и удобства в использовании.

200 Мб оперативной памяти требуется для создания и хранения моделей, в которых может быть до 1000 событий.

250 Мб свободного места на диске требуется для хранения пакета GraphViz, исполняемого файла приложения и динамической библиотеки с реализацией алгоритма “Fuzzy miner”.

1. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ
   1. **Предполагаемая потребность**

Данный продукт может быть востребован в качестве инструмента для построения модели процесса и её анализа в экономической и информационной областях. Аналогично, он может быть полезен для сообщества исследователей области Process mining.

* 1. **Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами**

На момент начала разработки отечественных аналогов обнаружено не было.

Зарубежными аналогами являются ProM, Disco и Celonis.

По сравнению с аналогами данное приложение:

* 1. Распространяется бесплатно;
  2. Крайне прост в использовании.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 15150-69 Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды. – М.: Изд-во стандартов, 1997.
11. Устинов В. Надежность оптических дисков: как их правильно хранить и использовать. //Журнал «625» №7. М.: Издательство «625», 2005.
12. Системные требования ОС Windows 7. [Электронный ресурс]// URL: <http://windows.microsoft.com/systemrequirements?4bcfd458> (Дата обращения: 21.11.2018, режим доступа: свободный).
13. ГОСТ Р 7.02-2006 Консервация документов на компакт-дисках. Общие требования. – М.: ИПК Издательство стандартов, 2006.
14. ГОСТ 19.602-78 Правила дублирования, учета и хранения программных документов, выполненных печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
15. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
16. Christian W. Gunther and Wil M.P. van der Aalst: Fuzzy Mining – Adaptive Process Simplification Based on Multi-Perspective Metrics // Eindhoven University of Technology — URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.1207&rep=rep1&type=pdf>

ПРИЛОЖЕНИЕ 1

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ**

Таблица 1 — Описание и функциональное назначение классов

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| Fuzzyminer | Главное окно интерфейса. |
| binary\_metric | Класс, реализующий метрики binFrequency, distanceSig, nameSig. |
| dotfile | Класс, осуществляющий перевод графа в dot-файл. |
| graph | Класс, реализующий граф. |
| name\_similarity | Класс, реализующий подсчёт разности между именами двух событий. |
| node | Класс, реализующий вершину графа. |
| unary\_metric | Класс, реализующий метрики unFrequency, routingSig. |

ПРИЛОЖЕНИЕ 2

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ, МЕТОДОВ И СВОЙСТВ**

Таблица 2.1 — Описание и функциональное назначение полей, методов и свойств класса FuzzyMiner

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | | Назначение |
| allowedCharacters | private | const string | | Содержит список разрешённых символов в имени событий. |
| imageName | private | const string | | Содержит имя картинки. |
| filename\_to\_cpp | private | string | | Содержит имя файла, передаваемое в библиотеку. |
| drawFlagEdges | private | bool | | Содержит сигнал о том – нужно ли начинать перерисовку модели при изменении параметра фильтрации рёбер. |
| drawFlagNodes | private | bool | | Содержит сигнал о том – нужно ли начинать перерисовку модели при изменении параметра фильтрации вершин. |
| drawFlagRatio | private | bool | | Содержит сигнал о том – нужно ли начинать перерисовку модели при изменении параметра решения конфликтов. |
| linesRead | private | string[] | | Содержит считанные из журнала событий трассы. |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| FuzzyMiner | public | - | - | Конструктор окна. |
| Calculate | unsafe private | void | object, EventArgs | Строит модель и выводит её на экран. |
| OpenFile | private | void | object, EventArgs | Открывает на чтение журнал событий. |
| SaveFile | private | void | object, EventArgs | Сохраняет полученную модель. |
| ParseFile | private | bool | string | Считывает журнал событий и разбивает его на трассы. |
| ParseLine | private | bool | string | Разбивает трассы на имена событий. |
| ParseWord | private | bool | string | Проверяет на корректность имена событий |
| OpenImage | private | void | - | Открывает изображение. |
| autoParam1\_MouseDown | private | void | object, EventArgs | Показывает, что пользователь нажал на ползунок параметра фильтрации рёбер. |
| autoParam1\_MouseUp | private | void | object, EventArgs | Показывает, что пользователь отпустил клавишу с ползунка параметра фильтрации рёбер. |
| autoParam1\_Scroll | private | void | object, EventArgs | Показывает, что пользователь работает с ползунком параметра фильтрации рёбер. |
| autoParam2\_Scroll | private | void | object, EventArgs | Показывает, что пользователь работает с ползунком параметра фильтрации вершин. |
| autoParam2\_MouseDown | private | void | object, EventArgs | Показывает, что пользователь нажал на ползунок параметра фильтрации вершин. |
| autoParam2\_MouseUp | private | void | object, EventArgs | Показывает, что пользователь отпустил клавишу с ползунка параметра фильтрации вершин. |
| FuzzyMiner\_Resize | private | void | object, EventArgs | Перерисовывает элементы формы в момент изменения её размера. |
| AutoParam3\_MouseUp | private | void | object, EventArgs | Показывает, что пользователь отпустил клавишу с ползунка параметра решения конфликтов. |
| AutoParam3\_MouseDown | private | void | object, EventArgs | Показывает, что пользователь нажал на ползунок параметра решения конфликтов. |
| AutoParam3\_Scroll | private | void | object, EventArgs | Показывает, что пользователь работает с ползунком параметра решения конфликтов. |
| FuzzyMiner\_FormClosing | private | void | object, EventArgs | Событие закрытия формы. |

