



Факультет компьютерных наук

Системное программирование

Москва 2024

Расширение статического анализа кода Java на основе пользовательских аннотаций

Extending Static Analysis of Java Code Based on User Annotations

Выполнил:

Пасилецкий Даниил Олегович

Руководитель:

Профессор Базовая кафедра
«Системное программирование» ИСП
РАН, факультета компьютерных наук,
Белеванцев Андрей Андреевич

Консультант:

Старший лаборант ИСП РАН,
Афанасьев Виталий Олегович



Основные термины, понятия и определения

Статический анализ кода – анализ исходного кода на предмет ошибок и недочётов без непосредственного выполнения анализируемых программ.

Java – анализ исходного кода на предмет ошибок и недочётов без непосредственного выполнения анализируемых программ.

Svace – анализ исходного кода на предмет ошибок и недочётов без непосредственного выполнения анализируемых программ.

Java Annotations – это специальная форма синтаксических метаданных, которая может быть добавлена в исходный код.

Моделирование – это специальный подход при котором исходный класс заменяется на его упрощенную модель, с которой умеет работать анализатор



Проблема

Анализатор не всегда может верно работать, иногда из-за ограничений на потребляемые время и память приходится что-то упрощать, а иногда код очень непонятный (либо вообще исходники отсутствуют). И одно из решений - это предоставить пользователю механизм, который подсказывает что-то анализатору.

```
1 public class SalaryCalculator {
2     public static statistics(int numberMounth) {
3         int countDay = Calendar.countDay(numberMounth)
4         int averangeDailySalary = calculateAverangeDailySalary(10000000, countDay)
5         // Что-то
6     }
7
8     public static double calculateAverangeDailySalary(double totalSalary, int totalDays) {
9         return totalSalary / totalDays; // Svace warning
10    }
11 }
12
13 public class Calendar {
14     public static int countDay(int numberMounth) {
15         // Сложная логика или нету исходников
16     }
17 }
```



Постановка задачи

Необходимо расширить возможности статического анализатора Svace, таким образом чтобы пользователи могли предоставлять дополнительную информацию анализатору, путем добавления специальных аннотаций в исходном коде анализируемой программы.

```
public class Calendar {  
    @Range(min = 29, max = 31)  
    public static int countDay(int numberMounth) {  
        // Сложная логика или нету исходников  
    }  
}
```



Моделирование

Используется специальный класс
который перекрывает настоящую
реализацию Calendar

```
public class Calendar {  
    public static int countDay(int numberMounth) {  
        int number = Spec.getAnyNumber()  
        number.setRange(29, 31)  
        return number  
    }  
}
```



План

- ☐ Определить набор аннотаций, которые могут быть использованы пользователями
- ☐ Улучшить работу статического анализатора с аннотациями
- ☐ Разработать пакет с набором аннотаций, которые используются для расширения информации анализатора.
- ☐ Доработать статический анализатор для реагирования на эти аннотации и учитывать их при анализе кода.



План

- ☒ Определить набор аннотаций, которые могут быть использованы пользователями
- ☒ Улучшить работу статического анализатора с аннотациями
- ☐ Разработать пакет с набором аннотаций, которые используются для расширения информации анализатора.
- ☐ Доработать статический анализатор для реагирования на эти аннотации и учитывать их при анализе кода.

} 70 %



Набор аннотаций

29

аннотаций без учета перегрузок

Можно помечать аннотациями:

- Параметры функций
- Поля
- Методы

@Sensitive

@Tainted

@FunHash

@Leaked

@NotNull

@FunPrintfLike



Моделирование

Используется специальный класс
который перекрывает настоящую
реализацию Calendar

```
public class Calendar {  
    @Range(min = 29, max = 31)  
    public static int countDay(int numberMounth) {  
        return Spec.getAnyNumber()  
    }  
}
```



Основные проблемы

Противоречия аннотациям

```
public class Calendar {  
    @NotNull  
    public static Object getCalendar() {  
        return null;  
    }  
}
```



Основные проблемы

Противоречия аннотациям

```
public class Calendar {  
    @NotNull  
    public static Object getCalendar() {  
        return null; // Warning: Deref null  
    }  
}
```



Основные проблемы

Default значения аннотаций

```
public @interface Range {  
    int min() default 0;  
    int max() default 100000;  
}  
  
public class Calendar {  
    @Range(min=29)  
    public static int countDay() {  
        return 30;  
    }  
}
```



Основные проблемы

Default значения аннотаций

```
public static int countDay();
descriptor: ()I
flags: (0x0009) ACC_PUBLIC, ACC_STATIC
Code:
    stack=1, locals=0, args_size=0
       0: bipush      30
       2: ireturn
LineNumberTable:
   line 4: 0
RuntimeInvisibleAnnotations:
   0: #14(#15=I#16)
      Range(
        min=29
      )
}
```

```
public @interface Range {
    int min() default 0;
    int max() default 100000;
}

public class Calendar {
    @Range(min=29)
    public static int countDay() {
        return 30;
    }
}
```



Факультет компьютерных наук

Системное программирование

Москва 2024

Расширение статического анализа кода Java на основе пользовательских аннотаций

Extending Static Analysis of Java Code Based on User Annotations

Выполнил:

Пасилецкий Даниил Олегович

Руководитель:

**Профессор Базовая кафедра
«Системное программирование» ИСП
РАН, факультета компьютерных наук,
Белеванцев Андрей Андреевич**

Консультант:

**Старший лаборант ИСП РАН,
Афанасьев Виталий Олегович**