**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**
**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Департамент программной инженерии

| Подп. и дата | |
|---|---|
| Инв. № дубл. | |
| Взам. инв. № | |
| Подп. и дата | |
| Инв. № подл | |

**СОГЛАСОВАНО**

Доцент департамента программной инженерии факультета компьютерных наук

_____ Л. В. Дворянский
«_____» _____ 2020 г.

**УТВЕРЖДАЮ**

Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук

_____ В. В. Шилов
«_____» _____ 2020 г.

# АНАЛИЗ ПОВЕДЕНИЯ ВРЕМЕННЫХ СИСТЕМ С ПОМОЩЬЮ ДИНАМИЧЕСКИХ МЕТРИЧЕСКИХ ГРАФОВ.

**Пояснительная записка**

**Лист Утверждения**

**RU.17701729.04.01-01 ПЗ 81 01-1-ЛУ**

Исполнитель: Студент группы БПИ172
_____ А. А. Измайлов
«_____» _____ 2020 г.

Москва 2020

**Утверждён**
**RU.17701729.04.01-01 ПЗ 81 01-1-ЛУ**

# АНАЛИЗ ПОВЕДЕНИЯ ВРЕМЕННЫХ СИСТЕМ С ПОМОЩЬЮ ДИНАМИЧЕСКИХ МЕТРИЧЕСКИХ ГРАФОВ.

| Подп. и дата | |
|---|---|
| Инв. № дубл. | |
| Взам. инв. № | |
| Подп. и дата | |
| Инв. № подл | |

**Пояснительная записка**

**RU.17701729.04.01-01 ПЗ 81 01-1**

**Листов 33**

Москва 2020

# Содержание

# 1. Введение

## 1.1. Наименование программы

### 1.1.1. Наименование на русском языке

Анализ поведения временных систем с помощью динамических метрических графов.

### 1.1.2. Наименование на английском языке

Behaviour analysis of real time systems via dynamic metric graphs.

## 1.2. Основание для разработки

Приказ декана факультета компьютерных наук И.В. Аржанцева "Об утверждении тем, руководителей курсовых работ студентов образовательной программы «Программная инженерия» факультета компьютерных наук"№ 2.3-02/1112-04 от 11.12.2019.

# 2. Назначение разработки

## 2.1. Назначение программы

### 2.1.1. Функциональное назначение

Программа представляет из себя транслятор, который принимает на вход временную сеть Петри[1], а на выход выдает метрический граф[2] или сообщение о том, что конвертация не возможна, а также наоборот. Программа не всегда может перевести сеть Петри в метрический граф, так как у этих моделей разная выразительность.

### 2.1.2. Эксплуатационное назначение

Данная программа может быть использована при исследовании свойств сетей Петри применительно к метрическим графам или наоборот. Такое может быть полезно, так как по отдельности эти две модели хорошо изучены, но методы исследования одной модели никто не применял к исследованию второй модели.

## 2.2. Краткая характеристика области применения

На данный момент временные сети Петри и метрические графы по отдельности хорошо изучены, но на самом деле между ними есть связь[3], что дает возможность применять методы анализа одной модели к другой, и наоборот. Так как данная сфера только начинает развиваться, на рынке нет автоматизированных решений для конвертации данных моделей друг к другу, зато уже давно есть программы для анализа временных сетей Петри, например UPPAAL[4]/TAPAAL[5], а также много результатов по изучению метрических графов.

Программа «Анализ поведения временных систем с помощью динамических метрических графов» позволяет проводить перевод сетей Петри в метрические графы и обртано, что должно помочь развивать данную сферу науки быстрее, так как не нужно будет в ручную переводить модели друг в друга для последующего анализа с помощью уже существующих решений.

# 3. Технические характеристики

## 3.1. Постановка задачи на разработку программы

Программа должна:

- уметь считывать временную сеть Петри с внешнего хранилища;
- уметь считывать метрический граф с внешнего хранилища;
- конвертировать временную сеть Петри в метрический граф;
- предоставлять отчет о том, почему нельзя перевести сеть Петри в метрический граф;
- конвертировать метрический граф во временную сеть Петри;

## 3.2. Описание алгоритма и функционирования программы

### 3.2.1. Внутреннее представляени метрического графа

Метрический граф в программе представлен, как взвешенный ориентированный граф. Для обозначение бесконечного ребра одна из вершин помечается как бесконечность (со стороны графа это просто вершина), а длина ребра равняется бесконечность ( *Double.POSITIVE_INFINITY* в языке программирования Java). Метрические графы по определению неориентированные, но для однозначности понимания того, что происходит при столкновении двух точек на ребре, ребро представляется как пара из двух ориентированных ребер одной длинны. Каждое ребро хранит список точек на нем.

### 3.2.2. Конвертация метрического графа в временную Сеть Петри[3]

1) Для каждого ребра $e$ соединяющего вершины $a$ и $b$ создается место $e_{ab}$ и место $e_{ba}$.
2) Для каждого ребра, у которого нет одного конца, т.е. пары ориентированных ребер из бесконечности (в нашей реализации у каждой бесконечности есть вершина, пусть это будет $a$) в $b$ и обратно в $a$.
   а) Создается место $e_{ba}$ и соединение $t_{ba}$ со срочностью 0, ребро с нулевым интервалом из $e_{ba}$ в $t_{ba}$. (Рис. 2)
   б) Для всех точек на ребре $ba$ создаются токены со временем 0 в месте $e_{ba}$
   в) Создается соединение $t_{ab}$ со срочностью 0.
   г) Для каждой точки $p_i$ на ребре $ab$
      i) Создается место $e^i_{ab}$ с одним токеном
      ii) Создается ребро из $e^i_{ab}$ в $t_{ab}$ с временным интервалом $[x(p_i), x(p_i)]$
   д) для каждого ребра $bc$ создается ребро из $t_{ab}$ в $e_{bc}$
3) Для каждого созданного места $e_{ab}$ где $a$ не бесконечность.
   а) Для каждой точки $p, x(p) = x_0$ на ориентированном ребре из $a$ в $b$ создается токен в месте $e_{ab}$ с значением таймера $x_0$.

