



Факультет компьютерных наук

Системное программирование

Москва 2025

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
обладать полной
информацией о коде.

И одно решение из решений
— предоставить
пользователям возможность
передавать информацию
анализатору.

```
1. private static int getWorkDays() {  
2.     // Very complicated implementation  
3. }  
  
5. private static int salaryInDay(int totalSalary,  
6.                                int workDays) {  
7.     return totalSalary / workDays; // No warning!  
8. }  
  
10. public static int calculateSalaryInDay( ) {  
11.     int dayInMonth = getWorkDays();  
12.     return salaryInDay(100000, dayInMonth);  
13. }
```

Листинг 1. Пример отсутствие предупреждения деления на ноль



Постановка задачи

```
1. @Interval(min=0, max=22)
2. private static int getWorkDays() {
3.     // Very complicated implementation
4. }

6. private static int salaryInDay(int totalSalary, int workDays) {
7.     return totalSalary / workDays;    // Warning!
8. }

10. public static int calculateSalaryInDay() {
11.     int dayInMonth = getWorkDays();
12.     return salaryInDay(100000, dayInMonth);
13. }
```

Листинг 2. Пример предупреждения деления на ноль.



Постановка задачи

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

Для достижение этой цели необходимо выполнить следующие задачи:

1. Разработать пакет, содержащий набор аннотаций и вспомогательные классы;
2. Добавить поддержку обработки пользовательских аннотаций;
3. Реализовать обработку параметров аннотаций, включая извлечение явно заданных значений и значений по умолчанию;
4. Интегрировать механизм обработки аннотаций в существующие этапы анализа, чтобы учитывать полученную информацию при интерпретации поведения кода;

Обзор существующих решений Спецификации

Создается отдельный файл, с упрощенной моделью, который заменяет реальную реализацию.

Отлично подходит для стандартных библиотек.

Минусы:

- Необходимо писать 2 файла;
- Вообще не анализируется файл, который замещен спецификацией;
- Не возможно использовать для полей.

```
1. public static int getWorkDays() {  
2.     int number = Spec.getAnyNumber();  
3.     Spec.setRange(number, 0, 22);  
4.     return number;  
5. }
```

Листинг 3. Пример использования спецфункций.



Обзор существующих решений

Checker Framework

```
1. private static int getWorkDays() {  
2.     // Very complicated implementation  
3. }  
4.  
5. private static int salaryInDay(@Min(1) int workDays,  
6.                               int totalSalary)  
{  
    return totalSalary / workDays;  
8. }  
9.  
10. public static int calculateSalaryInDay() {  
11.     int dayInMonth = getWorkDays();  
12.     return salaryInDay(dayInMonth, 100000); // Warning!  
13. }
```

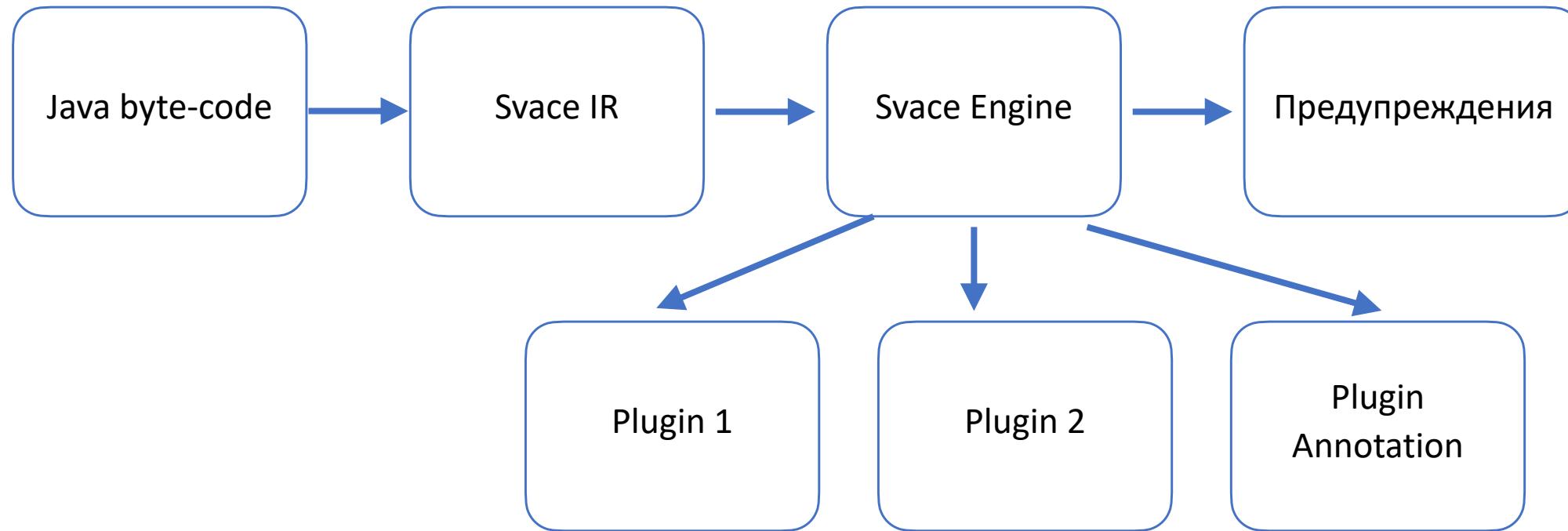
Листинг 4. Пример использования ограничений в Checker Framework.

