Разработка аспектно-ориентированного расширения для языка Kotlin

Скрипаль Б.А. Ицыксон В.М.

Санкт-Петербургский политехнический университет Петра Великого

21 апреля 2017 г.

Введение

- Аспектно-ориентированный подход позволяет описывать и внедрять сквозную функциональность
- Представлен в 1997 году Грегором Кичалесом
- Используется совместно с объектно-ориентированным подходом

Задачи, решаемые АОП

- Протоколирование
- Обработка ошибок
- Проверка прав доступа
- Проверка пост- и предусловий
- Трассировка
- ...

Реализации АОП для языков программирования

- Java
  - Spring AOP
  - Aspect
  - JAML
  - CaesarJ
  - ...
- C#
  - PostSharp
  - Aspect.NET
  - AspectC#
  - ...
- Python
  - Aspyct
  - PyPy
  - PEAK
  - Lightweight Python AOP
  - ...
- C/C++
  - AspectC
  - AspectC++
  - FeatureC++
  - ...

Kotlin?

Основные понятия

- Аспект (aspect) — сущность, инкапсулирующая в себе сквозную функциональность
- Точка внедрения (join point) — точка в программе, к которой должна быть применена сквозная функциональность
- Срез (pointcut) — множество всех точек внедрения, к которым должна быть применена сквозная функциональность
- Совет (advice) — сущность, содержащая функциональность, которая должна быть применена к точке внедрения

Способы описания аспектов

- Аннотации
- Расширения целевого языка
- Аннотирующие комментарии
- Специальные классы
- XML
- ...

Способы внедрения сквозной функциональности

- Статический:
  - На уровне исходных кодов
  - Во время компиляции
  - Сразу после компиляции
- Динамический:
  - При помощи прокси-объектов
  - Во время загрузки файлов в JVM
  - ...

Описание аспектов для Kotlin

- База — AspectJ
- Адаптация под особенности языка Kotlin

Адаптация синтаксиса AspectJ к языку Kotlin

- Приведение описания функций к виду, используемому в Kotlin
- Изменение стандартных типов Java на типы Kotlin
- Изменение модификаторов полей и методов
- Добавление возможности задания атрибутов аргументов методов
- Добавление поддержки функций-расширений (extension functions)

Пример описания аспекта для языка Kotlin

```
aspect A  {
    pointcut fooPC(): execution(fun Foo.*())
    pointcut printPC(): call(public !extension fun
        kotlin.io.pri*(Any?))

    before(): fooPC() && printPC() {
        println("Hello before!!")
    }

    after(): fooPC() && printPC() {
        println("Hello after!!")
    }
}
```

Реализованные возможности

- Вставка советов:
  - before
  - after
  - around
- Описание срезов:
  - call
  - execution
- Поддержка extension функций
- Задание nullability модификаторов

Внедрение аспектов

Статический способ внедрения на уровне промежуточного представления

1. Построение PSI
2. Построение модели аспектов
3. Разметка элементов PSI тегами, соответствующим срезам
4. Внедрение кода советов
5. Компиляция PSI

Внедрение аспектов

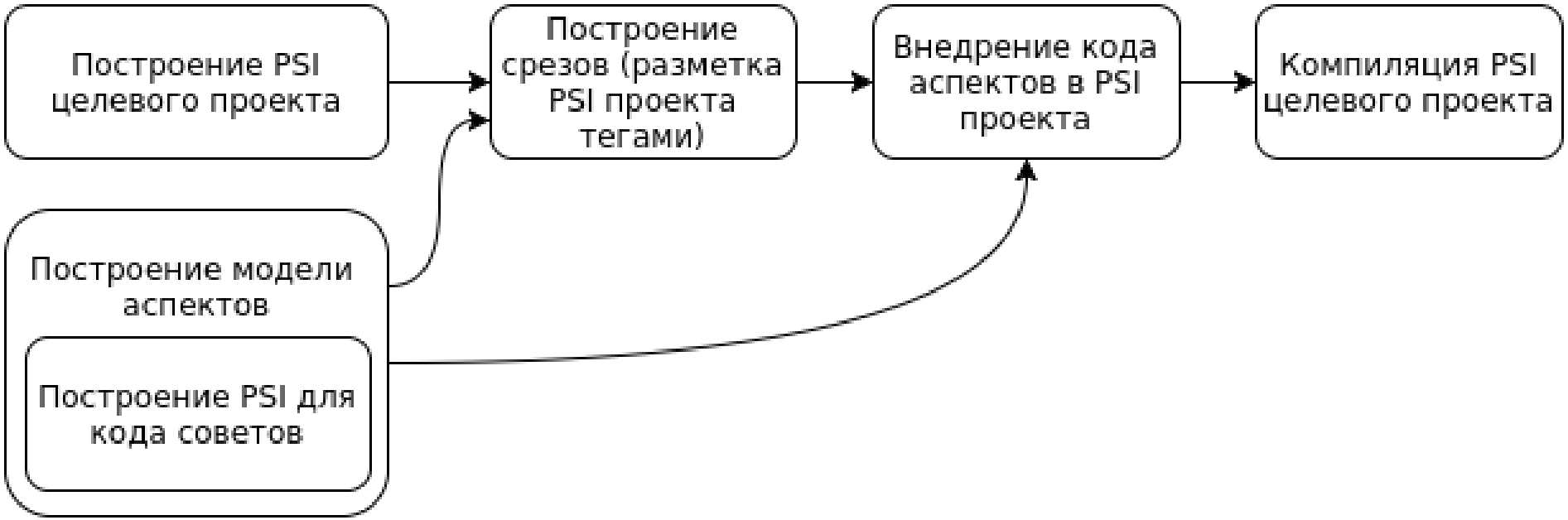

Рис.: Процесс внедрения аспектов

Внедрение кода советов

При применении совета к вложенному вызову используем лямбда-wrapper «run»
Выражение

```
val a = a.foo().bar()
```

преобразуется в:

```
val a = a.
        run{
                val buf = foo()
                advice_code
                buf}
        .bar()
```

Пример аспекта

```
aspect A  {
    pointcut fooPC(): execution(fun Foo.*())
    pointcut printPC(): call(public !extension fun
        kotlin.io.pri*(Any?))

    before(): fooPC() && printPC() {
        println("Hello before!!")
    }

    after(): fooPC() && printPC() {
        println("Hello after!!")
    }
}
```

Пример применения аспектов

```
class Foo {
  fun foo() {
    run{
      val ____a = run{
        println("Hello before!!")
        println("Hello foo!!")
      }
      println("Hello after!!")
      ____a
    }
  }

  fun bar() {
        println("Hello bar!!")
  }
}
```

Тестирование прототипа

- Искусственные примеры
- Студенческие проекты (размер в несколько сотен строк)

Время применения советов к программе занимает до нескольких секунд

Дальнейшие исследования

- Реализация возможностей, существующих в AspectJ
  - Структуры описания срезов (args, cflow)
  - Способы применения советов (afterthrowing, afterreturning)
  - Описание функций и переменных внутри аспекта
  - Обращение к аргументам функций
- Учет возможностей Kotlin
  - inline функции
  - ...
- Тестирование и отладка
  - Улучшение тестов
  - Увеличение размеров и сложности тестовых проектов

Контактные данные

Санкт-Петербургский политехнический университет
Петра Великого, кафедра КСПТ

- Скрипаль Б.А. *skripal@kspt.icc.spbstu.ru*
- Ицыксон В.М. *vlad@icc.spbstu.ru*