Проект Architecture Reloaded: автоматический поиск возможных рефакторингов в объектно-ориентированном коде

Тимофей Брыксин

27 марта 2018 г.

Мотивация

Проект Machine Learning for Solving Software Maintenance and Evolution Problems.
Университет Бабеш-Бойяи, Румыния

- *Marian, Z.* Aggregated metrics guided software restructuring. IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2012
- *Marian, Z. Czibula, G., Czibula, I.G.* Using Software Metrics for Automatic Software Design Improvement. Studies in Informatics and Control, 21(3), 2012
- *Marian, Z.* A study on hierarchical clustering based software restructuring. Studia Universitatis Babes-Bolyai, Informatica 57 (2), 2012

Подход №1: построение метрики для системы в целом

- Подсчёт 16 метрик
- Агрегированная метрика класса
  - выбор оптимальных и предельных значений
  - приведение всех метрик к $[0, 1]$, чем меньше, тем лучше
  - сумма всех нормализованных метрик
- Агрегированная метрика системы
  - среднее значение агрегированных метрик классов

Задача кластеризации

1. каждый метод помещаем в свой кластер (класс)
2. повторяем, пока не останется один кластер
   - рассматриваем все варианты слить два кластера; выбираем тот, который даёт меньшую агрегированную метрику системы
   - принимаем получившееся разбиение за текущее
3. из всех получившихся разбиений на шагах 1 и 2 выбираем разбиение с минимальной агрегированой метрикой

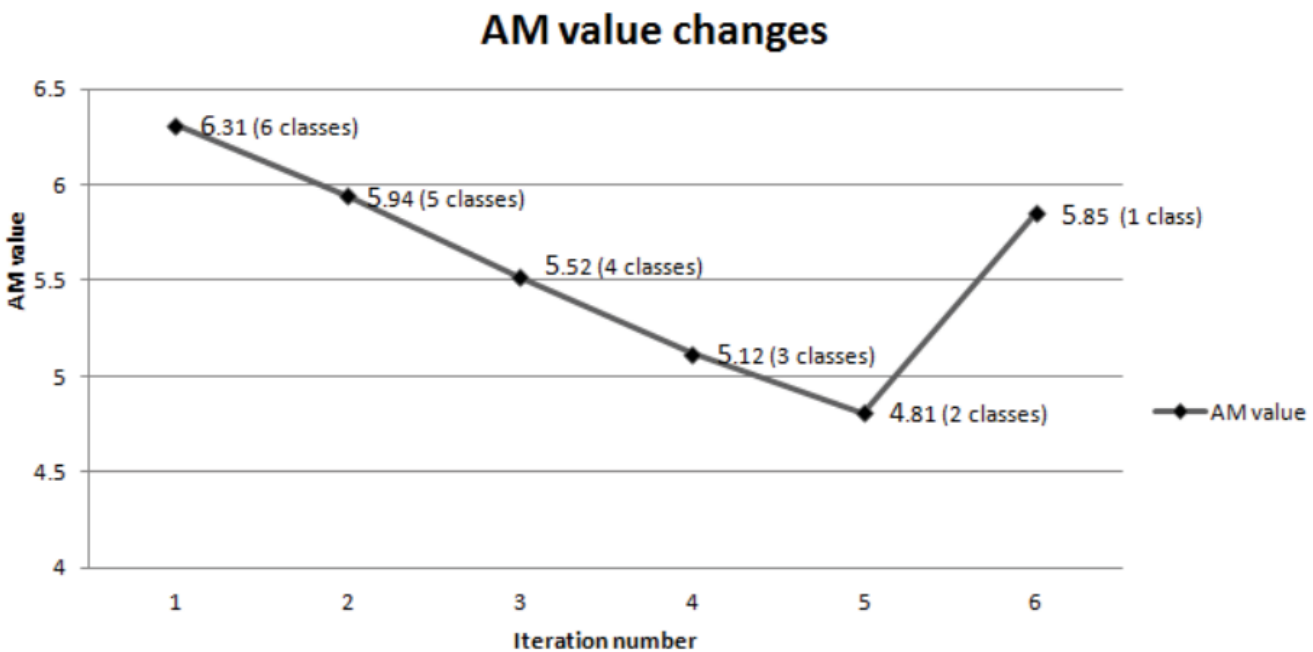
Пример

```java
public class Class_A {
    public static int attributeA1;
    public static int attributeA2;
    public static void mA1(){
        attributeA1 = 0;
        mA2();
    }
    public static void mA2(){
        attributeA2 = 0;
        attributeA1 = 0;
    }
    public static void mA3(){
        attributeA2 = 0;
        attributeA1 = 0;
        mA1();
        mA2();
    }
}
```

```java
public class Class_B {
    private static int attributeB1;
    private static int attributeB2;
    public static void mB1(){
        Class_A.attributeA1=0;
        Class_A.attributeA2=0;
        Class_A.mA1();
    }
    public static void mB2(){
        attributeB1=0;
        attributeB2=0;
    }
    public static void mB3(){
        attributeB1=0;
        mB1();
        mB2();
    }
}
```

