Ten Years of JDeodorant: Lessons Learned from the Hunt for Smells

Евгений Новожилов

Что такое JDeodorant

- Eclipse plugin
- 1й релиз – ноябрь, 2007 год
- “For the moment, the tool identifies five kinds of bad smells, namely Feature Envy, Type Checking, Long Method, God Class and Duplicated Code.”

Сценарий использования

1. Identify where the software should be refactored.
2. Determine which refactoring(s) should be applied to the identified places.
3. Guarantee that the applied refactoring preserves behavior.
4. Apply the refactoring.
5. Assess the effect of the refactoring on quality characteristics of the software or the process
6. Maintain the consistency between the refactored program code and other software artifacts

Обзор

1. Статистика использования JDeodorant и упоминания в литературе
2. Обзор аналогов и сравнение технологий
3. Наблюдения, полученные с использованием JDeodorant
4. Взгляд на 10 лет развития с точки зрения разработки

Использование и цитирование

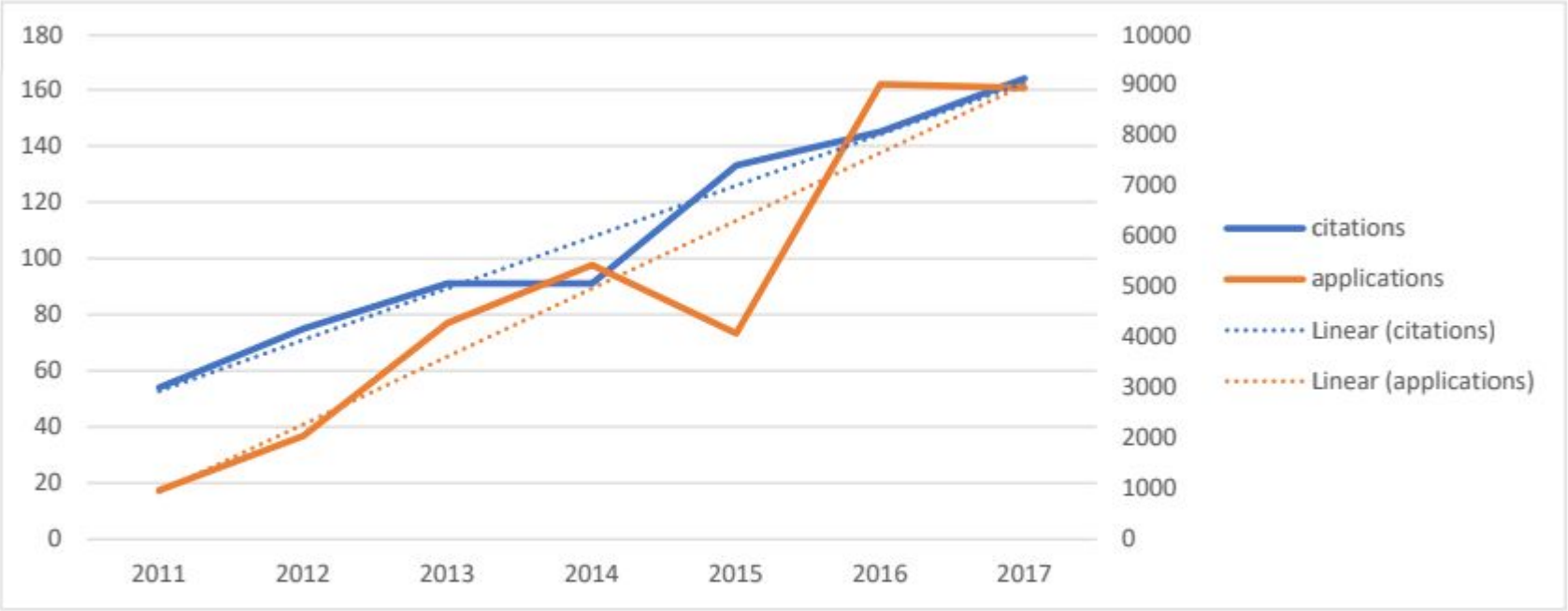

Fig. 1. Impact of JDeodorant.

