Анализ “мертвого” кода

Команда “Шерстяные волчары с мощными лапищами”

16 июня 2023 г.

Немного о нас

Рис. 1: Про название

Немного о нас

Разрабатываем ПО ATMoSphere для устройств самообслуживания

- Много кода на C++
- Есть юнит-тесты
- Есть высокоуровневые тесты на бизнес-сценарии

Немного о нас

Разрабатываем ПО ATMoSphere для устройств самообслуживания

- Много кода на C++
- Есть юнит-тесты
- Есть высокоуровневые тесты на бизнес-сценарии

Решили искать мёртвый код на примере нашей кодовой базы C++

“Мёртвый” код

```
1
2   /* I see dead code */
3                                              .....
4                                            X X  /
5                                           /<   /
6                                 ___ __________/_#__=o
7                               /(- /(\_\_______   \
8                                           \o     \
9                                           |'     |
10                                           |     _|
11                                          /o   __\
12                                         / '     |
13                                        / /      |
14                                       /_/\______|
15                                     (   _(    <
16  #SourceCodeComments                 \____\____\
17  ............................|___ |_______|.............................
18
```

Рис. 2: Плавно переходим к теме кода

“Мёртвый” код

Мёртвый код - это код, который не влияет на работу программы и может быть удалён.

“Мёртвый” код

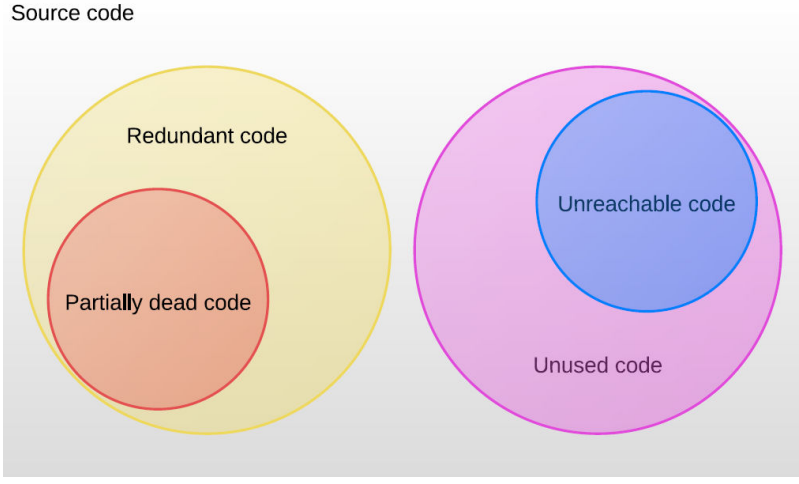

Рис. 3: Схемка из интернетов

“Мёртвый” код

Разные синтаксические единицы

- Переменные
- Функции
- Классы
- Библиотеки

“Мёртвый” код

Разные способы выявления

- Статический анализ
  - Флаги компилятора и линковщика
  - Анализаторы вроде CppCheck, Clang-Tidy, SonarCube, PvsStudio
  - Самостоятельный
- Динамический анализ
  - Code coverage tool (например - gcovr)

“Мёртвый” код

Разные способы выявления

- Статический анализ
  - Флаги компилятора и линковщика
  - Анализаторы вроде CppCheck, Clang-Tidy, SonarCube, PvsStudio
  - Самостоятельный
- Динамический анализ
  - ✓ Code coverage tool (например - gcovr)

“Мёртвый” код

Рис. 4: Поиск мёртвого кода по версии ИИ

“Мёртвый” код

Динамический анализ

- ✗ Синтаксические единицы
- ✓ Блоки кода

“Мёртвый” код

Что ещё можно было бы искать

- “Забытые”(не добавленные в cmake проект) файлы
- “Забытые”(не влитые в develop/master) ветки в git
- Unreachable(недостижимый) код

“Мёртвый” код

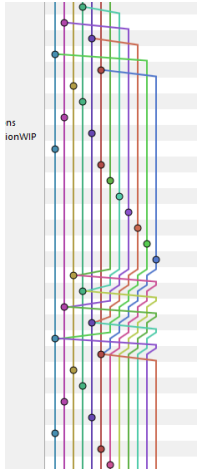

Рис. 5: *git*ar hero или мёртвые ветки в git

Подготовка данных

- Собираем репозиторий со специальным флагом сборки для coverage
- Запускаем тесты (желательно - тестирование бизнес-сценариев)
- Собираем данные по coverage - их мы и будем использовать в основе нашего анализа

Первичный анализ полученных данных

- Считаем, что покрытый тестами код точно не является мёртвым
- Непокрытый код либо мёртвый, либо на него не написаны тесты
  - Такой подход заставляет писать больше высокоуровневых тестов
- Дальше начинаем анализ не покрытых тестами блоков кода или файлов

Блоки кода vs файлы

Можно давать оценку “мёртвости” по файлам. Её легче получить, но она не слишком информативна. Это наш MVP1.

Следующая итерация - это оценка для отдельных блоков кода(под блоками понимаются подряд идущие не покрытые тестами строки кода). Это наш MVP2.

Система фильтров

Чтобы система получилась расширяемой, мы добавили фильтры, которые влияют на итоговую оценку

- Первый и основной фильтр - фильтр по данным от coverage
  - Для файлов смотрим процент непокрытых линий
  - Для блоков кода смотрим размер блока
- Вторым мы выбрали фильтр по данным от git
  - Для файлов дату последней модификации и частоту модификаций
  - Для блоков кода смотрим среднее/крайнее значение последней модификации и частоту модификаций
- Сюда же можно добавить какой-нибудь стат-анализ или любые другие метрики

Система фильтров

- Фильтр для каждого блока/файла возвращает какую-то оценку “в попугаях”
- Эти оценки умножаются на вес фильтра и затем суммируются
- Чем больше итоговое значение для файла/блока, тем больше вероятность того, что там есть мёртвый код
- Затем файлы/блоки кода сортируются на основе этой оценки

Заключение

Работает - не трогай!
Не работает? Это MVP...