Санкт-Петербургский государственный университет

Кафедра системного программирования

Рефакторинг кода средствами Clang

Сергеев Игнатий Федорович, группа 22.Б10-мм

Научный руководитель: К.К. Смирнов, старший преподаватель кафедры системного программирования

Санкт-Петербург
2024

Введение

- IDE - неотъемлемая часть разработки;
- "ООО Софтком" разрабатывает свою IDE;
- в ней необходима функция полуавтоматического рефакторинга;
- IDE разрабатывается в рамках инфраструктуры Clang.

Постановка задачи

Целью работы является выбор и обновление инструмента из проекта Clang для выполнения рефакторингов.
Задачи:

- Провести обзор средств рефакторинга в проекте Clang;
- Дополнить проект необходимыми рефакторингами;
- Протестировать написанные трансформации.

Описание инструментов рефакторинга в Clang

Clang --- проект, содержащий инструменты работы с кодом написанном на c/c++/obj c.
Инструменты рефакторинга в Clang:

- Clang-Tidy;
- Clang-Rename;
- Clang-Transformer.
- Clang-Refactor.

Выбор инструмента

Критерий	Clang-Refactor	Clang-Tidy	Clang-Transformer	Clang-Rename
Расширяемость	+-	+	+	-
Произвольные проверки	+	+	-	+
Отсутствие доп проверок	+	-	+	+

Таблица: Сравнение инструментов

Пример рефакторинга

Код до рефакторинга:

```
struct Str {
    int field;
    int unusedField;
};

void func() {
    Str obj;
    obj.field = 1;
}
```

Пример рефакторинга

Код после рефакторинга:

```
struct Str {
    int field;
};

void func() {
    Str obj;
    obj.field = 1;
}
```

Описание Clang Refactor

Инструмент Clang-Refactor реализует преобразования через специальный движок --- RefactoringEngine.
RefactoringEngine состоит из 3-х основных компонент:

- действий;
- правил;
- требований.