Применение рефакторингов

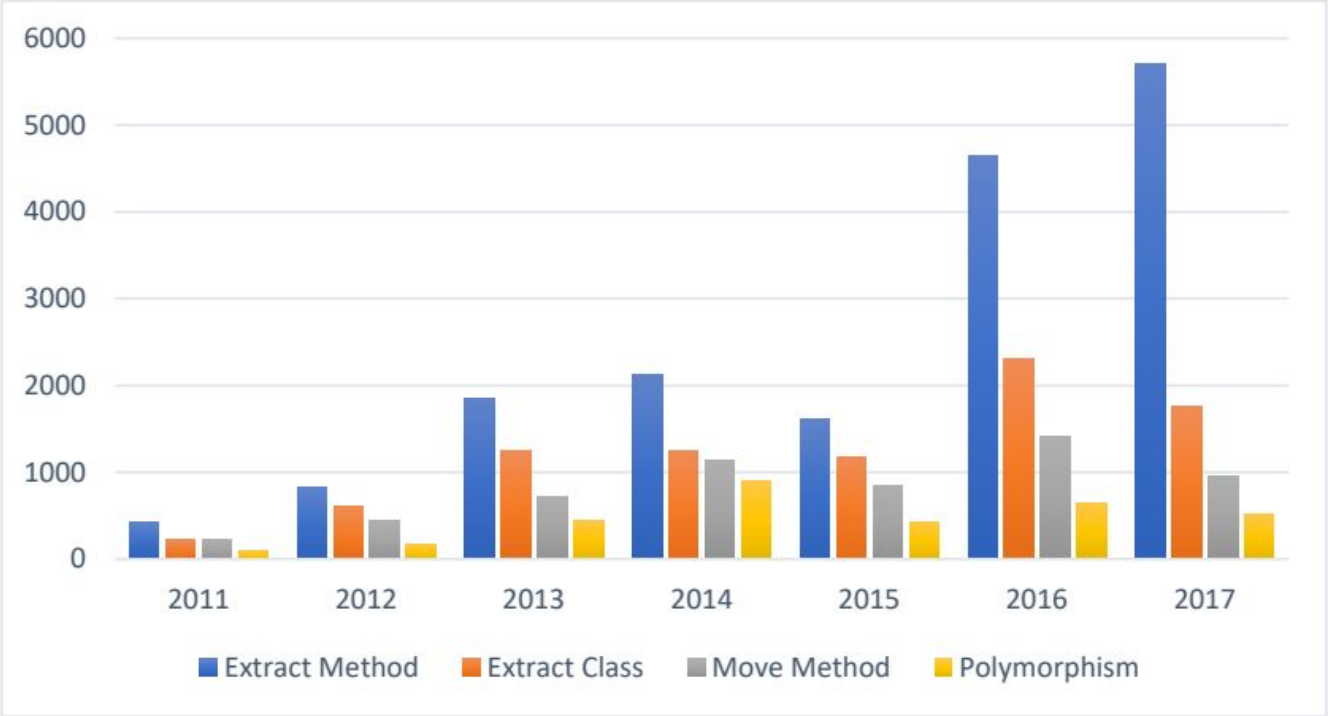

Fig. 2. Number of applied refactorings per refactoring type.