Результаты

No.	Partition	Value for metric AM
1	{{mA1()},{mA2()}, {mA3()}, {mB1()}, {mB2()}, {mB3()}}	6.31
2	{{mA1()},{mA2(), mA3()}, {mB1()}, {mB2()}, {mB3()}}	5.942
3	{{mA1()},{mA2(), mA3()}, {mB1()}, {mB2(), mB3()}}	5.52
4	{{mA1(), mB1()},{mA2(), mA3()}, {mB2(), mB3()}}	5.12
5	{{mA1(), mB1(), mA2(), mA3()}, {mB2(), mB3()}}	4.81
6	{{mA1(), mB1(), mA2(), mA3(), mB2(), mB3()}}	5.85



Подход №2

- Каждому классу и методу ставится в соответствие вектор свойств
- Вводится функция расстояния между этими векторами
- На базе неё осуществляется кластеризация

Компоненты вектора

- Relevant properties
  - для метода: сам метод, объемлющий класс, все поля объемлющего класса и методы системы, используемые в методе, все методы системы, перекрывающие метод
  - для класса: сам класс, все его поля и методы, все реализуемые им интерфейсы, все его классы-наследники
- Depth in Inheritance
- Number of Children
- Fan-In
- Fan-Out

Полу-метрика

$s_i = (s_{i1}, s_{i2}, s_{i3}, s_{i4}, s_{i5})$ – вектора свойств (RP + метрики)

$$d(s_i, s_j) = \begin{cases} 0, & \text{если } i = j \\ \sqrt{\frac{1}{m} \cdot \left(1 - \frac{|s_{i1} \cup s_{j1}|}{|s_{i1} \cap s_{j1}|} + \sum_{k=2}^{m} (s_{ik} - s_{jk})^2\right)}, & \text{если } s_{i1} \cap s_{j1} \neq \emptyset \\ \infty, & \text{иначе} \end{cases}$$

Automatic Refactoring Identification (ARI)

Начинаем с пустого разбиения. Для каждого метода:

1. ищем такой класс C, что $d(m, C) < d(m, C') \; \forall C' \in S, C \neq C'$
2. если $d(m, C) < 1$, то можно определить рефакторинг Move Method:
   - если C находится в некотором элементе разбиения k, то добавляем m туда же
   - если C не входит в разбиение, создаём новый кластер и добавляем в него C и m
3. если $d(m, C') = \infty \; \forall C' \in S$, то это рефакторинг Extract Class. Ищем метод m', ближайший к m
   - если такой нашёлся, помещаем m в кластер к нему
   - если $d(m, m') = \infty \; \forall m' \in S$, создаём новый кластер (класс) для m
4. сливаем все классы $C_1 C_2$ такие, что $d(C_1, C_2) < 1$ (рефакторинг Inline Class)