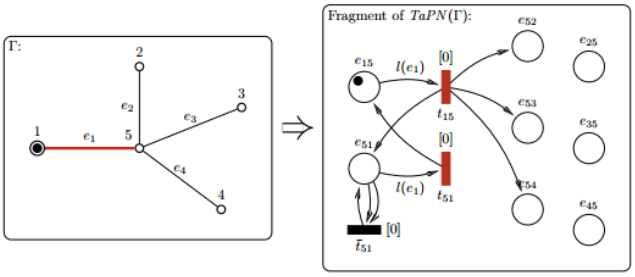


Рис. 1 — Конвертация ребра $e_1$

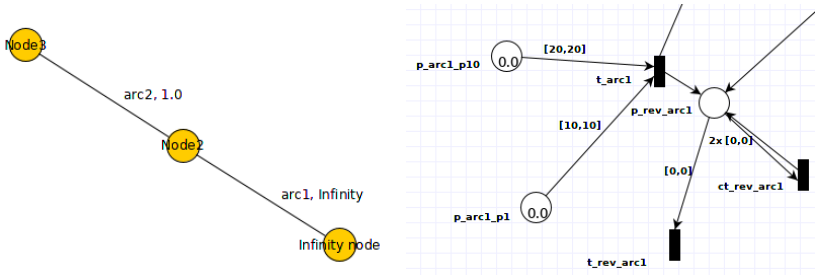


Рис. 2 — Конвертация ребра $arc1$, которое идет из бесконечности, и содержит 2 точки, $rev\_arc1$ - обратное ребро в бесконечность. Для каждой точки получается свое место

б) Далее для $e_{ab}$ рядом создается переход $t_{ab}$ со срочностью равной 0, если $b$ небесконечность

в) Далее для $e_{ab}$ соединяестя ребром с временным интервалом равным $[l(e), l(e)]$ с $t_{ab}$.

г) Для каждого ориентированного ребра выходящего из $b$ в вершину $c$ создается ребро из $t_{ab}$ в $e_{bc}$.

д) создается еще одно соединение $\bar{t}_{ab}$. Также создается ребро двойной кратности с нулевым временным интервалом из $e_{ab}$ в $\bar{t}_{ab}$ и обратно одинарной кратности.

4) Для получившейся сети запускается силовой алгоритм размещения Fruchterman and Reingold[6].

По ходу алгоритма, новым объектам присваиваются идентификаторы:

- p_*id ребра* - имя места, полученного из соответствующего ребра
- t_*id ребра* - имя соединения, полученного из соответсвующего ребра
- ct_*id ребра* - имя соединения, для схлопования токенов, полученного из соответствующего ребра

Итоговая сложность $O(|E| \cdot max(cnt(e \in E)))$ для построения сети Петри без размещения, и $O(|V|^2 + |E|)$ для размещения, где $E$ - ребра, $V$ - вершины, $cnt$ - количество точек на ребре.

### 3.2.3. Конвертация из временных сетей Петри в метрические графы[3]

Из-за меньшей выразительности метрических графов, не все временные сети Петри могут быть переведены в метрические графы. Временная сеть петри может быть переведена, если:

- Все соединения имеют срочность 0

- Все временные интервалы имеют длинну 0, то есть содержат только одну точку.

Данные условия проверяются при первой встречи той или инной сущности в работе алгоритма.

1) Для каждого соединения $t_i$ создается вершина $v_i$ в графе.

2) Для каждого соединения $t_i$, для всех исходящих из него ребер рассматривается место $p$, для всех исходящих ребер из $p$ расматривается соединение $t_j$.

   а) Между соответсвующими вершинами $v_i$ и $v_j$ для $t_i$, $t_j$ проводится ребро с длиной равной временному интервалу, или очень маленькому числу, если интервал равен [0, 0].

   В данном шаге могут образоваться кратные ребра или ребра из вершины в неё же. Разрешается это так (Рис. 4):

   - Для кратных ребер создается новая вершина, превращающая одно из ребер в два. Длина полученых ребер равна длине начального пополам.
   - Для циклов с самим собой создается две дополнительные вершины, ребро делится на три ребра, длиной в три раза меньше

   б) Для всех токенов из $p$ в начало полученного ребра из $v_i$ в $v_j$, то есть в вершину $v_i$, ставится точка.

3) Для всех $p$, у которых есть только входный или только выходные ребра

   а) для каждого соседнего соединения $t$ из соответсвуюшей вершины $v$ создается бесконечное ребро.

   б) если у $p$ только исходящие ребра, то для каждого токена в $p$, для каждого исходящего ребра с концом в $t_i$, в бесконечное ребро из соответствующей вершины $v_i$ ставится точка в сторону к $v_i$, на расстоянии от $v_i$ равном временному интервалу на ребре из $p$ в $t_i$.

По ходу алгоритма, новым объектам присваиваются идентификаторы:

- a_*id соединения*_*id места*_*id соединения* - имя ребра, полученного из соотвествующих соединений и места
- rev_*id* - имя обратного ребра для ребра id
- l_*id начала*_*id конца* - имя ребра, соединеного с бесконечностью
- inf_*id начала*_*id конца* - имя для вершины бесконечности
- br_*id*_part* - имя ребра для разлома цикла с самим собой или кратного ребра
- br_n_*id* - имя вершины для разлома цикла с самим собой или кратного ребра

Итоговая сложность $O(|E| \cdot max(cnt(p \in P)))$ для построения метрического графа, где $E$ - множество ребер, $cnt$ - количество токенов в месте.

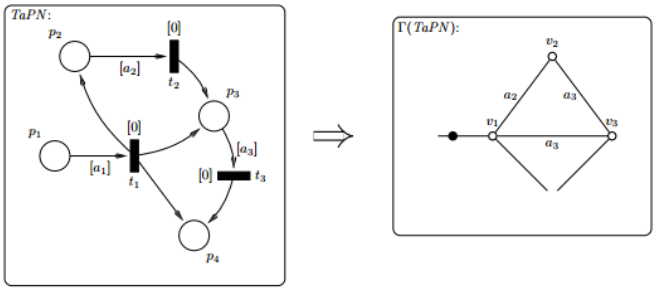


Рис. 3 — Конвертация сети Петри в Метрический граф

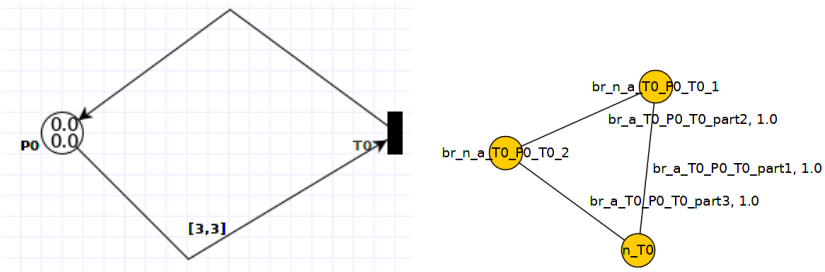


Рис. 4 — Обход ситуации с циклом с самим собой

### 3.2.4. Сериализация метрических графов в формат JSON[7]

Все ребра записываются в JSON, в виде объектов со всей дополнительной информацие, такой как **id** - идентификатор, **label** - метка, **comment** - комментарий. За ними записываются ребра, которые внутри себя содержат массив точек на этих ребрах и всю дополнительную информацию, как **id** - идентификатор, **label** - метка, **comment** - комментарий, **length** - длина.

