НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

НИУ ВШЭ - Пермь

ПГНИУ

И.Д. Ермаков, К.А. Проскуряков, Л.Н. Лядова

Языково-ориентированный подход к разработке средств визуализации данных: автоматизация разработки DSL и генерация кода для интерактивной визуализации

ТРИС, 2024

Содержание

Актуальность исследования

Цель и задачи исследования

Классификация методов визуализации данных

Визуализация данных: подход, основанный на использовании онтологии

Обобщённая структура средств визуализации данных

Процесс разработки онтологии языков

Автоматизация разработки языков

Генерация кода для визуализации данных

Заключение

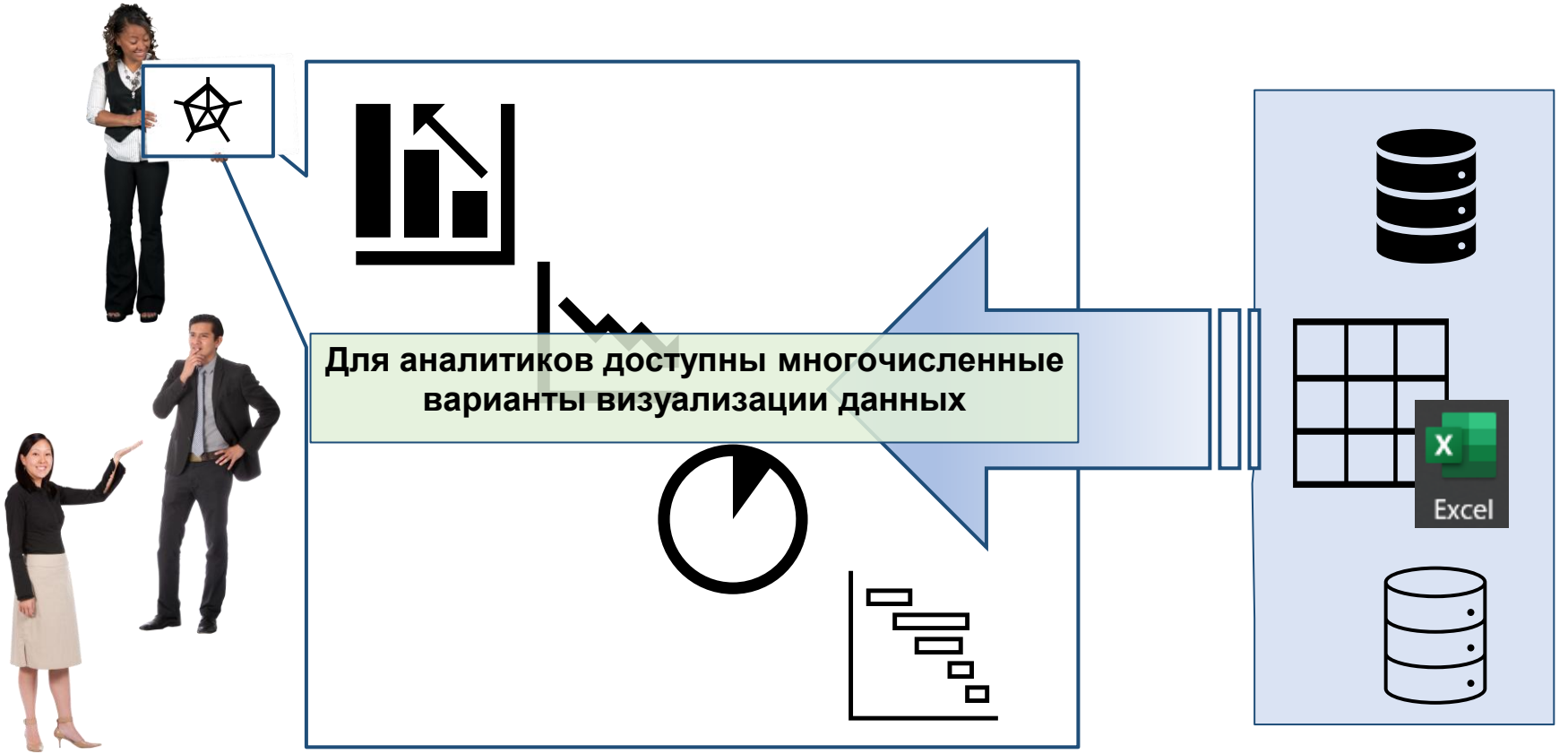
Приложения

Актуальность исследования

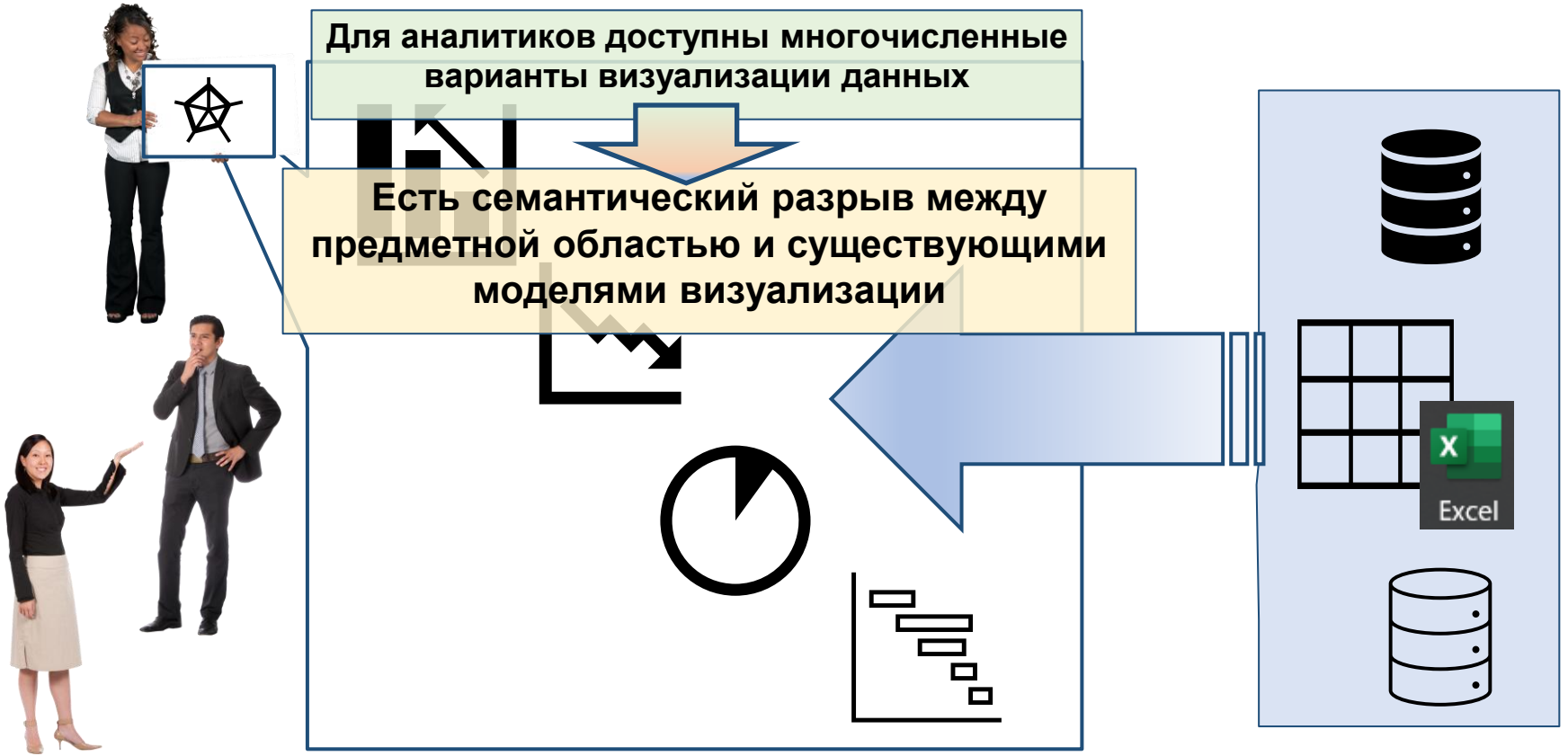
Инструменты визуализации данных как метода анализа данных получили широкое распространение в различных отраслях, бизнес-функциях и ИТ-дисциплинах, как в частном, так и в государственном секторе.

При этом потребности конечных пользователей (то есть аналитиков данных) включают необходимость создания пользовательских типов диаграмм для конкретных задач и предметных областей, поскольку базовые типы диаграмм с базовой геометрией могут ограничивать передачу информации и приводить к неэффективной визуализации, что, в свою очередь, может привести к ошибкам в принятии решений. Часто такая настройка требует использования языка программирования. А отсутствие глубоких знаний программирования у пользователей приводит к необходимости создания low-code платформ.