Эксперименты

Пример с Move Method

	Class_A	Class_B	mA1()	mA2()	mA3()	mB1()	mB2()	mB3()
Class_A	0	∞	0.329	0.397	0.528	0.471	∞	∞
Class_B	∞	0	∞	∞	∞	0.482	0.555	0.239
mA1()	**0.329**	∞	0	0.361	0.453	0.406	∞	∞
mA2()	**0.397**	∞	0.361	0	0.596	0.464	∞	∞
mA3()	**0.528**	∞	0.453	0.596	0	0.444	∞	∞
mB1()	**0.471**	0.482	0.406	0.464	0.444	0	0.474	0.471
mB2()	5.49	**0.555**	∞	∞	∞	0.474	0	0.567
mB3()	5.5	**0.239**	∞	∞	∞	0.471	0.567	0

$$K_1 = \{Class\_A, mA1(), mA2(), mA3(), mB1()\}$$

$$K_2 = \{Class\_B, mB2(), mB3()\}$$

Hierarchical Agglomerative Clustering (HAC)

Иерархическая агломеративная кластеризация с методом полной связи и эвристической функцией слияния кластеров

- $D(K_1, K_2) = \max_{x \in K_1, y \in K_2} d(x, y)$
- два кластера сливаются, только если расстояние между ними меньше 1

Большой пример

Фреймворк JHotDraw: 173 класса, 1375 методов, 475 полей классов

- ARI: найдено 20 рефакторингов Move Method, 9 некорректных
- HAC: найдено 158 кластеров, из них 3 новых
  - всего 12 методов, 5 некорректно перемещённых

Программная реализация

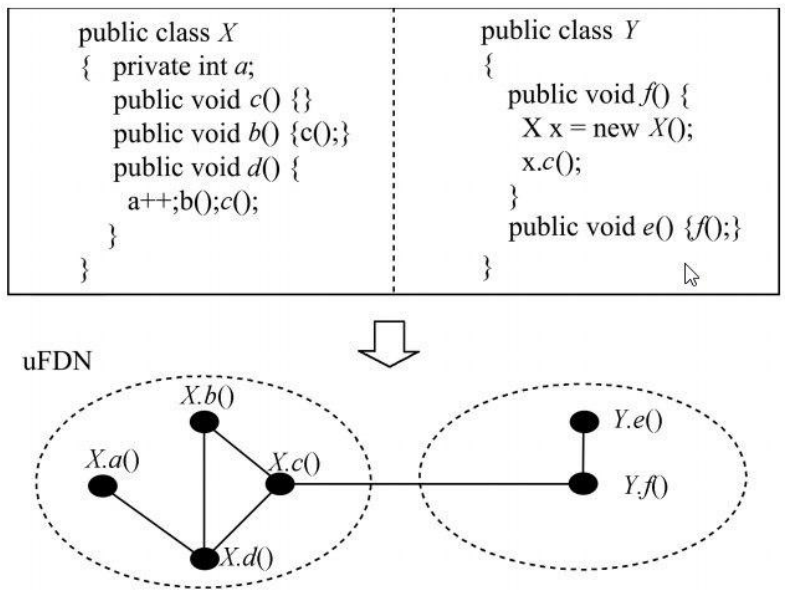
- Авторская реализация FAOS (закрытая и недоступная ☹)
- Плагин MetricsReloaded для IDEA
  - Наборы метрик Martin Packaging, MOOD, Chidamber-Kemerer, LOC, Complexity и др.
  - https://github.com/BasLeijdekkers/MetricsReloaded

Наши эксперименты на JHotDraw 5.1

- ARI: 91 рефакторинг против 20
- HAC: 85 рефакторингов против 12

CCDA: Constrained Community Detection Algorithm

- *Wei-Feng Pan, Bo Jiang, Bing Li*. Refactoring Software Packages via Community Detection in Complex Software Networks // International Journal of Automation and Computing, 2013, pp.157–166.