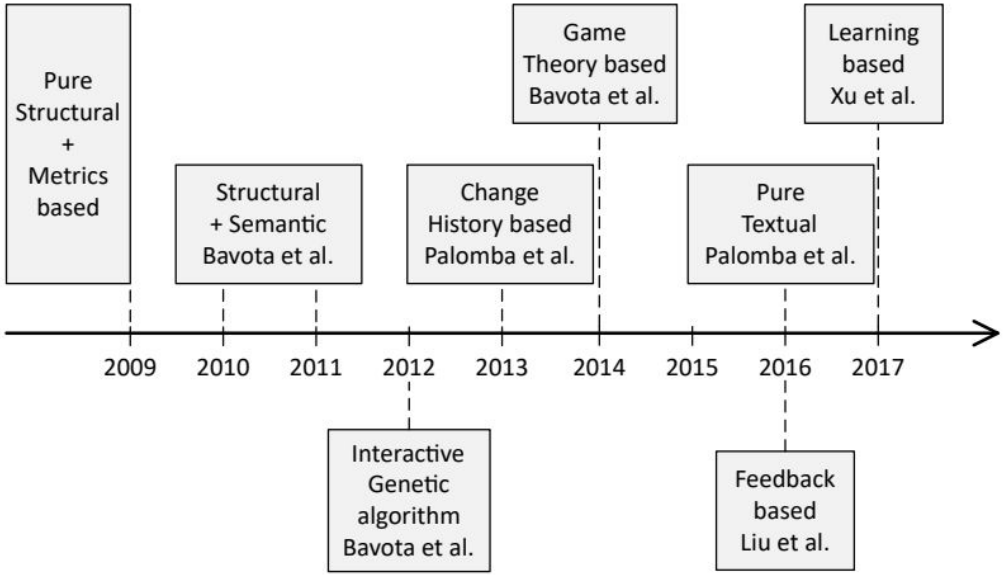

Fig. 3. Important milestones in the evolution of code smell detection and refactoring recommendation systems.

Move Method

- MethodBook
  - Структурный + семантический анализ
  - Relational Topic Models
- HIST
  - Первый инструмент, использующий информацию из change sets
  - “HIST achieved a precision of 78% and a recall of 77%, while JDeodorant achieved a precision of 65% and a recall of 71%. Moreover, there was a 54% overlap between the Feature Envy code smells detected by HIST and JDeodorant”

Move Method (cont.)

- JMove
- TACO
  - LSI
- RefactoringNavigator
  - (Code, Design) -> Guidance
  - Recommendations
- DominoEffect
  - Перемещение одного метода может вызвать перемещение других методов
- c-JRefRec
  - Геометрический коэффициент между вектором класса и метода

Метрики JDeodorant в исследованиях

- Общий вывод: все подходы лучше, чем JDeodorant

Алгоритм->	HIST	JMove	TACO	Domino	c-JRefRec
Precision	0.65	0.15	0.57	<< 0.76	0.38
Recall	0.71	0.4	0.69		0.25

Extract Method

- JExtract
  - Similarity of their dependency sets extracted from each individual statement
- SEMI
  - Cohesion between pairs of statements
- GEMS
  - More metrics + learning; 2 sets, 267 and 5598 records in each one
- Вывод: все предложенное выше снова лучше

Наблюдения в исследованиях

- Текстовые рефакторинги проще понимать и применять
- Структурные рефакторинги кажутся менее очевидными
- Применение любого вида рефакторингов положительно влияет на метрику Tight Class Cohesion
- Применение рефакторингов в мобильных приложениях увеличивает потребление энергии с 7% до 80%
- Long Method встречается в 2 раза чаще в мобильных приложениях

Взгляд разработчиков

- Инструмент должен быть разработан с учетом реальных сценариев использования и работы разработчиков
  - Предлагать делать рефакторинг при решении задачи, нежели заставлять делать рефакторинг отдельно
- Анализ исходного кода стоит производить с использованием абстракций на AST
- Стоит собирать статистику использования как можно раньше
- Инструмент, в первую очередь, должен быть плагином к среде разработки, а не отдельным приложением

Взгляд разработчиков (продолжение)

- Инструмент должен требовать минимальное ресурсов для установки и настройки
- Инструмент стоит проверить на коммерческих проектах
- Инструмент стоит выкладывать в open-source как можно раньше
- И, конечно же, должна быть документация и инструкция