

# XML-ФОРМАТ ОПИСАНИЯ ВИЗУАЛИЗАТОРОВ

Корнеев Г.А., СПбГУ ИТМО

Информацию об ошибках и комментарии отправляйте по адресу [kgeorgiy@rain.ifmo.ru](mailto:kgeorgiy@rain.ifmo.ru)

## Оглавление

ОГЛАВЛЕНИЕ .....	1
1. ОБЩАЯ СТРУКТУРА .....	2
2. ОПИСАНИЯ ТЕГОВ .....	2
2.1. <i>Общее описание визуализатора (tag visualizer)</i> .....	3
3. ОПИСАНИЕ ЛОГИКИ ВИЗУАЛИЗАТОРА .....	4
3.1. <i>Основные элементы</i> .....	4
Описание логики визуализатора (tag algorithm) .....	4
Описание импортируемых пакетов и классов (tag import) .....	5
Описание глобальных переменных (tag variable) .....	6
Описание глобальных методов (tag method) .....	6
3.2. <i>Описание модели данных</i> .....	7
Описание модели данных (tag data) .....	7
Описание переменных модели данных (tag variable) .....	8
Описание метода преобразования модели данных в строку (tag toString) .....	8
3.3. <i>Описание процедур</i> .....	9
Описание процедур (tag auto) .....	9
Описание локальных переменных (tag variable) .....	10
Описание начального и конечного состояния автомата (теги start и finish) .....	10
3.4. <i>Описание шагов алгоритма</i> .....	11
Простой шаг (tag step) .....	11
Описание действий (теги action, direct и reverse) .....	13
Вызов вложенного автомата (tag call-auto) .....	13
Оператор выбора (теги if, then, else) .....	14
Цикл с предусловием (tag while) .....	16
4. ОПИСАНИЕ КОНФИГУРАЦИИ ВИЗУАЛИЗАТОРА .....	17
Описание конфигурации визуализатора (tag configuration) .....	17
4.1. <i>Описание элементов интерфейса</i> .....	18
Описание панели (tag panel) .....	18
Описание кнопки (tag button) .....	18
Описание новой панели выбора (tag adjustable-panel) :TODO: .....	19
Описание старой панели выбора (tag spin-panel) .....	19
4.2. <i>Таблицы стилей</i> .....	20
Описание стиля (tag style) :TODO: .....	20
Описание таблицы стилей (style-set) .....	22
Описание шрифта (tag font) .....	22
Описание цвета (tag color) .....	23
4.3. <i>Группы, свойства и сообщения</i> .....	24
Описание группы (tag group) .....	24
Описание свойства (tag property) .....	25
Описание сообщений (tag message) .....	25
ССЫЛКИ .....	26

# 1. Общая структура

XML-описание визуализатора состоит из трех основных частей:

- общее описание визуализатора;
- описание логики визуализатора;
- описание конфигурации визуализатора.

Общее описание визуализатора содержит основную информацию о визуализаторе:

- название визуализируемого алгоритма;
- информацию об авторе визуализатора и его руководителе;
- название пакета, в котором находится визуализатор и его главного класса;
- предпочтительные размеры визуализатора.

Описание логики визуализатора содержит описания процедур (автоматов) реализующих визуализируемый алгоритм. Так же в описании логики визуализатора может присутствовать информация, облегчающая построение обратного автомата.

Описание конфигурации визуализатора содержит информацию о том, как визуализатор должен отображаться на экране, в том числе:

- размеры частей визуализатора,
- конфигурация нестандартных элементов управления;
- сообщения, выдаваемые пользователю (в том числе, сообщения об ошибках);
- таблицы стилей для элементов оформления;
- допустимые границы изменения параметров пользователем.

В следующем разделе подробно описываются все теги, которые могут быть использованы в описании визуализатора.

## 2. Описания тегов

В этом разделе описываются теги, которые могут быть использованы в XML-описании визуализатора. Для каждого тега приведено его краткое описание, описание атрибутов и вложенных тегов. Для формализации использования тегов для каждого из них приведены фрагменты определения типа документа [1] (Document Type Definition, DTD) и XML-схема [2, 3, 4].

Описание тегов и атрибутов содержит столбец, помеченный диэзом (#), обозначающий какое количество раз может встречаться этот тег. Используются следующие обозначения:

- \* — произвольное количество раз;
- + — не менее одного раза;
- 1 — ровно один раз.
- ? — не более одного раза.

Описание атрибутов так же включает столбец, помеченный буквой “И”. В этом столбце стоит “+”, если значение атрибута должно задаваться на нескольких языках. Имена атрибутов для различных

языков образуются из имени атрибута с добавленным к нему дефисом и двухбуквенным кодом языка. На пример, вместо атрибута message нужно будет задать атрибуты message-ru и message-en для русского и английского языка соответственно.

Для каждого тега приведены его атрибуты и описание содержимого. При описании вложенных тегов используется сокращение steps обозначающее одно из step, call-auto, if или while.

## 2.1. Общее описание визуализатора (тег visualizer)

Атрибуты тега visualizer содержат общее описание визуализатора, а его потомки содержат описание логики (тег algorithm) и конфигурации (тег configuration) визуализатора.

### DTD

```
<!ELEMENT visualizer
  ((algorithm, configuration) | (configuration, algorithm))
>

<!ATTLIST visualizer
  id                ID                #REQUIRED
  package           CDATA            #REQUIRED
  main-class        CDATA            #REQUIRED
  preferred-width   CDATA            #REQUIRED
  preferred-height  CDATA            #REQUIRED
  name-ru           CDATA            #REQUIRED
  name-en           CDATA            #REQUIRED
  copyright-ru      CDATA            #REQUIRED
  copyright-en      CDATA            #REQUIRED
  author-en         CDATA            #REQUIRED
  author-ru         CDATA            #REQUIRED
  author-email      CDATA            #REQUIRED
  supervisor-en     CDATA            #REQUIRED
  supervisor-ru     CDATA            #REQUIRED
  supervisor-email  CDATA            #REQUIRED
  xmlns:xsi         CDATA            #FIXED
                   "http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation CDATA #FIXED
                   "http://ips.ifmo.ru/vizi/schema/visualizer.xsd"
>
```

### XML-схема

```
<xsd:element name="visualizer" type="visualizer"/>
<xsd:complexType name="visualizer">
  <xsd:sequence>
    <xsd:element name="algorithm" type="algorithm"/>
    <xsd:element name="configuration" type="configuration"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="id" use="required"/>
  <xsd:attribute name="package" type="xsd:string" use="required"/>
  <xsd:attribute name="main-class" type="xsd:string" use="required"/>
  <xsd:attribute name="preferred-width" type="xsd:positiveInteger" use="required"/>
  <xsd:attribute name="preferred-height" type="xsd:positiveInteger" use="required"/>
  <xsd:attribute name="name-ru" type="xsd:string" use="required"/>
  <xsd:attribute name="name-en" type="xsd:string" use="required"/>
  <xsd:attribute name="copyright-ru" type="xsd:string" use="required"/>
  <xsd:attribute name="copyright-en" type="xsd:string" use="required"/>
  <xsd:attribute name="author-en" type="xsd:string" use="required"/>
  <xsd:attribute name="author-ru" type="xsd:string" use="required"/>
  <xsd:attribute name="author-email" type="xsd:string" use="required"/>
  <xsd:attribute name="supervisor-en" type="xsd:string" use="required"/>
  <xsd:attribute name="supervisor-ru" type="xsd:string" use="required"/>
  <xsd:attribute name="supervisor-email" type="xsd:string" use="required"/>
</xsd:complexType>
```

## Атрибуты

Атрибут	#	И	Описание
Id	1		Идентификатор визуализатора
Package	1		Пакет, в котором находится визуализатор
main-class	1		Имя основного класса визуализатора
preferred-width	1		Предпочтительная ширина визуализатора в пикселях
preferred-height	1		Предпочтительная высота визуализатора в пикселях
Name	1	+	Название визуализатора
Author	1	+	Информация об авторе визуализатора
author-email	1		E-mail автора визуализатора
Supervisor	1	+	Информация о руководителе проекта
Supervisor-email	1		E-mail руководителя проекта
Copyright	1	+	Информация о подразделении, создавшем визуализатор

## Содержимое

Вложенные элементы

Тег	#	Описание
algorithm	1	Описание логики визуализатора
configuration	1	Описание модели данных

## Пример

```
<visualizer
  id="FindMaximum"
  package="ru.ifmo.vizi.find_max"
  main-class="FindMaximumVisualizer"

  preferred-width="400"
  preferred-height="250"

  name-ru="Поиск максимума в массиве\nнатуральных чисел (пример)"
  name-en="Search for maximum element in the array\nof natural numbers (example)"

  author-ru="Георгий Корнеев"
  author-en="Georgiy Korneev"
  author-email="kgeorgiy@rain.ifmo.ru"

  supervisor-ru="Георгий Корнеев"
  supervisor-en="Georgiy Korneev"
  supervisor-email="kgeorgiy@rain.ifmo.ru"

  copyright-ru="Copyright \u00A9 Кафедра КТ, СПб ГИТМО (ТУ), 2003"
  copyright-en="Copyright \u00A9 Computer Technologies Department, SPb IFMO, 2003"
> ... </visualizer>
```

## 3. Описание логики визуализатора

### 3.1. Основные элементы

#### Описание логики визуализатора (тег algorithm)

Тег algorithm является контейнером для описаний автоматов и модели данных, используемой визуализатором.

## DTD

```
<!ELEMENT algorithm (import*, variable*, data, auto+, method*)>
<!ATTLIST algorithm>
```

## XML-схема

```
<xsd:complexType name="algorithm">
  <xsd:sequence>
    <xsd:element
      name      = "import"
      type      = "xsd:string"
```

```

        minOccurs = "0"
        maxOccurs = "unbounded"
    />
    <xsd:element
        name = "variable"
        type = "global-variable"
        minOccurs = "0"
        maxOccurs = "unbounded"
    />
    <xsd:element
        name = "data"
        type = "data"
    />
    <xsd:element
        name = "auto"
        type = "auto"
        maxOccurs = "unbounded"
    />
    <xsd:element
        name = "method"
        type = "method "
        minOccurs = "0"
        maxOccurs = "unbounded"
    />
</xsd:sequence>
</xsd:complexType>

```

### Атрибуты

Нет

### Содержимое

Вложенные элементы

Тег	#	Описание
import	*	Описание импортируемых пакетов и классов
variable	*	Описание глобальных переменных
data	1	Описание модели данных
method	*	Описание глобальных методов
auto	+	Описание автомата

### Пример

```
<algorithm> ... </algorithm>
```

#### Описание импортируемых пакетов и классов (тег import)

Тег import служит для описания пакетов и классов, импортируемых реализацией алгоритма. Тело тега должно содержать описание импортируемого пакета или класса в виде, определенном в разделе 7.5 [5].

Автоматически импортируются пакет ru.ifmo.vizi.base.auto и класс java.util.Locale.

### DTD

```

<!ELEMENT import (#PCDATA)*>
<!ATTLIST import>

```

### XML-схема

```

<xsd:element
    name = "import"
    type = "xsd:string"
    minOccurs = "0"
    maxOccurs = "unbounded"
/>

```

### Атрибуты

Нет

## Содержимое

Текст — описание импортируемого пакета или класса.

## Примеры

```
<import>java.io.*</import>
<import>java.util.List</import>
```

### Описание глобальных переменных (тег variable)

Тег variable непосредственно внутри тега algorithm служит для описания глобальных переменных, доступных во всех автоматах. Имена переменных должны быть корректными java-идентификаторами и не содержать символов подчеркивания.

## DTD

```
<!ELEMENT variable EMPTY>
<!ATTLIST variable
  description      CDATA      #REQUIRED
  name             CDATA      #IMPLIED
  type             CDATA      #IMPLIED
  value           CDATA      #IMPLIED
>
```

## XML-схема

```
<xsd:complexType name="global-variable">
  <xsd:attribute name="name" type="id" use="required"/>
  <xsd:attribute name="type" type="type" use="required"/>
  <xsd:attribute name="value" type="value" use="required"/>
  <xsd:attribute name="description" type="description" use="required"
/>
</xsd:complexType>
```

## Атрибуты

Атрибут	#	Описание
name	1	Имя переменной
type	1	Тип переменной
value	1	Исходное значение переменной
description	1	Словесное описание переменной

## Содержимое

Нет

## Примеры

```
<variable
  description = "Массив для поиска"
  name       = "a"
  type      = "int[]"
  value     = "new int[]{1, 2, 3, 1, 3, 5, 6}"
/>
```

### Описание глобальных методов (тег method)

Тег method служит для описания глобальных методов.

## DTD

```
<!ELEMENT method (#PCDATA)*>
<!ATTLIST method
  header      CDATA      #REQUIRED
  comment     CDATA      #REQUIRED
>
```

## XML-схема

```
<xsd:complexType name="method">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
```

```
<xsd:attribute name = "header" type = "xsd:string" use = "required"/>
<xsd:attribute name = "comment" type = "description" use = "required"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
```

### Атрибуты

Нет

### Содержимое

Текст — текст метода на *Java*.

### Пример

```
<method
  header = "int strToInt(String s)"
  comment = "Преобразования строки в целое"
>
  try {
    return Integer.parseInt(s);
  } catch (Exception e) {
    return 0;
  }
</method>
```

## 3.2. Описание модели данных

### Описание модели данных (тег data)

Тег data служит для описания модели данных, используемых визуализатором. В XML-описании может присутствовать только одно описание модели данных.

Для совместимости с предыдущими версиями, в теге data могут описываться переменные модели данных, для новых проектов рекомендуется описывать глобальные переменные в теге algorithm.

### DTD

```
<!ELEMENT data (variable*, toString)>
<!ATTLIST data>
```

### XML-схема

```
<xsd:complexType name="data">
  <xsd:sequence>
    <xsd:element
      name = "variable"
      type = "deprecated-variable"
      minOccurs = "0"
      maxOccurs = "unbounded"
    />
    <xsd:element
      name = "toString"
      type = "xsd:string"
    />
  </xsd:sequence>
</xsd:complexType>
```

### Атрибуты

Нет

### Содержимое

Вложенные элементы

Тег	#	Описание
variable	*	Описание глобальных переменных модели
toString	1	Метод преобразования модели в строку (для отладки)

## Пример

```
<data> ... </data>
```

### Описание переменных модели данных (тег variable)

Описание переменных внутри модели данных сохранено для совместимости с предыдущими версиями. В новых версиях следует использовать описание глобальных переменных в теге `algorithm` и локальных переменных в описаниях процедур (теги `auto`).

Тег `variable` внутри тега `data` описывает переменные модели данных. Имена переменных должны быть корректными java-идентификаторами и не содержать символов подчеркивания.

## DTD

```
<!ELEMENT variable (#PCDATA)*>
<!ATTLIST variable
  description          CDATA          #REQUIRED
>
```

## XML-схема

```
<xsd:complexType name="deprecated-variable">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="description" type="description" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

## Атрибуты

Атрибут	#	Описание
description	1	Словесное описание переменной

## Содержимое

Текст — определение переменной.

## Примеры

```
<variable description = "Массив для поиска">
  int[] a = new int[]{1, 2, 3, 1, 3, 5, 6}
</variable>
```

### Описание метода преобразования модели данных в строку

#### (тег toString)

Тег `toString` служит для описания процедуры для преобразования текущего состояния модели данных в строку. Метод преобразование модели данных в строку используется для проверки корректности сгенерированного обратного автомата. Для дополнительной информации смотри раздел `:TODO:.`

## Атрибуты

Нет

## Содержимое

Тест — текст процедуры

## Пример

```
<toString>
  StringBuffer s = new StringBuffer();
```



```

s.append("max = ").append(@Main@max).append("\n");
s.append("i = ").append(@Main@i).append("\n");
return s.toString();
</toString>

```

### 3.3. Описание процедур

#### Описание процедур (тег auto)

Тег auto служит для описания процедур (автоматов), реализующих визуализируемый алгоритм. Каждая процедура имеет уникальный идентификатор, который должен быть корректным Java-идентификатором класса и не содержать символов подчеркивания.

#### DTD

```

<!ELEMENT auto (variable*, start?, (%steps;)+, finish?)>
<!ATTLIST auto
  id          ID          #REQUIRED
  description CDATA      #REQUIRED
>

```

#### XML-схема

```

<xsd:complexType name="auto">
  <xsd:sequence>
    <xsd:element
      name      = "variable"
      type      = "local-variable"
      minOccurs = "0"
      maxOccurs = "unbounded"
    />
    <xsd:element
      name      = "start"
      type      = "start-finish"
      minOccurs = "0"
    />
    <xsd:group
      ref       = "automata-steps"
      maxOccurs = "unbounded"
    />
    <xsd:element
      name      = "finish"
      type      = "start-finish"
      minOccurs = "0"
    />
  </xsd:sequence>
  <xsd:attribute name = "id"          type = "id"          use = "required"/>
  <xsd:attribute name = "description" type = "description" use = "required"
/>
</xsd:complexType>

```

#### Атрибуты

Атрибут	#	Описание
id	1	Идентификатор процедуры
description	1	Словесное описание процедуры

#### Содержимое

Вложенные элементы

Тег	#	Описание
variable	*	Описание локальных переменных
start	?	Описание начального состояния автомата
steps	+	Описание шагов автомата
finish	?	Описание конечного состояния автомата

#### Примеры

```

<auto id="Main" description="Поиск максимума в массиве">
...

```

</auto>

## Описание локальных переменных (тег variable)

Тег variable внутри тега auto служит для описания локальных переменных, доступных только внутри этого автомата. Если определены локальная и глобальная переменные с совпадающими именами, то в автомате будет доступна только локальная переменная.

Имена переменных должны быть корректными java-идентификаторами и не содержать символов подчеркивания.

### DTD

```
<!ELEMENT variable EMPTY>
<!ATTLIST variable
  description      CDATA      #REQUIRED
  name             CDATA      #IMPLIED
  type             CDATA      #IMPLIED
>
```

### XML-схема

```
<xsd:complexType name="local-variable">
  <xsd:attribute name="name" type="id" use="required"/>
  <xsd:attribute name="type" type="type" use="required"/>
  <xsd:attribute name="description" type="description" use="required"/>
</xsd:complexType>
```

### Атрибуты

Атрибут	#	Описание
name	1	Имя переменной
type	1	Тип переменной
description	1	Словесное описание переменной

### Содержимое

Нет

### Примеры

```
<variable
  description = "Переменная цикла"
  name       = "i"
  type      = "int"
/>
```

## Описание начального и конечного состояния автомата (теги start и finish)

Тег start (finish) служит для задания метода отображения и комментария к начальному (конченому) состоянию автомата.

### DTD

```
<!ELEMENT start (draw?)>
<!ATTLIST start
  comment-ru      CDATA      #IMPLIED
  comment-en      CDATA      #IMPLIED
  comment-args    CDATA      #IMPLIED
>
<!ELEMENT finish (draw?)>
<!ATTLIST finish
  comment-ru      CDATA      #IMPLIED
  comment-en      CDATA      #IMPLIED
  comment-args    CDATA      #IMPLIED
>
```

### XML-схема

```
<xsd:complexType name="start-finish">
  <xsd:sequence>
    <xsd:element
```

```

    name      = "draw"
    type      = "draw"
    minOccurs = "0"
  />
</xsd:sequence>
<xsd:attributeGroup ref="comment" />
</xsd:complexType>

```

### Атрибуты

Атрибут	#	И	Описание
comment	1	+	Комментарий к начальному состоянию
comment-args	?		Аргументы комментария

### Содержимое

Вложенные элементы

Тег	#	Описание
draw	?	Метод отображения начального состояния.

### Примеры

```

<start
  comment-ru="На экране изображен массив, в котором будет осуществляться поиск
    максимума"
  comment-en="There is an array on the display"
>
  ...
</start>

<finish
  comment-ru="Максимум найден ({0})"
  comment-en="Maximum found ({0})"
  comment-args="new Integer(@max)"
>
  <draw>
    @visualizer.updateArray(0, 0);
  </draw>
</finish>

```

## 3.4. Описание шагов алгоритма

Каждый шаг алгоритма, кроме вызова вложенного автомата (процедуры) имеет идентификатор, описание и уровень шага. Идентификатор шага должен быть корректным java-идентификатором, и быть уникальным в рамках процедуры.

Уровень шага задает, при шагах какого размера этот шаг отображается визуализатором. Значение уровня по умолчанию — 0, соответствует малым шагам.

### Простой шаг (тег step)

Простой шаг может содержать одно или несколько присваиваний переменным модели.

### DTD

```

<!ELEMENT step (draw?, (action | (direct, reverse?)))>
<!ATTLIST step
  id          NMTOKEN          #IMPLIED
  description CDATA            #REQUIRED
  level       CDATA            "0"
  comment-ru  CDATA            #IMPLIED
  comment-en  CDATA            #IMPLIED
  comment-args CDATA            #IMPLIED
>

```

### XML-схема

```

<xsd:complexType name="step">
  <xsd:sequence>
    <xsd:element
      name      = "draw"

```

```

    type          = "draw"
    minOccurs     = "0"
  />
  <xsd:choice>
    <xsd:element
      name       = "action"
      type      = "action"
    />
    <xsd:group ref="direct-reverse"/>
  </xsd:choice>
</xsd:sequence>
<xsd:attribute name = "id"           type = "id"/>
<xsd:attribute name = "description" type = "description"/>
<xsd:attribute name = "level"       type = "level" default = "0"/>
<xsd:attributeGroup ref="comment"/>
</xsd:complexType>

```

## Атрибуты

Атрибут	#	И	Описание
id	1		Идентификатор шага
description	1		Описание шага
level	?		Уровень шага
comment	1	+	Комментарий к шагу
comment-args	?		Аргументы комментария

## Содержимое

### Вложенные элементы

Тег	#	Описание
draw	?	Метод отображения состояния
action	?	Автоматически обращаемое действие
direct	?	Действие, обращаемое в ручную
reverse	?	Ручное обращение действия

Элементы `direct` и `action` не могут использоваться одновременно. Элемент `direct` может сопровождаться элементом `reverse`, для указания ручного обращения действия.

## Примеры

```

<step
  id="newMax"
  description="Обновление максимума"
  comment-ru="Обновляем текущий максимум"
  comment-en="Updating current maximum"
>
  <draw>
    @visualizer.updateArray(@i, 2);
  </draw>
  <direct>
    stack.pushInteger(@max);
    @max = @a[@i];
  </direct>
  <reverse>
    @max = stack.popInteger();
  </reverse>
</step>

<step
  id="newMax"
  description="Обновление максимума"
  comment-ru="Обновляем текущий максимум"
  comment-en="Updating current maximum"
>
  <draw>
    @visualizer.updateArray(@i, 2);
  </draw>
  <action>

```

```
@max @= @a[@i];
</action>
</step>
```

### Описание действий (теги action, direct и reverse)

Теги action, direct и reverse служат для описания действий, выполняемых в простом шаге. В теге action описываются действия, которые будут обращены в автоматическом режиме (см. раздел :TODO:). В тегах direct и reverse описываются действия, выполняемые на прямом и обратном проходе соответственно.

### DTD

```
<!ELEMENT direct (#PCDATA)*>
<!ATTLIST direct>

<!ELEMENT reverse (#PCDATA)*>
<!ATTLIST reverse>

<!ELEMENT action (#PCDATA)*>
<!ATTLIST action>
```

### XML-схема

```
<xsd:simpleType name="action">
  <xsd:list itemType="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="direct">
  <xsd:list itemType="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="reverse">
  <xsd:list itemType="xsd:string"/>
</xsd:simpleType>
```

### Атрибуты

Нет

### Содержимое

Текст — описание действий на языке java (см. раздел :TODO:).

### Примеры

```
<action>
  @i @= @i + 1;
</action>

<direct>
  @i++;
</direct>

<reverse>
  @i--;
</reverse>
```

### Вызов вложенного автомата (тег call-auto)

Тег call-auto служит для вызова вложенного автомата (процедуры).

### DTD

```
<!ELEMENT call-auto EMPTY>
<!ATTLIST call-auto
  id IDREF #REQUIRED
>
```

### XML-схема

```
<xsd:complexType name="call-auto">
  <xsd:attribute name="id" type="id" use="required"/>
</xsd:complexType>
```

```
</xsd:complexType>
```

## Атрибуты

Атрибут	#	И	Описание
id	1		Идентификатор вызываемого автомата

## Содержимое

Нет

## Примеры

```
<call-auto id="factorial"/>
```

### Оператор выбора (теги if, then, else)

Тег if служит для описания оператора ветвления. Он должен содержать элемент then, в котором описываются действия, выполняемые при истинности условия. Так же он может содержать элемент else, в котором описываются действия, выполняемые при ложности условия.

Проверяемое условие задается атрибутом test. Условие, проверяемое при обратном проходе (“обратное условие”) задается не обязательным атрибутом rtest.

## DTD

```
<!ELEMENT if (draw?, then, else?)>
<!ATTLIST if
  id          NMTOKEN          #IMPLIED
  description CDATA            #REQUIRED
  test        CDATA            #REQUIRED
  rtest       CDATA            #IMPLIED
  level       CDATA            "0"
  true-comment-ru CDATA        #IMPLIED
  true-comment-en CDATA        #IMPLIED
  false-comment-ru CDATA        #IMPLIED
  false-comment-en CDATA        #IMPLIED
  comment-args CDATA            #IMPLIED
>
```

```
<!ELEMENT then (%steps;)*>
<!ATTLIST then>
```

```
<!ELEMENT else (%steps;)*>
<!ATTLIST else>
```

## XML-схема

```
<xsd:complexType name="if">
  <xsd:sequence>
    <xsd:element
      name      = "draw"
      type      = "draw"
      minOccurs = "0"
    />
    <xsd:element
      name      = "then"
      type      = "then"
    />
    <xsd:element
      name      = "else"
      type      = "else"
      minOccurs = "0"
    />
  </xsd:sequence>
  <xsd:attribute name = "id"          type = "id"/>
  <xsd:attribute name = "description" type = "description"/>
  <xsd:attribute name = "rtest"       type = "test"/>
  <xsd:attribute name = "test" type = "test" use = "required"/>
  <xsd:attribute name = "level" type = "level" default = "0"/>
  <xsd:attributeGroup ref="choice-comment"/>
</xsd:complexType>
```

```

</xsd:complexType>

<xsd:complexType name="then">
  <xsd:group
    ref          = "automata-steps"
    minOccurs    = "0"
    maxOccurs    = "unbounded"
  />
</xsd:complexType>

<xsd:complexType name="else">
  <xsd:group
    ref          = "automata-steps"
    minOccurs    = "0"
    maxOccurs    = "unbounded"
  />
</xsd:complexType>

```

## Атрибуты

Для тега if:

Атрибут	#	И	Описание
id	1		Идентификатор шага
description	1		Описание шага
test	1		Условие
rtest	1		Обращенное условие
level	?		Уровень шага
true-comment	1	+	Комментарий, отображаемый, когда условие выполняется
false-comment	1	+	Комментарий, отображаемый, когда условие не выполняется
comment-args	?		Аргументы комментариев

У тегов then и else атрибуты отсутствуют.

## Содержимое

Для тега if:

Тег	#	Описание
draw	?	Метод отображения состояния
then	1	Действия, выполняемые при истинности условия
ekse	?	Действия, выполняемые при ложности условия

Теги тегов then и else могут содержать описания произвольных шагов (step, call-auto, if и while).

## Примеры

```

<if
  id="Cond"
  description="Условие"
  test="@max < @a[@i]"
  true-comment-ru="{0} больше текущего максимума ({1})"
  true-comment-en="{0} greater than current maximum ({1})"
  false-comment-ru="{0} не больше текущего максимума ({1})"
  false-comment-en="{0} not greater than current maximum ({1})"
  comment-args="new Integer(@a[@i]), new Integer(@max)"
>
  <draw>visualizer.updateArray(@i, 1); draw<
  <then> ... </then>
  <else> ... </else>
</if>

```

## Цикл с предусловием (тег while)

Тег while служит для описания циклов с предусловием. Проверяемое условие задается атрибутом test. Условие, проверяемое при обратном проходе (“обратное условие”) задается не обязательным атрибутом rtest.

### DTD

```
<!ELEMENT while (draw?, (%steps;)+)>
<!ATTLIST while
  id                NMTOKEN      #IMPLIED
  description       CDATA        #REQUIRED
  test              CDATA        #REQUIRED
  rtest            CDATA        #IMPLIED
  level            CDATA        "0"
  true-comment-ru  CDATA        #IMPLIED
  true-comment-en  CDATA        #IMPLIED
  false-comment-ru CDATA        #IMPLIED
  false-comment-en CDATA        #IMPLIED
  comment-args     CDATA        #IMPLIED
>
```

### XML-схема

```
<xsd:complexType name="while">
  <xsd:group
    ref          = "automata-steps"
    maxOccurs    = "unbounded"
  />
  <xsd:attribute name = "id"          type = "id"/>
  <xsd:attribute name = "description" type = "description"/>
  <xsd:attribute name = "test"       type = "test"          use = "required"/>
  <xsd:attribute name = "rtest"      type = "test"/>
  <xsd:attribute name = "level"      type = "level"         default = "0"/>
  <xsd:attributeGroup ref="choice-comment"/>
</xsd:complexType>
```

### Атрибуты

Атрибут	#	И	Описание
id	1		Идентификатор шага
description	1		Описание шага
test	1		Условие
rtest	1		Обращенное условие
level	?		Уровень шага
true-comment	1	+	Комментарий, отображаемый, когда условие выполняется
false-comment	1	+	Комментарий, отображаемый, когда условие не выполняется
comment-args	?		Аргументы комментариев

### Содержимое

Тег	#	Описание
draw	?	Метод отображения состояния
steps	*	Действия, выполняемые при истинности условия

### Пример

```
<while
  id="Loop"
  description="Цикл"
  test="@i < @a.length"
  level="-1"
> ... </while>
```



## 4. Описание конфигурации визуализатора

Корневым элементом конфигурации является тег `configuration`. Конфигурация может содержать следующие элементы:

1. Описание элементов интерфейса
  - Описание панели (тег `panel`)
  - Описание кнопки (тег `button`)
  - Описание новой панели выбора (тег `adjustable-panel`)
  - Описание старой панели выбора (тег `spin-panel`)
2. Описания таблиц стилей
  - Описание стиля (тег `style`)
  - Описание таблицы стилей (`style-set`)
  - Описание шрифта (тег `font`)
  - Описание цвета (тег `color`)
3. Описания групп, свойств и сообщений
  - Описание группы (тег `group`)
  - Описание свойства (тег `property`)
  - Описание сообщений (тег `message`)

### Описание конфигурации визуализатора (тег `configuration`)

Тег `configuration` является контейнером для элементов, описывающих конфигурацию визуализатора.

#### DTD

```
<!ELEMENT configuration (%elements;)*>
<!ATTLIST configuration>
```

#### XML-схема

```
<xsd:complexType name="configuration">
  <xsd:group
    ref          = "elements"
    minOccurs    = "0"
    maxOccurs    = "unbounded"
  />
</xsd:complexType>

<xsd:group name="elements">
  <xsd:choice>
    <xsd:element name = "font"           type = "font"           />
    <xsd:element name = "color"          type = "color"          />
    <xsd:element name = "property"       type = "property"      />
    <xsd:element name = "message"        type = "message"       />
    <xsd:element name = "panel"          type = "panel"         />
    <xsd:element name = "style"          type = "style"         />
    <xsd:element name = "button"         type = "button"        />
    <xsd:element name = "spin-panel"     type = "spin-panel"    />
    <xsd:element name = "styleset"       type = "styleset"      />
    <xsd:element name = "group"          type = "group"         />
    <xsd:element name = "choice"         type = "choice"        />
  </xsd:choice>
</xsd:group>
```

## Атрибуты

Нет

## Содержимое

Вложенные элементы: любые конфигурационные элементы

## Пример

```
<configuration> ... </configuration>
```

### 4.1. Описание элементов интерфейса

#### Описание панели (тег panel)

Тег panel предназначен для описания вида панелей (наследников класса java.awt.Panel).

## DTD

```
<!ELEMENT panel (font)>
<!ATTLIST panel
  description      CDATA          #IMPLIED
  param            NMTOKEN       #REQUIRED
  foreground       CDATA          #REQUIRED
  background       CDATA          #REQUIRED
>
```

## XML-схема

```
<xsd:complexType name="panel">
  <xsd:sequence>
    <xsd:element name="font" type="font"/>
  </xsd:sequence>
  <xsd:attribute name="description" type="xsd:string"/>
  <xsd:attribute name="param" type="xsd:string" use="required"/>
  <xsd:attribute name="foreground" type="color-type" use="required"/>
  <xsd:attribute name="background" type="color-type" use="required"/>
</xsd:complexType>
```

## Атрибуты

Атрибут	#	Описание
description	?	Описание конфигурируемой панели
param	1	Имя параметра для записи конфигурации
foreground	1	Основной цвет (цвет надписей)
background	1	Цвет фона

## Содержимое

Тег	#	Описание
font	1	Шрифт надписей

## Пример

```
<panel
  description = "Конфигурация клиентской области"
  param      = "client"
  foreground = "000000"
  background = "ffffff"
>
  <font face="Serif" size="14" style="plain"/>
</panel>
```

#### Описание кнопки (тег button)

Тег button предназначен для описания кнопок (наследников класса java.awt.Button).

## DTD

```
<!ELEMENT button EMPTY>
<!ATTLIST button
  description      CDATA          #IMPLIED
  param            NMTOKEN       #REQUIRED
  caption-ru      CDATA          #REQUIRED
>
```

```
caption-en          CDATA          #REQUIRED
hint-ru            CDATA          #REQUIRED
hint-en            CDATA          #REQUIRED
```

>

## XML-схема

```
<xsd:complexType name="button">
  <xsd:attribute name="description" type="xsd:string"/>
  <xsd:attribute name="param" type="xsd:string" use="required"/>
  <xsd:attribute name="caption-ru" type="xsd:string" use="required"/>
  <xsd:attribute name="caption-en" type="xsd:string" use="required"/>
  <xsd:attribute name="hint-ru" type="xsd:string" use="required"/>
  <xsd:attribute name="hint-en" type="xsd:string" use="required"/>
</xsd:complexType>
```

## Атрибуты

Атрибут	#	И	Описание
description	?		Описание конфигурируемой кнопки
param	1		Имя параметра для записи конфигурации
caption	1	+	Надпись на кнопке
hint	1	+	Подсказка для кнопки

## Содержимое

Нет

## Пример

```
<button
  description = "Next-button"
  param       = "next"
  caption-ru  = ">>"
  caption-en  = ">>"
  hint-ru     = "Шаг вперед"
  hint-en     = "Step forward"
/>
```

### Описание новой панели выбора (тег adjustable-panel) :TODO:

### Описание старой панели выбора (тег spin-panel)

Старая панель выбора сохранена для обратной совместимости. В новых проектах следует использовать новые панели выбора (adjustable-panel).

Тег spin-panel описывает конфигурацию панели выбора. Для дополнительной информации смотри раздел :TODO:).

## DTD

```
<!ELEMENT spin-panel (button, button)>
<!ATTLIST spin-panel
  description          CDATA          #IMPLIED
  param                NMTOKEN       #REQUIRED
  caption-ru           CDATA          #REQUIRED
  caption-en           CDATA          #REQUIRED
  hint-ru              CDATA          #REQUIRED
  hint-en              CDATA          #REQUIRED
  value                CDATA          #REQUIRED
  max-value            CDATA          #REQUIRED
  min-value            CDATA          #REQUIRED
  step                 CDATA          #REQUIRED
```

>

## XML-схема

```
<xsd:complexType name="spin-panel">
  <xsd:sequence>
    <xsd:element
      name       = "button"
      type       = "button"
      minOccurs = "2"
```

```

        maxOccurs    = "2"
    />
</xsd:sequence>
<xsd:attribute name = "description"    type = "xsd:string"/>
<xsd:attribute name = "param"         type = "xsd:string" use = "required"/>
<xsd:attribute name = "caption-ru"    type = "xsd:string" use = "required"/>
<xsd:attribute name = "caption-en"    type = "xsd:string" use = "required"/>
<xsd:attribute name = "hint-ru"       type = "xsd:string" use = "required"/>
<xsd:attribute name = "hint-en"       type = "xsd:string" use = "required"/>
<xsd:attribute name = "value"         type = "xsd:double" use = "required"/>
<xsd:attribute name = "max-value"     type = "xsd:double" use = "required"/>
<xsd:attribute name = "min-value"     type = "xsd:double" use = "required"/>
<xsd:attribute name = "step"         type = "xsd:double" use = "required"/>
</xsd:complexType>

```

## Атрибуты

Атрибут	#	И	Описание
description	?		Описание конфигурируемой панели
param	1		Имя параметра для записи конфигурации
caption	1	+	Шаблон надписи на панели
hint	1	+	Подсказка для панели
value	1		Начальное значение
min-value	1		Минимальное значение
max-value	1		Максимальное значение
step	1		Шаг изменения значения

## Содержимое

Тег	#	Описание
button	2	Конфигурация кнопок увеличения/уменьшения значений.

## Пример

```

<spin-panel
  description = "'Delay' button"
  param      = "delay"
  caption-ru = "Задержка: {0,number,####}"
  caption-en = "Delay: {0,number,####}"
  hint-ru    = "Задержка между шагами\nв автоматическом режиме"
  hint-en    = "Delay between steps\nin automated mode"
  value      = "1000" step = "100"
  min-value  = "100"max-value  = "5000"
>
  <button
    param      = "button-less"
    caption-ru = "&lt;&lt;" hint-ru = "Уменьшить задержку"
    caption-en = "&lt;&lt;" hint-en = "Decrease delay"
  />
  <button
    param      = "button-more"
    caption-ru = "&gt;&gt;" hint-ru = "Увеличить задержку"
    caption-en = "&gt;&gt;" hint-en = "Increase delay"
  />
</spin-panel>

```

## 4.2. Таблицы стилей

### Описание стиля (тег style) :TODO:

## DTD

```

<!ELEMENT style (font?)>
<!ATTLIST style
  description      CDATA          #IMPLIED
  param            NMTOKEN       #IMPLIED
  text-color       CDATA          #IMPLIED
  text-align       CDATA          #IMPLIED
  message-align    CDATA          #IMPLIED

```

```

border-color          CDATA          #IMPLIED
border-status        %boolean;       #IMPLIED
fill-color           CDATA          #IMPLIED
fill-status          %boolean;       #IMPLIED
aspect               CDATA          #IMPLIED
aspect-status        %boolean;       #IMPLIED
padding              CDATA          #IMPLIED

```

>

## XML-схема

```

<xsd:complexType name="style">
  <xsd:sequence>
    <xsd:element name = "font" type = "font" minOccurs = "0"/>
  </xsd:sequence>
  <xsd:attribute name = "description" type = "xsd:string"/>
  <xsd:attribute name = "param" type = "xsd:string"/>
  <xsd:attribute name = "text-color" type = "color-type"/>
  <xsd:attribute name = "text-align" type = "align"/>
  <xsd:attribute name = "message-align" type = "align"/>
  <xsd:attribute name = "border-color" type = "color-type"/>
  <xsd:attribute name = "border-status" type = "xsd:boolean"/>
  <xsd:attribute name = "fill-color" type = "color-type"/>
  <xsd:attribute name = "fill-status" type = "xsd:boolean"/>
  <xsd:attribute name = "aspect" type = "positiveRealNumber"/>
  <xsd:attribute name = "aspect-status" type = "xsd:boolean"/>
  <xsd:attribute name = "padding" type = "positiveRealNumber"/>
</xsd:complexType>

```

## Атрибуты

Атрибут	#	И	Описание
description	?		Описание стиля
param	1		Имя параметра для записи конфигурации
text-color	?	+	Цвет текста
text-align	?	+	Выравнивание текста в сообщении
message-align	?		Выравнивание сообщения относительно фигуры
border-color	?		Цвет рамки
border-status	?		Отображать ли рамку
fill-color	?		Цвет фона
fill-status	?		Отображать ли фон
aspect	?		Отношение длины к ширине
aspect-status	?		Использовать ли отношение длины к ширине
padding	?		Отступ от границы до текста (в пикселях)

Выравнивание текста сообщения и сообщения задаются числом в диапазоне 0..1. При этом, если выравнивание равно  $a$ , то свободное место будет распределено на лево и на право в пропорции  $a:(1-a)$ .

Таким образом:

- 0 — выравнивание по левому краю;
- 0.5 — выравниванию по центру;
- 1 — выравнивание по правому краю.

## Содержимое

Тег	#	Описание
font	1	Шрифт надписей

## Пример

```

<style
  description      = "Обычная ячейка"
  text-color       = "000000"
  text-align       = "0.5"
  border-color     = "000000"
  border-status    = "true"
  fill-color       = "8080ff"

```

```

fill-status      = "true"
aspect-status    = "false"
padding          = "0.2"
>
<font face="Serif" size="12" style="plain"/>
</style>

```

### Описание таблицы стилей (style-set)

Таблица стилей задает набор стилей (тег style), используемых для отображения одного элемента в зависимости от состояния. Если в стиле не определены некоторые свойства, то их значения берутся из описания первого стиля в наборе. Стили нумеруются, начиная с 0.

### DTD

```

<!ELEMENT styleset (style*)>
<!ATTLIST styleset
  description      CDATA          #IMPLIED
  param            NMTOKEN       #REQUIRED
>

```

### XML-схема

```

<xsd:complexType name="styleset">
  <xsd:sequence>
    <xsd:element
      name      = "style"
      type      = "styleInStyleset"
      minOccurs = "0"
      maxOccurs = "unbounded"
    />
  </xsd:sequence>
  <xsd:attribute name = "description" type = "xsd:string"/>
  <xsd:attribute name = "param"      type = "xsd:string" use = "required"/>
</xsd:complexType>

```

### Атрибуты

Атрибут	#	И	Описание
description	?		Описание таблицы стилей
param	1		Имя параметра для записи конфигурации

### Содержимое

Тег	#	Описание
style	1	Стили в таблице

### Пример

```

<styleset
  description = "Таблица стилей для ячеек массива"
  param      = "array"
> ... </styleset>

```

### Описание шрифта (тег font)

Тег font служит для описания типа и размера шрифта, обычно он вложен в описание других элементов.

### DTD

```

<!ELEMENT font EMPTY>
<!ATTLIST font
  description      CDATA          #IMPLIED
  param            NMTOKEN       "font"
  face             %fonts;       #IMPLIED
  size             CDATA          #IMPLIED
  style            %font-style;  #IMPLIED
>

```

### XML-схема

```

<xsd:complexType name="font">
  <xsd:attribute name = "description" type = "xsd:string"/>

```

```

<xsd:attribute name = "param"           type = "xsd:string"       default="font"/>
<xsd:attribute name = "face"            type = "font-name"/>
<xsd:attribute name = "size"            type = "xsd:positiveInteger"/>
<xsd:attribute name = "style"           type = "font-style"/>
</xsd:complexType>

<xsd:simpleType name="font-name">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Serif"/>
    <xsd:enumeration value="SansSerif"/>
    <xsd:enumeration value="Symbol"/>
    <xsd:enumeration value="Monospaced"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="font-style">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="plain"/>
    <xsd:enumeration value="bold"/>
    <xsd:enumeration value="italic"/>
    <xsd:enumeration value="bolditalic"/>
  </xsd:restriction>
</xsd:simpleType>

```

## Атрибуты

Атрибут	#	И	Описание
description	?		Описание шрифта
param	1		Имя параметра для записи конфигурации
face	1		Начертание шрифта
size	1		Размер шрифта в пунктах
style	1		Стиль шрифта

Определены следующие начертания шрифтов:

- **Serif** — с засечками (Times New Roman, Roman).
- **SansSerif** — без засечек (Arial, Helvetica).
- **Symbol** — для специальных символов (Symbol).
- **Monospaced** — моноширинной (Courier New, Courier New).

Для стиля шрифта определены следующие значения:

- **Plain** — обычное начертание.
- **Bold** — полужирное начертание.
- **Italic** — курсивное начертание.
- **BoldItalic** — полужирный курсив.

## Содержимое

Нет

## Пример

```
<font face="Serif" size="12" style="plain"/>
```

### Описание цвета (тег color)

Тег color служит для описания цветов элементов, обычно он вложен в описание других элементов.

Цвет представляется в виде 6-значного шестнадцатеричного числа: RRGGBB.

## DTD

```

<!ELEMENT color EMPTY>
<!ATTLIST color

```

```

description          CDATA          #IMPLIED
param                NMTOKEN        #REQUIRED
value                CDATA          #REQUIRED

```

>

### XML-схема

```

<xsd:complexType name="color">
  <xsd:attribute name="description" type="xsd:string"/>
  <xsd:attribute name="param" type="xsd:string" use="required"/>
  <xsd:attribute name="value" type="color-type" use="required"/>
</xsd:complexType>

<xsd:simpleType name="color-type">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0-9a-fA-F]{6}"/>
  </xsd:restriction>
</xsd:simpleType>

```

### Атрибуты

Атрибут	#	И	Описание
description	?		Описание цвета
param	1		Имя параметра для записи конфигурации
value	1		Цвет

### Содержимое

Нет

### Пример

```
<color param="foreground" value="000000"/>
```

## 4.3. Группы, свойства и сообщения

Группы служат для объединения свойств и сообщений, имеющих общий смысл, на пример сообщения, отображаемые одним элементом управления. Так же группы могут использоваться для конфигурирования нестандартных компонент.

### Описание группы (тег group)

### DTD

```

<!ELEMENT group (%elements;)*>
<!ATTLIST group
  description          CDATA          #IMPLIED
  param                NMTOKEN        #REQUIRED

```

>

### XML-схема

```

<xsd:complexType name="group">
  <xsd:group
    ref          = "elements"
    minOccurs    = "0"
    maxOccurs    = "unbounded"
  />
  <xsd:attribute name="description" type="xsd:string"/>
  <xsd:attribute name="param" type="xsd:string" use="required"/>
</xsd:complexType>

```

### Атрибуты

Атрибут	#	И	Описание
description	?		Описание группы
param	1		Имя параметра для записи элементов группы

### Содержимое

Вложенные элементы: любые конфигурационные элементы



## Пример

```
<group
  description = "Конфигурация подсказок"
  param      = "hint"
> ... </group>
```

### Описание свойства (тег property)

Свойства описывают нестандартные конфигурационные компоненты, не зависящие от языка

## DTD

```
<!ELEMENT property EMPTY>
<!ATTLIST property
  description      CDATA          #IMPLIED
  param            NMTOKEN       #REQUIRED
  value           CDATA          #REQUIRED
>
```

## XML-схема

```
<xsd:complexType name="property">
  <xsd:attribute name="description" type="xsd:string"/>
  <xsd:attribute name="param" type="xsd:string" use="required"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>
```

## Атрибуты

Атрибут	#	И	Описание
description	?		Описание свойства
param	1		Имя параметра для записи свойства
value	1		Значение свойства

## Содержимое

Нет

## Пример

```
<property
  description = "Вертикальное расстояние от текста до границы"
  param      = "vgap"
  value      = "-1"
/>
```

### Описание сообщений (тег message)

Тег message служит для описания сообщений, выводимых пользователю. Обычно сообщения объединяются в группы.

## DTD

```
<!ELEMENT message EMPTY>
<!ATTLIST message
  description      CDATA          #IMPLIED
  param            NMTOKEN       #REQUIRED
  message-ru      CDATA          #REQUIRED
  message-en      CDATA          #REQUIRED
>
```

## XML-схема

```
<xsd:complexType name="message">
  <xsd:attribute name="description" type="xsd:string"/>
  <xsd:attribute name="param" type="xsd:string" use="required"/>
  <xsd:attribute name="message-ru" type="xsd:string" use="required"/>
  <xsd:attribute name="message-en" type="xsd:string" use="required"/>
</xsd:complexType>
```

## Атрибуты

Атрибут	#	И	Описание
description	?		Описание свойства
param	1		Имя параметра для записи свойства
message	1	+	Шаблон сообщения

## Содержимое

Нет

## Пример

```
<message  
  description = "Заголовок окна"  
  param      = "title"  
  message-ru = "О визуализаторе"  
  message-en = "About this visualizer"  
>
```

## Ссылки

1. Extensible Markup Language (XML) 1.0 (Second Edition) // <http://www.w3.org/TR/2000/REC-xml-20001006/>.
2. XML Schema Part 0: Primer // <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/>
3. XML Schema Part 1: Structures // <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>
4. XML Schema Part 2: Data Types // <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>
5. *B. Joy, G. Steele, J. Gosling, G. Bracha* Java Language Specification (Second Edition). Addison-Wesley. 2000. // <http://java.sun.com/docs/books/jls/>