Сложность данного алгоритма $O(|V| + |E| \cdot max(cnt(e \in E)))$, где $V$ - множество вершин, $E$ - множество ребер, $cnt$ - количество точек на ребре.

### 3.2.5. Десериализация метрических графов из формата JSON

- Входные данные валидируются по схеме[8] см. Приложение 4
- С помощью SAX[9] парсера данные достаются из файла, и складываются в промежуточные списки ребер и вершин
- В объект *builder*, который строит граф, кладутся сначала вершины, потом ребра.
- *builder* строит граф, или сообщает об ошибке.

Сложность данного алгоритма $O(N + |V| + |E| \cdot max(cnt(e \in E)))$, где $N$ - размер входного файла, $V$ - множество вершин, $E$ - множество ребер, $cnt$ - количество точек на ребре.

### 3.2.6. Сериализация метрических графов в формат yEd GraphML[10]

Данная сериализация почти не отличается от JSON, кроме формата XML, и того, что для более красивого представления, обратные ребра нужно спрятать. Для этого, по ходу записи, ребра складываются в множество $s$. И при записи текущего ребра, если обратное ему содержится в $s$, то текущему ребру присваивается свойство **visible** = *false*.

Так же для присвоения вершина координат, используется силовой алгоритм размещения Fruchterman and Reingold, перед записью.

Сложность записи $O(N+|V|+|E|\cdot max(cnt(e \in E)))$, сложность расположения $O(|V|^2+|E|)$, где $N$ - размер входного файла, $V$ - множество вершин, $E$ - множество ребер, $cnt$ - количество точек на ребре.

### 3.2.7. Построение метрического графа

Для построения графа используется шаблон проектирования Builder[11]. Для внутреннего хранения используются структуры из JGraphT[12].

Builder графа хранит внутри себя промежуточный граф, состоящий из вершин, и сырых ребер, которые тоже являются Builder, только для ребер.

При любом изменении, Builder графа проверяет консистентность структуры графа:

- каждый объект: вершина, ребро, точка, имеют уникальный идентификатор
- нет кратных ребер
- нет ребер из бесконечности в бесконечность
- всегда есть обратное ребро

В Builder графа можно добавить:

- вершину
- ребро, между двумя добавленными вершинам, сразу с обратным ребром
- точку в уже добавленное ребро, это возможно благодаря тому, что хранится не ребро, а сырое ребро

После построения Builder порождает неизменяемый объект графа, который соотвествует всем свойствам метрического графа.

### 3.2.8. Механизм конвертации

С помощью шаблона проектирования Chain of Responsibility[13], конвертации можно выстроить в цепочку (или композицию, если говорить в функциональном стиле).

То есть можно сделать цепочки:

- файл → метрический граф → временная сеть Петри → файл
- файл → временная сеть Петри → метрический граф → файл

так как объекты реализуют интерфейс функции *Function<TIn, TOut>*[14].

### 3.2.9. Расширение TAPAAL

Чтобы расширить функциональность TAPAAL, и не получить циклических зависимостей был выбран механизм точек расширения[15].

В разных местах внутри кода TAPAAL объявляются точки расширения, которые являются проводником для дополнительной логики из вне. TAPAAL объявляет имя и интерфейс для общения с расширениями, это и называется точкой расширения.

В другом месте, вне TAPAAL, можно объявить расширения, объявив тем самым новую логику, не изменяя код TAPAAL и не добавляя зависимостей в модуль TAPAAL.

В данной программе, реализовано 3 расширения, которые добавляют новый элементы для вызова конвертаций в меню.

### 3.2.10. Обоснование выбора алгоритмов решения задания

- Алгоритмы конвертации являются на данный момент единственными, поэтому они были выбраны.
- SAX десериализация была выбрана, так как она за один проход дает возможность считать данные, без хранения лишних данных.
- Шаблон Builder для построения графа, позволяет инкапсулировать логику валидации, и сделать создание графа атомарным действием, то есть либо графа не сущесивуеи, либо он есть сразу со всеми содержимым.

### 3.2.11. Возможные взаимодействия программы с другими программами

Данная программа может быть вызвана через CLI из другой прогрыммы, так же другие программы могут объявлять свои точки расширения для TAPAAL, добавляя свою логику. Созданные программой файлы .tapn можно открыть в TAPAAL, .graphml в yEd Graph Editor и в других программах, поддерживающих данный формат.

## 3.3. Описание и обоснование выбора метода организации входных и выходных данных

### 3.3.1. Описание метода организации входных и выходных данных

Входные/выходные данные для временных сетей Петри ораганизованы в XML[16] формате TAPN, который является подмножеством PNML[17], стандартный формат данных в программе для анализа временных сетей Петри TAPAAL.

Входные/Выходные данные представляются ввиде JSON, со своей схемой для валидации.

Выходные данные для метрических графов так же могут быть в формате yEd GraphML, который является расширением GraphML[18], стандартным форматом для yEd Graph Edito[19].

### 3.3.2. Обоснования выбора метода организации входных и выходных данных

Формат TAPN выбран для совместимости с TAPAAL. Формат yEd GraphML выбран для использования программы yEd Graph Editor для визуализации полученных графов.

Формат JSON выбран потому что,

1) JSON позволяет быстро считатывать и записывать данные, так как хранит только то, что нужно

2) В данное время есть денденция выносить вычисления в облачные сервисы, где основным стандартом формата данных является JSON, нельзя исключать такой вариант развития данной программы.

## 3.4. Описание и обоснование выбора состава технических и программных средств

### 3.4.1. Состав технических и программных средств

Для работы программы необходим следующий состав технических средств[20]:

- Процессор архитектуры AMD или Intel с частотой не менее 2 266 МГц;
- Не менее 512 МБ ОЗУ;
- Не менее 200МБ на жестком диске;
- Клавиатура;

Для работы программы необходим следующий состав программных средств[20]

- Одна из ниже перечисленных операционных систем:
  - Windows 10
  - Windows 8.x (настольная версия)
  - Windows 7 с пакетом обновления 1 (SP1)
  - Windows Vista SP2
  - Windows Server 2008 R2 с пакетом обновления 1 (SP1) (64-разрядная версия)
  - Windows Server 2012 и 2012 R2 (64-разрядная версия)
  - Mac OS X 10.8.3+, 10.9+
  - Oracle Linux 5.5+1
  - Oracle Linux 6.x (32-разрядная версия), 6.x (64-разрядная версия)2
  - Oracle Linux 7.x (64-разрядная версия)2
  - Red Hat Enterprise Linux 5.5+1, 6.x (32-разрядная версия), 6.x (64-разрядная версия)2
  - Red Hat Enterprise Linux 7.x (64-разрядная версия)2
  - Suse Linux Enterprise Server 10 SP2+, 11.x
  - Suse Linux Enterprise Server 12.x (64-разрядная версия)2
  - Ubuntu Linux 12.04 LTS, 13.x, 14.x, 15.x, 16.x, 18.x
- Установленная Java SE Runtime Environment 11 или выше

### 3.4.2. Обоснование выбора технических и программных средств

Программа реализована на языке Java версии 11[21], с использованием библиотек, полностью написанных на Java. В программе используется много особенностей добавленных в Java 11, поэтому версия ниже не подойдёт.