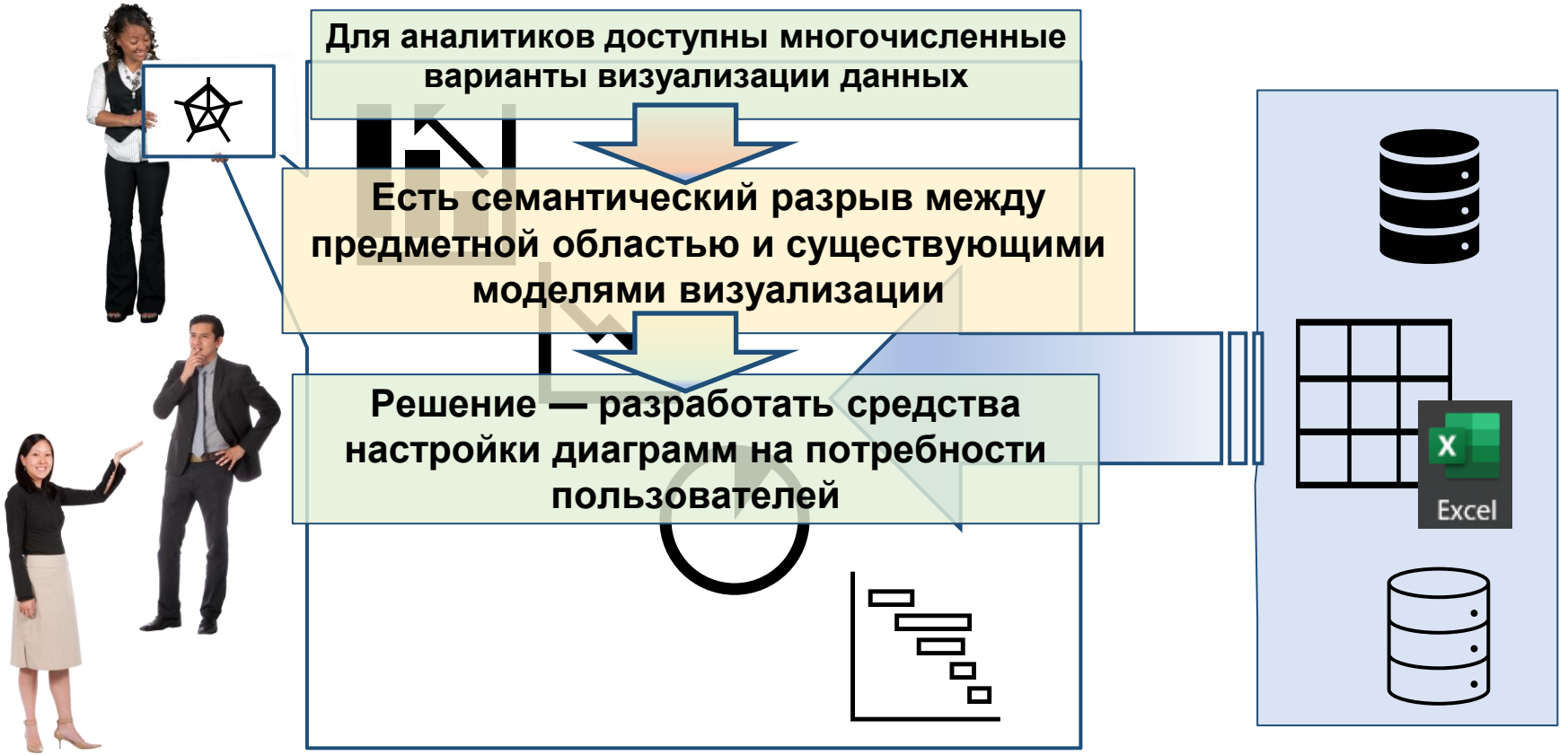
Актуальность исследования



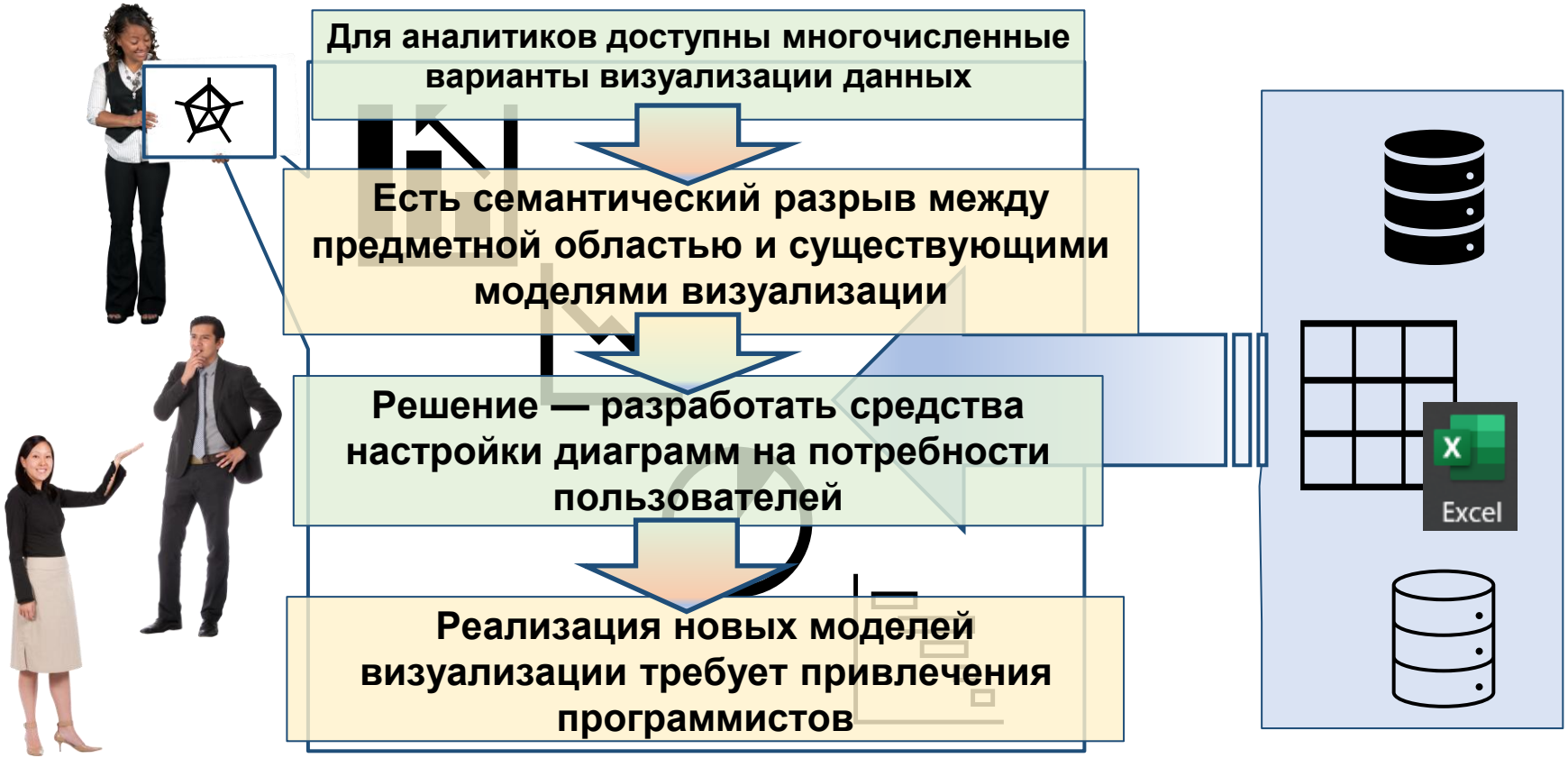
Актуальность исследования



Актуальность исследования



Актуальность исследования



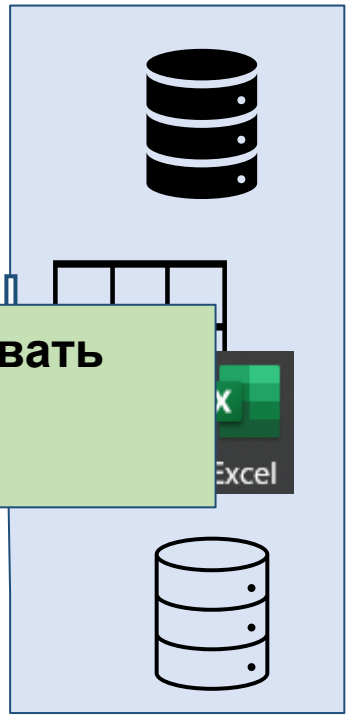
Актуальность исследования

Для аналитиков доступны многочисленные варианты визуализации данных

Есть семантический разрыв между предметной областью и существующими моделями визуализации

Предлагается использовать DSM-подход – разрабатывать специализированные языки с использованием DSM-платформы, основанной на знаниях

Реализация новых моделей визуализации требует привлечения программистов



Актуальность исследования

Доступные на данный момент системы инструментов визуализации данных можно разделить на следующие группы:

(1) электронные таблицы (например, Excel, Google Sheets),
(2) аналитические платформы (например, Microsoft Power BI, Tableau),
(3) редакторы диаграмм (например, Miro, ChartBlocks).

Стандартные инструменты первых двух групп ограничены базовыми типами диаграмм и возможностями настройки визуальных эффектов. Инструменты третьей группы позволяют создавать собственные визуализации, редактировать расположение элементов, но не предоставляют их настройки на предметные области.

Использование *языков программирования общего назначения* (например, библиотек Python для визуализации данных: Matplotlib, Seaborn, Plotly и др.) позволяет создавать выразительных визуализаций для решения конкретных задач, но требует глубоких знаний программирования от разработчиков диаграмм.

Актуальность исследования

Таким образом, конечные пользователи сталкиваются с двумя проблемами:

1. Как автоматизировать разработку визуализаций, чтобы снизить уровень требований к знаниям пользовательского программирования?
2. Как обеспечить настройку визуализаций для конкретных предметных областей?

Многообещающий подход к решению этих проблем предполагает использование многогранной онтологии.

Цель и задачи исследования

Цель – преодолеть ряд ограничений существующих средств визуализации и предоставить пользователям возможность настраивать методы визуализации данных для различных предметных областей, под специфику решаемых задач. При этом снизив требования к навыкам программирования конечного пользователя системы при создании диаграмм.

Данная цель может быть достигнута путем создания специальных языков – предметно-ориентированных языков (DSL) и средств автоматизации их разработки, управляемых знаниями.

В предыдущих работах был предложен подход к автоматизации создания DSL, основанный на использовании многоаспектной онтологии. С использованием данного подхода были разработаны языки описания структур данных, бизнес-процессов, алгоритмов управления…

В данном исследовании проведена апробация подходя для разработки моделей визуализации данных.

Цель и задачи исследования

Для реализации данного подхода необходимо было решить следующие ***задачи***:

1. Определить специфические требования пользователей к настройке визуализации данных для решаемых задач и предметных областей. Выявить проблемы, с которыми сталкиваются пользователи, чтобы устранить эти ограничения путем разработки DSL.
2. Проанализировать и классифицировать диаграммы визуализации данных, чтобы выявить основу для разработки языков визуализации данных и формализовать результаты для разработки онтологий.
3. Определить общую структуру системы визуализации данных.
4. Разработать онтологию языков визуализации данных на основе выполненной классификации диаграмм.
5. Описать базовую метамодель языка для разработки новых языков визуализации данных.
6. Привести пример кастомизации модели визуализации данных.
7. Описать подход к генерации кода для визуализации данных на основе созданных моделей.

Требования к средствам визуализации

Специфика *требований* пользователя заключается в:

- необходимости выявления новых типов диаграмм;
- создании визуализаций, адаптированных к специализированным задачам и областям;
- моделировании ранее созданных диаграмм и внедрении спецификаций визуализации в систему;
- быстром создании интерактивных визуализаций.

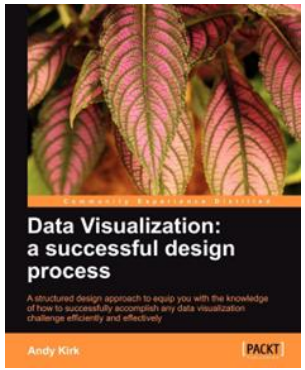
В области визуализации данных существуют *проблемы*:

- ограниченная степень гибкости в манипулировании элементами диаграмм;
- отсутствие фокусировки на реальных потребностях пользователя;
- потеря эффективности передачи данных при выборе неподходящего метода визуализации.

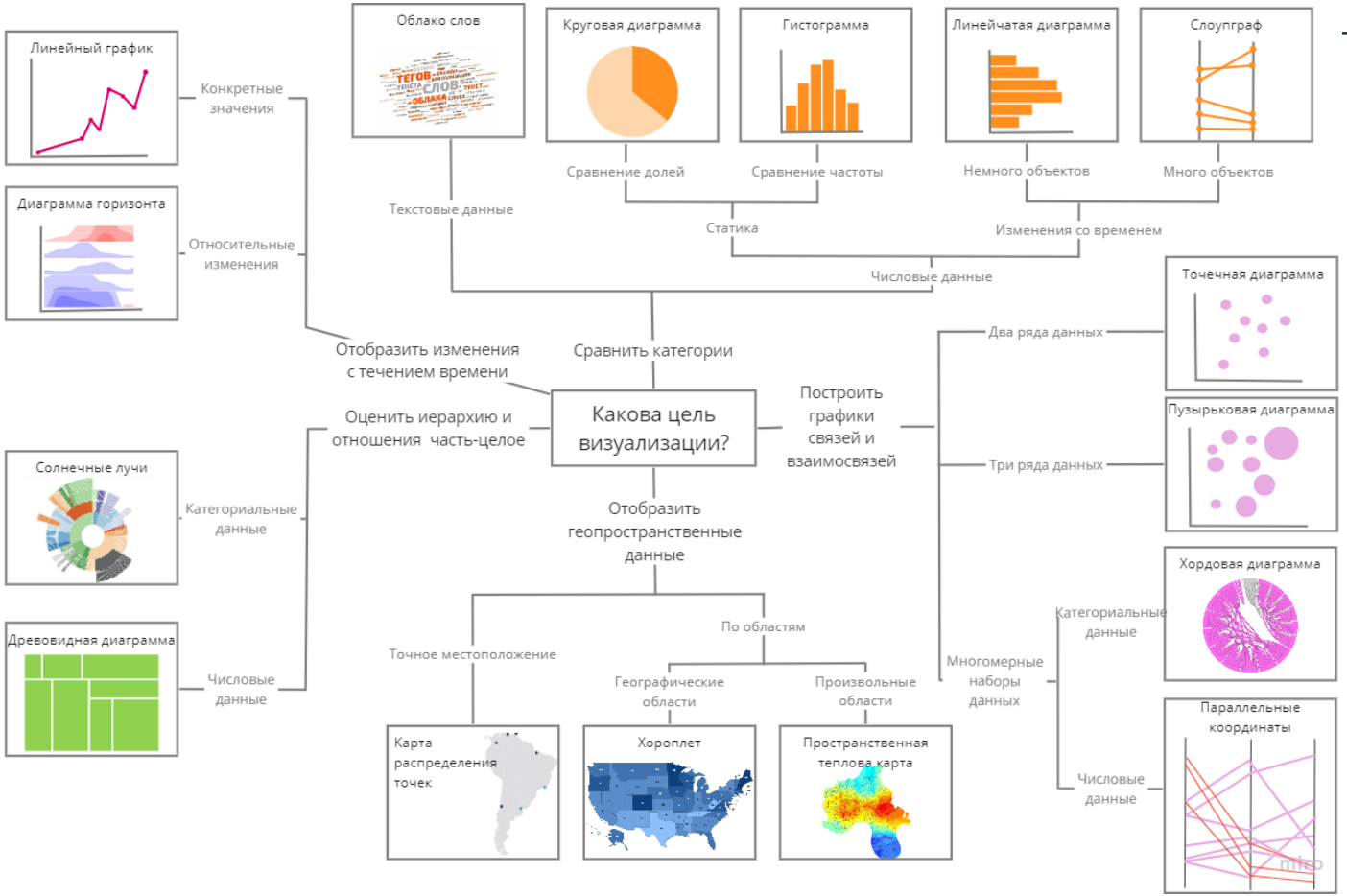
Классификация методов визуализации

Использована для классификации *таксономия методов визуализации* данных, предложенной Энди Кирком:

(1) сравнение категорий,
(2) оценка иерархий и отношений «часть-целое»,
(3) демонстрация изменений во времени,
(4) построение графиков связей и отношений,
(5) отображение геопространственных данных.



Классификация методов визуализации



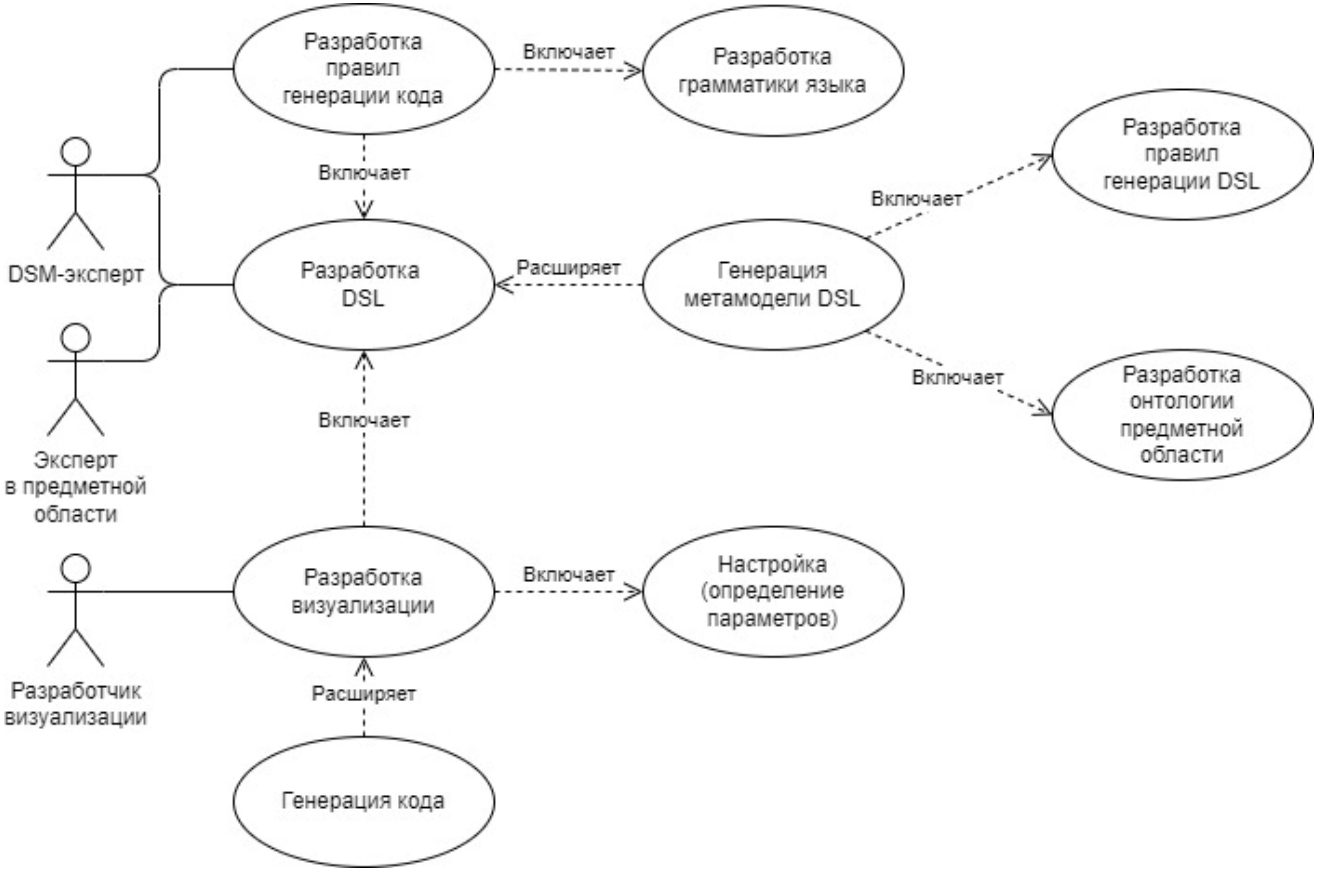
Визуализация данных – разработка моделей: подход, основанный на использовании онтологии

Публикаций, посвященных созданию DSL для визуализации данных, немного. Согласно проведенному обзору, большинство DSL ориентированы на небольшой набор стандартных диаграмм (например, на круговые диаграммы и гистограммы) или визуализацию конкретных типов данных (например, геопространственных).

