Инструментальная среда для анализа программных систем

А.М. Половцев, гр. 63501/13
научный руководитель: к.т.н. доцент В.М. Ицыксон

Направление: 230100 – Информатика и вычислительная техника
Магистерская программа: 230100.68.15 — Технологии проектирования системного и прикладного программного обеспечения

Санкт-Петербургский государственный политехнический университет

10 июня 2014 г.

Актуальность

- Растет сложность и размер программных систем
- Снижаются сроки разработки
- Эти факторы ведут к падению уровня качества
- В различных методах повышения качества часто решаются похожие задачи:
  - Построение моделей программы (AST, CFG, ...)
  - Построение метрик
  - Реинжиниринг программного обеспечения (оптимизация, рефакторинг, ...)
  - Визуализация свойств системы

Обычно эти задачи решаются вручную для каждого языка программирования

Рассматриваемый подход

- Решение данной проблемы основывается на использовании промежуточного представления, не зависящего от языка программирования, на котором написана анализируемая система
- **Метамодель** - модель, описывающая язык для формулировки моделей
- Поиск компромисса между степенью детализации и уровнем абстракции

Решаемые задачи

- В итоге требуется решить следующие задачи:
  - Проектирование промежуточного представления, не зависящего от языка программирования анализируемой системы, для применения обобщенных процедур анализа
  - Разработка графической инструментальной среды на основе промежуточного представления для визуализации моделей, метрик и свойств программных систем с целью повышения уровня качества

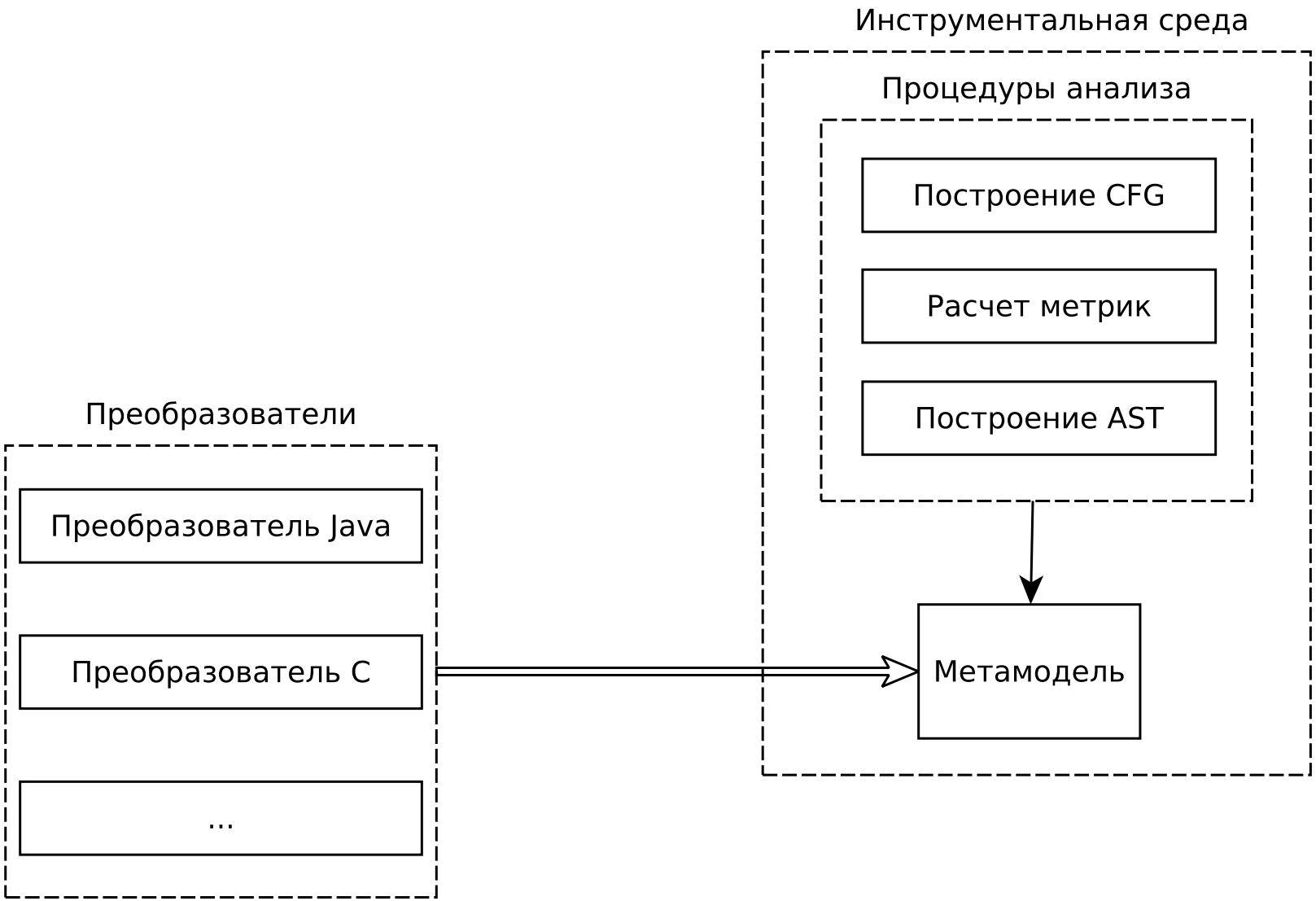
Требования

- Требования к промежуточному представлению
  - Независимость от языка описания анализируемой системы
  - Расширяемость
  - Простота в использовании
  - Полнота
- Требования к инструментальной среде
  - Визуализация извлеченных моделей
  - Подсчет метрик
  - Визуализация дополнительных свойств