Архитектура Refactoring Engine

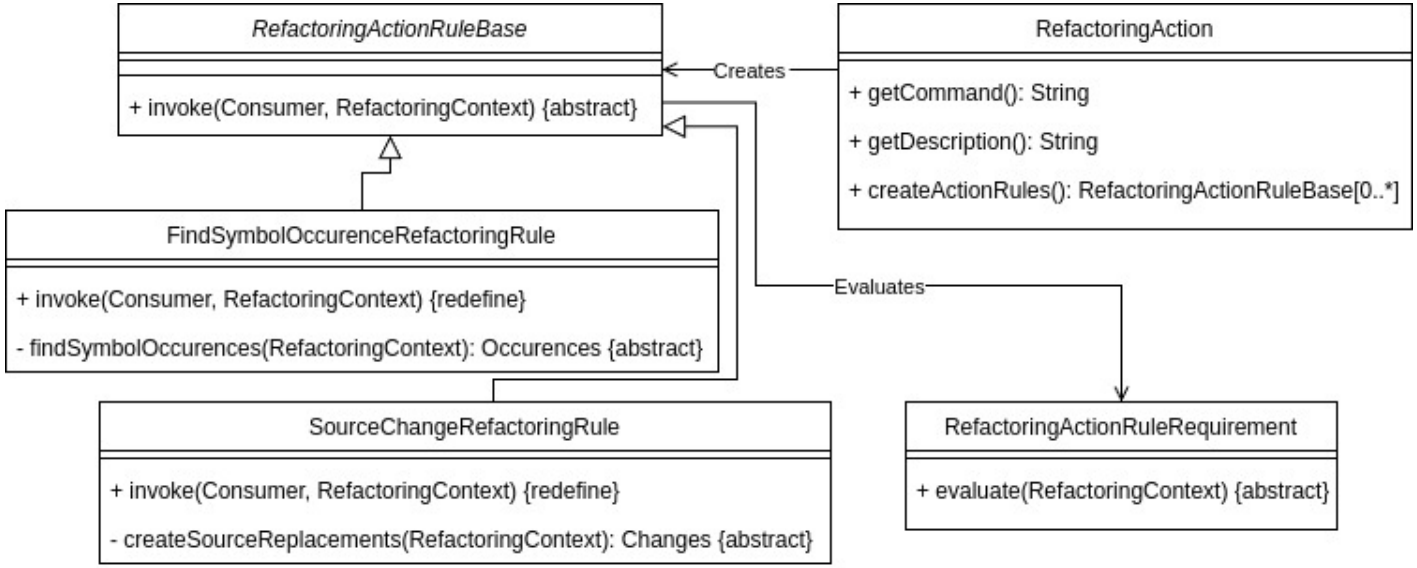

Рис.: Refactoring Engine

Архитектура решения

Задача требует указания места, где проводить рефакторинг:

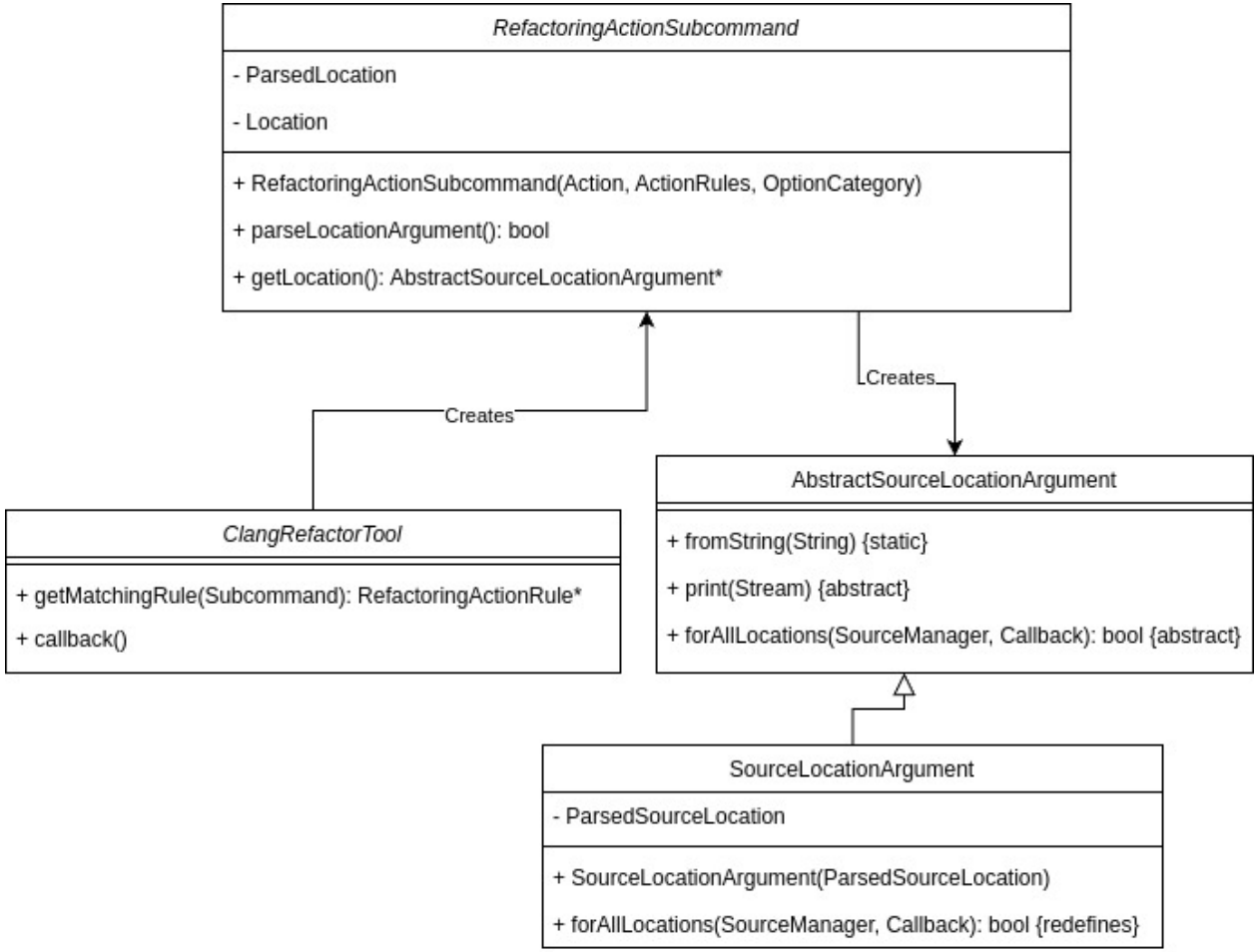

Рис.: Опция локации

Архитектура решения

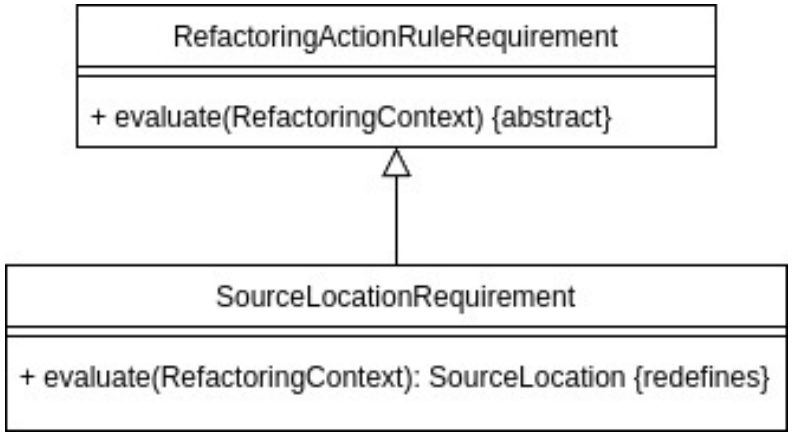

Рис.: Требование локации

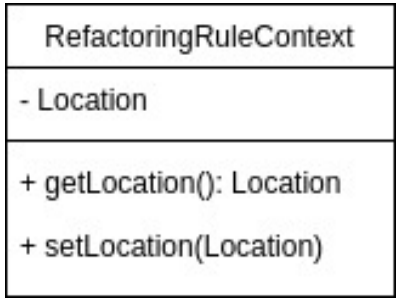

Рис.: Изменение контекста

Архитектура решения

Clang-Transformer использует 2 удобные абстракции для изменения AST:

- AST Matcher --- ifStmt(varDecl().bind("var"));
- AST Edit --- changeTo(node("var"), cat(" ")).

Для интеграции в Refactoring Engine требуется добавить:

- правило --- для возможности использования из действий;
- требования --- для возможности предоставления правилам.

Архитектура решения

Предоставление AST Matcher и Edit для использования в правилах:

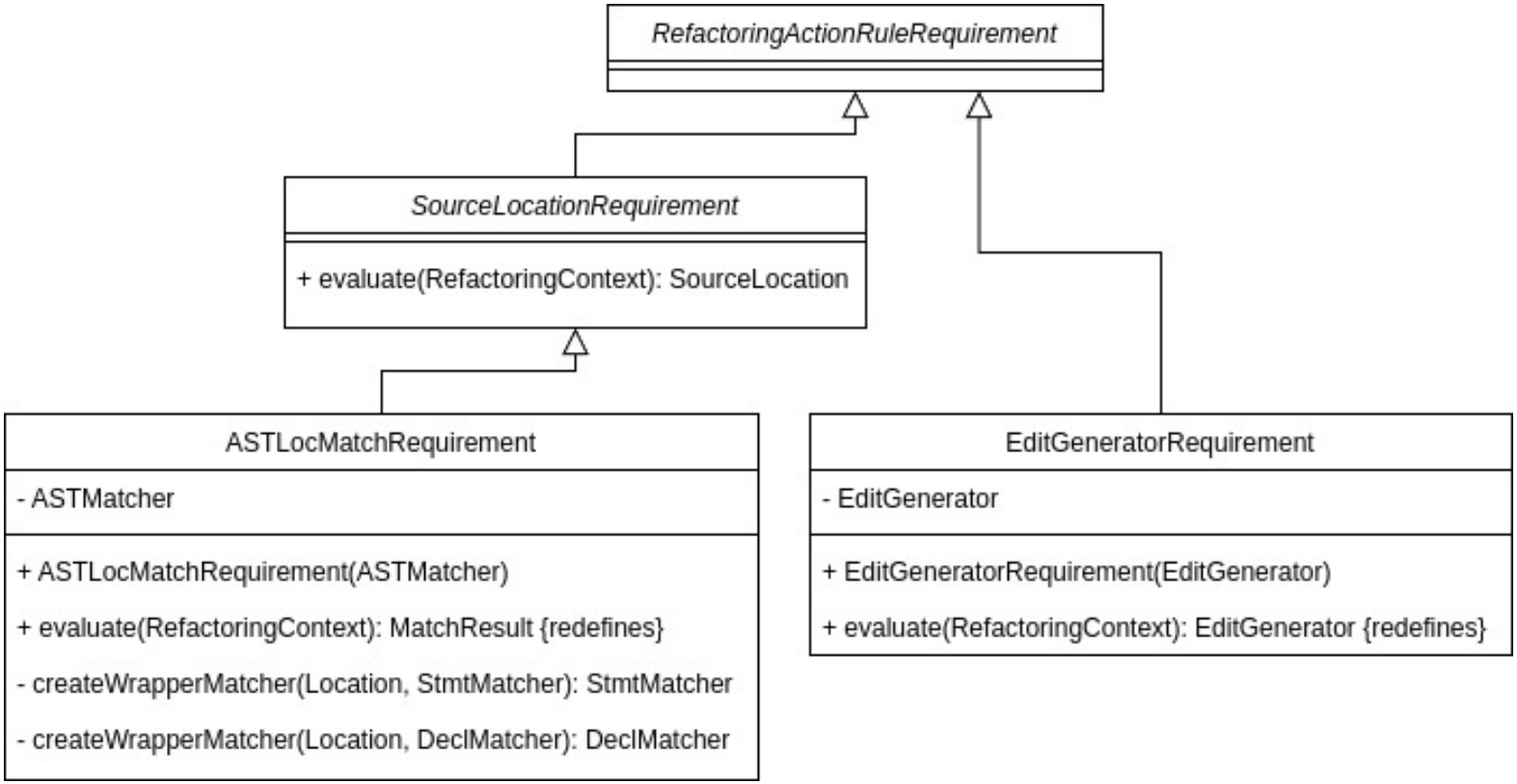

Рис.: Требования для AST Matcher и Edit

Архитектура решения

Правило на основе AST Matcher и Edit:

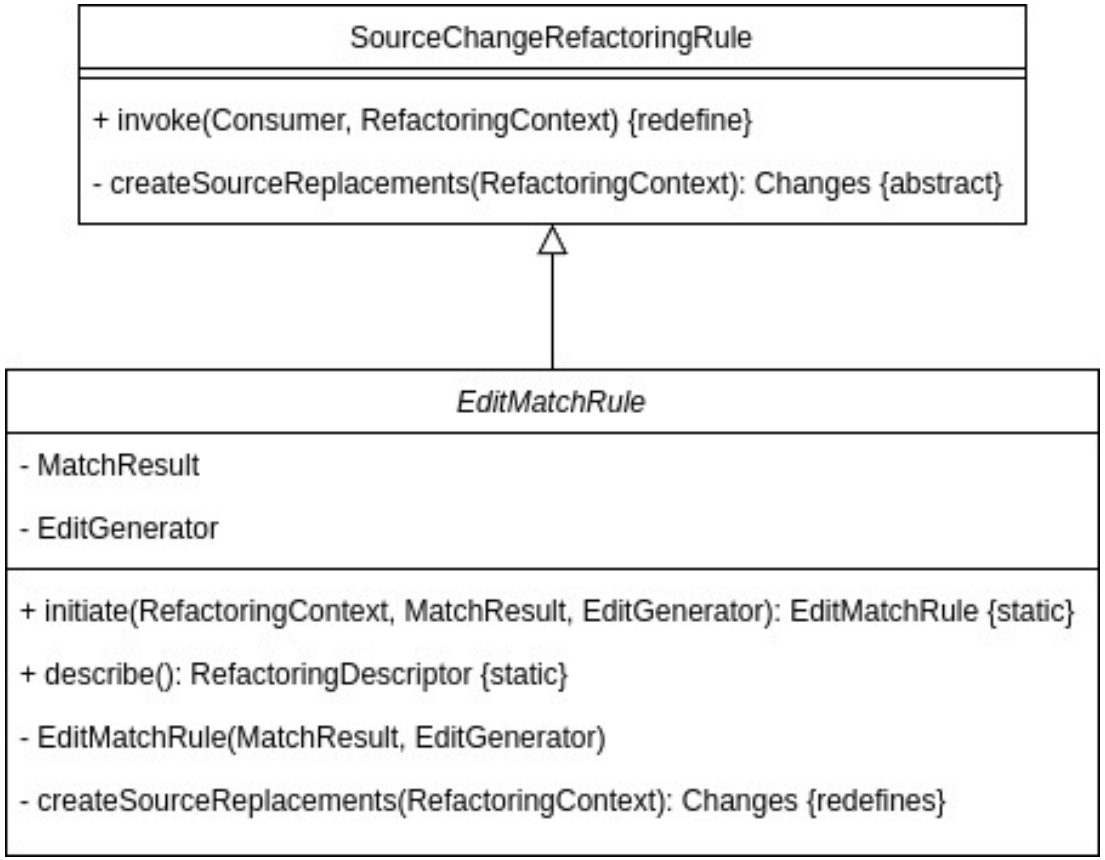

Рис.: Правило EditMatch

Архитектура решения

Действия, выполняющие требуемые рефакторинги:

- redundand-if;
- redundand-while;
- simplify-expr;
- unused-field;
- unused-method.

Пример архитектуры одного из действий:

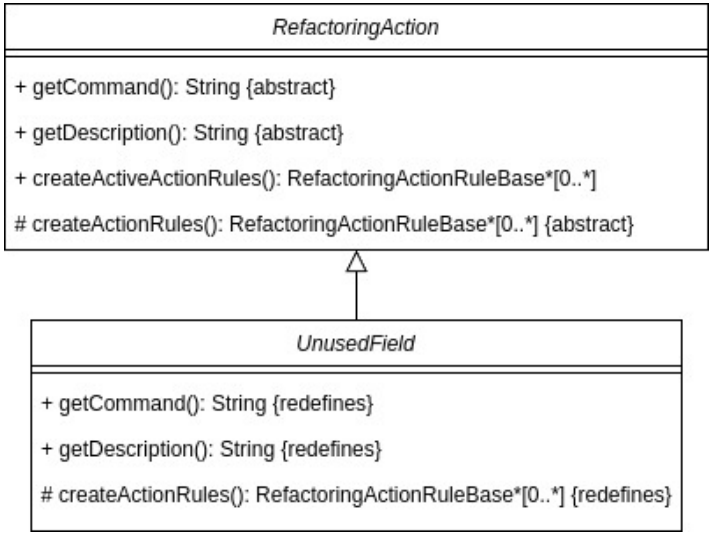

Рис.: UnusedField действие

Тестирование

Основное тестирование Clang-Refactor является интеграционным, в LLVM оно проводится с помощью следующий утилит:

- LIT(LLVM Integrated Tester);
- FileCheck.

Для упрощения использования вызова команд, необходимо поддержать указание опции location из тестов.

Тестирование

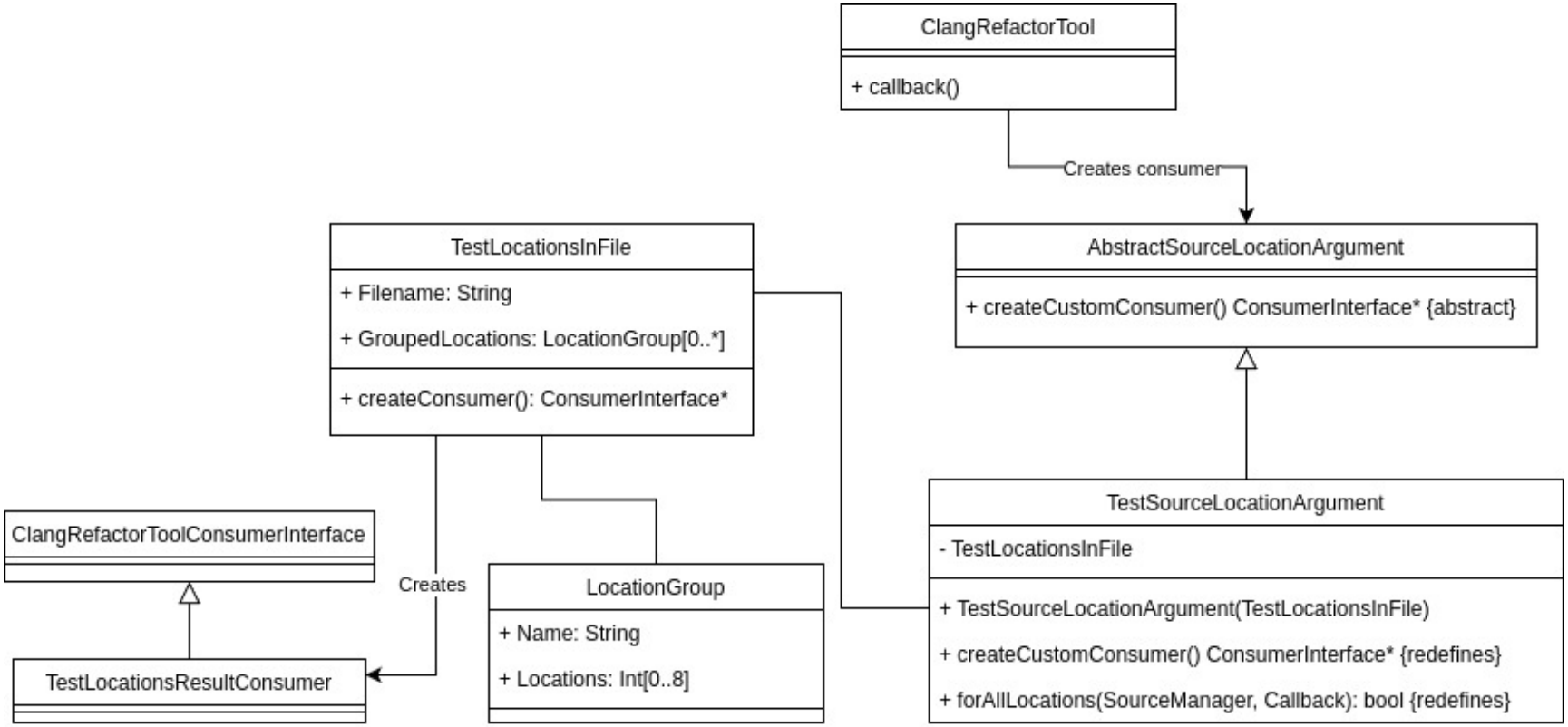

Рис.: Поддержка опции location в тестах

Тестирование

Написание тестовых сценариев делится на 4 этапа:

- написание кода тестового сценария;
- написание комментариев с указанием опций локации;
- написание комментариев LIT с вызовом команд;
- написание комментариев FileCheck с описанием правильного вывода.

Тестирование

Пример тестов для действия unused-field:

```
// RUN: clang-refactor unused-field -location=test:%s %s -- 2>&1
// RUN: | grep -v CHECK | FileCheck %s

struct Bar {
    /*loc used=*/int member;
    /*loc unused=*/int unusedMember;
};

void foo() {
    Bar var;
    var.member = 1;
}

// CHECK: 1 'unused' results:
// CHECK: struct Bar {
// CHECK-NEXT: /*loc used=*/int member;
// CHECK-NEXT: /*loc unused=*/
// CHECK-NEXT: };

// CHECK: 1 'used' results:
// CHECK-NEXT: refactoring action can't be initiated without a matching location
```

Результаты

- был проведен обзор проекта Clang и его инструментов для рефакторинга;
- проект был дополнен необходимыми рефакторингами;
- написанные трансформации были протестированы.

Исходный код можно найти по ссылкам:

- https://github.com/IgnatSergeev/llvm-project/pull/3;
- https://github.com/IgnatSergeev/llvm-project/pull/4;
- https://github.com/IgnatSergeev/llvm-project/pull/5.