Coverity

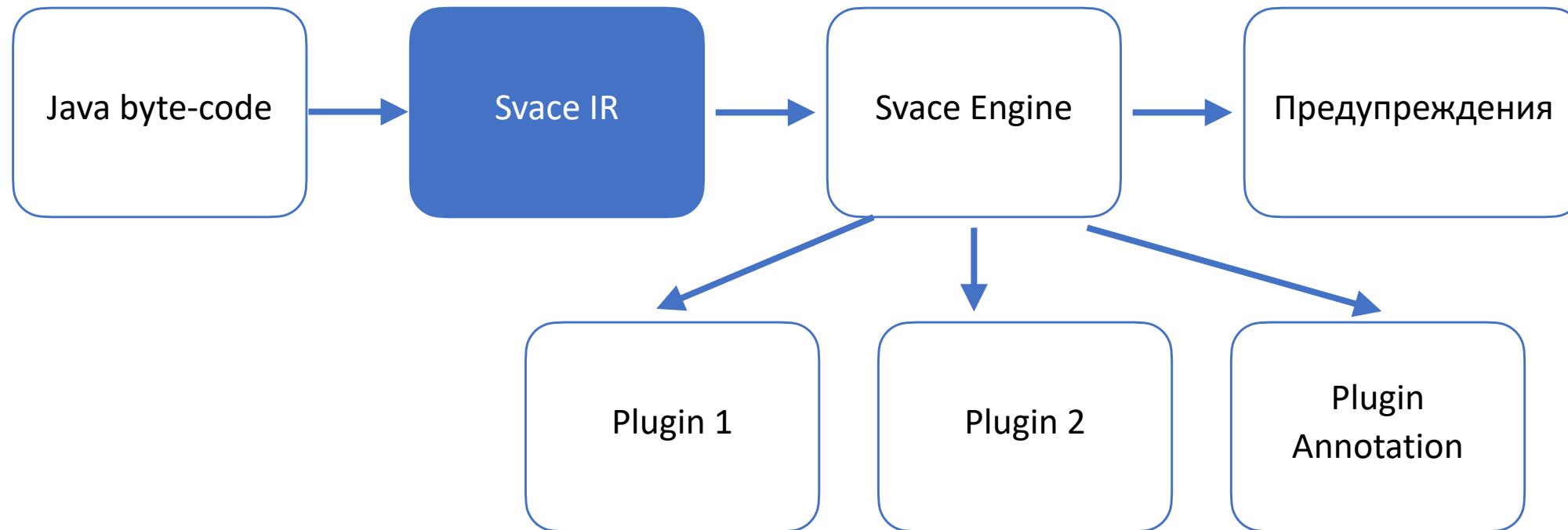
```
1. @Range(min = 0, max = 22)  
2. private static int getWorkDays() {  
3.     // Very complicated implementation  
4. }
```

Листинг 5. Пример использования аннотаций в Coverity.