Таблица 2.2 — Описание и функциональное назначение полей, методов и свойств класса

binary\_metric

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | | Назначение |
| binary\_frequency | private | double | | Cодержит значение метрики binFrequency. |
| binary\_significance | private | double | | Cодержит итоговую важность связи между двумя вершинами. |
| name\_similar | private | double | | Cодержит значение метрики nameSig. |
| binary\_distance | private | double | | Cодержит значение метрики distanceSig. |
| aggregate | private | double | | Вспомогательная метрика.Служит для подсчёта distanceSig. |
| user\_frequency | private | double | | Значимость метрики binFrequency, введённая пользователем. |
| user\_distance | private | double | | Значимость метрики distanceSig, введённая пользователем. |
| user\_nameSig | private | double | | Значимость метрики nameSig, введённая пользователем. |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| binary\_metric | public | - | - | Конструктор без параметров. |
| count\_binary\_frequency | public | void | int, int | Считает метрику binFrequency. |
| get\_binary\_frequency | public | double | - | Возвращает значение метрики binFrequency. |
| get\_binary\_significance | public | double | - | Возвращает значимость свзяи между двумя вершинами. |
| count\_binary\_significance | public | void | - | Считает значимость свзяи между двумя вершинами. |
| count\_name\_similarity | public | void | string, string | Считает метрику nameSig. |
| count\_binary\_distance | public | void | - | Считает метрику distanceSig. |
| get\_name\_similarity | public | double | - | Возвращает значение метрики nameSig. |
| get\_binary\_distance | public | double | - | Возвращает значение метрики distanceSig. |
| normalize\_binary\_distance | public | void | double | Нормализует метрику distSig. |
| count\_aggregate | public | void | - | Считает вспомогательную метрику aggregate. |
| normalize\_aggregate | public | void | double | Нормализует вспомогательную метрику aggregate. |
| get\_aggregate | public | double | - | Возвращает значение вспомогательной метрики aggregate. |
| normalize\_binary\_significance | public | void | double | Нормализует важность связи между двумя вершинами. |
| set\_user\_values | public | void | double, double, double | Устанавливает введённые пользователем значения важности метрик. |

Таблица 2.3 — Описание и функциональное назначение полей, методов и свойств класса dotfile

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | | Назначение |
| fout | private | ofstream | | Содержит поток вывода. |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| create\_dot | public | void | string, graph | Переводит граф в формат .dot и записывает его в файл. |

Таблица 2.4 — Описание и функциональное назначение полей, методов и свойств класса graph

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | | Назначение |
| all\_names | private | map<std::string, int> | | Содержит имена всех событий. |
| size | private | int | | Количество вершин в графе. |
| nodes | private | vector<node> | | Вершины графа. |
| parse\_line | private | vector<string> | | Содержит трассы. |
| node\_visited | private | vector<bool> | | Нужен для dfs. Показывает посещали ли мы вершину. |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| graph | public | - | vector<std::string> | Конструктор, получающий граф по журналу событий. |
| graph | public | - | - | Пустой конструктор класса. |
| get\_node | public | node | int | Возвращает вершину. |
| get\_size | public | int | - | Возвращает количество вершин в графе. |
| count\_binary\_frequency | public | void | - | Считает метрику binFrequency для всех связей между вершинами графа. |
| count\_unary\_frequency | public | void | - | Считает метрику unFrequency для всех вершин графа. |
| count\_unary\_routing | public | void | - | Считает метрику routingSig для всех вершин графа. |
| cut\_nodes | public | void | - | Запускает процесс фильтрации вершин. |
| count\_all\_metrics | public | void | - | Запускает подсчёт всех метрик. |
| count\_name\_similarity | public | void | - | Запускает подсчёт метрики nameSig. |
| count\_binary\_distance | public | void | - | Запускает подсчёт метрики nameSig. |
| count\_aggregate | public | void | - | Запускает подсчёт вспомогательной метрики aggregate для ребра в графе. |
| count\_unary\_aggregate | public | void | - | Запускает подсчёт вспомогательной метрики aggregate для вершины в графе. |
| count\_binary\_significance | public | void | - | Запускает подсчёт итоговой важности рёбер. |
| count\_unary\_significance | public | void | - | Запускает подсчёт итоговой важности вершин. |
| solve\_conflicts | public | void | double | Ищет конфликты между вершинами. |
| solve\_conflict | public | void | int, int, double | Решает конфликты между вершинами. |
| count\_rel | public | double | int, int, double | Считает значение rel для двух вершин (см. пункт 3.2) |
| cut\_edges | public | void | double | Удаляет вершины, не прошедшие фильтрацию, из графа. |
| check\_number\_of\_components | public | double | - | Считает количество компонент в графе. |
| dfs | public | void | int | Алгоритм dfs (поиск в глубину) |
| push\_user\_values | public | void | double, double, double, double, double | Запускает процесс записывания введённых пользователем весов важностей метрик. |

Таблица 2.5 — Описание и функциональное назначение полей, методов и свойств класса name\_similarity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| get\_string\_sim | public | double | string, string | Сравнивает имена двух строк. |

Таблица 2.6 — Описание и функциональное назначение полей, методов и свойств класса node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | | Назначение |
| relations | private | vector<pair<int, int>> | | Содержит все смежные вершины. |
| binary | private | vector<binary\_metric> | | Содержит класс binary\_metric для вершин, в которые есть ребро из рассматриваемой. |
| relations\_in | private | vector<int> | | Содержит вершины, из которых есть ребро в рассматриваемою. |
| visible\_relations | private | vector<bool> | | Показывает нужно ли отображать ребро в визуализации модели, исходящее из рассматриваемой вершины. |
| unary | private | unary\_metric | | Содержит класс unary\_metric для вершины. |
| name | private | string | | Содержит имя вершины. |
| visible | private | bool | | Показывает нужно ли отображать данную вершину в визуализации модели или нет. |
| frequency | private | int | | Частота вызова события в журнале событий. |
| amount\_of\_edges\_in | private | int | | Количество рёбер, входящих в данную вершину. |
| amount\_of\_edges\_out | private | int | | Количество рёбер, исходящих из данной вершины. |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы |  |
| node | public | - | - | Пустой конструктор класса. |
| node | public | - | int, name | Конструктор класса. |
| get\_relation | public | int | int | Возвращает номер вершины, связанной с рассматриваемой ребром. |
| get\_relation\_frequency | public | int | int | Возвращает частоту вызова ребра между двумя вершинами в журнале событий. |
| get\_name | public | string | - | Возвращает имя вершины. |
| get\_node\_frequency | public | int | - | Возвращает частоту вызова вершины в журнале событий. |
| inc\_frequency | public | void | - | Увеличивает на единицу значение частоты вызова вершины в журнале событий. |
| inc\_in\_relations | public | void | - | Увеличивает на единицу количество рёбер, входящих в рассматриваемую вершину. |
| inc\_out\_relations | public | void | - | Увеличивает на единицу количество рёбер, выходящих из рассматриваемой вершины. |
| push\_relation | public | void | int | Добавляет ребро из рассматриваемой вершины в другую. |
| push\_relation\_in | public | void | int | Добавляет ребро из вершины в рассматриваемою. |
| get\_number\_of\_relations | public | int | - | Возвращает количество рёбер, исходящих из рассматриваемой в другие. |
| get\_unary\_significanse | public | double | - | Возвращает значение итоговой важности вершины. |
| count\_unary\_frequency | public | void | int | Запускает подсчёт метрики unFrequency. |
| get\_unary\_frequency | public | double | - | Возвращает значение метрики unFrequency. |
| count\_binary\_frequency | public | void | int | Запускает подсчёт метрики binFrequency. |
| get\_binary\_frequency | public | double | int | Возвращает значение метрики binFrequency. |
| get\_visible | public | bool | - | Возвращает значение, обозначающее будет ли рассматриваемая вершина входить в итоговую модель процесса. |
| set\_visible | public | void | bool | Устанавливает значение, обозначающее будет ли рассматриваемая вершина входить в итоговую модель процесса. |
| count\_name\_similarity | public | void | graph\* | Запускает подсчёт метрики nameSig. |
| count\_unary\_routing | public | void | graph\*, int | Запускает подсчёт метрики routingSig. |
| get\_unary\_routing | public | double | - | Возвращает значение метрики routingSig. |
| normalize\_unary\_routing | public | void | double | Нормализует значение метрики routingSig. |
| normalize\_binary\_distance | public | void | double | Нормализует значение метрики distanceSig. |
| count\_binary\_distance | public | void | graph\*, int | Запускает подсчёт метрики distanceSig. |
| count\_aggregate | public | void | - | Запускает подсчёт вспомогательной метрики aggregate. |
| get\_max\_aggregate | public | double | - | Возвращает максимум из всех метрик aggregate в графе. |
| normalize\_aggregate | public | void | double | Нормализует значение вспомогательной метрики aggregate. |
| count\_unary\_aggregate | public | void | - | Запускает подсчёт вспомогательной метрики unary\_aggregate. |
| get\_unary\_aggregate | public | double | - | Возвращает значение вспомогательной метрики unary\_aggregate. |
| normalize\_unary\_aggregate | public | void | double | Нормализует значение вспомогательной метрики aggregate. |
| get\_max\_distance | public | double | - | Возвращает максимум из всех метрик distanceSig в графе. |
| normalize\_distance | public | void | double | Нормализует метрику distanceSig. |
| count\_binary\_significance | public | void | - | Запускает подсчёт значения итоговой важности рёбер в графе. |
| get\_max\_binary\_significance | public | double | - | Возвращает максимум из всех значений итоговой важности рёбер в графе. |
| normalize\_binary\_significance | public | void | double | Нормализует значения итоговой важности рёбер в графе. |
| count\_unary\_significance | public | void | - | Запускает подсчёт итоговых значений важностей вершин графа. |
| normalize\_unary\_significance | public | void | double | Нормализует итоговые значения важностей вершин графа. |
| get\_number\_of\_in\_relations | public | int | - | Возвращает количество рёбер, входящих в рассматриваемую вершину. |
| get\_in\_relation | public | int | int | Возвращает номер вершины, ребро из которой входит в рассматриваемую. |
| get\_binary\_significance | public | double | int | Возвращает итоговое значение важности вершины. |
| set\_rel\_visible | public | void | int, bool | Устанавливает значение отображения ребра, исходящего из рассматриваемой вершины, в итоговую модель процесса |
| get\_relation\_visible | public | bool | int | Возвращает значение, обозначающее будет ли ребро входить в итоговую модель процесса. |
| push\_user\_values | public | void | double, double, double, double, double | Запускает процесс установки значений весов важностей метрик, введённых пользователем. |