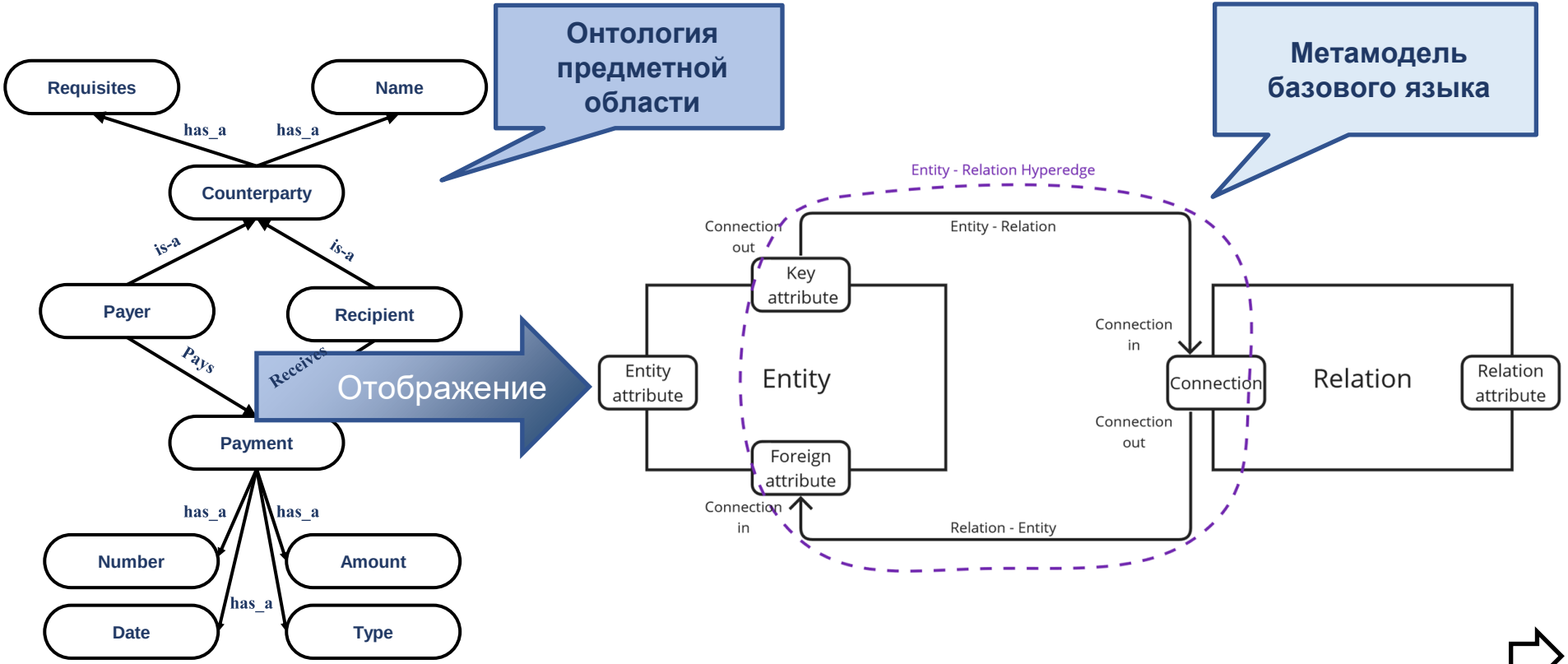
Языки различаются по уровням абстракции, контекстам использования и возможностям реализации.

Тем не менее, в рассмотренных инструментах адаптация диаграмм под конкретные предметные области не найдена. Исходя из этого, мы делаем вывод, что языково-ориентированный подход к реализации инструмента визуализации данных может стать основным при разработке системы визуализации данных

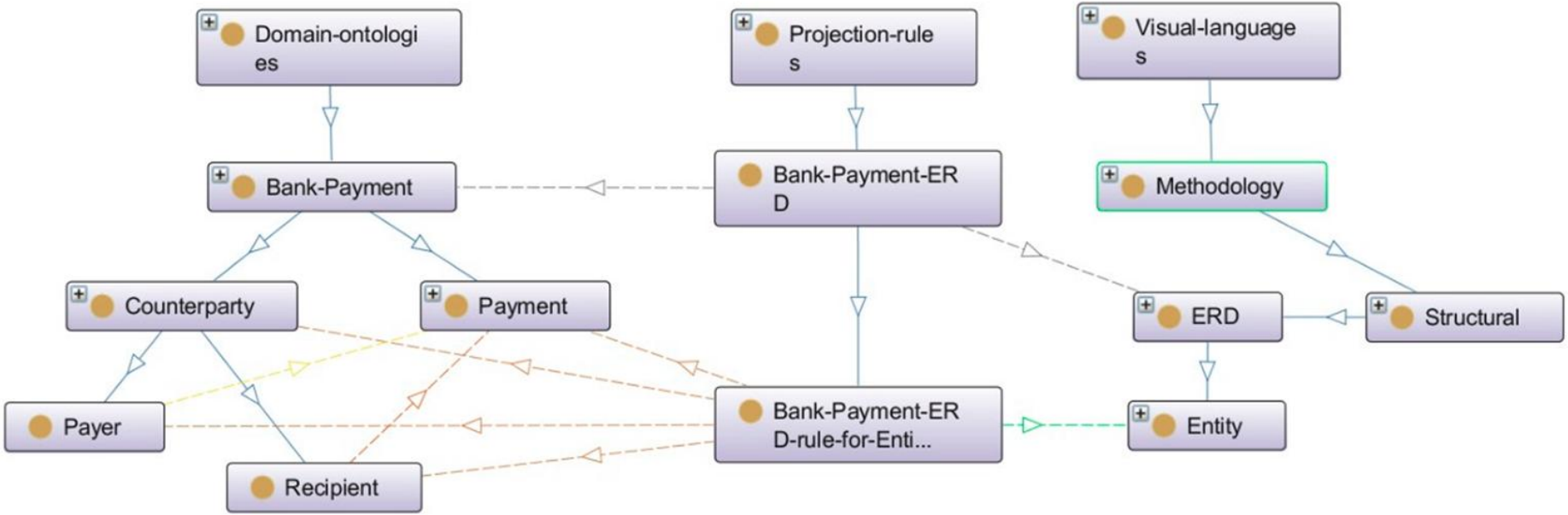
Визуализация данных – разработка моделей: подход, основанный на использовании онтологии



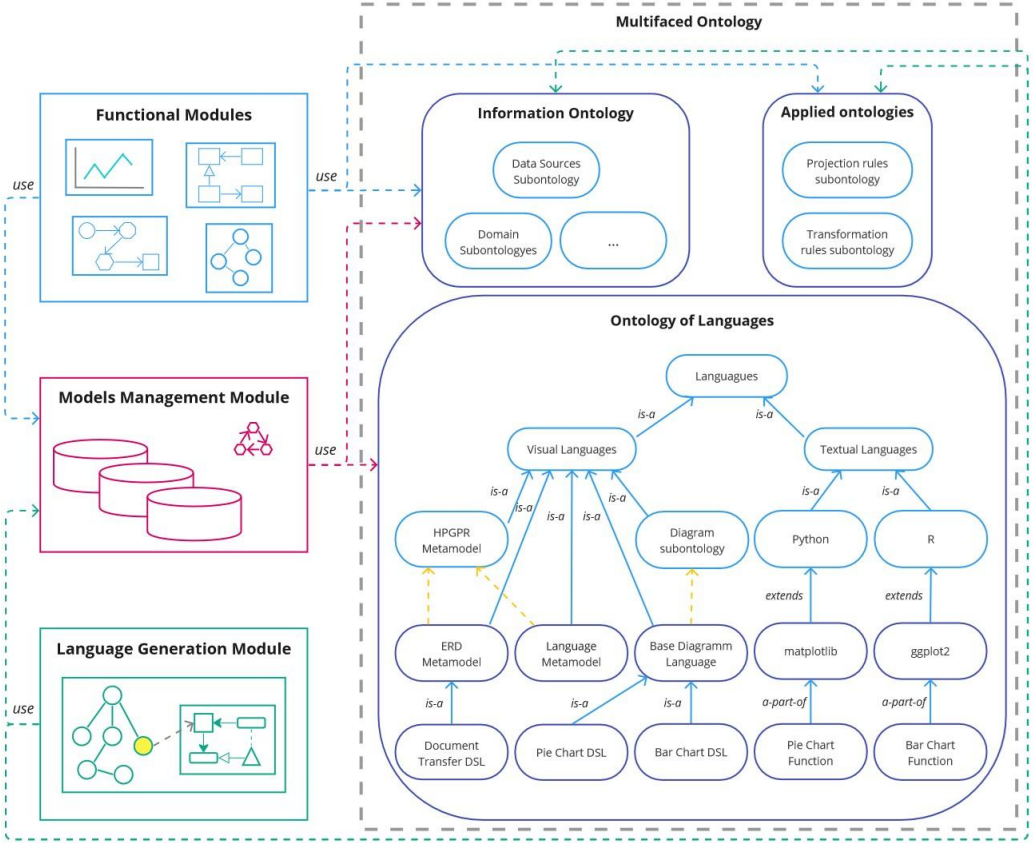
Визуализация данных – разработка моделей: подход, основанный на использовании онтологии



Визуализация данных – разработка моделей: подход, основанный на использовании онтологии

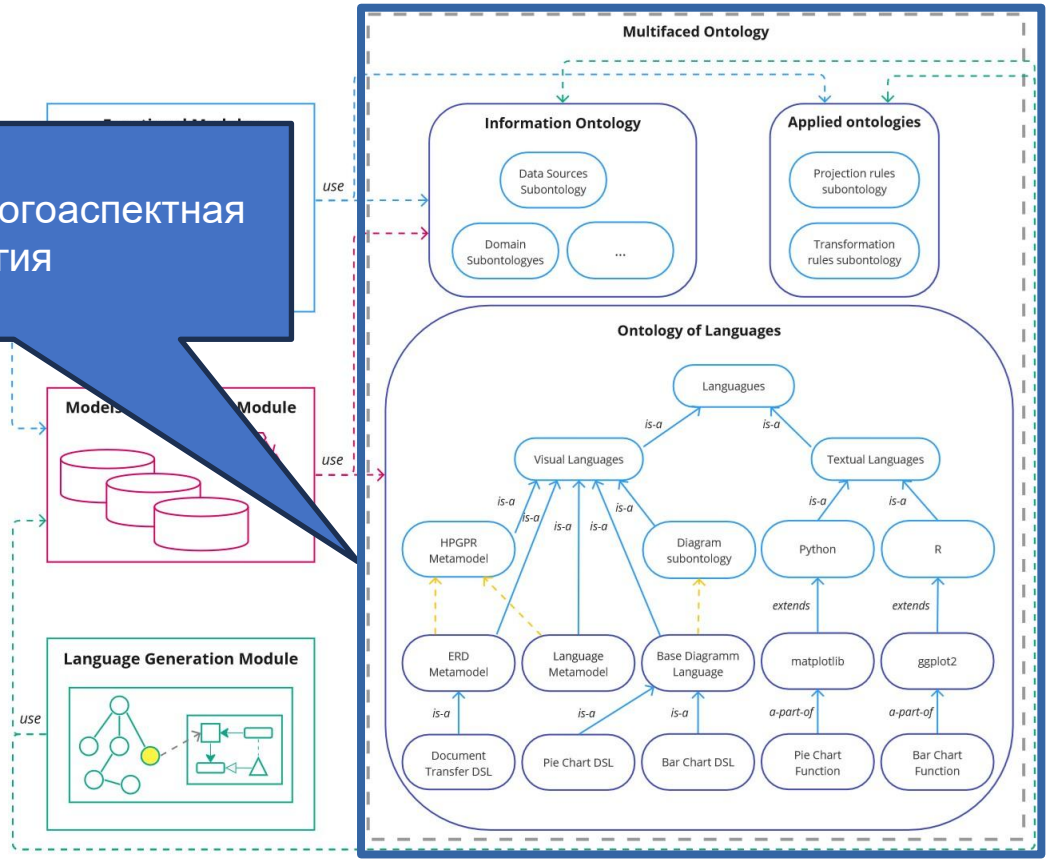


Обобщённая структура средств визуализации данных



Обобщённая структура средств визуализации данных

Ядро системы многоаспектная онтология

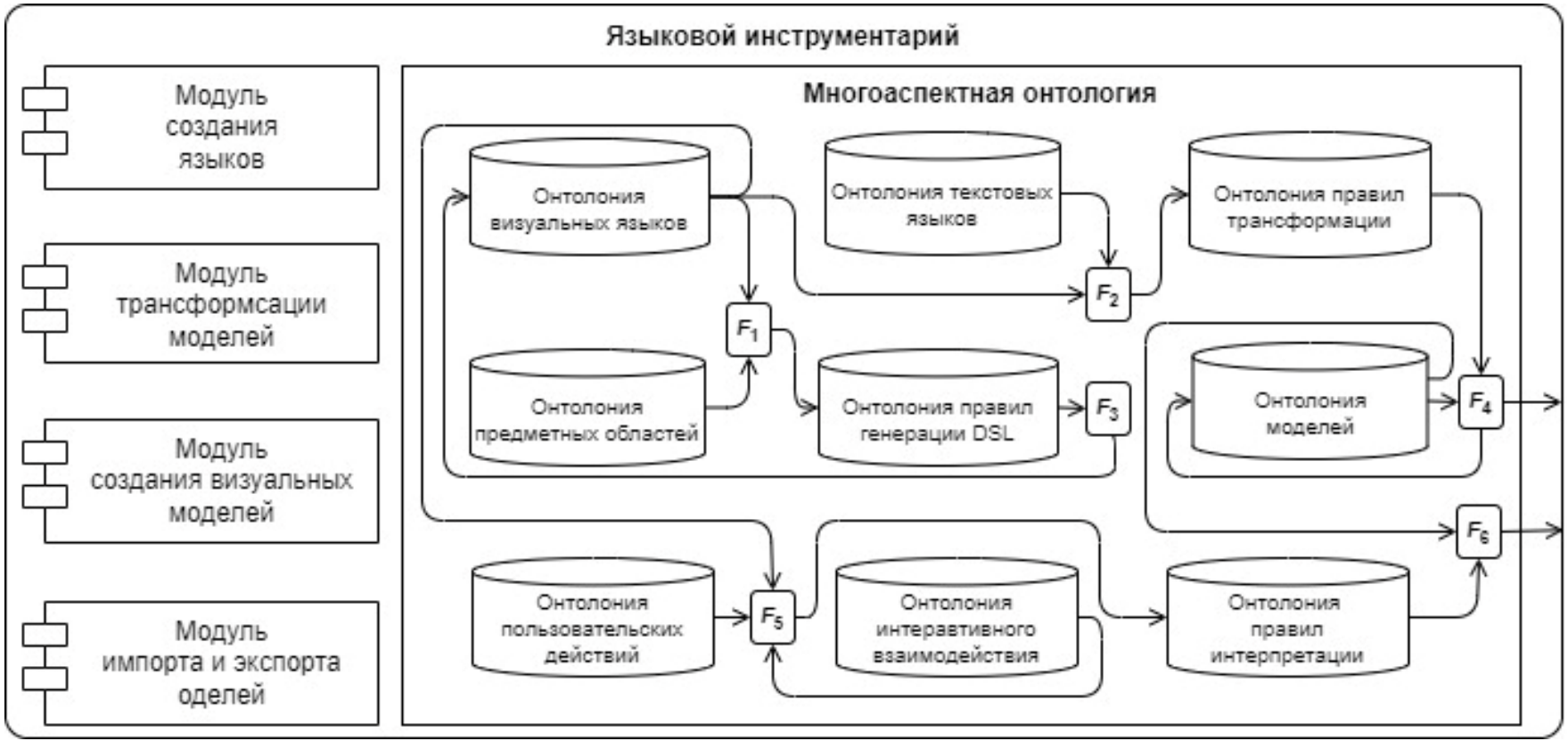


Обобщённая структура средств визуализации данных

Обобщённая структура средств визуализации данных

Обобщённая структура средств визуализации данных

Обобщённая структура средств визуализации данных



Процесс разработки онтологии языков

Процесс построения онтологии языков визуализации данных включает следующие этапы:

1. Формализация абстрактной диаграммы, которая будет включать свойства, общие для всех типов диаграмм (заголовок, легенда, ширина, высота и т. д.).
2. Выделение разных типов диаграмм в отдельные классы на основе разработанной ранее классификации диаграмм.
3. Добавление уникальных элементов диаграмм в классы.
4. Доопределение отношений между элементами.

Процесс разработки онтологии языков

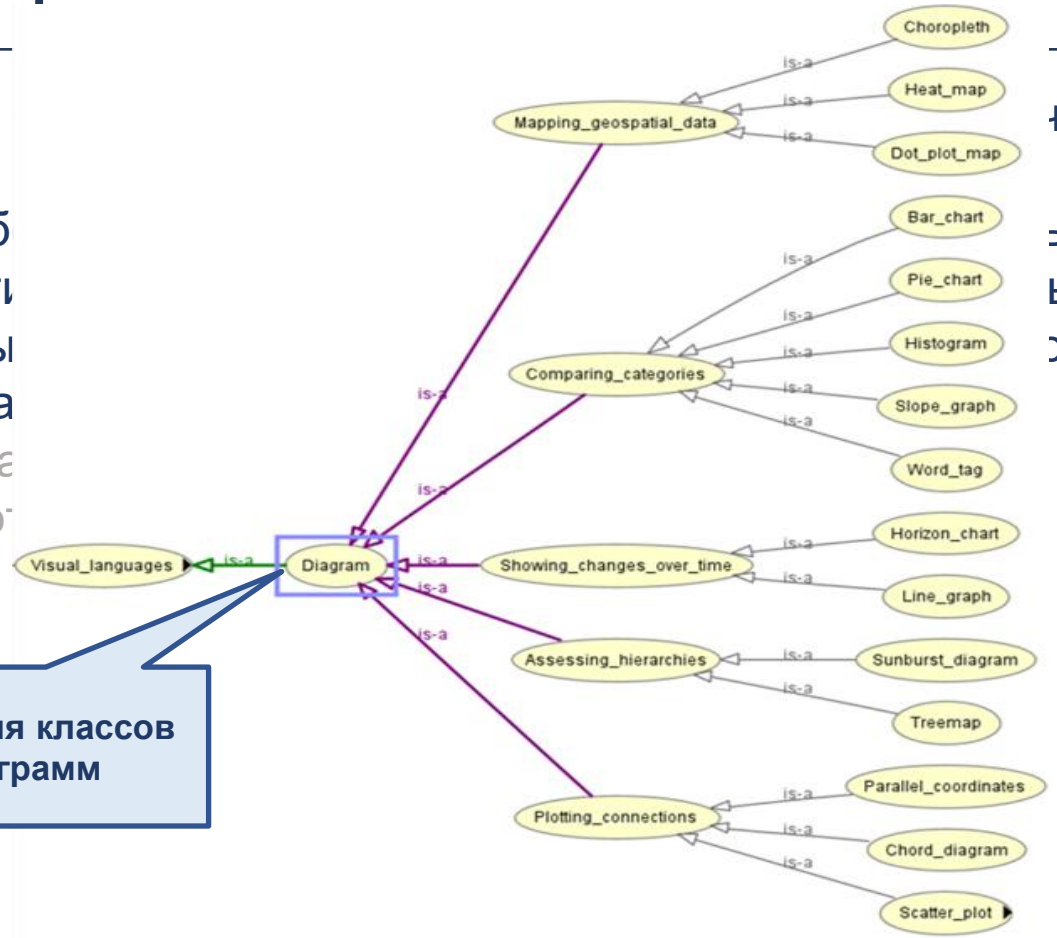
Процесс построения онтологии языков визуализации данных включает следующие этапы:

1. Формализация абстрактной диаграммы, которая будет включать свойства, общие для всех типов диаграмм (заголовок, легенда, ширина, высота и т. д.).
2. Выделение разных типов диаграмм в отдельные классы на основе разработанной ранее классификации диаграмм.
3. Добавление уникальных элементов диаграмм в классы.
4. Доопределение отношений между элементами.

Процесс разработки онтологии языков

Процесс построения нает
следующие этапы:

1. Формализация аб ь свойства,
   общие для всех ти ысота и т. д.).
2. Выделение разны ове
   разработанной ра
3. Добавление уника
4. Доопределение о



Иерархия классов диаграмм

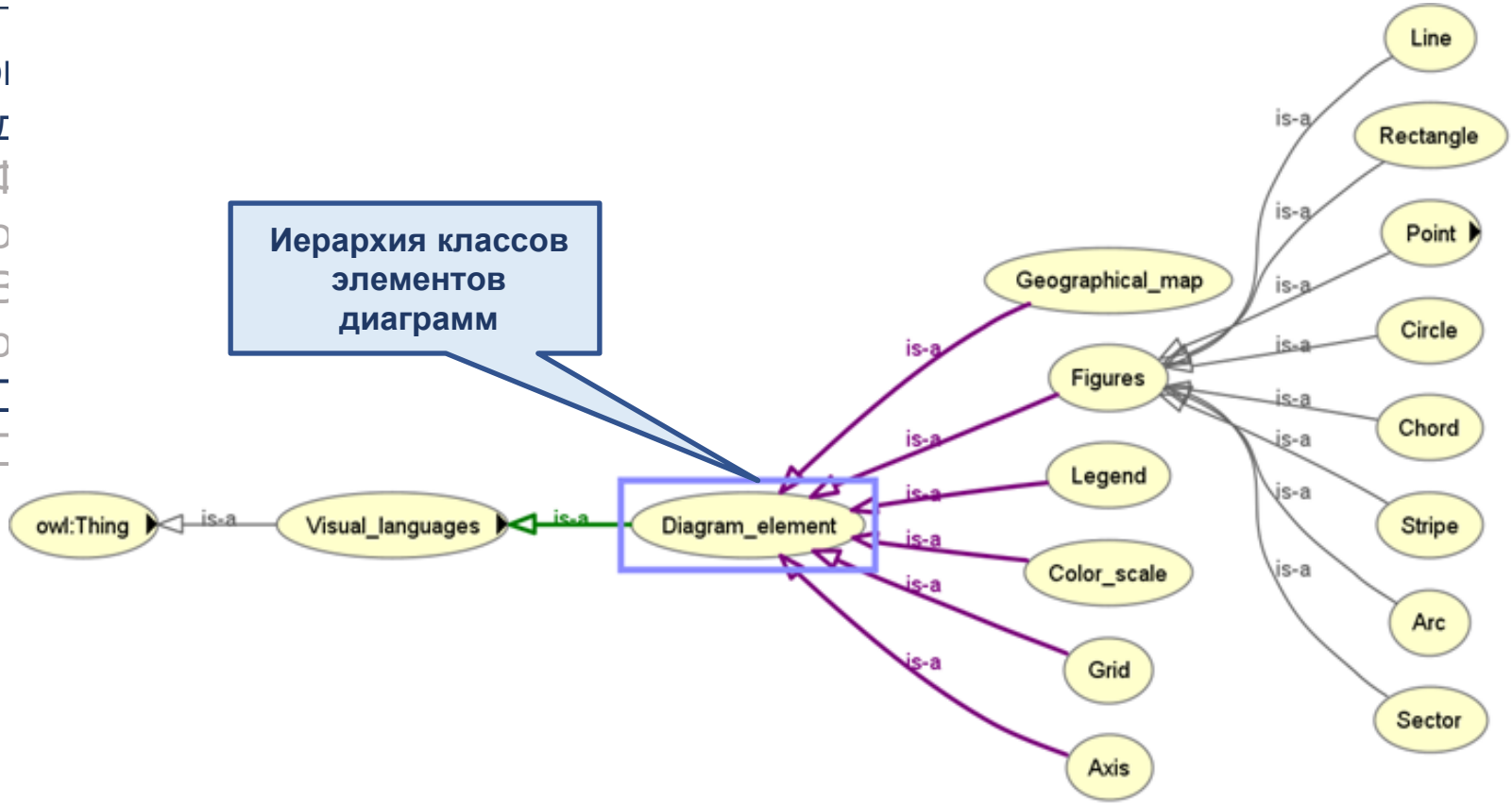
Процесс разработки онтологии языков

Процесс построения онтологии языков визуализации данных включает следующие этапы:

1. Формализация абстрактной диаграммы, которая будет включать свойства, общие для всех типов диаграмм (заголовок, легенда, ширина, высота и т. д.).
2. Выделение разных типов диаграмм в отдельные классы на основе разработанной ранее классификации диаграмм.
3. Добавление уникальных элементов диаграмм в классы.
4. Доопределение отношений между элементами.

Процесс разработки онтологии языков

Про
след
1. Ф
   о
2. Е
   р
3. Д
4. Д



Процесс разработки онтологии языков

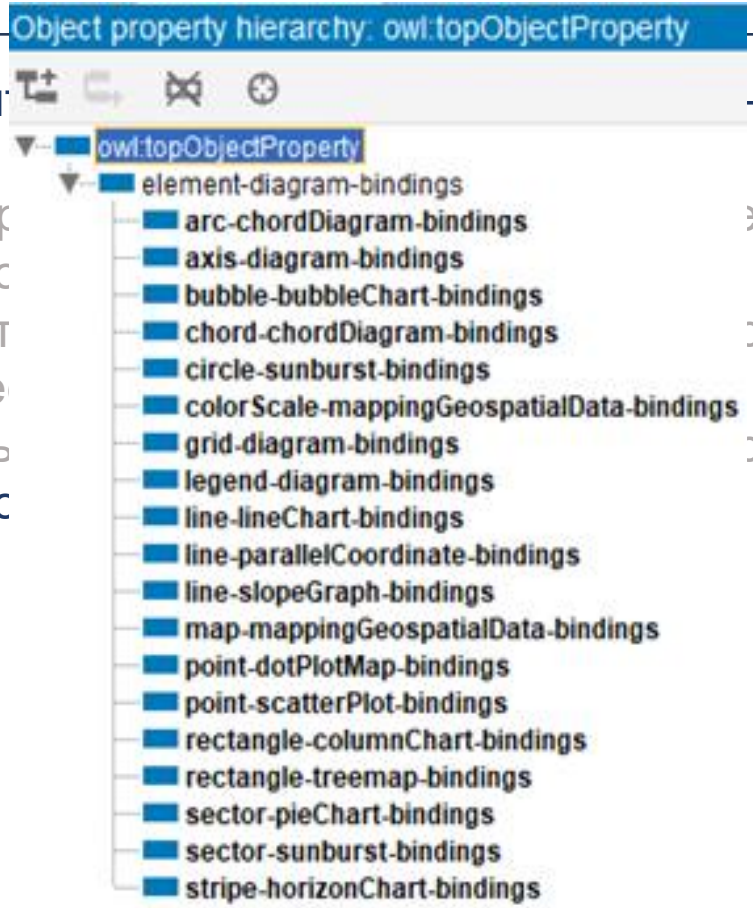
Процесс построения онтологии языков визуализации данных включает следующие этапы:

1. Формализация абстрактной диаграммы, которая будет включать свойства, общие для всех типов диаграмм (заголовок, легенда, ширина, высота и т. д.).
2. Выделение разных типов диаграмм в отдельные классы на основе разработанной ранее классификации диаграмм.
3. Добавление уникальных элементов диаграмм в классы.
4. Доопределение отношений между элементами.

Процесс разработки онтологии языков

Процесс построения он нных включает следующие этапы:

1. Формализация абстр т включать свойства, общие для всех типо ширина, высота и т. д.).
2. Выделение разных т ссы на основе разработанной ране
3. Добавление уникаль сы.
4. Доопределение отно

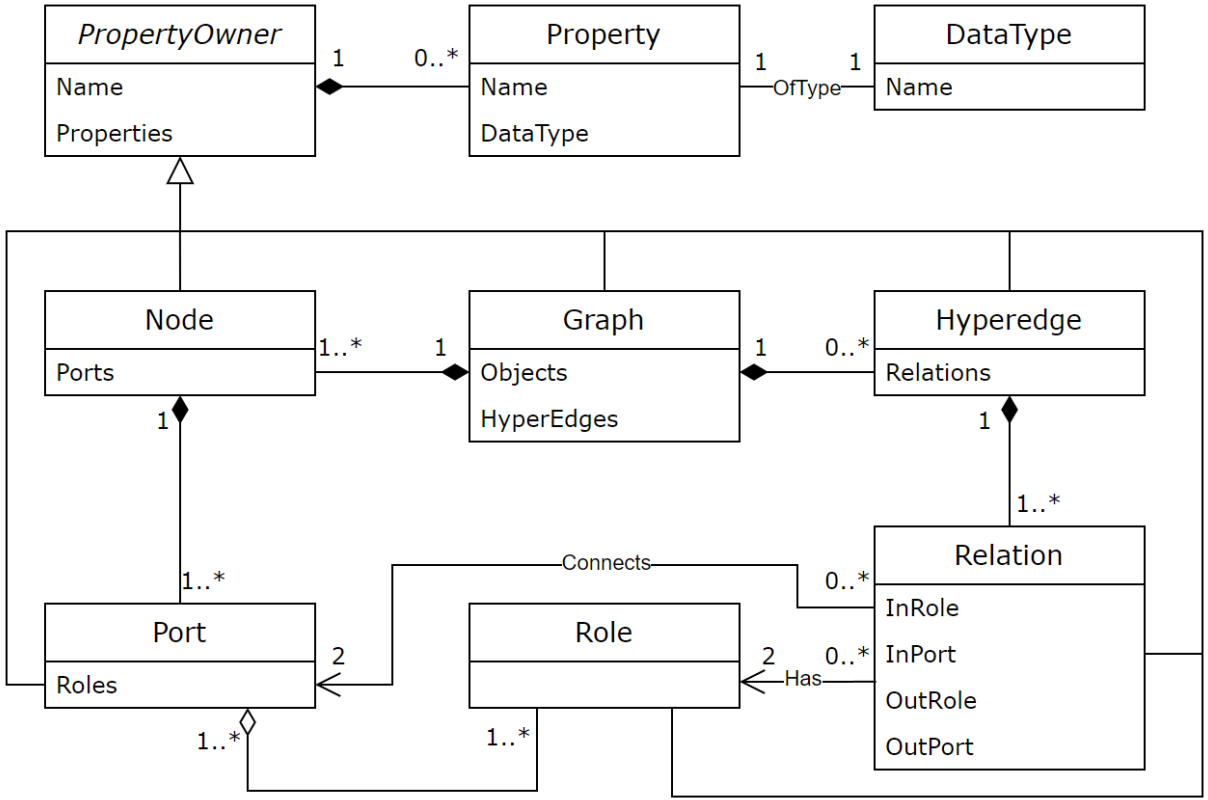


Автоматизация разработки языков: выбор метаязыка

Метаязык	Модель	Объект модели	Связи между объектами	Роли	Полюсы	Свойства
EMOF	–	Class	Property, Association	–	–	Property
Ecore	–	EClass	EReference	–	–	EAttribute
GOPPRR	**Graph**	**Object**	**Relationship**	**Role**	**Port**	**Property**
ArkM3	–	Class	Association, Composition	–	–	Property
WebGME MML	FCO	FCO	Pointer, Set, Connection, Containment, Inheritance, Mixin	Connection Role	–	Attribute

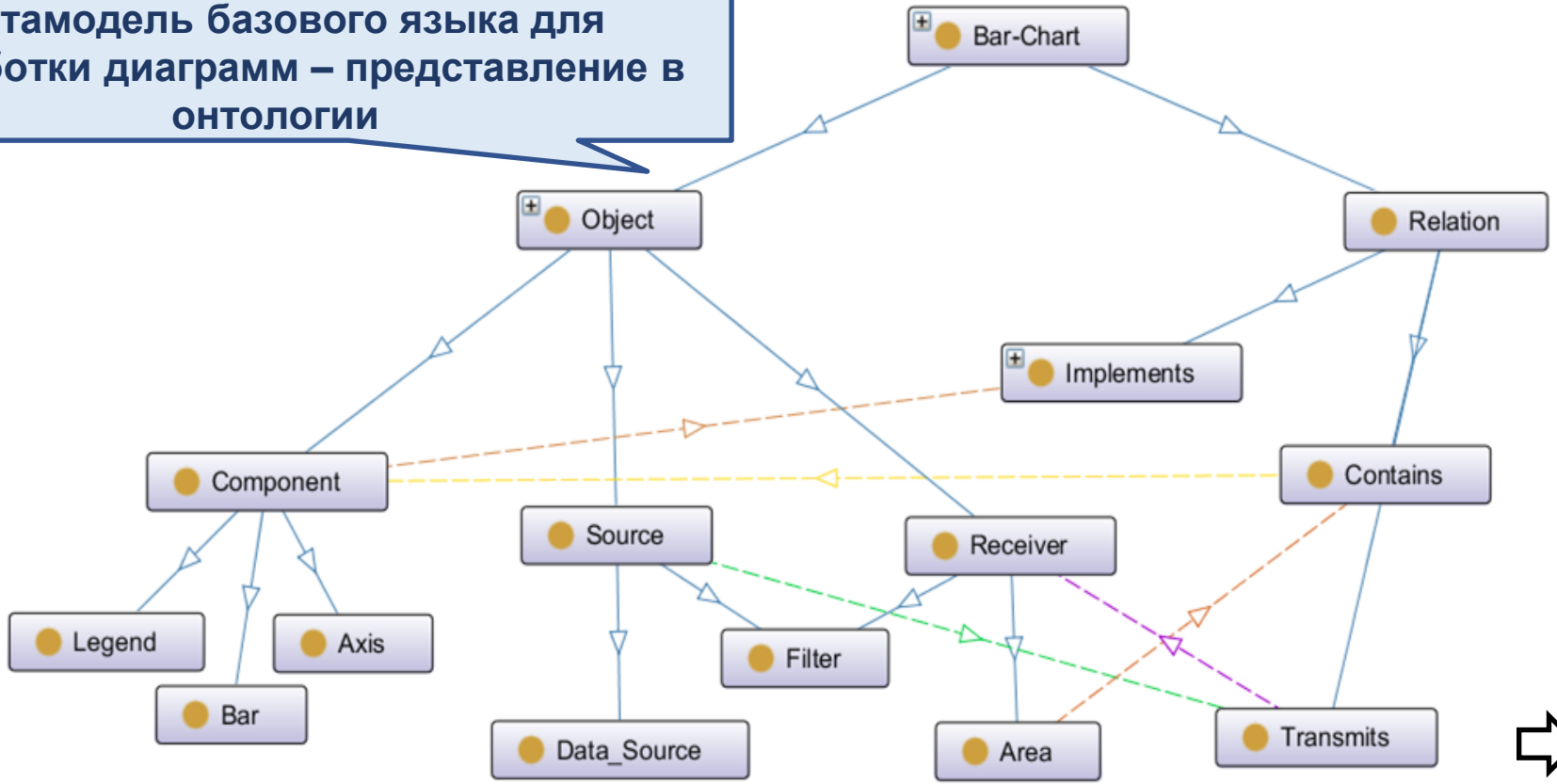
Автоматизация разработки языков: метамодель

Метамодель визуального метаязыка



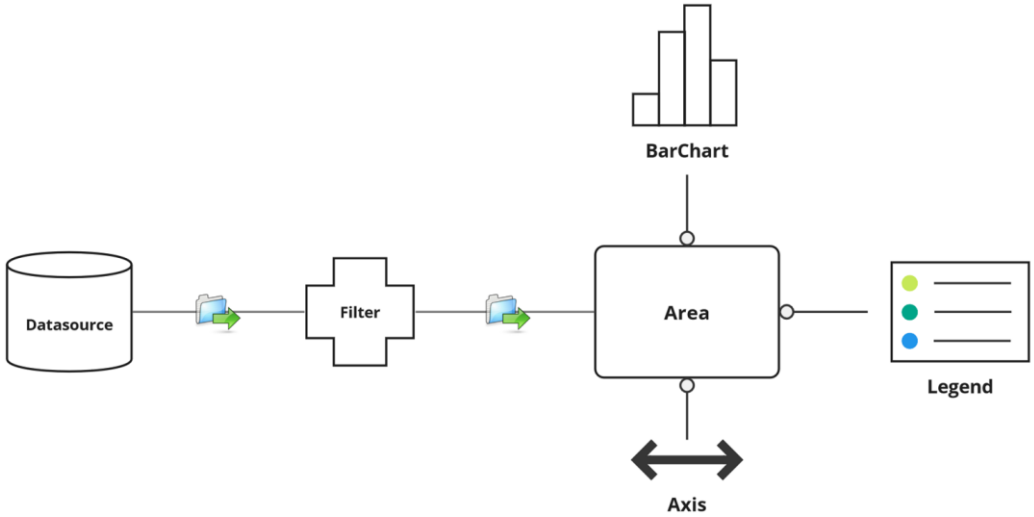
Автоматизация разработки языков:
Метамодель базового языка

Метамодель базового языка для разработки диаграмм – представление в онтологии

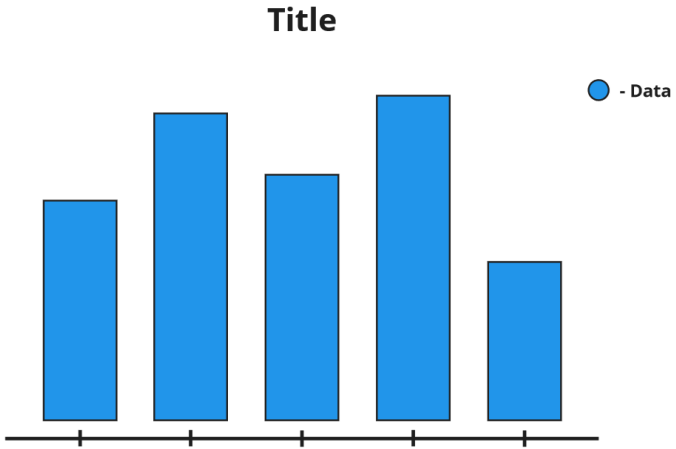


Автоматизация разработки языков

Модель визуализации, созданная с использованием базового DSL

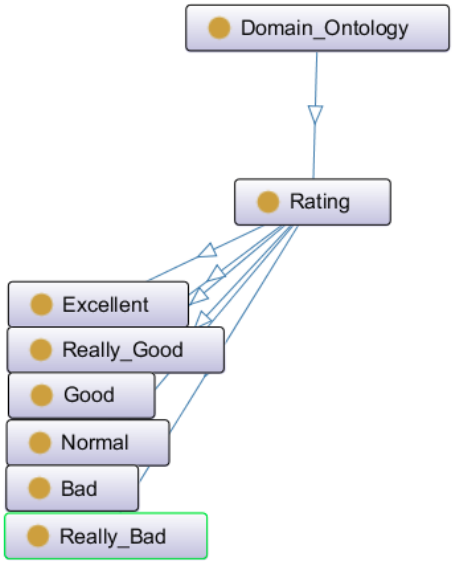


Построенная диаграмма

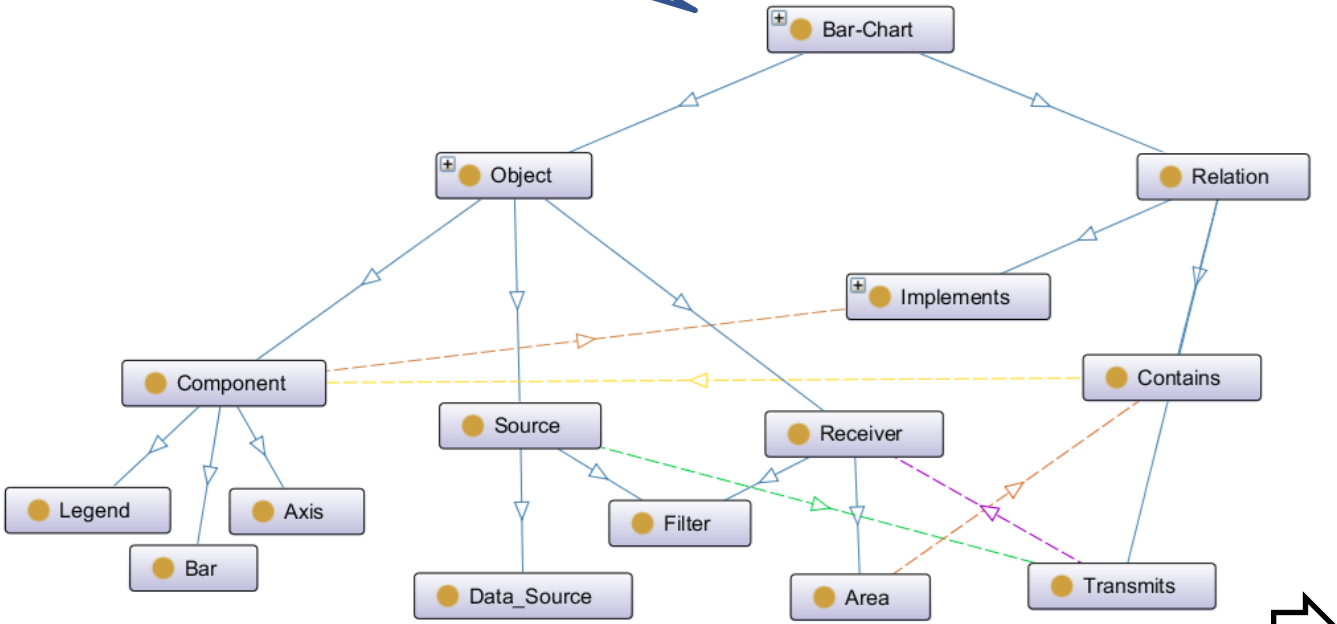


Автоматизация разработки языков: настройка языка

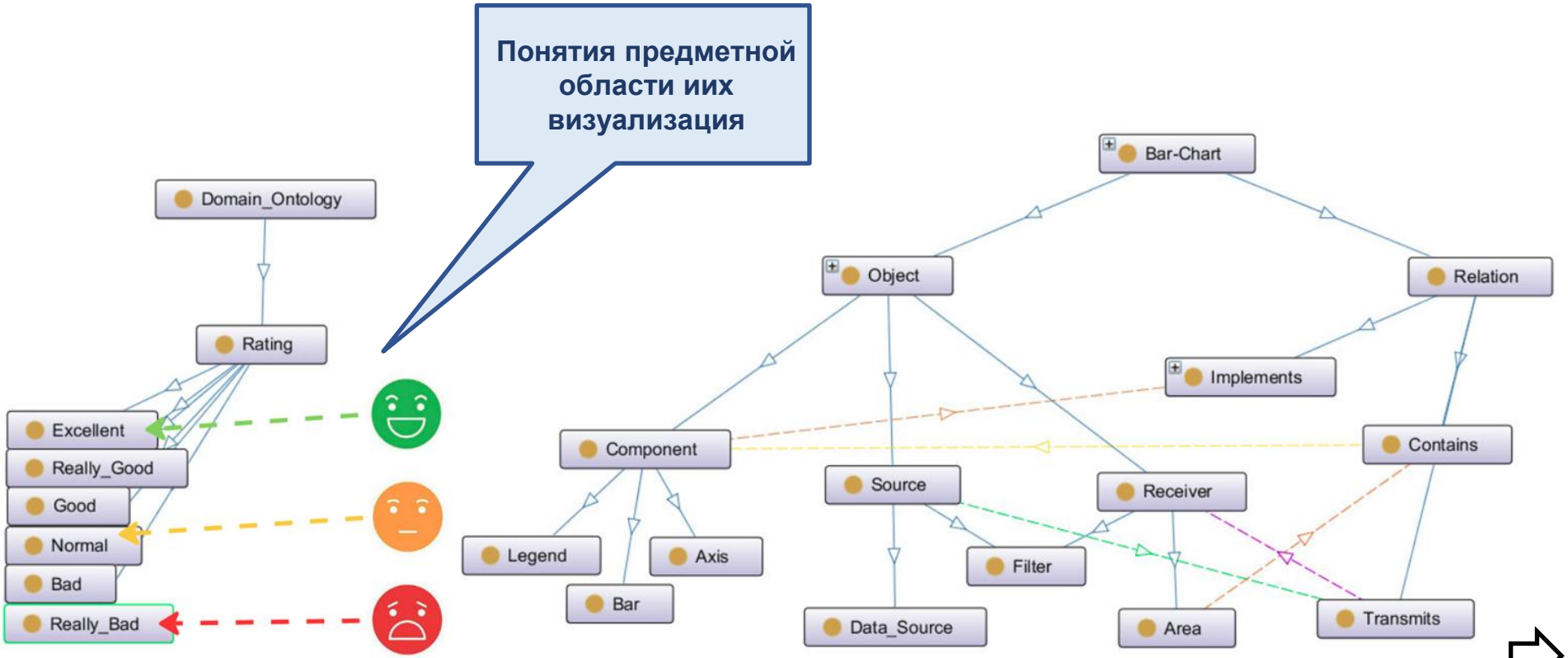
Онтология предметной области



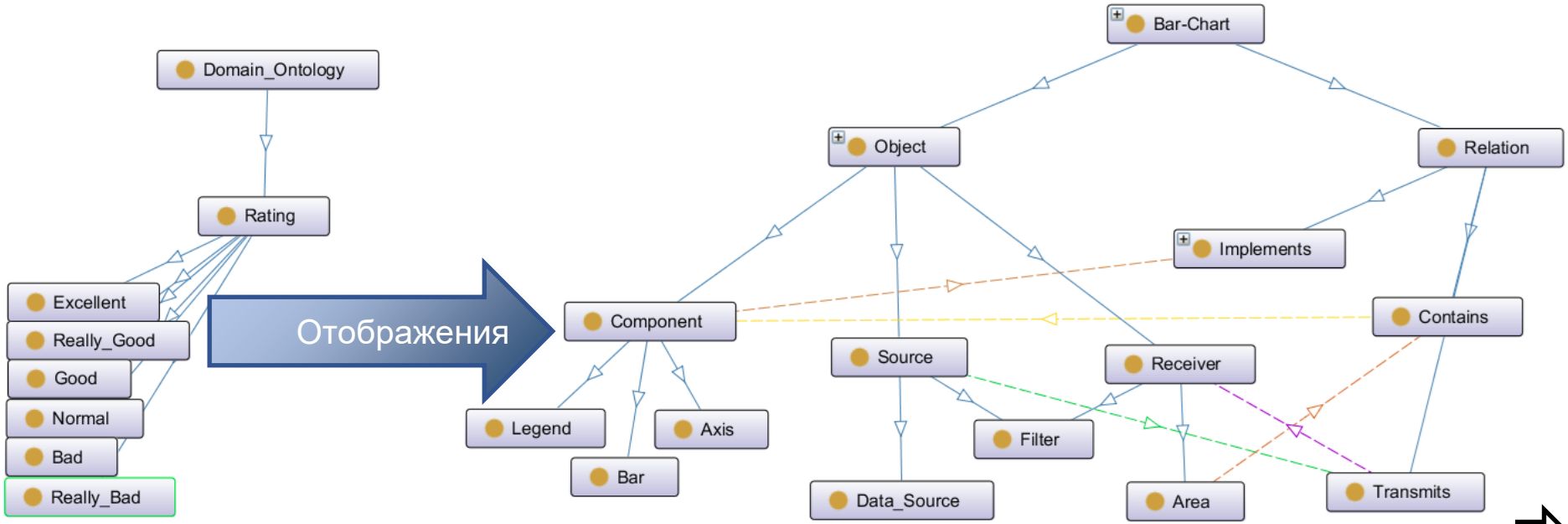
Метамодель базового языка



Автоматизация разработки языков: настройка языка

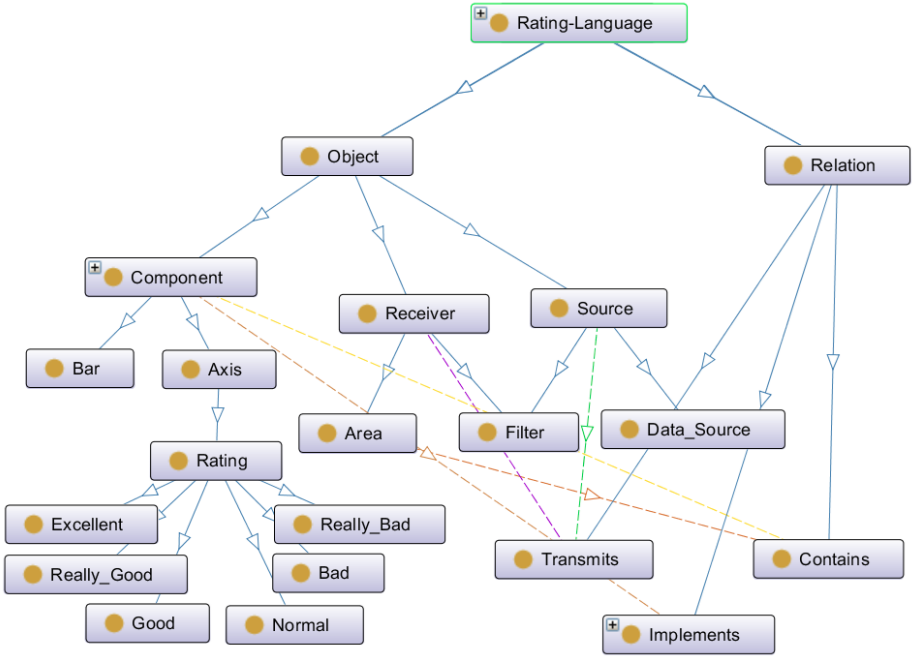


Автоматизация разработки языков: настройка языка



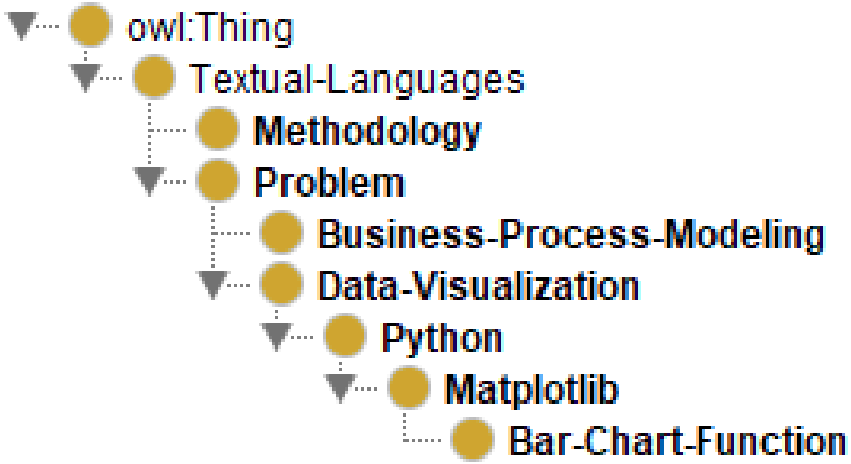
Автоматизация разработки языков: настройка языка

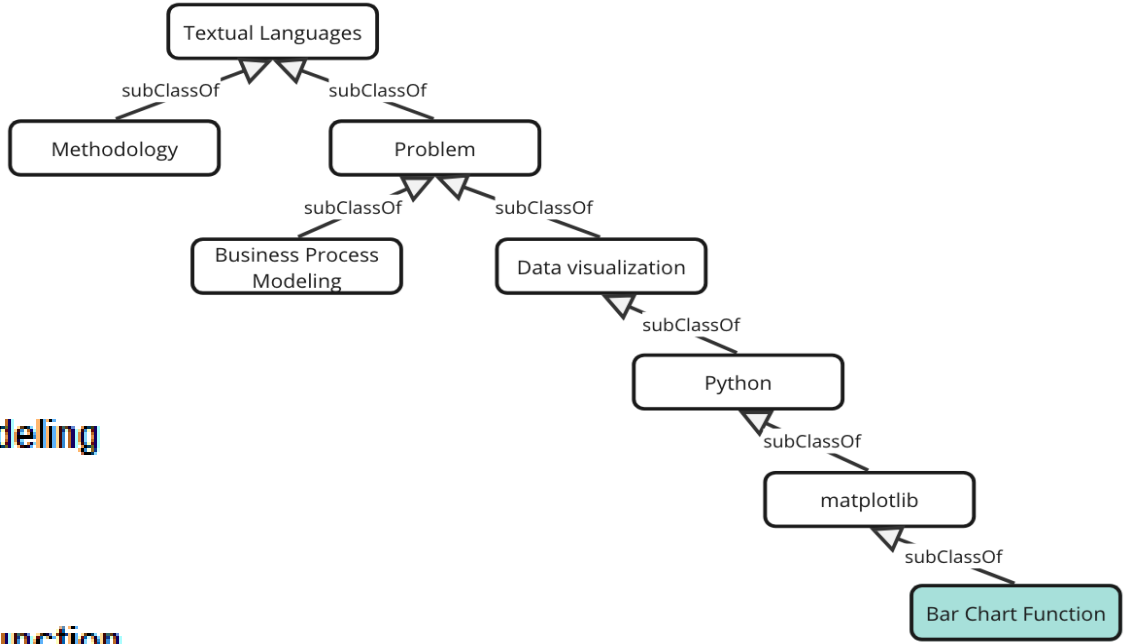
Сгенерированный DSL оценки



Диаграмма, созданная на новом языке

Генерация кода для визуализации данных: иерархия текстовых языков в онтологии

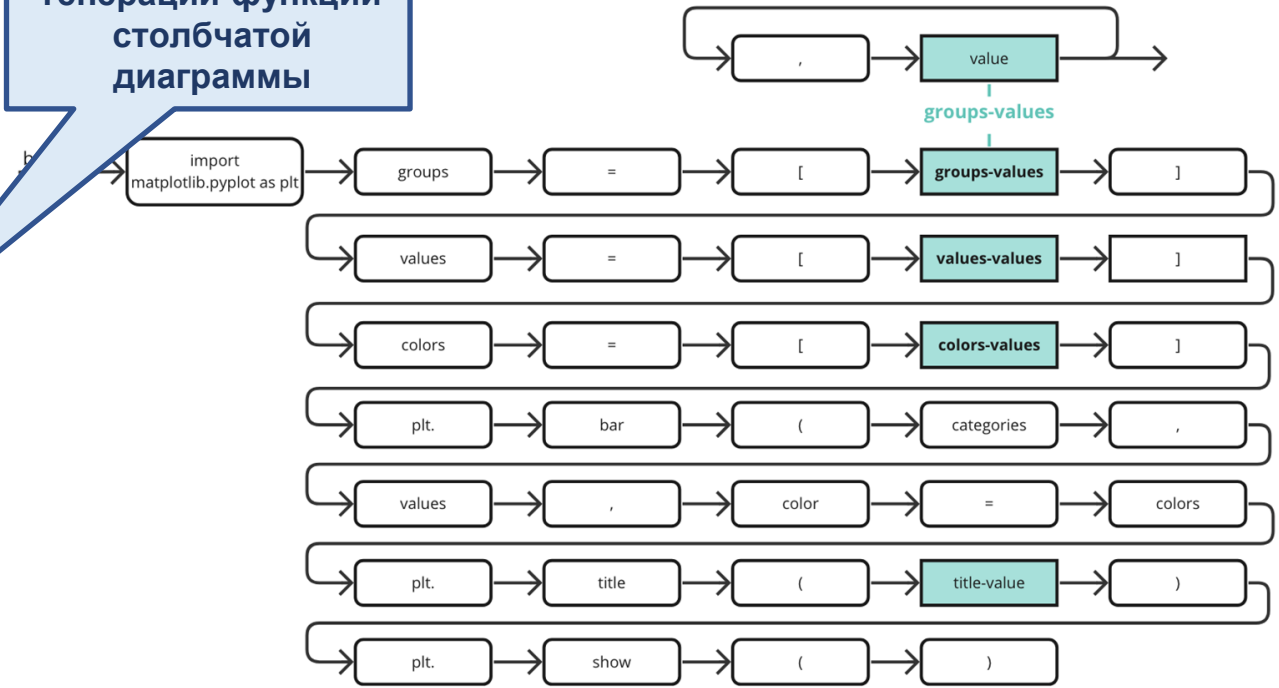




Генерация кода для визуализации данных

Шаблон для генерации функции столбчатой диаграммы

```
import matplotlib.pyplot as plt

groups = [groups-values]
values = [values-values]
colors = [colors-values]

plt.bar(categories, values, color=colors)

plt.title(title-value)

plt.show()
```

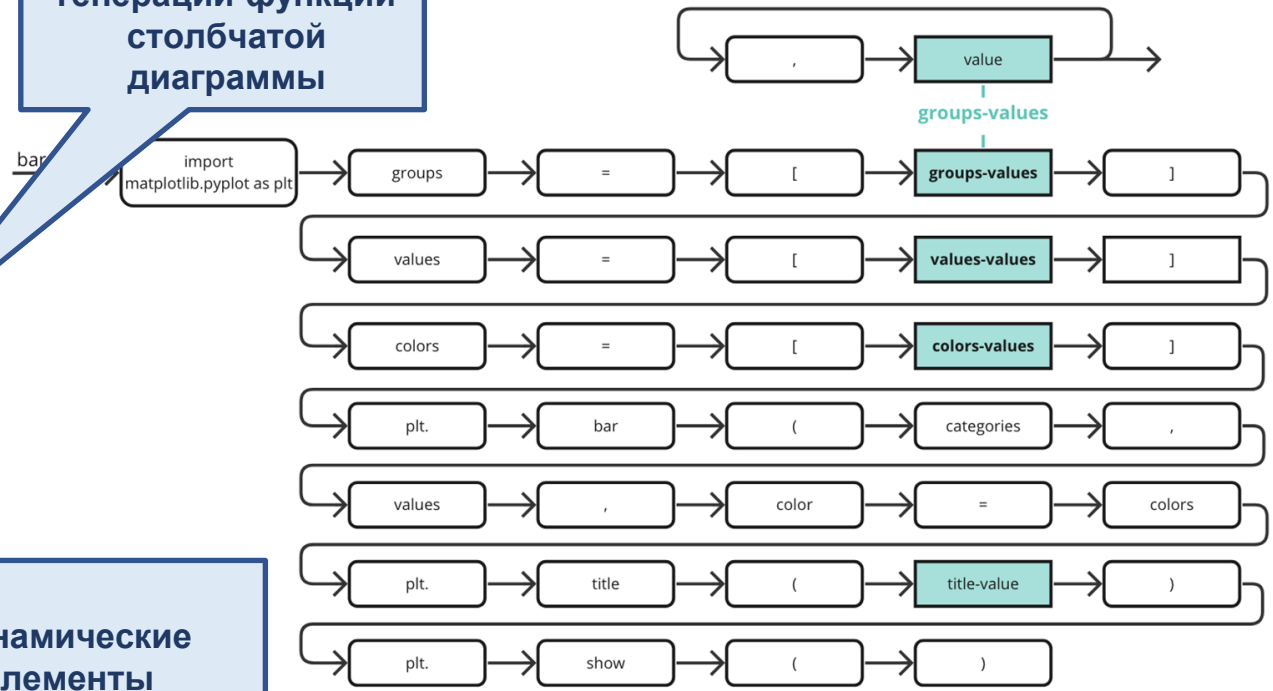


Генерация кода для визуализации данных

Шаблон для генерации функции столбчатой диаграммы

```
import matplotlib.pyplot as plt

groups = [groups-values]
values = [values-values]
colors = [colors-values]

plt.bar(categories, values, color=colors)

plt.title(title-value)

plt.show()
```

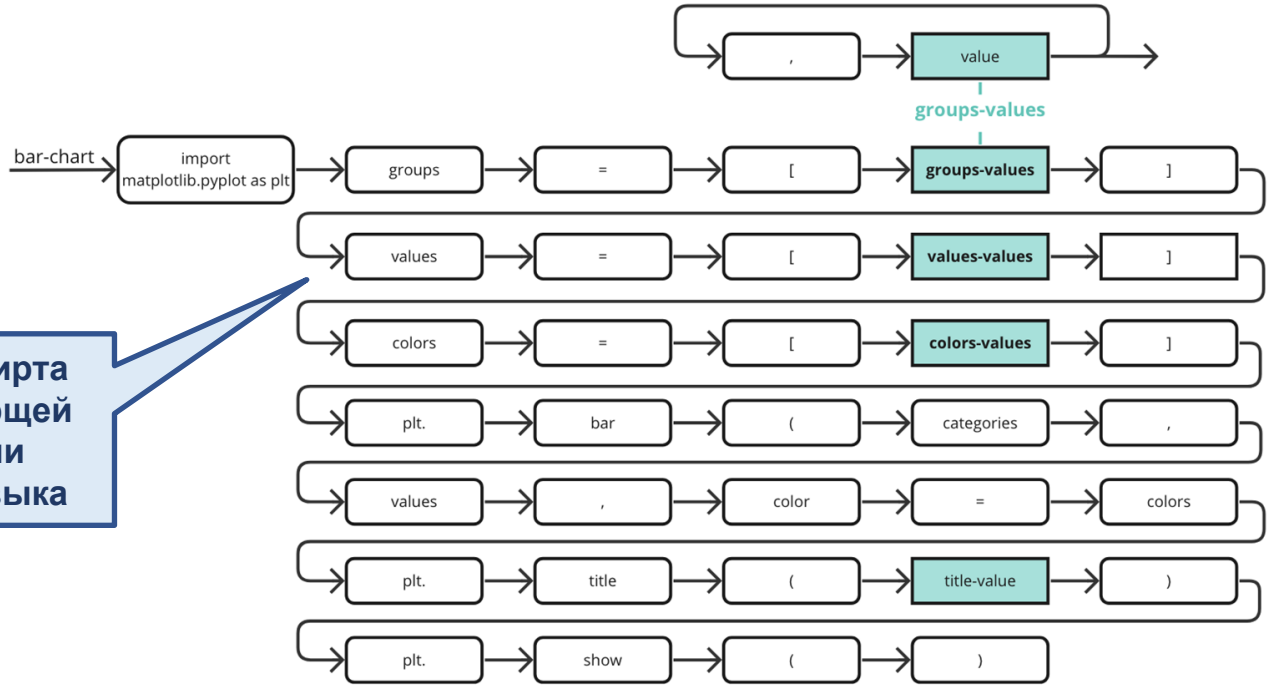
Динамические элементы



Генерация кода для визуализации данных

```
import matplotlib.pyplot as plt

groups = [groups-values]
values = [values-values]
colors = [colors-values]

plt.bar(categories, v

plt.title(title-value)

plt.show()
```

Диаграмма Вирта соответствующей конструкции текстового языка



Генерация кода для визуализации данных

```
import matplotlib.pyplot as plt

groups = [groups-values]
values = [values-values]
colors = [colors-values]

plt.bar(categories, values, color=colors)

plt.title(title-value)

plt.show()
```

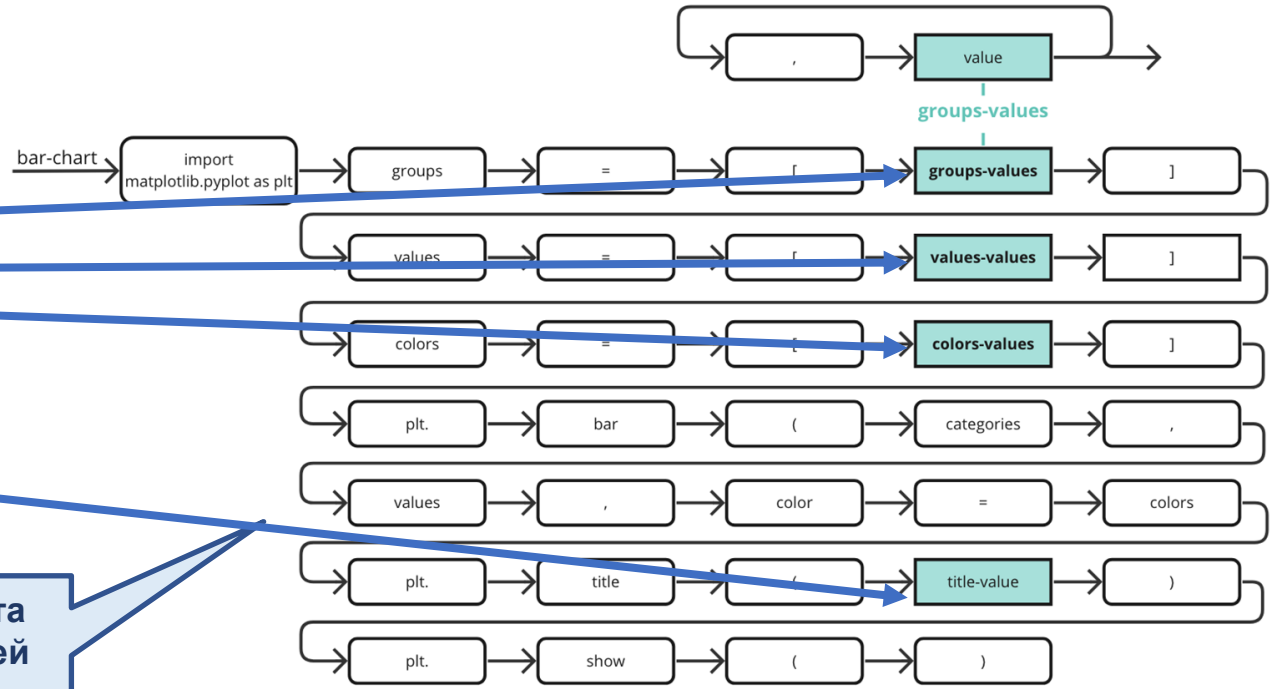


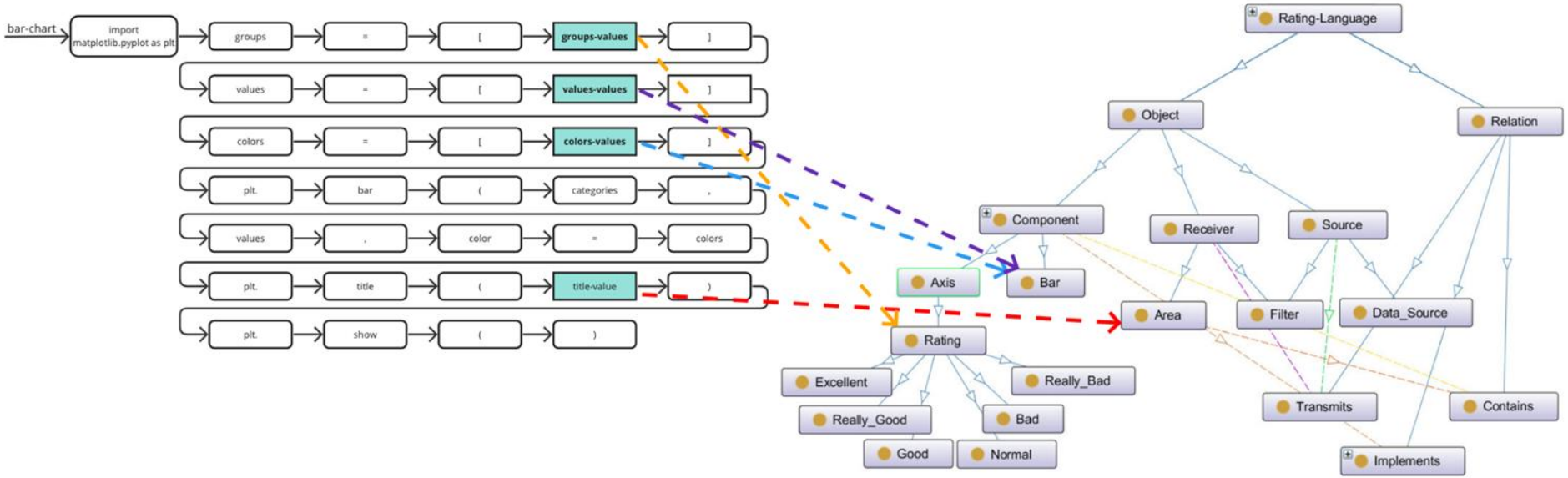
Диаграмма Вирта соответствующей конструкции текстового языка

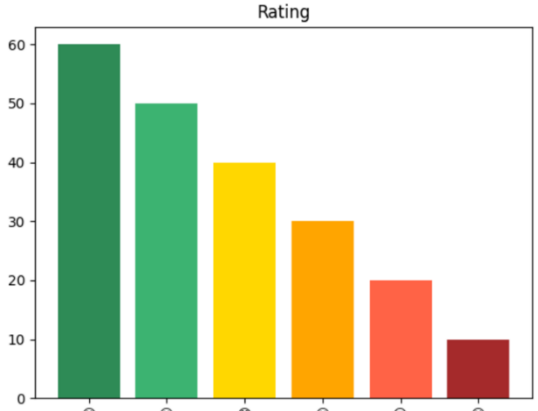
Генерация кода для визуализации данных: правила трансформации

Правила трансформаций ставят в соответствие объектам метамодели DSL и их свойствам нетерминальные символы синтаксических диаграмм. При этом если нетерминальный символ простой, то ему должно соответствовать свойство объекта, а если составной, то другой объект метамодели языка.

- Правило преобразования связывает объект метамодели визуального языка и конструкцию текстового языка.
- Каждый простой нетерминальный символ конструкции текстового языка связан со свойством объекта метамодели визуального языка.
- Каждый составной нетерминальный символ конструкции текстового языка связан с объектом метамодели визуального языка.

Генерация кода для визуализации данных: обработка нетерминальных символов

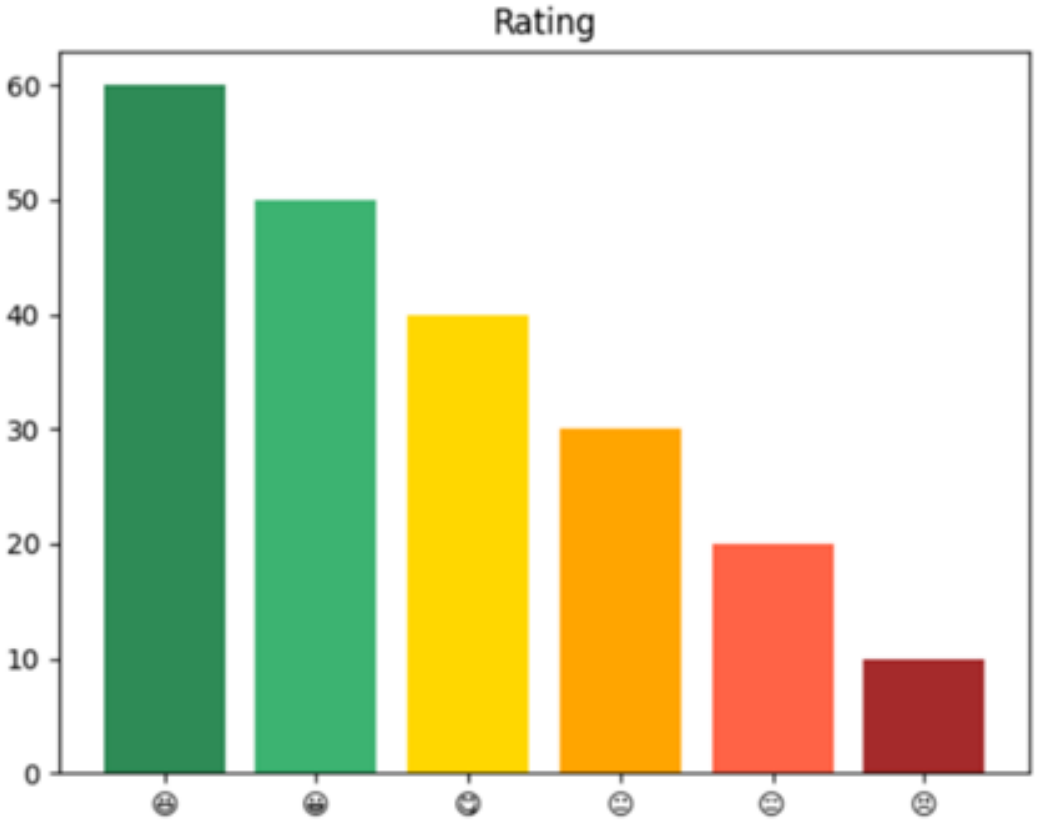




да для визуализации данных: Сгенерированный код

```
...o.pyplot as plt

...'😀', '😋', '😐', '😔', '😣']
values = [..., ..., 40, 30, 20, 10]
colors = ['seagreen', 'mediumseagreen', 'gold', 'orange',
'tomato', 'brown']

plt.bar(groups, values, color=colors)

plt.title('Rating')

plt.show()
```

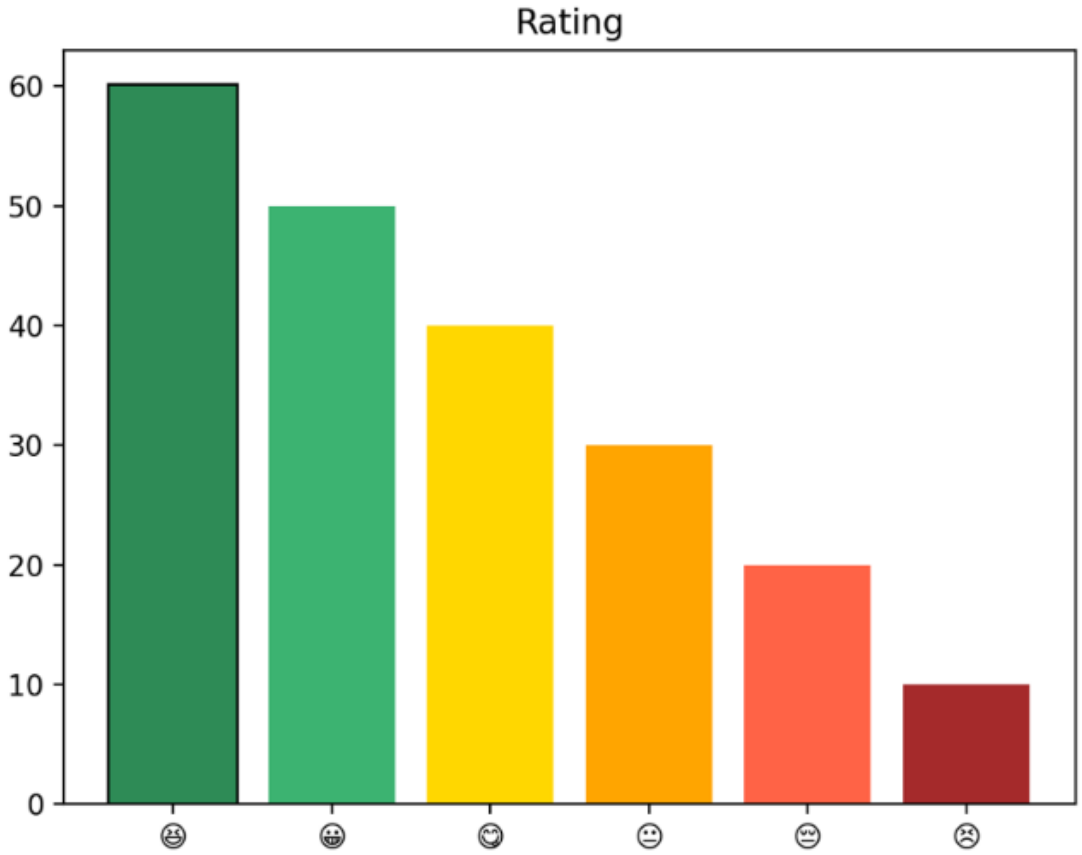
Генерация кода для визуализации данных: Построенная диаграмма



Генерация кода для визуализации данных: Интерактивная визуализация

```
%matplotlib notebook
import matplotlib.pyplot as plt
groups = ['😆', '😀', '😊', '😐', '😔', '😣']
values = [60, 50, 40, 30, 20, 10]
colors = ['seagreen', 'mediumseagreen', 'gold', 'orange', 'tomato', 'brown']
fig, ax = plt.subplots()
bars = ax.bar(groups, values)

def on_hover(event):
    for bar in bars:
        if bar.contains(event)[0]:
            bar.set_alpha(1.0)
            bar.set_edgecolor('black')
            bar.set_linewidth(2)
        else:
            bar.set_alpha(0.8)
            bar.set_edgecolor('none')
            bar.set_linewidth(1)
    fig.canvas.draw_idle()

fig.canvas.mpl_connect('motion_notify_event', on_hover)
plt.bar(groups, values, color=colors)
plt.title('Rating')
```

Генерация кода для визуализации данных: Интерактивная визуализация



Заключение

В данном исследовании предложена структура инструмента визуализации данных, основанного на знаниях, на основе языкового подхода. Она позволяет преодолеть ряд ограничений существующих инструментов визуализации и предоставить пользователям возможность настраивать модели визуализации данных для различных предметных областей путем создания специальных языков, а также снижает требования к навыкам программирования конечного пользователя для создания DSL

Практическая применимость данного подхода продемонстрирована на примере создания диаграммы для оценки качества обслуживания клиентов.

Следующим этапом исследования является реализация описанных идей и расширение возможностей разработки диаграмм разных типов, интерактивной визуализации путем интерпретации созданных моделей

Спасибо за внимание!

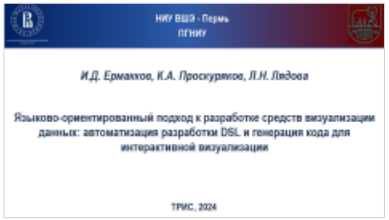
Докладчик: ***Л.Н. Лядова***

E-mail: *LLyadova@hse.ru*

ТРИС, 2024

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

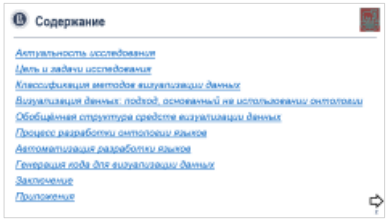
HSE University
Perm State University


НИУ ВШЭ - Пермь
ПГНИУ

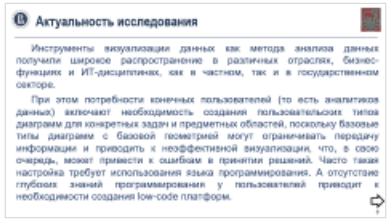
И.Д. Ермаков, К.А. Проскурнов, Л.Н. Лядова

Языково-ориентированный подход к разработке средств визуализации данных: автоматизация разработки DSL и генерация кода для интерактивной визуализации

ТРИС, 2024


Содержание

Актуальность исследования
Цель и задачи исследования
Классификация методов визуализации данных
Визуализация данных: подход, основанный на использовании онтологии
Обобщённая структура средств визуализации данных
Процесс разработки онтологии языков
Автоматизация разработки языков
Генерация кода для визуализации данных
Заключение
Приложение


Актуальность исследования

Инструменты визуализации данных как метода анализа данных получили широкое распространение в различных отраслях, бизнес-функциях и ИТ-дисциплинах, как в частном, так и в государственном секторе.

При этом потребности конечных пользователей (то есть аналитиков данных) включают необходимость создания пользовательских типов диаграмм для конкретных задач и предметных областей, поскольку базовые типы диаграмм с базовой геометрией могут ограничивать передачу информации и приводить к неэффективной визуализации, что, в свою очередь, может привести к ошибкам в принятии решений. Часто такая настройка требует использования языка программирования. А отсутствие глубоких знаний программирования у пользователей приводит к необходимости создания low-code платформ.

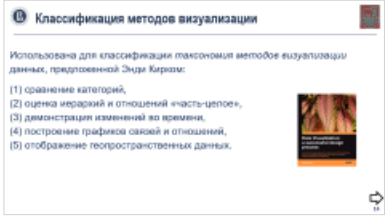

Цель и задачи исследования

Цель – преодолеть ряд ограничений существующих средств визуализации и предоставить пользователям возможность настраивать методы визуализации данных для различных предметных областей, под специфику решаемых задач. При этом снизив требования к навыкам программирования конечного пользователя системы при создании диаграмм.

Данная цель может быть достигнута путем создания специальных языков – предметно-ориентированных языков (DSL) и средств автоматизации их разработки, управляемых знаниями.

В предыдущих работах был предложен подход к автоматизации создания DSL, основанный на использовании многоаспектной онтологии. С использованием данного подхода были разработаны языки описания структур данных, бизнес-процессов, алгоритмов управления…

В данном исследовании проведена апробация подхода для разработки моделей визуализации данных.


Классификация методов визуализации

Использована для классификации таксономия методов визуализации данных, предложенной Энди Кирком:

(1) сравнение категорий,
(2) оценка иерархий и отношений «часть-целое»,
(3) демонстрация изменений во времени,
(4) построение графиков связей и отношений,
(5) отображение геопространственных данных.


Визуализация данных – разработка моделей: подход, основанный на использовании онтологии

Публикаций, посвященных созданию DSL для визуализации данных, немного. Согласно проведенному обзору, большинство DSL ориентированы на небольшой набор стандартных диаграмм (например, на круговые диаграммы и гистограммы) или визуализацию конкретных типов данных (например, геопространственных).

Языки различаются по уровням абстракции, контекстам использования и возможностям реализации.

Тем не менее, в рассмотренных инструментах адаптация диаграмм под конкретные предметные области не найдена. Исходя из этого, мы делаем вывод, что языково-ориентированный подход к реализации инструмента визуализации данных может стать основным при разработке системы визуализации данных


Обобщённая структура средств визуализации данных


Процесс разработки онтологии языков

Процесс построения онтологии языков визуализации данных включает следующие этапы:

1. Формализация абстрактной диаграммы, которая будет включать свойства, общие для всех типов диаграмм (заголовок, легенда, ширина, высота и т. д.).
2. Выделение разных типов диаграмм в отдельные классы на основе разработанной ранее классификации диаграмм.
3. Добавление уникальных элементов диаграмм в классы.
4. Доопределение отношений между элементами.


Автоматизация разработки языков: выбор метаязыка

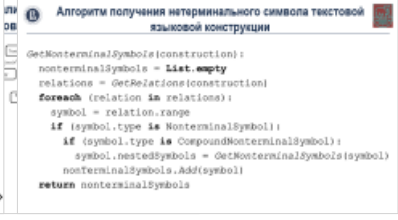
Метаязык	Модель	Объект модели	Связи между объектами	Роли	Полюсы	Свойства
EMOF	–	Class	Property, Association	–	–	Property
Ecore	–	EClass	EReference	–	–	EAttribute
GOPPRR	Graph	Object	Relationship	Role	Port	Property
ArkM3	–	Class	Association, Composition	–	–	Property
WebGME MML	FCO	FCO	Pointer, Set, Connection, Containment, Inheritance, Mixin	Connection Role	–	Attribute


Заключение

В данном исследовании предложена структура инструмента визуализации данных, основанного на знаниях, на основе языкового подхода. Она позволяет преодолеть ряд ограничений существующих инструментов визуализации и предоставить пользователям возможность настраивать модели визуализации данных для различных предметных областей путем создания специальных языков, а также снижает требование к навыкам программирования конечного пользователя для создания DSL.

Практическая применимость данного подхода продемонстрирована на примере создания диаграммы для оценки качества обслуживания клиентов.

Следующим этапом исследования является реализация описанных идей и расширение возможностей разработки диаграмм разных типов, интерактивной визуализации путем интерпретации созданных моделей


Алгоритм получения нетерминального символа текстовой языковой конструкции

```
GetNonterminalSymbols(construction):
  nonterminalSymbols = List.empty
  relations = GetRelations(construction)
  foreach (relation in relations):
    symbol = relation.range
    if (symbol.type is NonterminalSymbol):
      if (symbol.type is CompoundNonterminalSymbol):
        symbol.nestedSymbols = GetNonterminalSymbols(symbol)
      nonTerminalSymbols.Add(symbol)
  return nonterminalSymbols
```

ТРИС, 2024

Алгоритм получения нетерминального символа текстовой языковой конструкции

```
GetNonterminalSymbols(construction):
  nonterminalSymbols = List.empty
  relations = GetRelations(construction)
  foreach (relation in relations):
    symbol = relation.range
    if (symbol.type is NonterminalSymbol):
      if (symbol.type is CompoundNonterminalSymbol):
        symbol.nestedSymbols = GetNonterminalSymbols(symbol)
      nonTerminalSymbols.Add(symbol)
  return nonterminalSymbols
```

Алгоритм добавления нового правила трансформации в онтологию

```
AddRule(visualLanguage, textualLanguage, rule, statements):
  class = CreateClass(visualLanguage, textualLanguage, rule)
  relation = CreateRelation(class, rule)
  foreach (statement in statements):
    symbol = statement.nonterminalSymbol
    object = statement.object
    ruleRelation = CreateRuleRelation(symbol, object, relation)
    while (symbol.type is CompoundNonterminalSymbol)
      statement = statements.next
      subRelation = CreateRuleRelation(symbol, object, ruleRelation)
      ruleRelation.subRelations.Add(subRelation)
    relation.subRelations.Add(ruleRelation)

CreateRuleRelation(symbol, object, parentRelation):
  ruleRelation = CreateSubRelation(symbol, object, parentRelation)
  return ruleRelation
```

Алгоритм удаления правила трансформации из онтологии

```
RemoveRule(visualLanguage, textualLanguage, rule):
  class = GetClass(visualLanguage, textualLanguage, rule)
  relation = GetRelation(class, rule)
  RemoveRuleRelation(relation)
  RemoveClass(class)

RemoveRuleRelation(relation):
  subRelations = GetSubRelations(relation)
  if (subRelations is List.empty):
    RemoveRelation(relation)
  else:
    foreach (subRelation in subRelations):
      RemoveRuleRelation(subRelation)
```

Алгоритм применения правил трансформации к модели

```
ExecuteRule(model, rule, file):
  metamodelObject = GetMetamodelObject(rule)
  modelObjects = GetModelObjects(model, metamodelObject)
  construction = GetConstruction(rule)
  relations = GetRelations(construction)
  foreach (modelObject in modelObjects):
    foreach (relation in relations):
      ProcessRelation(relation, modelObject, model, rule, file)

ProcessRelation(relation, modelObject, model, rule, file):
  symbol = relation.range
  if (symbol.type is TerminalSymbol):
    file.AppendText(symbol)
  else:
    metamodelObject = GetMappedMetamodelObject(symbol)
    if (metamodelObject is null):
      return
    if (symbol.type is CompoundNonterminalSymbol):
      subRule = GetSubRule(rule, symbol)
      ExecuteRule(model, subRule, file)
    else:
      propertyValue = GetPropertyValue(modelObject, symbol)
      file.AppendText(propertyValue)
```