Общий принцип работы Svace



Техническая реализация





Обработка параметров

Значения по умолчанию

```
1. public @interface Interval {  
2.     int min() default -2147483648;  
3.     int max() default 2147483647;  
4. }  
5.  
6. @Interval(min=0)  
7. private static int getWorkDays() {  
8.     ...  
9. }
```

Листинг 6. Пример аннотации Interval.

```
10. RuntimeInvisibleAnnotations:  
11. 0: #14(#15=I#16)  
12.     Lru/ispras/svace/Interval;(  
13.         min=0  
14.     )
```

Листинг 7. Пример аннотации Interval в
bytecode.

Обработка параметров Представления в SvalceIR

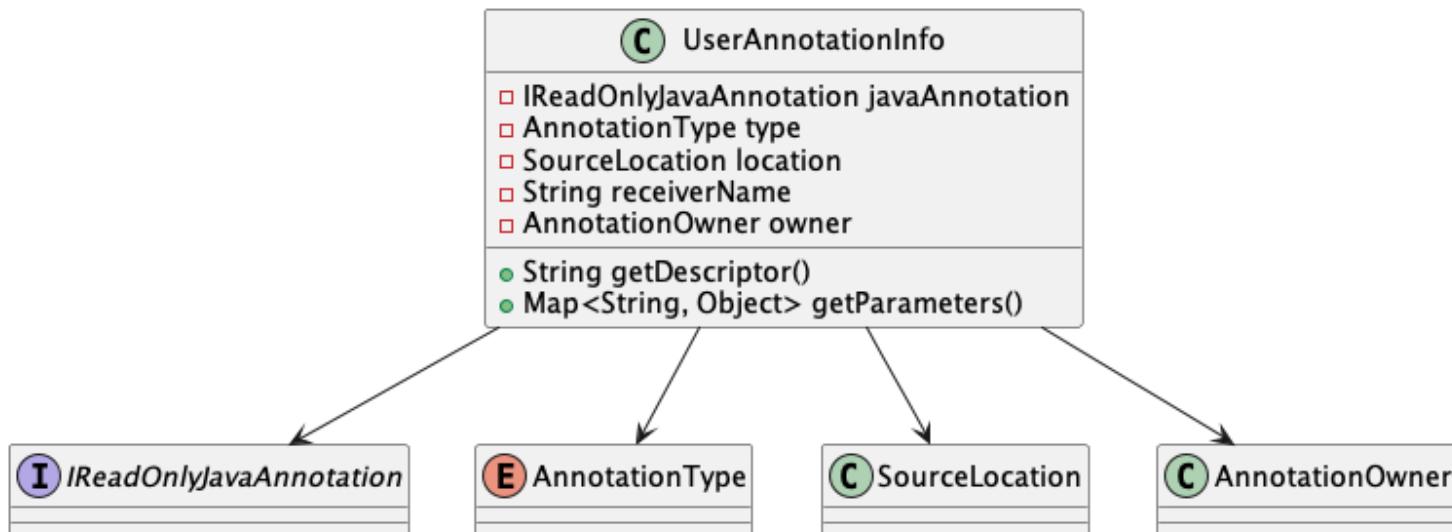


Рисунок 2. Диаграмма классов UserAnnotationInfo

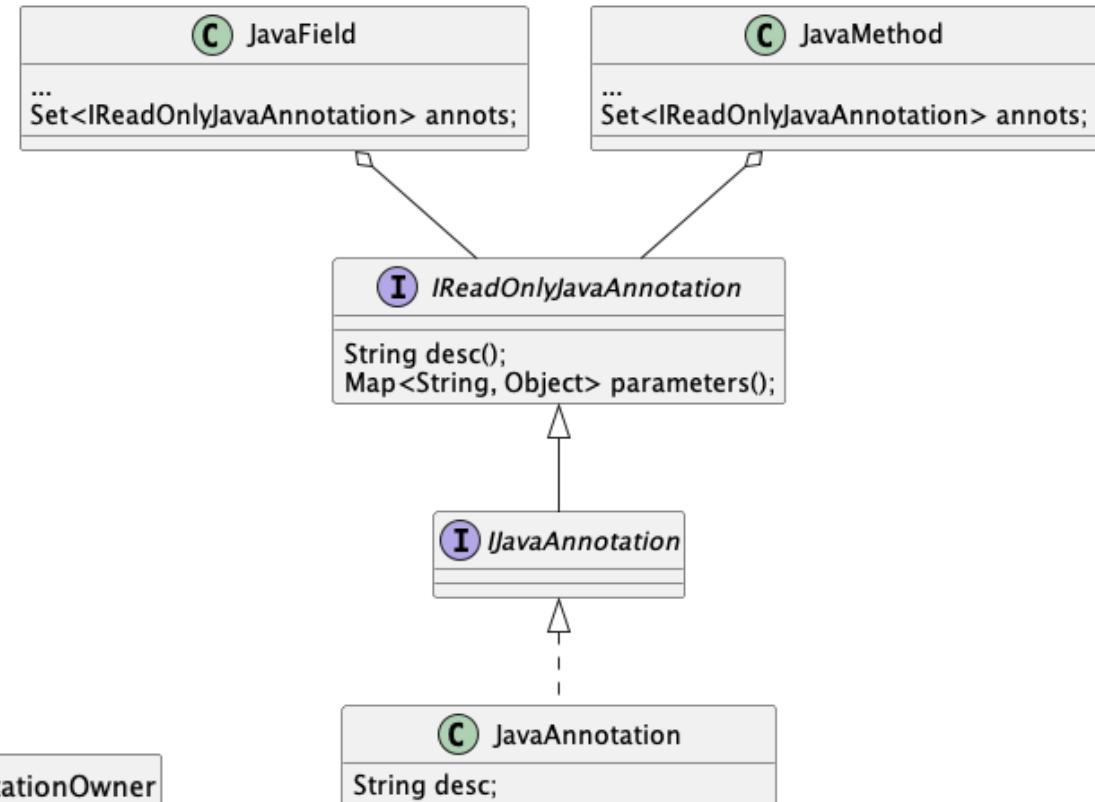
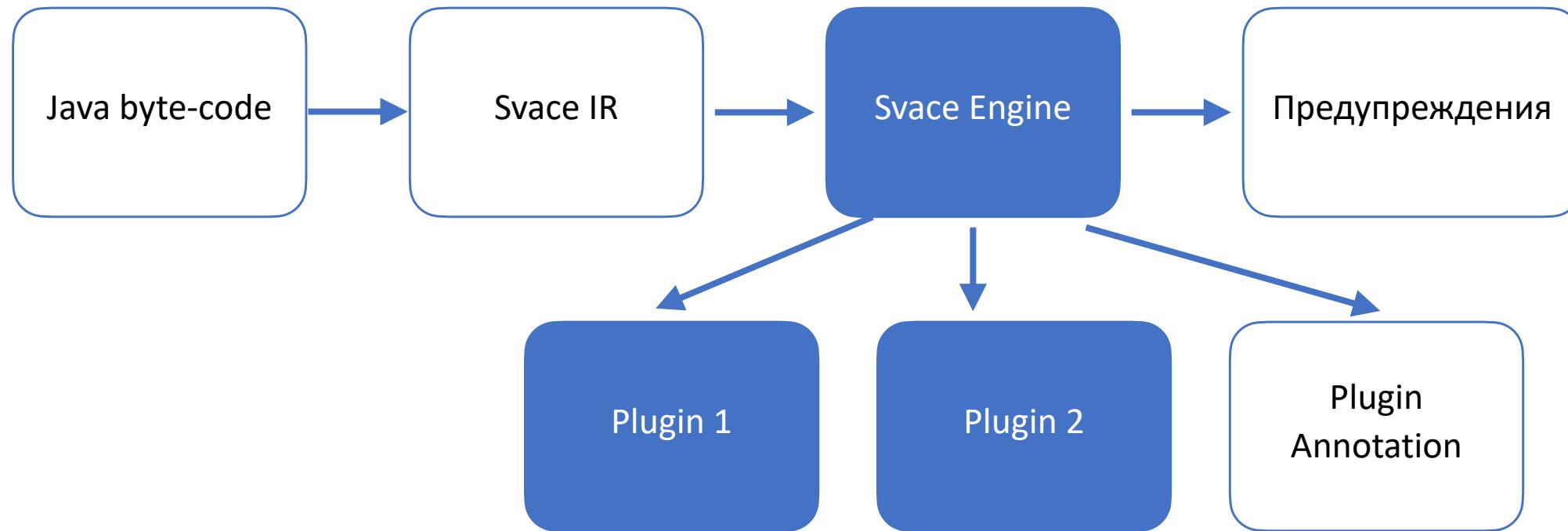
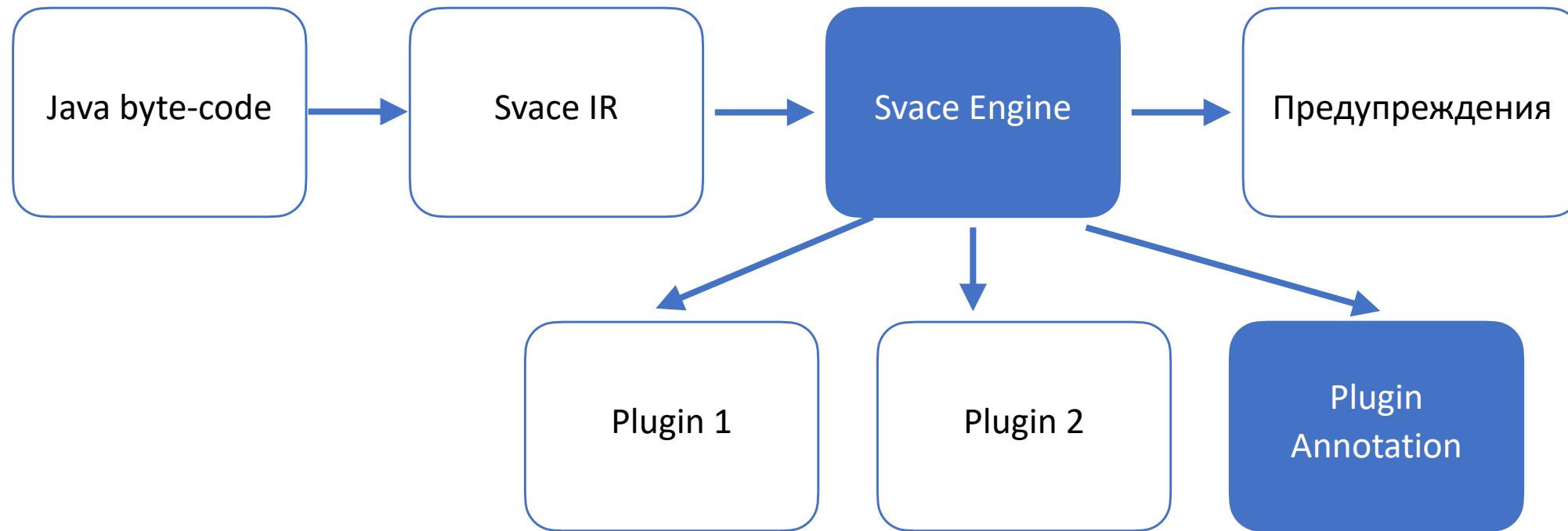