# 4. Ожидаемые технико-экономические показатели

## 4.1. Предполагаемая потребность

Данная программа может быть полезна при изучении совместных свойств метрических графов и временных сетей Петри.

## 4.2. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами

Данная программа является первой реализацией конвертаций метрических графов в сети Петри и обратно, так как эта сфера только начинает развиваться.

# Список используемых источников

[1] Petri Net. Дата посещения 1.04.2020. URL: https://en.wikipedia.org/wiki/Petri_net.

[2] Peter Kuchment Gregory Berkolaiko. Introduction to Quantum Graphs. American Mathematical Soc. T. Vol. 186.

[3] Leonid Dworzanski Vsevolod Chernyshev. On Correspondence Between Metric Graphs and Timed-Arcs Petri Nets.

[4] UPPAAL. Home. Дата посещения 1.04.2020. URL: http://www.uppaal.org/.

[5] TAPAAL. Introduction. Дата посещения 1.04.2020. URL: http://www.tapaal.net/introduction/.

[6] Class IndexedFRLayoutAlgorithm2D. Дата посещения 1.04.2020. URL: https://jgrapht.org/javadoc/org/jgrapht/alg/drawing/IndexedFRLayoutAlgorithm2D.html.

[7] JSON. Дата посещения 1.04.2020. URL: https://ru.wikipedia.org/wiki/JSON.

[8] JSON Schema. Дата посещения 1.04.2020. URL: https://json-schema.org/.

[9] SAX. Дата посещения 1.04.2020. URL: https://ru.wikipedia.org/wiki/SAX.

[10] yEd GraphML. Дата посещения 1.04.2020. URL: http://docs.yworks.com/yfiles/doc/developers-guide/graphml.html.

[11] Builder pattern. Дата посещения 1.04.2020. URL: https://en.wikipedia.org/wiki/Builder_pattern.

[12] a Java library of graph theory data structures and algorithms. Дата посещения 1.04.2020. URL: https://jgrapht.org/.

[13] Chain-of-responsibility pattern. Дата посещения 1.04.2020. URL: https://en.wikipedia.org/wiki/Chain-of-responsibility_pattern.

[14] Interface Function<T,R>. Дата посещения 1.04.2020. URL: https://docs.oracle.com/javase/8/docs/api/java/util/function/Function.html.

[15] extension point (точка расширения). Дата посещения 1.04.2020. URL: https://openu.ru/Books/UML/Extension_point.asp.

[16] XML. Дата посещения 1.04.2020. URL: https://ru.wikipedia.org/wiki/XML.

[17] Petri Net Markup Language. Дата посещения: 22.11.2018. URL: http://www.pnml.org.

[18] The GraphML File Format. Дата посещения 1.04.2020. URL: http://graphml.graphdrawing.org/.

[19] yEd Graph Editor. Дата посещения 1.04.2020. URL: https://www.yworks.com/products/yed.

[20] Каковы системные требования для Java? Дата посещения: 1.04.2020. URL: https://www.java.com/ru/download/help/sysreq.xml.

[21] Java™ Platform, Standard Edition 11 API Specification. Дата посещения 1.04.2020. URL: https://docs.oracle.com/javase/11/docs/api/overview-summary.html.