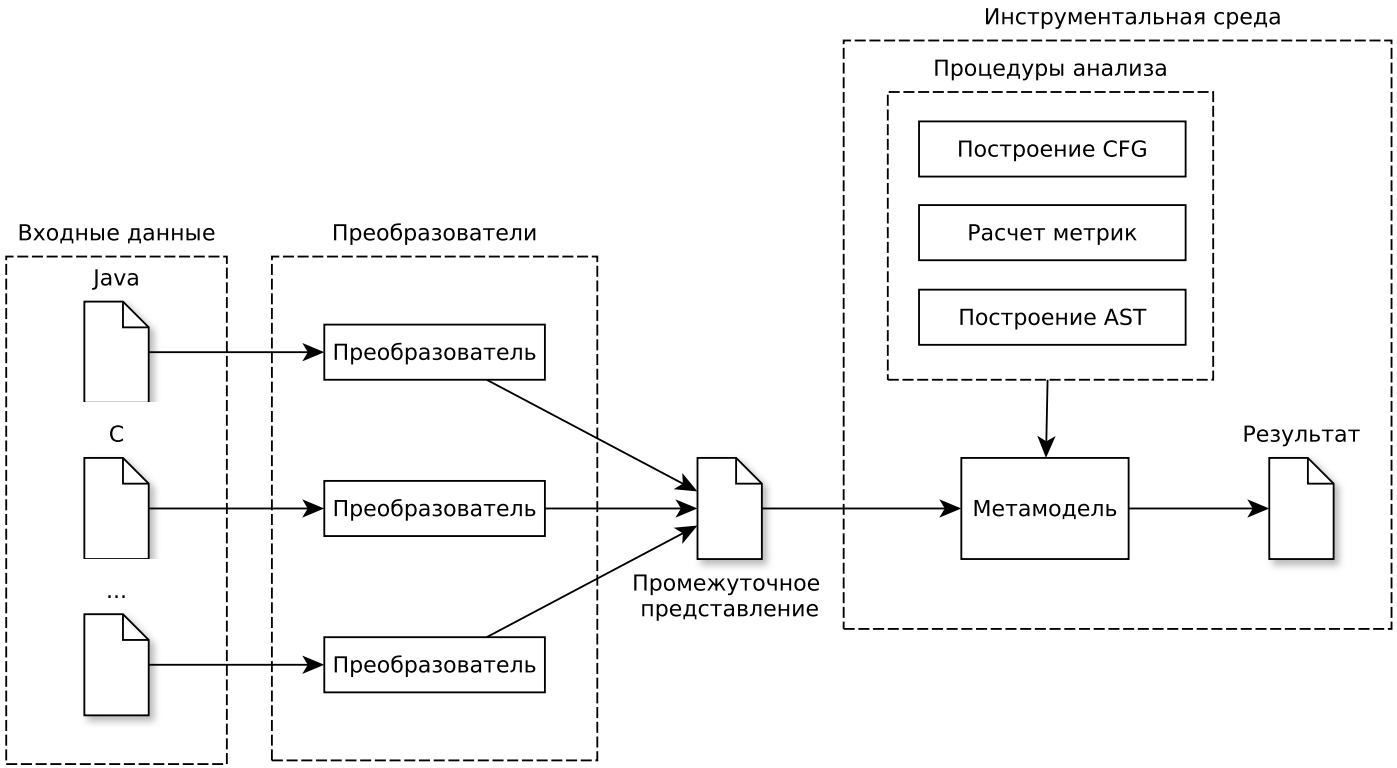
Существующие решения

- Moose - слишком громоздкая метамодель
- LLVM - низкоуровневое промежуточное представление
- SMIILE - ориентирован только на метрики
- Ulf-Ware - представление только для языков Java и C++

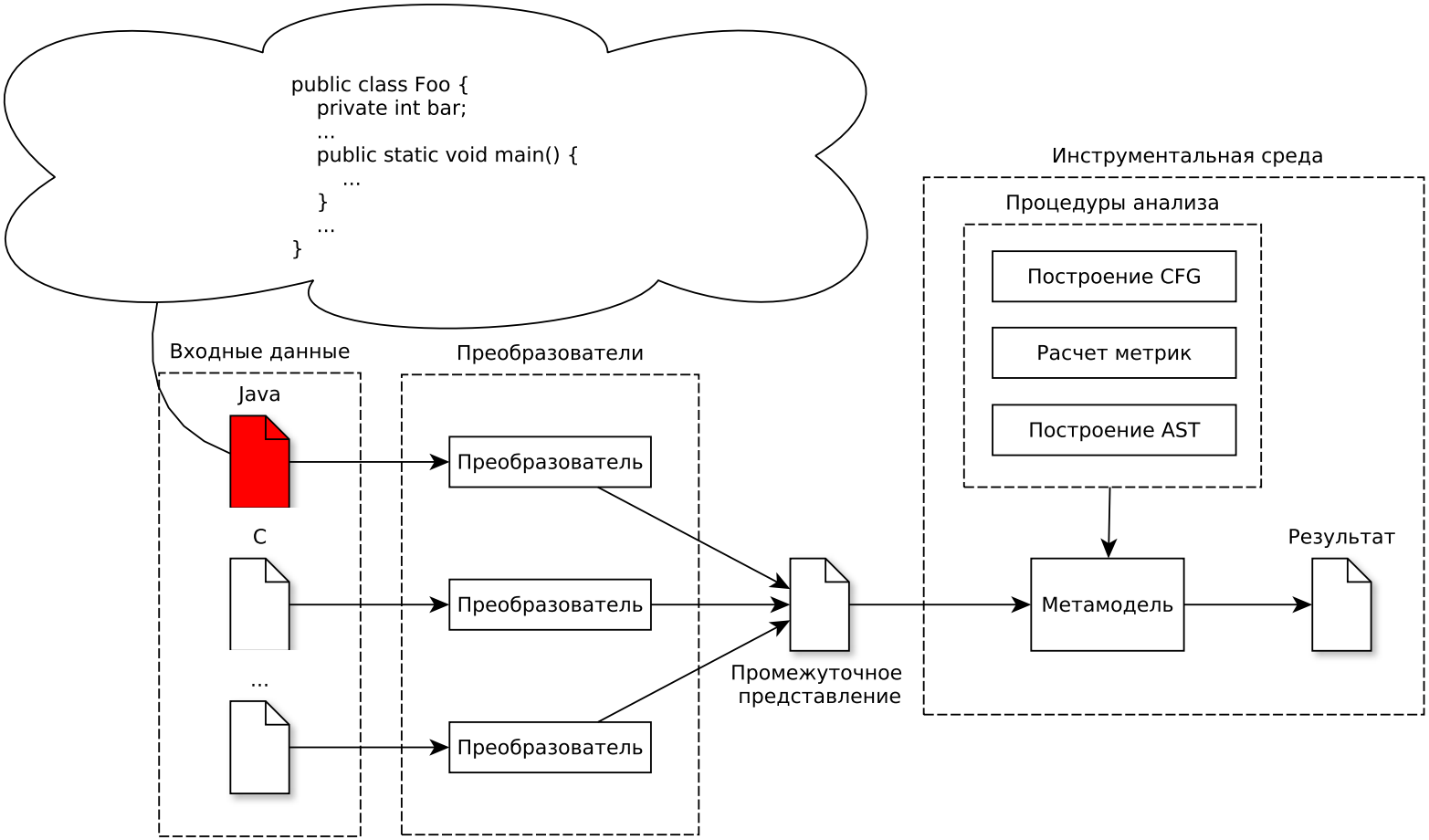
Архитектура программной системы



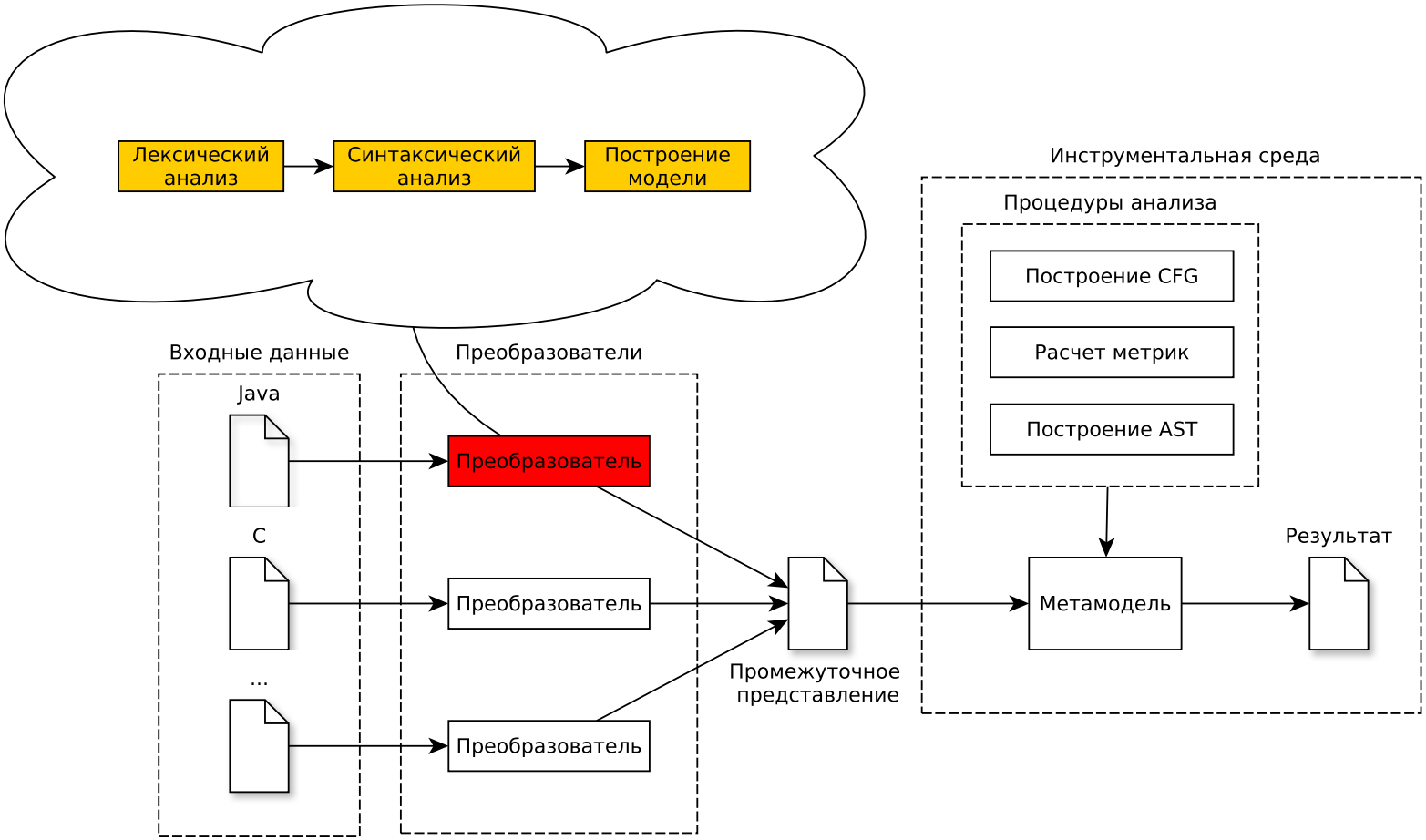
Принцип работы



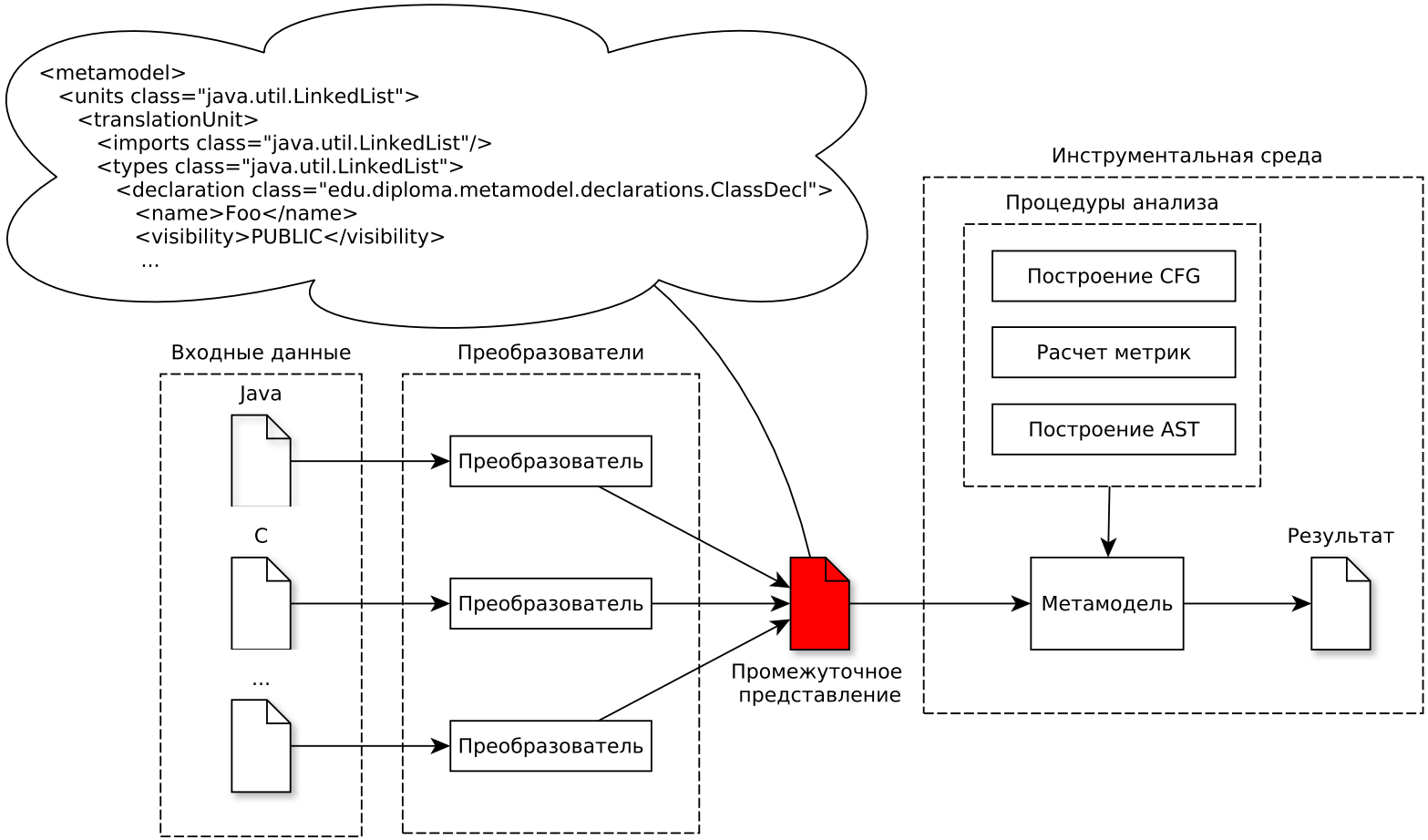
Принцип работы



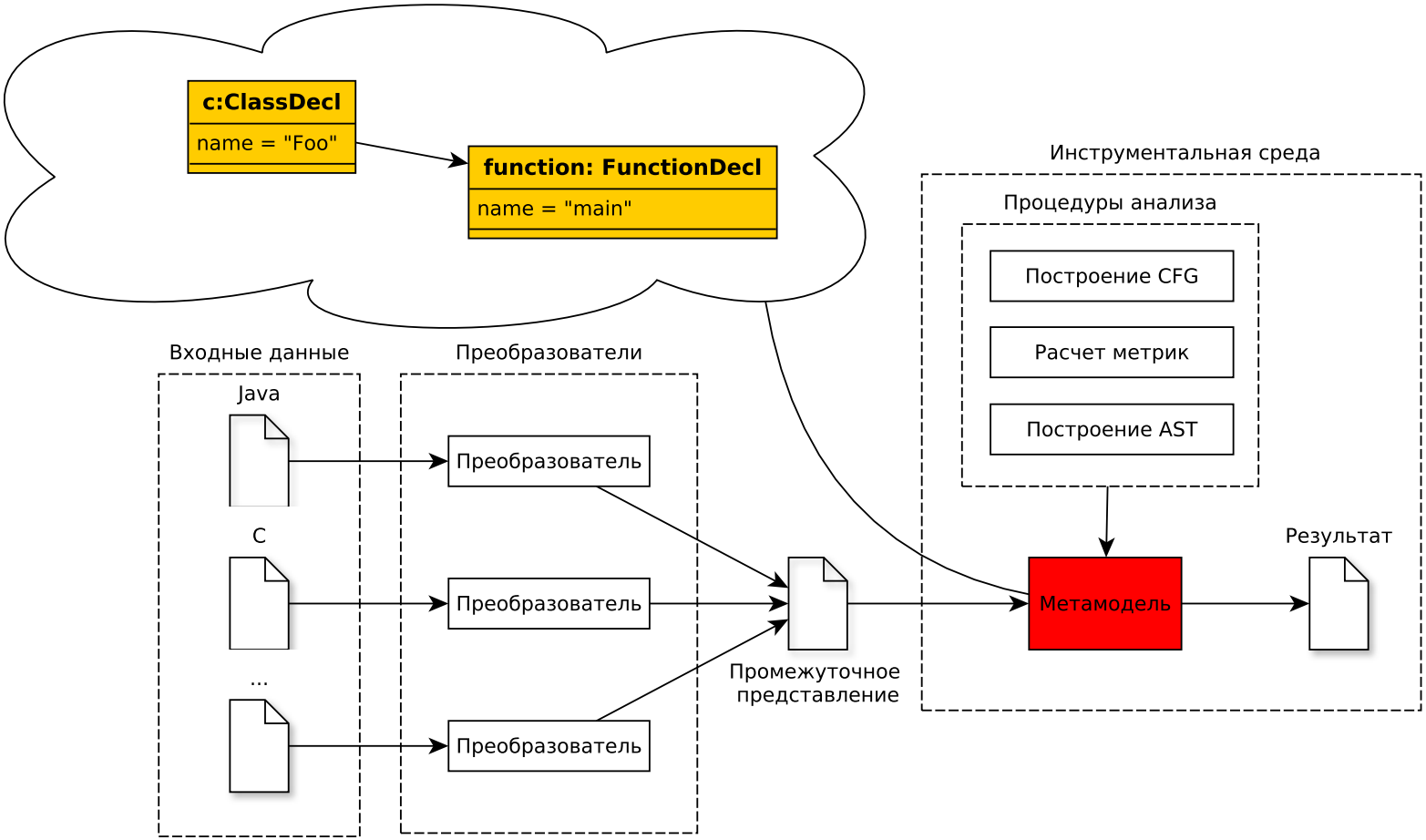
Принцип работы



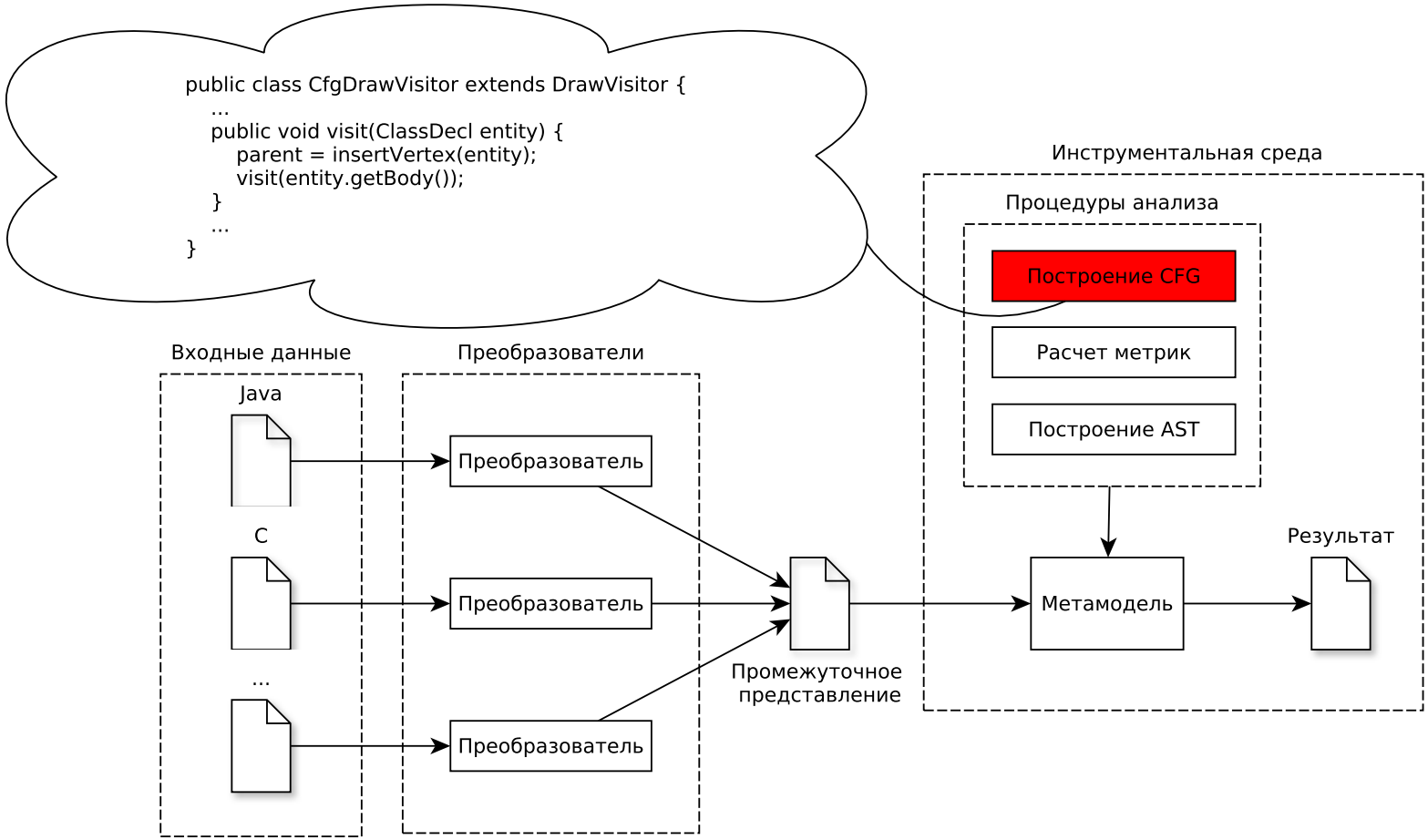
Принцип работы



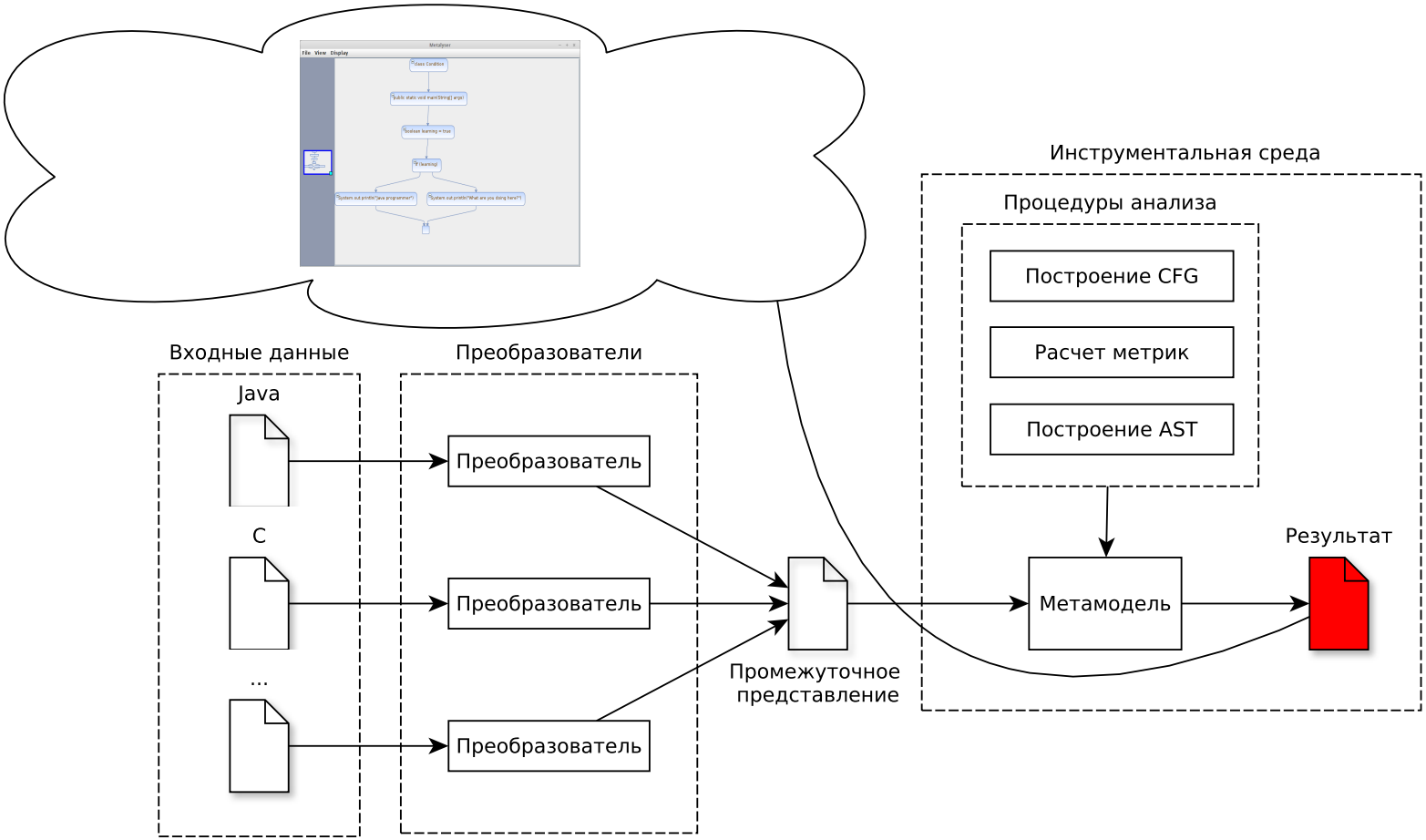
Принцип работы



Принцип работы



Принцип работы



Разработка метамодели

- Стандарт MOF - мета-метамодель
- Архитектура была модифицирована - убраны классы для отношений между сущностями
- Расширяемость - все сущности наследуются от одного класса
- API для обхода модели программы - шаблон “Посетитель” (Visitor)

Разработка преобразователей

- Разрабатываются отдельно для каждого языка программирования
- Разработаны преобразователи для языков Java и C
- Для создания лексических и синтаксических анализаторов использовался генератор парсеров ANTLR

Разработка процедур визуализации и анализа

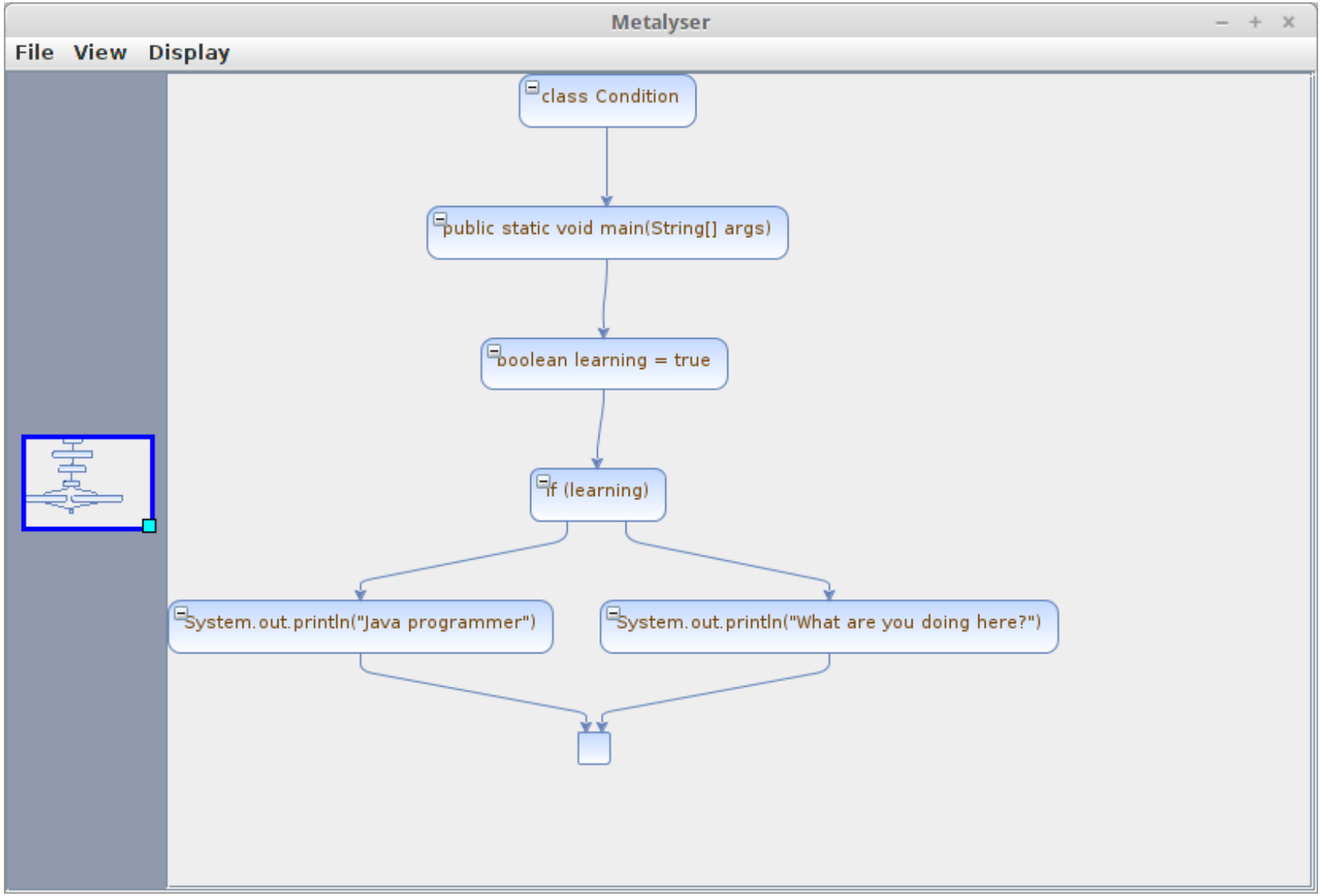
- Графический интерфейс на Swing
- Для отрисовки графов используется библиотека JGraphX
- Визуализация AST и CFG
- Построение UML-диаграмм классов
  - Существуют ограничения в силу недостаточного количества информации о семантике отношений
- Подсчет метрик Лоренца и Кидда

Демонстрация

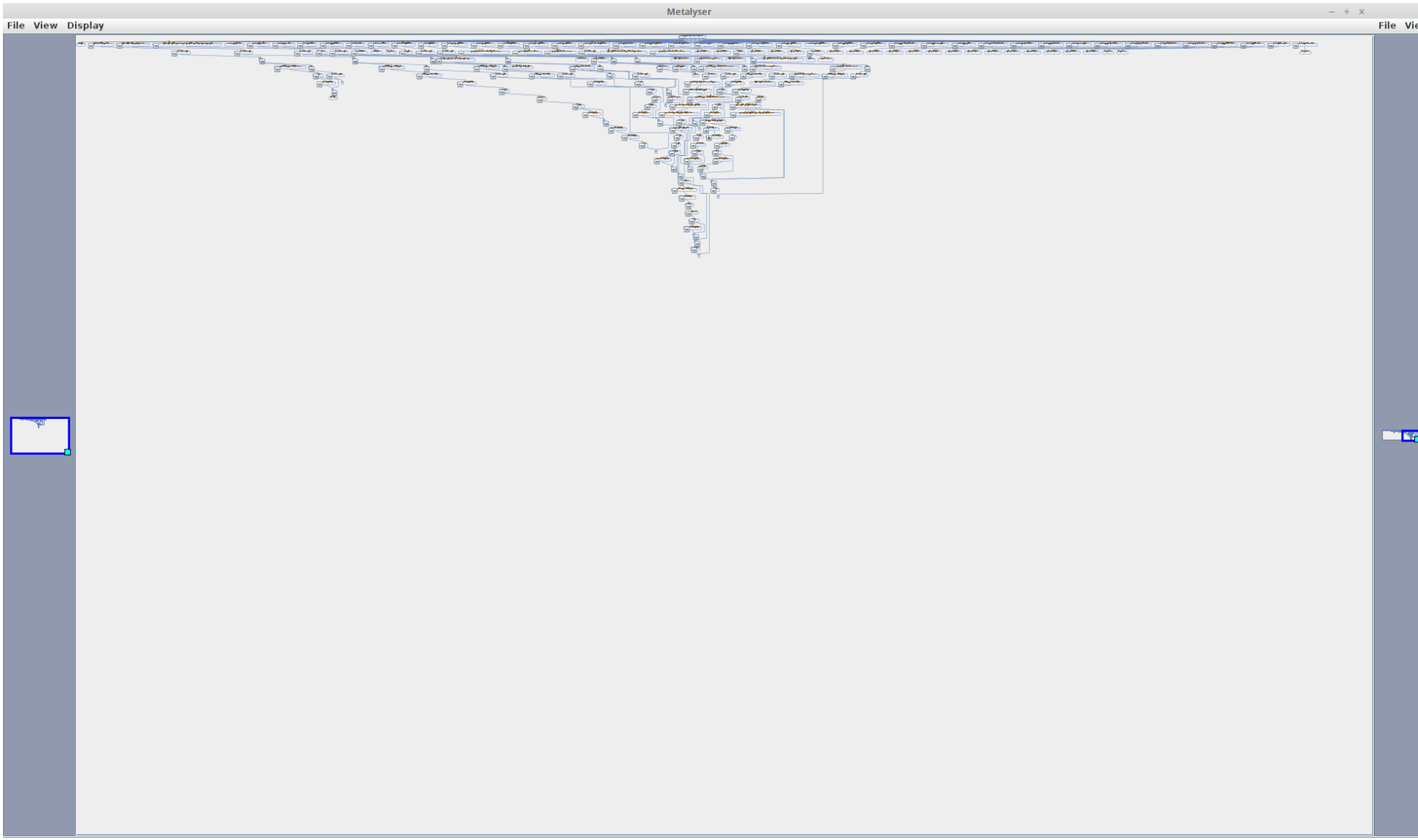
Рассмотрим пример:

```
class Condition {
  public static void main(String[] args) {
    boolean learning = true;

    if (learning) {
      System.out.println("Java programmer");
    } else {
      System.out.println("What are you doing here?");
    }
  }
}
```

Демонстрация. Визуализация CFG



Демонстрация. Визуализация CFG

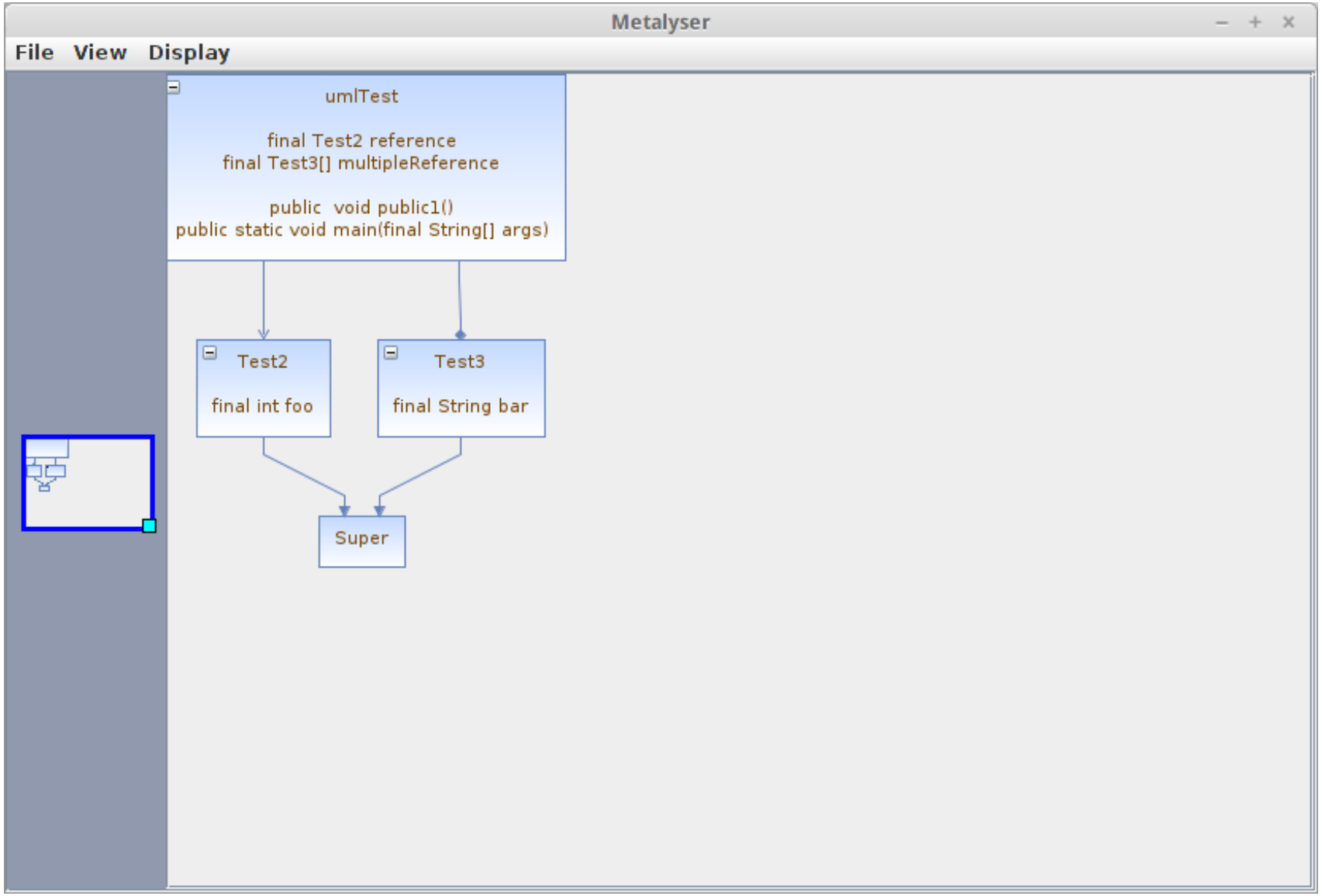


Демонстрация

Еще один пример:

```
public class umlTest {
    private final Test2 reference;
    private final Test3[] multipleReference;

    public void public1() {}

    public static void main(final String[] args) {}
}

class Test2 extends Super {
    private final int foo;
}

class Test3 extends Super {
    private final String bar;
}

class Super {
}
```

Демонстрация. UML-диаграмма классов



Результаты

- Разработана языконезависимая метамодель для унификации процедур анализа и визуализации
- Разработана графическая инструментальная среда для демонстрации возможностей предложенного представления
- Проведено тестирование разработанной системы
- Разработанную метамодель можно применять в качестве библиотеки при создании анализаторов

Направления дальнейшей разработки

- Разработка преобразователей для других языков программирования
- Извлечение новых видов моделей
- Создание языка запросов к метамодели
- Разработка новых видов визуализации
- Расчет дополнительных метрик

Спасибо за внимание!