Рисунок 1. Диаграмма классов JavaAnnotation

Техническая реализация



Техническая реализация



Противоречие между аннотацией и фактическим кодом

```
1. @NotNull
2. public static Object getObject() {
3.     return null;
4. }

6. public static void main(String[ ] args) {
7.     Object ob = getObject();
8.     if (ob == null) {
9.         // Branch 1
10.    } else {
11.        // Branch 2
12.    }
13. }
```

Листинг 8. Пример противоречия между аннотацией и кодом.

Противоречие между аннотацией и фактическим кодом

```
1. @NotNull
2. public static Object getObject() {
3.     return null;
4. }
5.
6. public static void main(String[ ] args) {
7.     Object ob = getObject();
8.     if (ob == null) {
9.         // Branch 1
10.    } else {
11.        // Branch 2
12.    }
13. }
```

Разименование нулевого указателя

Всегда false

Недостигимый участок кода

Листинг 8. Пример противоречия между аннотацией и кодом.



Результаты

- Реализована обработка параметров аннотаций
- Добавлена поддержка обработки пользовательских аннотаций
- Разработан плагин для учета аннотаций при анализе;
- Добавлены аннотации и их обработка:
 - `@NotNull`
 - `@Nullable`
 - `@Interval`
 - `@TaintedInt`
 - `@TaintedPtr`
 - `@TrustedSink`
- Написано более 100 синтетических тестов
- Написана программная документация;



Факультет компьютерных наук

Системное программирование

Москва 2025

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