# Приложение 1. Диаграмма модулей

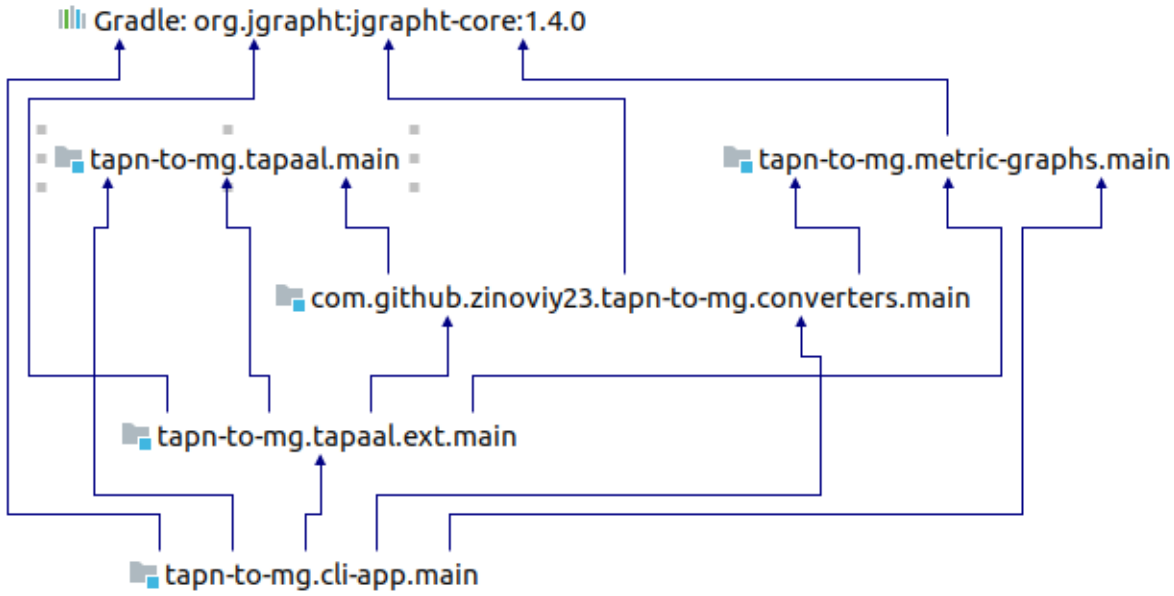


Рис. 5 — Диаграмма модулей

# Приложение 2. Диаграмма классов

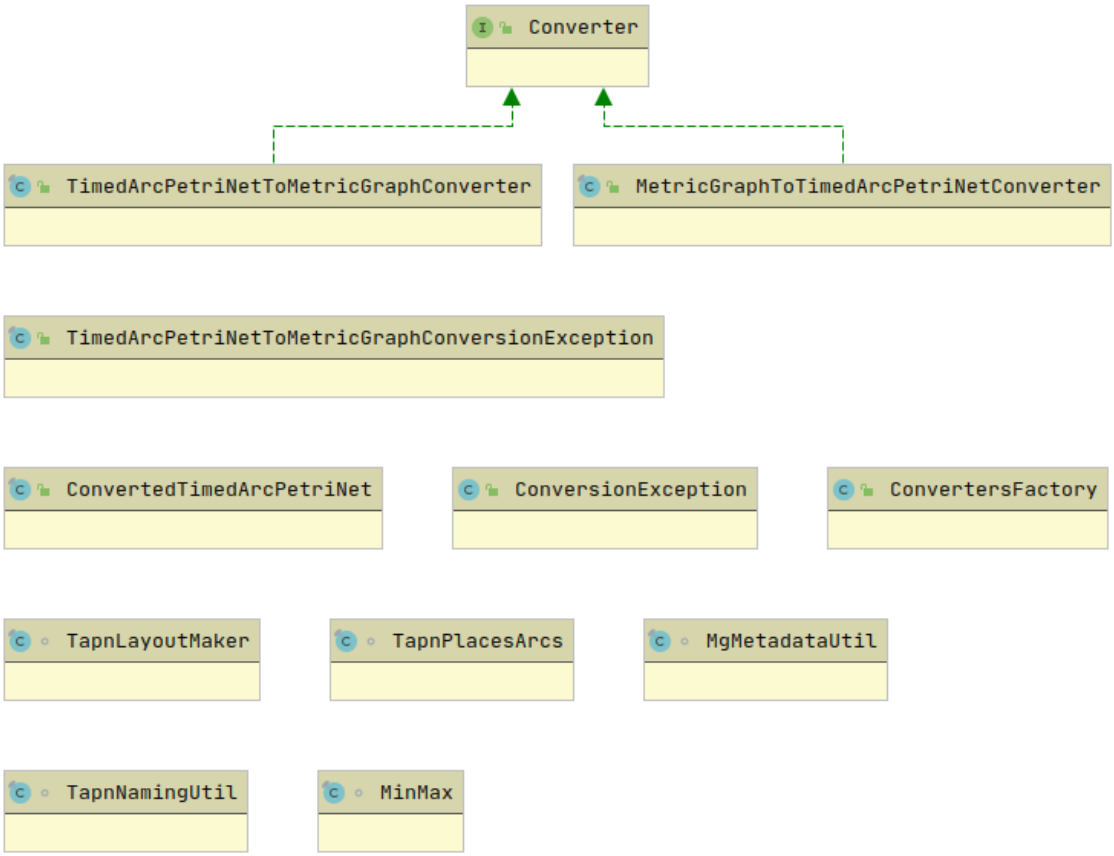


Рис. 6 — Диаграмма классов модуля converters

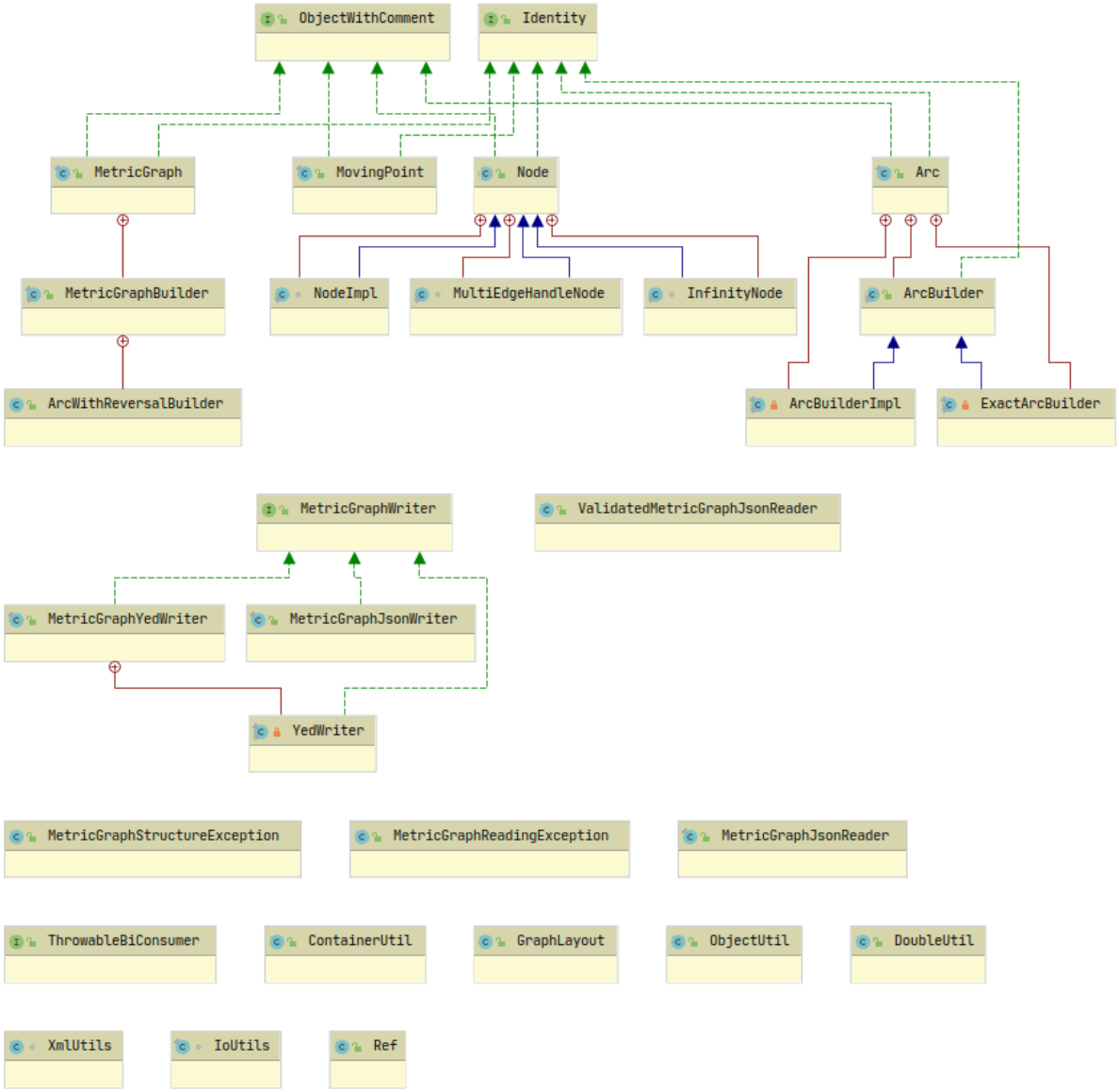


Рис. 7 — Диаграмма классов модуля metric-graphs

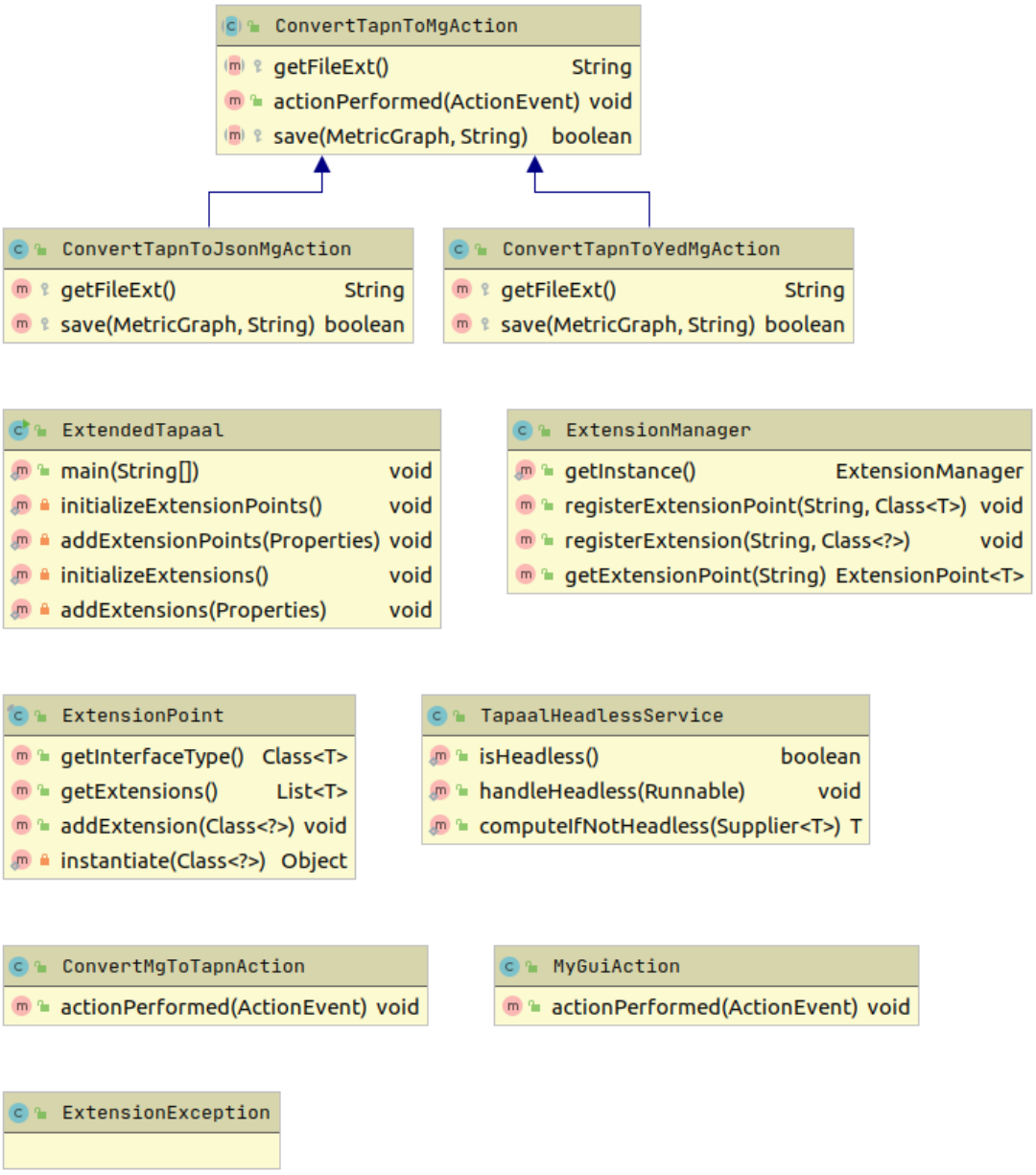


Рис. 8 — Диаграмма классов модуля tapaal.ext

# Приложение 3. JSON схема для метрических графов

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://jsongraphformat.info/v2.0/json-graph-schema.json",
  "title": "JSON Metric Graph Schema",
  "oneOf": [
    {
      "type": "object",
      "properties": {
        "graph": {
          "$ref": "#/definitions/graph"
        }
      },
      "additionalProperties": false,
      "required": [
        "graph"
      ]
    }
  ],
  "definitions": {
    "graph": {
      "type": "object",
      "required": ["id"],
      "additionalProperties": false,
      "properties": {
        "id": {
          "type": "string"
        },
        "label": {
          "type": "string"
        },
        "directed": {
          "type": [
            "boolean"
          ],
          "default": true
        },
        "type": {
          "type": "string"
        },
        "metadata": {
          "type": "object",
          "properties": {
            "comment": {
              "type": "string"
            }
          },
          "additionalProperties": true
        },
        "nodes": {
```

```
  "type": "object",
  "additionalProperties": {
    "type": "object",
    "properties": {
      "label": {
        "type": "string"
      },
      "metadata": {
        "type": "object",
        "properties": {
          "comment": {
            "type": "string"
          }
        },
        "additionalProperties": true
      }
    },
    "additionalProperties": false
  }
},
"edges": {
  "type": [
    "array"
  ],
  "items": {
    "type": "object",
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string"
      },
      "source": {
        "type": "string"
      },
      "target": {
        "type": "string"
      },
      "relation": {
        "type": "string"
      },
      "directed": {
        "type": [
          "boolean"
        ],
        "default": true
      },
      "label": {
        "type": "string"
      },
      "metadata": {
        "type": [
```