- Максимизация показателя качества $Q = \sum_{i=1}^{n}(e_{ii} - a_i^2)$

Алгоритм AKMeans

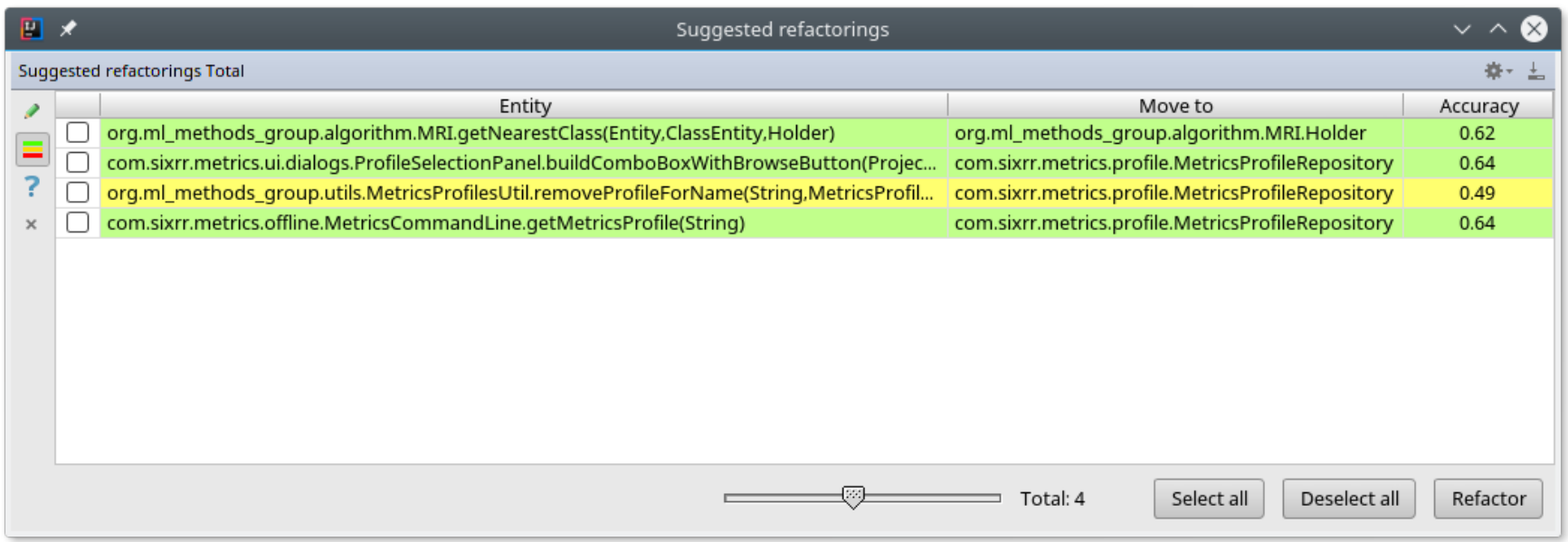
- Илия Кузьмина, ВКР бакалавра, 2017 г.
- Совмещение иерархической кластеризации и метода K-средних
  - случайная инициализация центров кластеров
  - вычисление расстояний от каждой сущности до всех кластеров (метод полной связи)
  - перемещение сущности с ближайшим расстоянием в соответствующий кластер
    - пока не останется нераспределённых сущностей
  - повтор итерации с разбиением N раз

Эксперименты и модификации

- Разные OO-метрики
- Состав множеств Relevant Properties
- Введение весов для свойств Relevant Properties
- Метрика accuracy для предлагаемых рефакторингов
- Механизм голосования

ArchitectureReloaded

- https://github.com/ml-in-programming/ArchitectureReloaded
- https://plugins.jetbrains.com/plugin/10411-architecturereloaded



Эксперименты

- JHotDraw v5.1
  - 117 классов, 860 методов, 294 полей
  - предложено 7 рефакторингов с accuracy >= 0.6
    - 5 из них применимы
- ArchitectureReloaded v0.1
  - 1061 классов, 3247 методов, 642 полей (включая плагин MetricsReloaded)
  - предложено 3 рефакторинга с accuracy >= 0.6
    - 2 из них применимы

Выводы

- Выделение рефакторингов – крайне неоднозначная область
- Рефакторинг vs Направление для улучшения
- Метрика accuracy и пороги для фильтрации
- Интеграция с IDE

Текущая работа и идеи на будущее

- Другие алгоритмы (генетические алгоритмы, теория игр, алгоритмы поиска, data mining, ...)
- Другие рефакториги (Extract Class, Move Field, Inline Class, ...)
- Более умный ансамбль
  - выделение сильных сторон алгоритмов
  - динамический выбор алгоритмов
- Дообучение на данных
  - предпочтения пользователя по рефакторингам
  - предпочтения пользователя по метрикам
- Интеграция с IDEA
  - Аннотатор
  - Инкрементальный пересчёт
- Платформа для комплексного сравнения алгоритмов

Генерация кода в IDEA при помощи Bayou

- Генерация кода на Java по набору вызовов API и типов данных
- Bayesian Sketch Learning подход
  - Murali, V. et. al. *Neural Sketch Learning for Conditional Program Generation.* ICLR 2018 (https://arxiv.org/abs/1703.05698)
- Плагин к IDEA
  - поддержка Android SDK и Java stdlib
  - https://github.com/ml-in-programming/bayou-integration
- Владислав Танков, ВКР бакалавра
- Публикация на SEIM-18

Поиск аномалий в коде на Kotlin

- Аномалии как редко встречающиеся структурные конструкции в коде
  - излишне сложные конструкции
  - нестандартные использования элементов языка
  - ...
- Данные – все репозитории на Kotlin с Github
  - 27 типов аномалий, с примерами кода
- Классификация аномалий и определение их сути
- https://github.com/ml-in-programming/kotlin-code-anomaly
- Кирилл Смиренко, ВКР бакалавра

Исправление ошибок в решениях на Stepik

- Большая база пар (<неуспешное решение>, <успешное решение>)
- Выделение исправлений и определение сути исправлений
  - функция расстояния на основе разницы в AST
  - кластеризация
- Поиск ближайшего верного решения, получения сути требуемого исправления
- Current status: эксперименты и настройка алгоритмов
- TODO: внедрение в Stepik и апробация на людях

Контекстная помощь разработчикам в IDEA

- Как помочь разработчику?
  - отображение релевантной документации
  - отображение релевантных участков кода
- Определение локального контекста и понимание его смысла
  - определение сути проблемы разработчика
- Ранжирование результатов выдачи
- Как тестировать?
  - метрики
  - usability-тестирование
- Михаил Кита, ВКР бакалавра

Анализ структуры и стиля кода

- Сведение к классической задаче ML
  - объекты — участки кода
  - характеристики — особенности стиля и структуры
  - стандартные методы классификации
- Применения
  - поиск на github программистов с похожим стилем
  - детектор плагиата в коде на Java
  - ...
- Основная сложность: векторизация исходного кода
  - лексические характеристики
  - синтаксические характеристики
  - неявные характеристики (LSA, deep features, ...)

Выделение имён методов по их реализации

- Имя метода = action + target
- Задача – выделить action и target из реализации метода
  - Yu, S., Zhang, R„ Guan, J. Properly and Automatically Naming Java Methods: A Machine Learning Based Approach. // International Conference on Advanced Data Mining and Applications, 2012
- Лексические и структурные характеристики + NLP (TF-IDF и т.п.)
- Java vs Python

Генерация условий задач по заданным критериям

- Концепция адаптивных курсов для MOOC
  - генерация условий задач
  - входные данные: история пользователя, сложность, теги
- Где взять данные? (много данных)
  - задачи на Stepik, Codeforces, HackerRank, ...
  - комментарии к коду на Stackoverflow
  - python docstrings в коде с Github
- Задача генерации текста по набору характеристик
  - RNN, VAE, GAN, ...

Покоммитный анализ репозиториев на github

- Подсчёт комплексных метрик, расширенная статистика по проекту
  - выявление предрасположенностей разработчиков
- Предсказание событий на основе исторической информации
  - какие методы поменяются в следующих 5 коммитах, кто сделает коммит и т.п.
- Анализ ревью кода в pull-request'ах
  - и распространение изменений на остальной проект

Применение методов машинного обучения в области программной инженерии

- Межвузовская исследовательская группа
- Еженедельный семинар
  - понедельник, 19:00, БЦ «Таймс», 4 этаж (в этом семестре)
- https://research.jetbrains.org/ru/groups/ml_methods
- https://github.com/ml-in-programming
- timofey.bryksin@gmail.com