Автоматическое изменение кода на Python в соответствии в графовыми шаблонами

Автор: Смирнов Олег Евгеньевич, 18.Б11-мм
Научный руководитель: к. т. н., доцент Брыксин Т. А.

Санкт-Петербургский государственный университет
Кафедра системного программирования

12 июня 2021г.

Введение

- Автоматическое исправление и рефакторинг кода — важная и актуальная задача, особенно для разработчиков IDE
- Её можно решать на этапе статического анализа кода
- Чтобы понять, какие изменения в коде разработчики совершают действительно часто, можно обратиться к истории коммитов в открытых репозиториях
- А чтобы локализовать сложные и даже *распределённые* шаблоны возможных ошибок, можно использовать графовые представления кода
  - Под *распределёнными* здесь понимаются шаблоны, в которых потенциально участвуют изолированные элементы из нескольких строк кода, связанные зависимостями по данным или потоку управления

Цель и задачи

- **Цель:** разработка инструмента для автоматического исправления кода на Python в соответствии с графовыми шаблонами изменений в IDE PyCharm
- **Задачи:**
  - Проанализировать существующие подходы к поиску и автоматическому исправлению ошибок в коде
  - Проанализировать существующие подходы к сбору графовых шаблонов изменений в коде
  - Реализовать алгоритм локализации графовых шаблонов в коде
  - Реализовать механизм исправления локализованного фрагмента в соответствии с шаблоном
  - Представить указанную функциональность в виде плагина для IDE PyCharm
  - Провести апробацию плагина с участием реальных разработчиков

Обзор предметной области

- **Локализация** «ошибок» в коде
  - Поиск по шаблону (сравнение AST, регулярные выражения)
  - Вероятностные методы
  - Символьные вычисления
  - ...
- **Изменение** кода в соответствии с шаблоном
  - Авто-замены фрагментов кода
  - Применение сценариев редактирования
  - Методы нейронного машинного перевода

Обзор предметной области

- **Поиск** шаблонов изменений кода
  - CPatMiner[1] — поиск часто встречающихся «изоморфных» подграфов в графовых представлениях изменений кода на Java
  - PythonChangeMiner[2] — адаптация указанного подхода для языка Python

[1] Nguyen H. A. et al. Graph-based mining of in-the-wild, fine-grained, semantic code change patterns //2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). – IEEE, 2019. – С. 819-830.

[2] `https://github.com/JetBrains-Research/python-change-miner`

Предлагаемый подход к использованию

1. Собрать и отфильтровать набор шаблонов изменений кода на Python.
2. Загрузить их в скрипт автоматической сборки плагина для PyCharm.
   - Либо использовать готовый прототип плагина с 9-ю шаблонами улучшений кода
3. При запуске плагин будет:
   - Локализовать подграфы шаблонов в коде разработчика
   - При подтверждении исправлять код в соответствии с шаблоном

Пример шаблона изменения

До:

```
for i in range(len(static)):
    dim = static[i]
```

После:

```
for i, dim in enumerate(static):
```

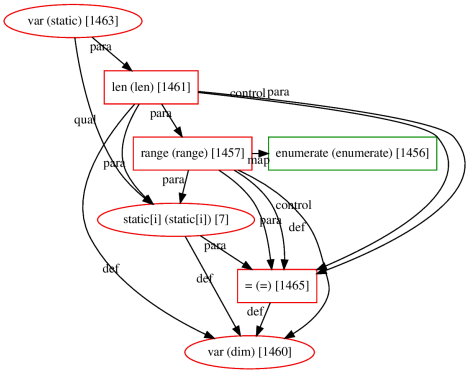

Рис.: Пример графового шаблона изменения кода на Python.

Детали реализации

- Плагин запускает *инспекцию*, которая:
  - Автоматически строит граф зависимостей для открытого в редакторе кода
  - Ищет в нем подграфы шаблонов из части «до» изменения с помощью проверок на изоморфизм (`JGraphT`)
- При подтверждении пользователем происходит автоматическое исправление фрагмента кода, реализованное с помощью последовательного применения сценариев редактирования AST (`GumTree`)
- Предварительная обработка включает в себя:
  - Выделение общей подпоследовательности изменений AST в каждом шаблоне
  - Автоматическое обобщение имён переменных
  - Подготовку текста всплывающей подсказки в IDE

Архитектура

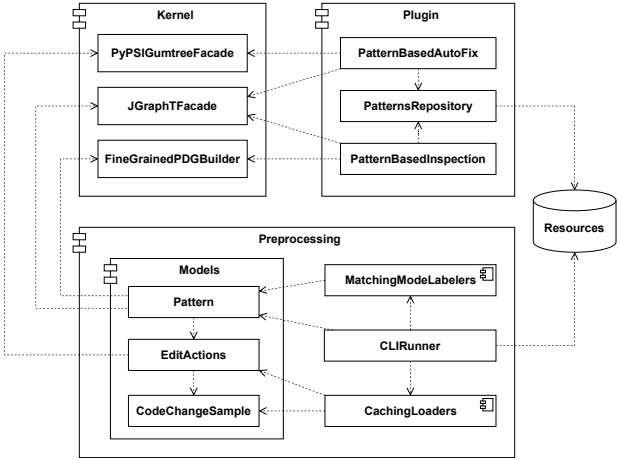

Рис.: Диаграмма компонентов плагина.

Пример работы плагина

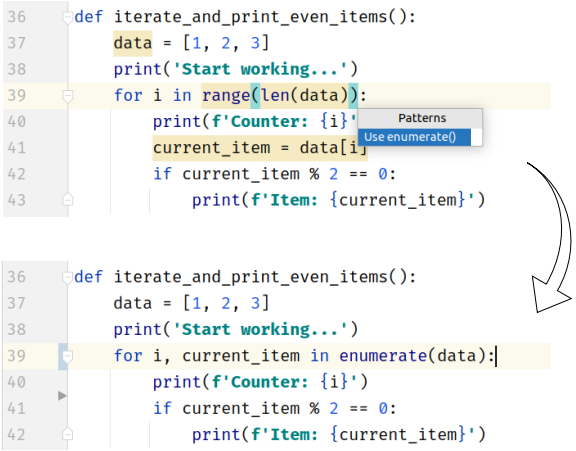

Рис.: Локализация и исправление шаблона в IDE PyCharm.

Апробация

- Была проведена среди 9 разработчиков с профессиональным опытом от двух до пяти лет
- Участники опроса должны были оценить различные аспекты работы плагина по шкале от 1 (неудовлетворительно) до 4 (отлично)
- Средние оценки:
  - Корректность производимых плагином эквивалентных преобразований кода: 3.66 из 4
  - Отсутствие влияния на производительность IDE: 3.88 из 4
  - Визуализация подсказок в редакторе кода: 3.33 из 4
  - Удобство использования автоматических исправлений: 3.77 из 4
- Все участники опроса согласились с потенциальной пользой идеи рекомендации подсказок в IDE, полученных из шаблонов частых изменений кода на GitHub

Результаты

- Предложен новый расширяемый подход к улучшению кода на основе графовых шаблонов частых изменений
- Реализован алгоритм локализации графового шаблона в коде на Python с использование проверок на изоморфизм подграфов
- Реализован процесс внесения исправлений в код с использованием сценариев редактирования
- Реализован скрипт автоматической предобработки шаблонов, полученных с помощью PythonChangeMiner
- Разработан плагин[3] для IDE PyCharm
- Проведена апробация плагина с участием реальных разработчиков

[3] https://github.com/JetBrains-Research/revizor