```
            "object"
          ],
          "properties": {
            "point": {
              "type": "array",
              "items": {
                "type": "object",
                "properties": {
                  "id": {
                    "type": "string"
                  },
                  "position": {
                    "type": "number"
                  },
                  "metadata": {
                    "type": "object",
                    "properties": {
                      "comment": {
                        "type": "string"
                      }
                    },
                    "additionalProperties": true
                  }
                },
                "additionalProperties": false,
                "required": ["id", "position"]
              }
            },
            "length": {
              "type": ["number", "string"]
            }
          },
          "additionalProperties": true
        }
      },
      "required": [
        "source",
        "target",
        "id"
      ]
    }
  }
 }
}
```

# Приложение 4. Описание и функциональное назначение классов

Таблица 4.1

| | | | | |
|---|---|---|---|---|
| Изм. | Лист | № докум. | Подп. | Дата |
| RU.17701729.04.01-01 ПЗ 81 01-1 | | | | |
| Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата |

| **Класс** | **Назначение** |
|---|---|
| TapnToMg | Главный класс CLI |
| ConvertedTimedArcPetriNet | Результат конвертации из метрического графа в сеть Петри |
| MetricGraphToTimedArcPetriNetConverter | Конвертер из метрического графа в сеть Петри |
| MinMax | Класс для агрегации из коллекции минимума и максимума |
| TapnLayoutMaker | Класс для расположения вершин сети Петри |
| TapnNamingUtil | Класс для именования объектов в сети Петри |
| MgMetadataUtil | Класс для именования объектов в метрическом графе |
| TapnPlacesArcs | Класс для обращения к ребрам инцидентным местам в сети Петри |
| TimedArcPetriNetToMetricGraphConversionException | Исключение при конвертации из сети Петри в метрический граф |
| TimedArcPetriNetToMetricGraphConverter | Конвертер из сетей Петри в метрические графы |
| ConversionException | Исключение при конвертации |
| Converter | Общий класс для конвертаций |
| ConvertersFactory | Фабрика конвертеров |
| ObjectWithComment | Объект с комментарием |
| MetricGraphYedWriter | Класс для записи в формате yEd |
| XmlUtils | Методы для работы с XML |
| MetricGraphJsonReader | Класс для считывания с JSON |
| MetricGraphJsonWriter | Класс для записи в JSON |
| MetricGraphReadingException | Исключение при считывание |
| MetricGraphWriter | Общий класс для записи графов |
| ValidatedMetricGraphJsonReader | Класс для считывания с валидацией |
| ContainerUtil | Методы для работы с коллекциями |
| DoubleUtil | Методы для работы с числами |
| GraphLayout | Класс для расположения графов |
| ObjectUtil | Методы для работы с объектами |
| Ref | Ссылка на значение |
| ThrowableBiConsumer | Функциональный интерфейс для принятия 2х значений |
| Arc | Класс ребро |
| MetricGraph | Класс граф |
| MetricGraphStructureException | Ошибка в структуре графа |
| MetricGraphBuilder | Класс для построения графа |
| ArcBuilder | Общий класс для построения ребра |
| ArcBuilderImpl | Реализация класс для построения ребра |
| ExactArcBuilder | Реализация, которая возвращает заданное ребро |

Продолжение таблицы 4.1

| **Класс** | **Назначение** |
|---|---|
| YedWriter | Класс для записи в yEd XML |
| MovingPoint | Класс точка |
| Node | Класс вершина |
| ExtensionException | Ошибка при расширении |
| ExtensionManager | Класс для управления расширениями |
| ExtensionPoint | Представление точки расширений |
| TapaalHeadlessService | Класс для контроля headless режима |
| ConvertMgToTapnAction | Класс для действия конвертации в сеть Петри |
| ConvertTapnToJsonMgAction | Класс для действия конвертации в JSON |
| ConvertTapnToMgAction | Общий класс для действий конвертации |
| ConvertTapnToYedMgAction | Класс для действия конвертации в yEd |
| ExtendedTapaal | Главный класс расширенного TAPAAL |
| Identity | объект с id |

# Приложение 5. Описание и функциональное назначение полей и методов

Таблица 5.1

Класс TapnToMg

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| main | public | главный метод |

Таблица 5.2

Класс ConvertedTimedArcPetriNet

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getNetwork | public | возвращает временную сеть Петри |
| getTemplate | public | возвращает шаблон |

Таблица 5.3

Класс MetricGraphToTimedArcPetriNetConverter

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| addPlacesForEachArc | private | добавляет места для каждого ребра |
| processArc | private | обрабатывает ребро |
| addArcForNeighbours | private | добавляет ребра для соседей |
| handleInfinity | private | обрабатывает бесконечные вершины |

Таблица 5.4

Класс MinMax

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| reduce | package-private | изменяет мин/макс на входящее число |
| combine | package-private | комбинирует 2 объетка |
| avg | package-private | среднее из мин макс |

Таблица 5.5

Класс TapnLayoutMaker

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getTextFromId | private | возвращает описание по id |
| createDataLayer | public | создает расположение графа |
| createNote | private | создает описание сети Петри |
| createPetriNetObject | public | по абстрактному объекту в графе расположения создает элемент сети Петри |

Таблица 5.6

Класс TapnNamingUtil

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getTapnName | package-private | имя сети Петри |
| nameForPlace | package-private | имя для места |
| nameForTransition | package-private | имя для соединения |
| nameForCollapsingTransition | package-private | имя для схлопывающего места |
| nameForInfinitePlace | package-private | имя для бесконечного места |

Таблица 5.7

Класс MgMetadataUtil

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getNodeName | private-package | имя для вершины |
| getNodeComment | private-package | коммент для вершины |
| getArcName | private-package | имя для ребра |
| getNameForReversal | private-package | имя для обратного ребра |
| getTokenName | private-package | имя для точек по токену |
| getLeadName | private-package | имя для ребра с одной вершиной |
| getInfName | private-package | имя для бесконечной вершины |
| getGraphName | private-package | имя графа |
| getMultiedgeHandlerNodeName | private-package | имя для вершины ломающей кратное ребро |
| getMultiedgeHandlerArcsNames | private-package | имя для ребер ломающих кратное ребро |
| getSelfLoopHandleArcsNames | private-package | имя для ребер ломающих цикл с самим собой |
| getComment | private-package | комментарий к графу |

Таблица 5.8

Класс TapnPlacesArcs

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getOutputArcs | public | возвращает выходные ребра для места |
| getInputArcs | public | возвращает входные ребра для места |
| assertInputArc | private | провряет выходное ребро |

Таблица 5.9

Класс TimedArcPetriNetToMetricGraphConverter

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| addNodes | private | добавляет вершины |
| addEdges | private | добавляет ребра |
| addEdge | private | добавляет ребро |
| fixSelfLoop | private | возвращает ребра для починки цикла с самим собой |
| fixLength | private | возвращает длину ребра, всегда больше 0 |
| getPointsFromPlace | private | точки из токенов в текущем месте |
| addLeads | private | добавляет ребра с одной вершиной |
| getPointsFromInputArc | private | добавляет точки из входного ребра |
| assertTransition | private | проверяет соединения |

Таблица 5.10

Класс Converter

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| convert | public | конвертирует A в B |

Таблица 5.11

Класс ConvertersFactory

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| createMetricGraphToTimedArcPetriNetConverter | public | создает конвертер из файла с графом в файл с сетью Петри |
| createFileToMetricGraphConverter | public | создает конвертер из файла в метрический граф |
| createTimedArcPetriNetToFileConverter | public | создает конвертер из сети Петри в файл |
| createFileToTimedArcPetriNetConverter | public | создает конвертер из файла в сеть петри |

Таблица 5.12

Класс Identity

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getId | public | возвращает id |

Таблица 5.13

Класс ObjectWithComment

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getComment | public | возвращает комментарий |

Таблица 5.14

Класс MetricGraphYedWriter

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| addIndents | private | добавляет отступы к XML |

Таблица 5.15

Класс MetricGraphJsonReader

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| | | |
| read | public | считывает граф |
| internalRead | private | считывает граф |
| readGraph | private | считывает и собирает граф |
| addArcAndReversal | private | добавляет ребро и обратное к нему |
| readEdges | private | считывает ребра |
| readEdge | private | считывает ребро |
| readArcMetadata | private | считвыает метаинформацию ребра |
| readPoints | private | считывает точки ребра |
| readPoint | private | считывает точку |
| readNodes | private | считывает вершины |
| readNode | private | считывает вершину |
| fetchMetadata | private | достае метаинформацию |
| lastLocation | public | последнее положение в файле до ошибки |

Таблица 5.16

Класс MetricGraphJsonWriter

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| internalWrite | private | записывает граф |
| writeGraphInfo | private | записывает дополнительную информацию о графе |
| writeNodes | private | записывает вершины |
| writeEdges | private | записывает ребра |
| writeArcMetadata | private | записывает метаинформацию ребра |
| writeComment | private | записывает комментарий |

Таблица 5.17

Класс MetricGraphWriter

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| write | public | записывает граф |

Таблица 5.18

Класс ValidatedMetricGraphJsonReader

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| validate | private | валидирует входной поток по схеме |
| getReader | public | возвращает внутренний считыватель |

Таблица 5.19

Класс ContainerUtil

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| first | public | первый элемент в коллекции |
| second | public | второй элемент в коллекции |
| getFromTable | public | достает элемент из таблицы |
| iterate | public | конвертирует Stream в Iterable |

Таблица 5.20

Класс DoubleUtil

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| equals | public | сравнивает два вещественных числа с учетом бесконечности и погрешности |

Таблица 5.21

Класс GraphLayout

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| createLayout | public | создает расположения графа |
| calcSize | private | считает размер области, в которой нужно разместить граф |

Таблица 5.22

Класс ObjectUtil

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| doIfNotNull | public | делает переданное действие, если первый аргумент не null |

Таблица 5.23

Класс Ref

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getData | public | возвращает содержимое |
| setData | public | обновляет содержимое |
| update | public | обновляет содержимое с помощью переданной функции |

Таблица 5.24

Класс ThrowableBiConsumer

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| consume | public | принимает два аргумента и обрабатывает 2 аргумента |

Таблица 5.25

Класс Arc

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| createBuilder | public | создает Builder ребра |
| verifyLength | private | проверяет длину ребра |
| checkPointsOnArc | package-private | проверяет что точки находятся на ребре |
| getLabel | public | возвращает метку |
| getSource | public | возвращает начальную вершину |
| getTarget | public | возвращает конечную вершину |
| getLength | public | возвращает длину |
| getPoints | public | возвращает точки |
| isDistanceToTarget | public | возвращает, считается ли расстояние с конца ребра |
| toBuilder | package-private | преварщает ребро в Builder ребра |

Таблица 5.26

Класс MetricGraph

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| createBuilder | public | создает Builder графа |
| getGraph | public | возвращает структуру графа |
| getLabel | public | возвращает метку |
| getReversal | public | обратное ребро |
| getNode | public | возвращает вершину по id |
| getArc | public | возвращает ребро по id |

Таблица 5.27

Класс MetricGraphBuilder

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| buildGraph | public | строит граф |
| addNode | public | добавляет вершину |
| addArc | public | добавляет ребро |
| setLabel | public | добавляет метку |
| setComment | public | присваивает комментарий |
| setId | public | присваивает идентификатор |
| addPoints | public | добавляет точки |
| containsEdge | public | проверяет есть ли ребро |
| verifyPoints | private | проверяет точки |
| addPoints | private | добавляе идентификаторы точек |
| verifyId | private | проверят идентификатор |
| addId | private | добавляет идентификатор |
| failAlreadyExists | private | кидает исключение, если объект уже есть |

Таблица 5.28

Класс ArcBuilder

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| setSource | public | присваивает начало ребра |
| getSource | public | возвращает начало ребра |
| setTarget | public | присваивает конец ребра |
| getTarget | public | возвращает конец ребра |
| setLength | public | присваивает длину |
| getLength | public | возвращает длину |
| addPoint | public | добавляет точку |
| createArc | public | создает ребро |
| setId | public | присваивает идентификатор |
| getPoints | public | возвращает точки |
| setPoints | public | присваивает точки |
| getLabel | public | возвращает метку |
| setLabel | public | присваивает точку |
| getComment | public | возвращает коммент |
| setComment | public | присваивает коммент |
| copy | public | копирует объект |
| makeString | protected | создает строку для toString |

Таблица 5.29

Класс ArcBuilderImpl

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| verifyPointId | private | проверяет идентификатор точки |

Таблица 5.30

Класс YedWriter

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| writeYedSchemes | private | записывает схемы в начало |
| writeDataElements | private | записывает ключи данных |
| writeDataDef | private | записывает определение ключа данных |
| writeGraph | private | записывает граф |
| writeArc | private | записывает ребро |
| writePoint | private | записывает точку |
| writeArcGraphics | private | записывает визуальное представление ребра |
| writeComment | private | записывает коммент |
| writeNode | private | записывает вершину |
| writeNodeGraphics | private | записывает визуальное прдставление вершины |

Таблица 5.31

Класс MovingPoint

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getPosition | public | позиция точки на ребре |

Таблица 5.32

Класс Node

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| createNode | public | создает вершину |
| createInfinity | public | создает бесконечную вершину |
| createMultiEdgeHandler | public | создает вершину, для разлома кратных ребер |
| getLabel | public | возвращает метку |
| isInfinity | public | проверяет вершину на бесконечность |

Таблица 5.33

Класс ExtensionManager

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getInstance | public | возвращает текущий экземпляр |
| registerExtensionPoint | public | регистрирует точку расширения |
| registerExtension | public | регистрирует расширение |
| getExtensionPoint | public | возвращает точку расширения |

Таблица 5.34

Класс ExtensionPoint

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getInterfaceType | public | возвращает интерфейс точки расширения |
| getExtensions | public | возвращает расширения |
| addExtension | public | добавляет расширение |
| instantiate | private | создает объект переданного класса |

Таблица 5.35

Класс TapaalHeadlessService

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| isHeadless | public | проверяет, что приложение находится в headless режиме |
| handleHeadless | public | делает действие, если режим не headless |
| computeIfNotHeadless | public | выполняет вычисление, если режим не headless |

Таблица 5.36

Класс ConvertTapnToMgAction

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| getFileExt | protected | возвращает расширение для результирующего файла |
| save | protected | сохраняет файл |

Таблица 5.37

Класс ExtendedTapaal

| **Имя** | **Модификатор доступа** | **Назначение** |
|---|---|---|
| main | | |
| initializeExtensionPoints | private | инициализирует точки расширения |
| addExtensionPoints | private | добавляет точки расширения |
| initializeExtensions | private | иницилизирует расширения |
| addExtensions | private | добавляет точки расширения |

# Лист регистрации изменений

| Изм. | Номера листов | | | | Всего листов в документе | № документа | Входящий № сопровод. докум. и дата | Подпись | Дата |
|---|---|---|---|---|---|---|---|---|---|
| | измененных | замененных | новых | аннулированных | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |