Анализ программного кода методами машинного обучения

Илия Кузьмина, 444
научный руководитель: к.т.н., доц. Брыксин Т.А.
рецензент: СПбАУ РАН, ст. преп. Шпильман А.А.

Возможные направления

- Поиск одинаковых участков кода
- Оценка качества кода
- Поиск и исправление ошибок в коде
- Поиск (анти)паттернов проектирования
- Автоматическая реструктуризация программ

Реструктуризация программ

- Процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения
- Цель: облегчить понимание логики программы
- Особенности: так же как и оценка результатов, часто производится вручную

$\Rightarrow$ необходимо автоматизированное решение

Поставленные задачи

- Изучить, как в настоящий момент используется машинное обучение при анализе кода
- Выбрать алгоритмы и метрики, на основе которых будет осуществляться реструктуризация
- Реализовать алгоритмы автоматической реструктуризации
- Провести тестирование алгоритмов на "ручных" примерах и реальных программных продуктах

Уровни реструктуризации

- Методы (изменение структуры операторов)
- Классы (изменение структуры методов и атрибутов внутри классов)
- Пакеты (изменение структуры классов внутри пакетов)

Обзор существующих решений

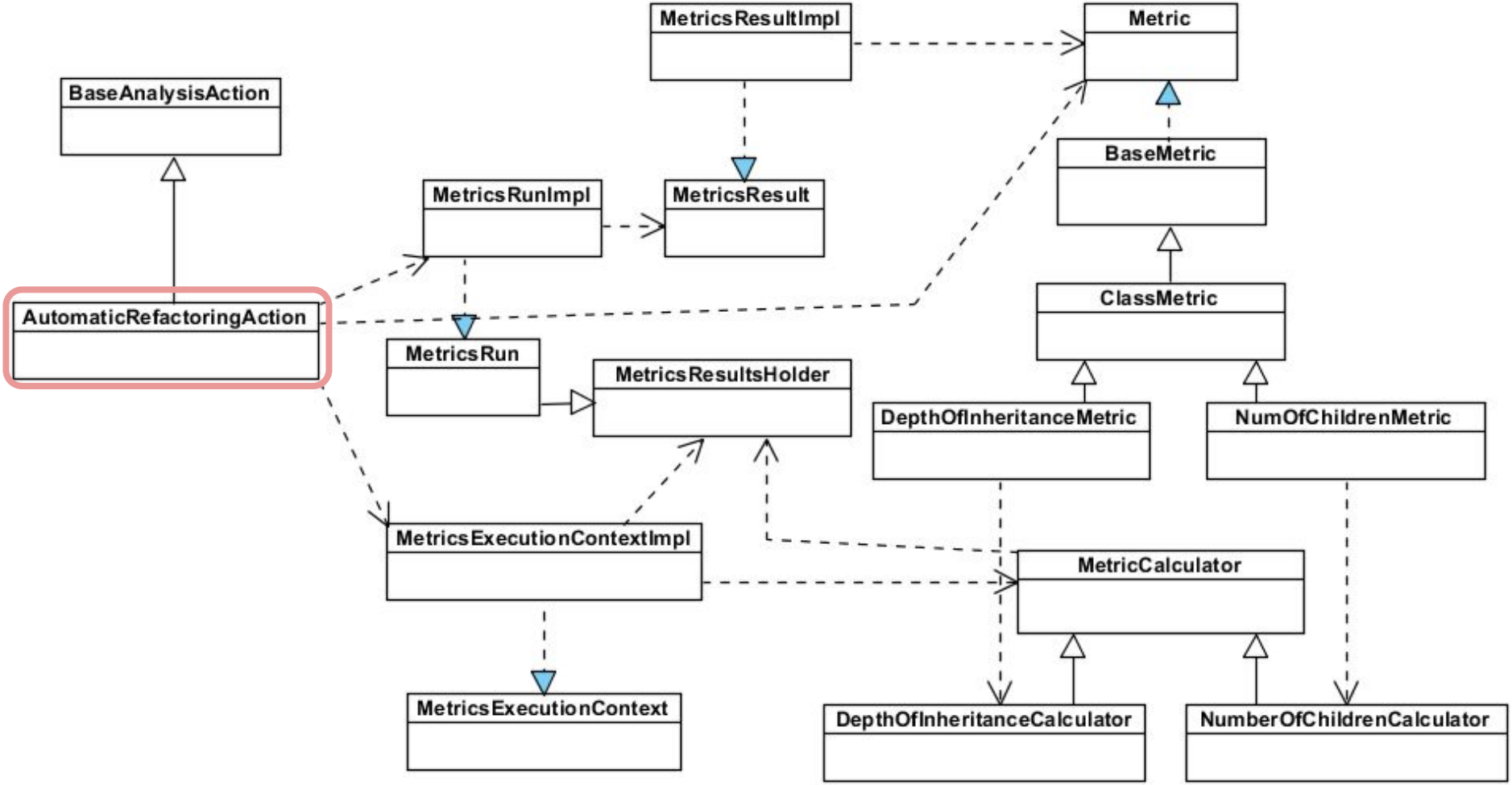
- JDeodorant, 2010
- ARI (Automatic Refactorings Identification)
  - Основная идея - векторизация сущностей и кластеризация
- CCDA (Constrained Community Detection Algorithm)
  - Основная идея - построение графа зависимостей между классами

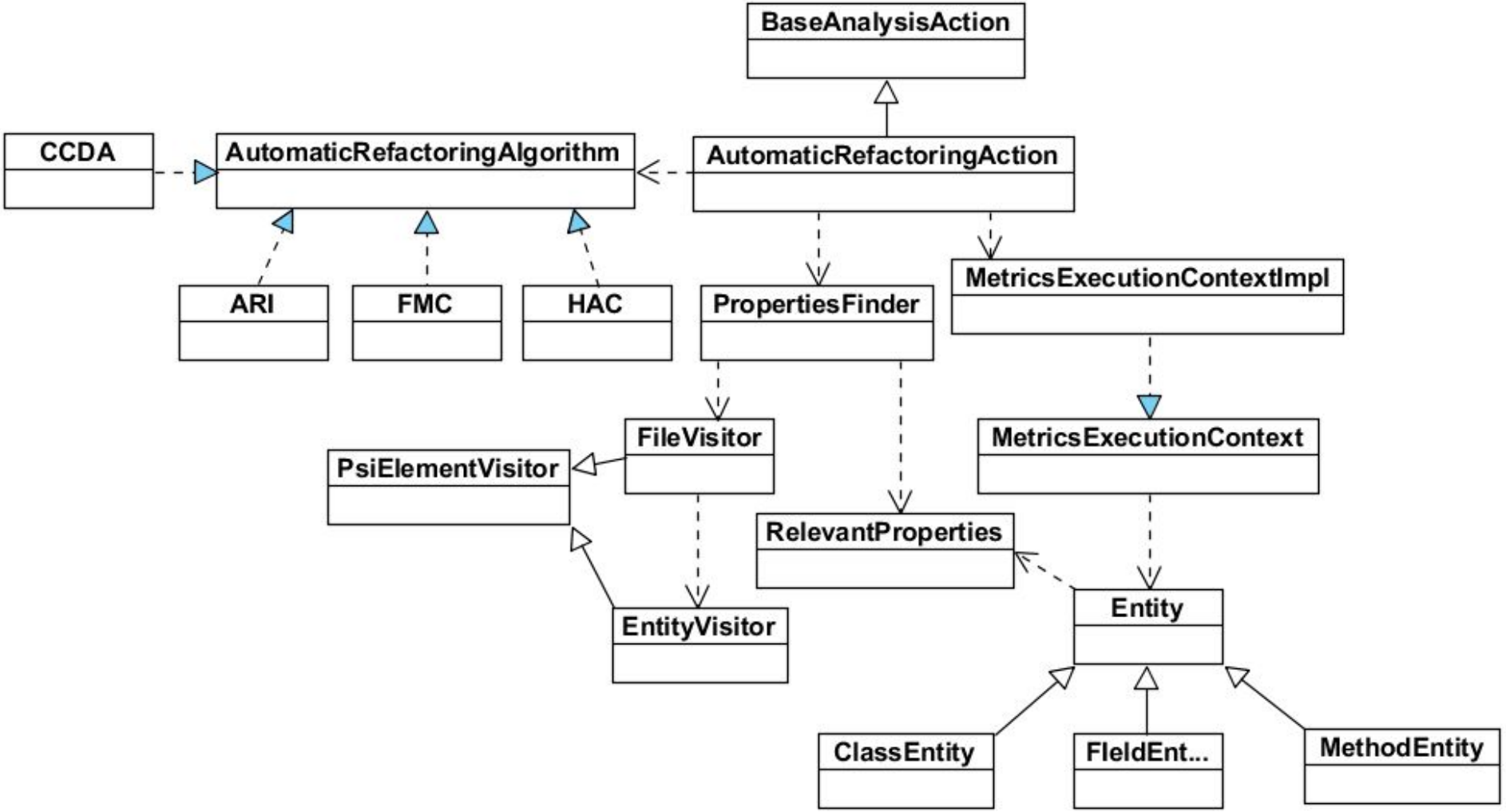
Алгоритмы кластеризации

1. Constrained Community Detection Algorithm (CCDA)
2. Automatic Refactorings Identifications (ARI)
3. Hierarchical Agglomerative Clustering (HAC)
4. Fields and Methods Clustering (FMC)

Реализация

- Вычисление метрик: плагин MetricsReloaded
- Построение RP и графа зависимостей: средства Project Structure Interface (PSI) из IntelliJ Platform SDK


MetricsResultImpl
Metric
BaseAnalysisAction
BaseMetric
MetricsRunImpl
MetricsResult
AutomaticRefactoringAction
ClassMetric
MetricsRun
MetricsResultsHolder
DepthOfInheritanceMetric
NumOfChildrenMetric
MetricsExecutionContextImpl
MetricCalculator
MetricsExecutionContext
DepthOfInheritanceCalculator
NumberOfChildrenCalculator


BaseAnalysisAction
CCDA
AutomaticRefactoringAlgorithm
AutomaticRefactoringAction
ARI
FMC
HAC
PropertiesFinder
MetricsExecutionContextImpl
FileVisitor
PsiElementVisitor
MetricsExecutionContext
RelevantProperties
EntityVisitor
Entity
ClassEntity
FIeldEnt...
MethodEntity

Апробация: первый пример

```
class class_A
{
    static void methodA1()
    {
        attributeA1=0;
        methodA2();
    }
    static void methodA2()
    {
        attributeA2=0;
        attributeA1=0;
    }
    static void methodA3()
    {
        attributeA1=0;
        attributeA2=0;
        methodA1();
        methodA2();
    }
    static int attributeA1;
    static int attributeA2;
}
```

```
class class_B
{
    static void methodB1()
    {
        class_A.attributeA1=0;
        class_A.attributeA2=0;
        class_A.methodA1();
    }
    static void methodB2()
    {
        attributeB1=0;
        attributeB2=0;
    }
    static void methodB3()
    {
        attributeB1=0;
        methodB1();
        methodB2();
    }
    static int attributeB1;
    static int attributeB2;
}
```

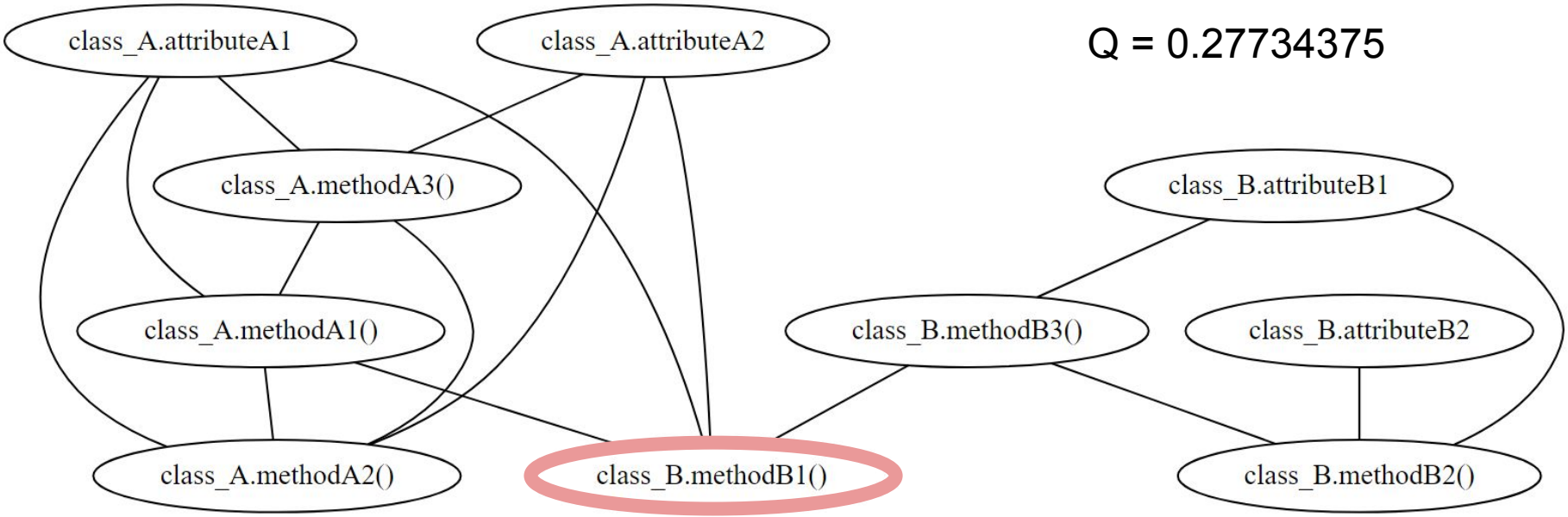
CCDA, первый пример

Quality Index $Q = \sum_{i=1}^{n} (e_{ii} - a_i^2)$

Q = 0.08984375

⇩

Q = 0.27734375



Векторизация сущностей

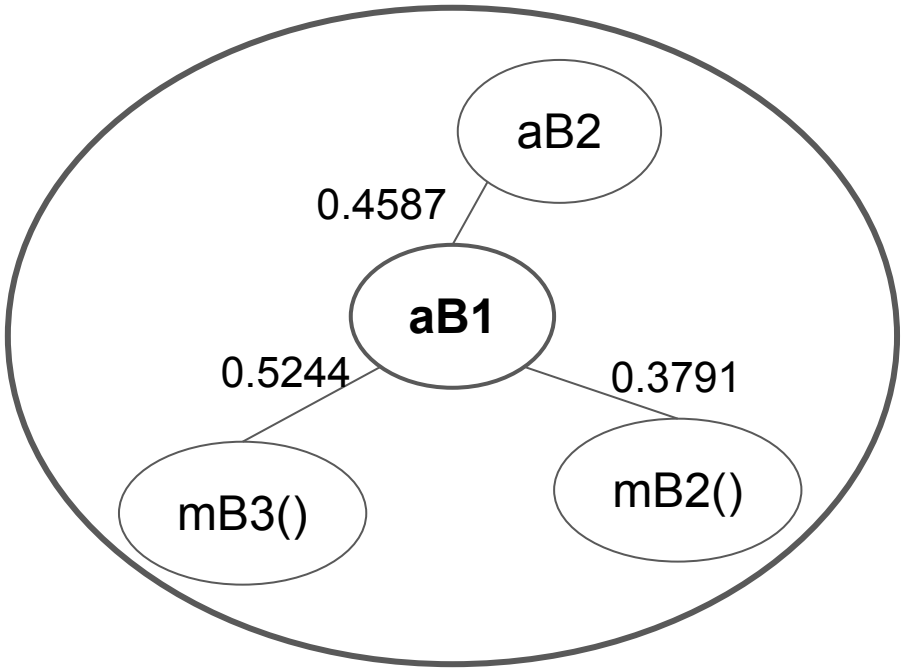
1. Depth in Inheritance Tree (DIT)
2. Number of Children (NOC)
3. Fan-In (FIC/FIM/FIF)
4. Fan-Out (FOC/FOM)
5. Множество связанных сущностей (Relevant Properties, RP)

$$d(X,Y) = \begin{cases} \sqrt{\frac{1}{m+1} * (2(1 - \frac{|RP_X \cap RP_Y|}{|RP_X \cup RP_Y|}) + \sum_{i=1}^{4}(v_{X_i} - v_{Y_i})^2)}, RP_X \cap RP_Y \neq \emptyset \\ \infty, otherwise \end{cases}$$

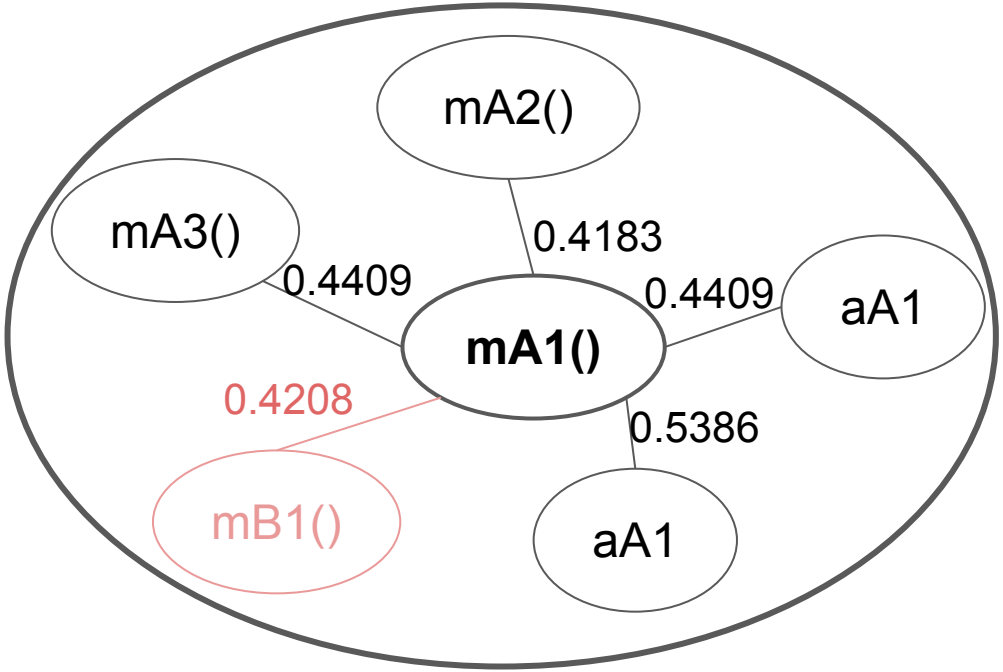
ARI, первый пример

	aA1	aA2	mA1()	mA2()	mA3()	aB1	aB2	mB1()	mB2()	mB3()
class_A	0,4347	0,4295	0,4040	0,3486	0,4208	inf	inf	**0,5**	inf	inf
class_B	0,7150	0,6599	inf	inf	inf	0,4409	0,4677	**0,5193**	0,4040	0,3118

FMC, первый пример

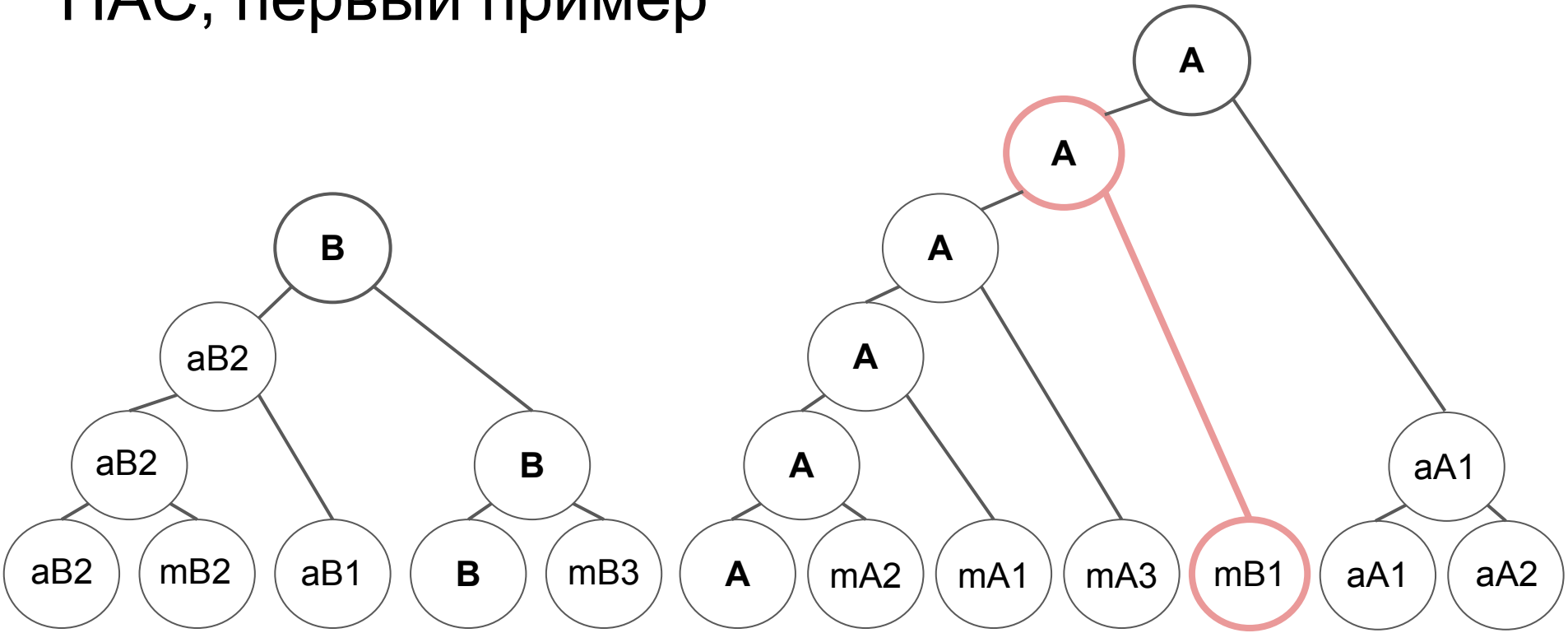


class_B

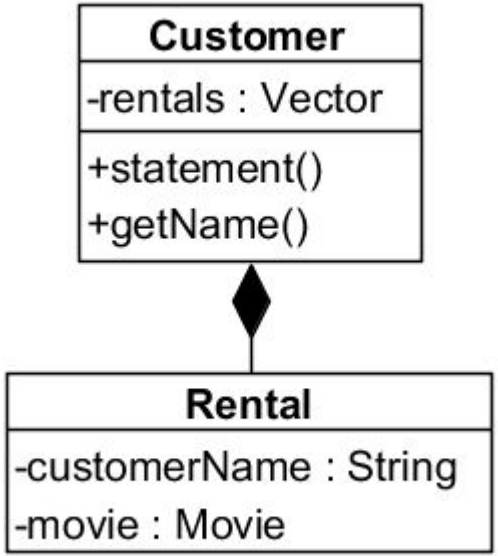


class_A

НАС, первый пример



Второй пример



- CCDA:
  Rental.customerName → Customer
- ARI: ∅
- FMC:
  Rental.customerName → Customer
- HAC: ∅

Третий пример

Исходный код проекта JHotDraw

- 298 классов
- 3242 метода
- 790 поля

1. CCDA: 122 Move Methods
2. ARI: 91 Move Methods, 12 Move Fields
3. FMC: 323 Move refactorings
4. HAC: 85 Move Methods, 4 Move Fields

4 общих, 2 из них корректны

59 общих (3/7 корректны)

Результаты

- Произведён анализ области применения машинного обучения для рефакторинга объектно-ориентированного кода
- Реализован инструмент для автоматической реструктуризации в виде плагина к среде разработки IntelliJ IDEA
  - Модифицированы и реализованы алгоритм выделения сообществ (CCDA) и два алгоритма кластеризации (ARI, HAC)
  - Разработан и реализован алгоритм кластеризации методов и полей (FMC)
- Произведена апробация алгоритмов на трёх примерах, включая исходный код проекта JHotDraw