Таблица 2.7 — Описание и функциональное назначение полей, методов и свойств класса unary\_metric

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | | Назначение |
| normalized\_frequncy | private | double | | Значение метрики unFrequency. |
| normalized\_routing | private | double | | Значение метрики routingSig. |
| aggregate | private | double | | Значение вспомогательной метрики aggregate. |
| unary\_significance | private | double | | Итоговое значение важности вершины. |
| user\_frequency | private | double | | Значение важности метрики unFrequency, введённое пользователем. |
| user\_routing | private | double | | Значение важности метрики routingSig, введённое пользователем |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| unary\_metric | public | - | - | Конструктор класса. |
| count\_frequency | public | void | int, int | Выполняет подсчёт метрики unFrequency. |
| get\_normalized\_frequency | public | double | - | Нормализует метрику unFrequency. |
| get\_normalized\_routing | public | double | - | Возвращает значение метрики routingSig. |
| count\_unary\_routing | public | void | double | Выполняет подсчёт метрики routingSig. |
| normalize\_unary\_routing | public | void | double | Нормализует метрику routingSig. |
| count\_aggregate | public | void | - | Выполняет подсчёт вспомогательное метрики aggregate. |
| get\_aggregate | public | double | - | Возвращает значение метрики aggregate. |
| normalize\_aggregate | public | void | double | Нормализует вспомогательную метрику aggregate. |
| count\_unary\_significance | public | void | - | Выполняет подсчёт итогового значения важности вершины. |
| get\_unary\_significance | public | double | - | Возвращает итоговой важности вершины. |
| normalize\_unary\_significance | public | void | double | Нормализует итоговую важность вершины. |
| set\_user\_values | public | void | double, double | Выставляет значения весов важности метрик, введённых пользователем |

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |