Software Engineering

Лекция 10: Качество и метрики ПО

Тимофей Брыксин
timofey.bryksin@gmail.com

Что такое качество ПО?

- Обывательский подход
- Профессиональный подход
  - соответствие требованиям (Crosby, 1979)
  - пригодность к использованию (Juran, Gryna, 1970)
- Жизненный подход
  - соответствие всем требованиям, явным и неявным

Стоимость качества

- Выработка требований к качеству
- Действия в процессе разработки
  - предупреждение “дефектов”
  - оценка “дефектов”
  - исправление внешних “дефектов”
  - исправление внутренних “дефектов”

Модель качества ПО

- Характеристики качества
- Атрибуты характеристик качества
- Метрики качества

Характеристики качества ПО

- Функциональность
- Надежность
- Удобство использования
- Эффективность
- Сопровождаемость
- Переносимость

Функциональность

- Функциональная полнота
- Правильность (точность)
- Функциональная совместимость (интероперабельность)
- Защищенность

Надежность

- Безотказность
- Устойчивость к ошибкам
- Восстанавливаемость
- Пригодноспособность
  - готовность к использованию
  - готовностью к непрерывному функционированию
  - безопасность для окружающей среды
  - устойчивость к самопроизвольному изменению
  - простота выполнения операций обслуживания

Удобство использования

- Понимаемость
- Легкость изучения
- Оперативность
- Согласованность

Эффективность

- Реактивность
- Эффективность ресурсов

Сопровождаемость

- Анализируемость
- Изменяемость
- Стабильность
- Тестируемость

Переносимость

- Адаптивность
- Настраиваемость
- Сосуществование
- Заменяемость

Метрики качества ПО

- Функциональность: метрики тестирования
- Надежность: метрики тестирования, динамические метрики
- Удобство использования: метрики эргономики
- Эффективность: динамические метрики
- Сопровождаемость: метрики кода
- Переносимость: метрики кода

Классификация метрик

- Метрики программного продукта
  - внешние
    - надежность
    - функциональность
    - сопровождение
    - стоимость
  - внутренние
    - размер
    - сложность
    - стиль
- Метрики процесса
- Метрики использования

Что можно измерять

- Размер
  - число классов, строк в программе, объём памяти, …
- Переиспользуемость кода
  - переиспользуемые классы, наследуемые классы, зависимости, ...
- Время
  - отклика, общего функционирования системы, выполнения компонента, …
- Усилия
  - производительность труда, трудоемкость, …
- Ошибки
  - количество ошибок, число отказов, …

Простые метрики

- Число строк кода (LOC/KLOC)
- Производительность = LOC / Затраты
- Удельная стоимость = Затраты / LOC
- Качество кода = Число ошибок / LOC
- Документированность = Число страниц документации / LOC

Больше метрик богу метрик!

- Метрики Холстеда
- Метрики С. Чидамбера и К. Кемерера
- Метрики Ф. Абреу
- Метрики Л. Константейна и Э. Йордана
- Метрики Л. Отта и Б. Мехра
- Метрики Д. Биемена и Б. Кенга
- Метрики М. Лоренца и Д. Кидда
- Метрики Р. Байндера
- ...

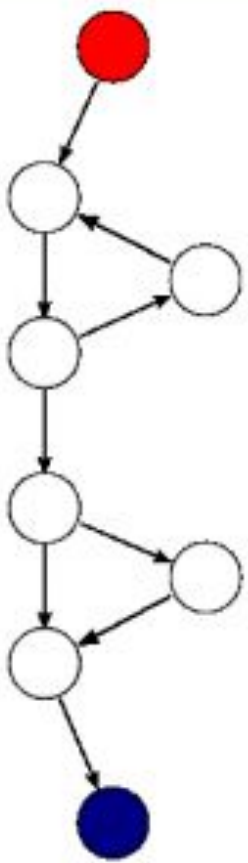
Метрики Холстеда

- Number of Unique Operators (NUOprtr)
- Number of Unique Operands (NUOprnd)
- Number of Operators (Noprtr)
- Number of Operands (Noprnd)

- Halstead Program Volume (HPVol) =
  $$(\text{Noprtr} + \text{Noprnd}) \times \log_2(\text{NUOprtr} + \text{NUOprnd})$$
- Halstead Difficulty (HDiff) = (NUOprtr/2) × (Noprnd / NUOprnd)
- Halstead Effort (HEff) = HDiff × HPVol

Цикломатическая сложность

- C = E – N + 2P
  - E -- число ребер
  - N -- число узлов
  - P -- число компонентов связности



Метрики С. Чидамбера и К. Кемерера

- Weighted Methods Per Class (WMC)
- Depth of Inheritance Tree (DIT)
- Number of children (NOC)
- Coupling between object classes (CBO)
- Response For a Class (RFC)
- Lack of Cohesion in Methods (LCOM)
  - НЕ СВЯЗАНЫ —количество пар методов без общих экземплярных переменных
  - СВЯЗАНЫ — количество пар методов с общими экземплярными переменными
  - I_j — набор экземплярных переменных, используемых методом M_j

$$LCOM = \begin{cases} \text{НЕ СВЯЗАНЫ - СВЯЗАНЫ, если (НЕСВЯЗАНЫ} > \text{СВЯЗАНЫ);} \\ 0 \text{ в противном случае.} \end{cases}$$

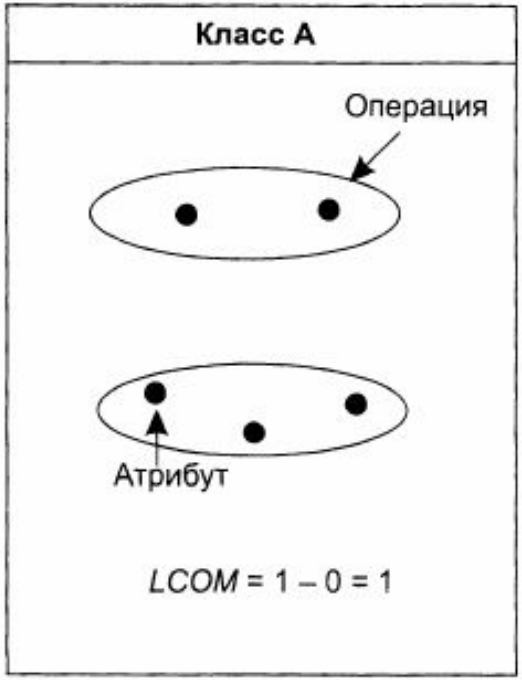
Полезные модификации WMC

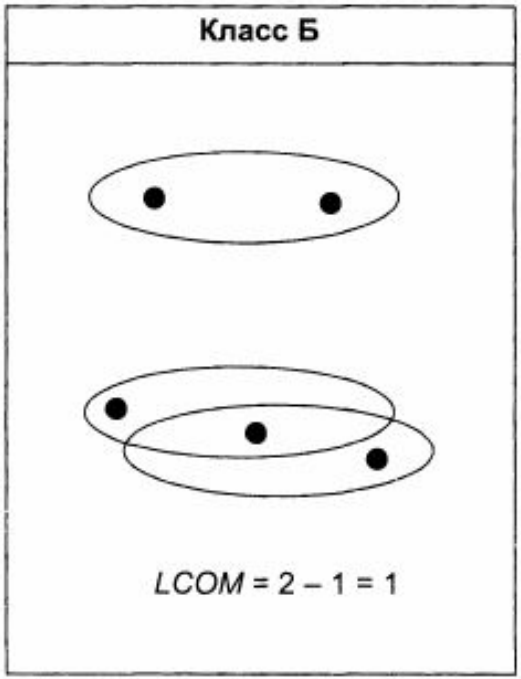
- WMC2 = Σ_i количество параметров метода i
- ANAM (Average Number of Arguments per Method) = WMC2/WMC

```
Set_interval (min, max),
Set_method (method),
Set_precision (precision),
Set_function_to_integrate (function),
Integrate;
```

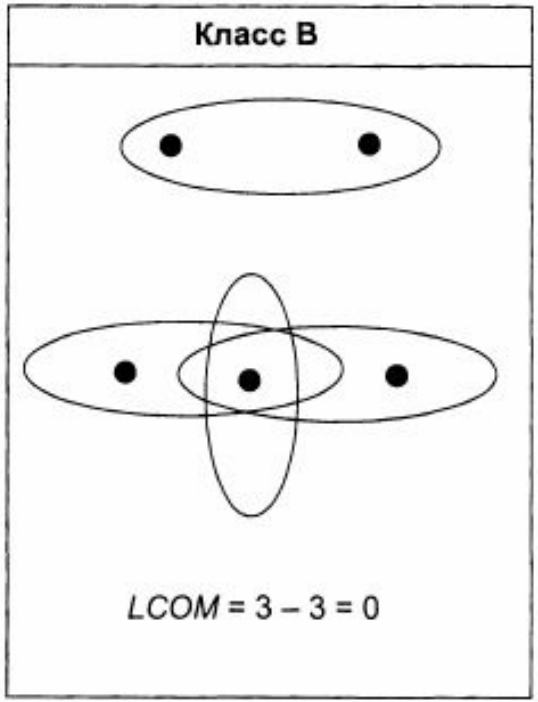
```
Integrate (function, min, max, method, precision)
```

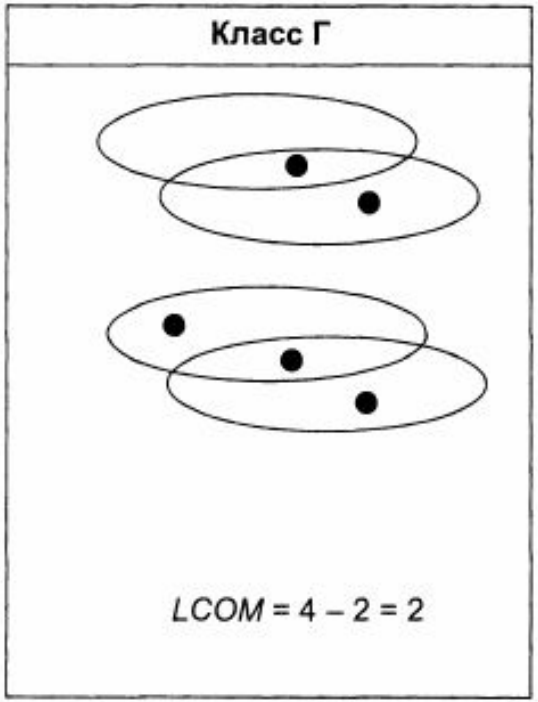
LCOM: недостатки





LCOM: недостатки (2)





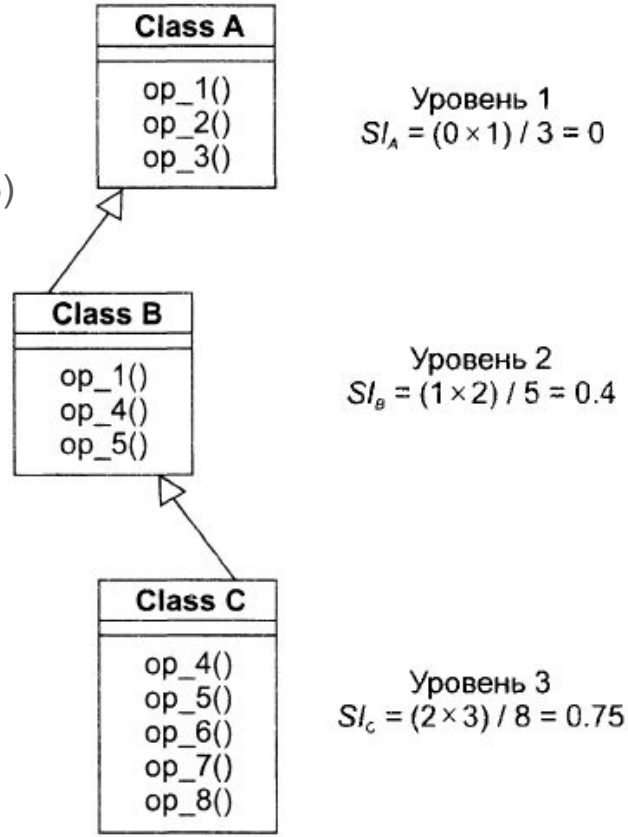
Модификация LCOM*

$$LCOM^* = \frac{\left(\frac{1}{a}\sum_{j=1}^{a} m(A_j)\right) - m}{1 - m}$$

- m -- количество методов класса
- a -- количество атрибутов класса
- $m(A_j)$ -- количество методов, которые имеют доступ к атрибуту A

Метрики Лоренца и Кидда

- Метрики, ориентированные на классы
  - Class Size (CS, <= 20)
  - Number of Operations Overridden by a Subclass (NOO, <= 3)
  - Number of Operations Added by a Subclass (NOA, <= 4)
  - Specialization Index (SI, <= 0.15)
    - SI = (NOO * уровень) / $M_{общ}$
- Метрики, ориентированные на операции
  - Average Operation Size (OS_{avg}, <=9)
  - Operation Complexity (OC)
  - Average Number of Parameters per operation (NP_{avg})



Набор метрик Фернандо Абреу (MOOD)

- фактор закрытости метода (MHF)
- фактор закрытости атрибута (AHF)
- фактор наследования метода (MIF)
- фактор наследования атрибута (AIF)
- фактор полиморфизма (POF)
- фактор сопряжения (COF)

Фактор закрытости метода (MHF)

$$MHF = \frac{\sum_{i=1}^{TC} M_h(C_i)}{\sum_{i=1}^{TC} M_d(C_i)}$$

- $M_h(C_i)$ -- количество скрытых методов в классе C_i
- $M_d(C_i)$ -- общее количество методов в классе C_i (без унаследованных)

Фактор закрытости атрибута (AHF)

$$AHF = \frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_d(C_i)}$$

- $A_h(C_i)$ -- количество видимых атрибутов в классе C_i
- $A_d(C_i)$ -- общее количество атрибутов в классе C_i (без унаследованных)

Фактор наследования метода (MIF)

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

- $M_i(C_i)$ -- количество унаследованных и не переопределённых методов в C_i
- $M_a(C_i)$ -- общее количество методов, доступных в классе C_i

Фактор наследования атрибута (AIF)

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

- $M_i(C_i)$ -- количество унаследованных и не переопределённых атрибутов в C_i
- $M_a(C_i)$ -- общее количество атрибутов, доступных в классе C_i

Фактор полиморфизма (POF)

$$POF = \frac{\sum_{i=1}^{TC} M_0(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]}$$

- $M_0(C_i)$ — количество унаследованных и переопределенных методов в C_i
- $M_n(C_i)$ — количество новых (не унаследованных и переопределенных) методов в C_i
- $DC(C_i)$ — количество потомков класса C_i

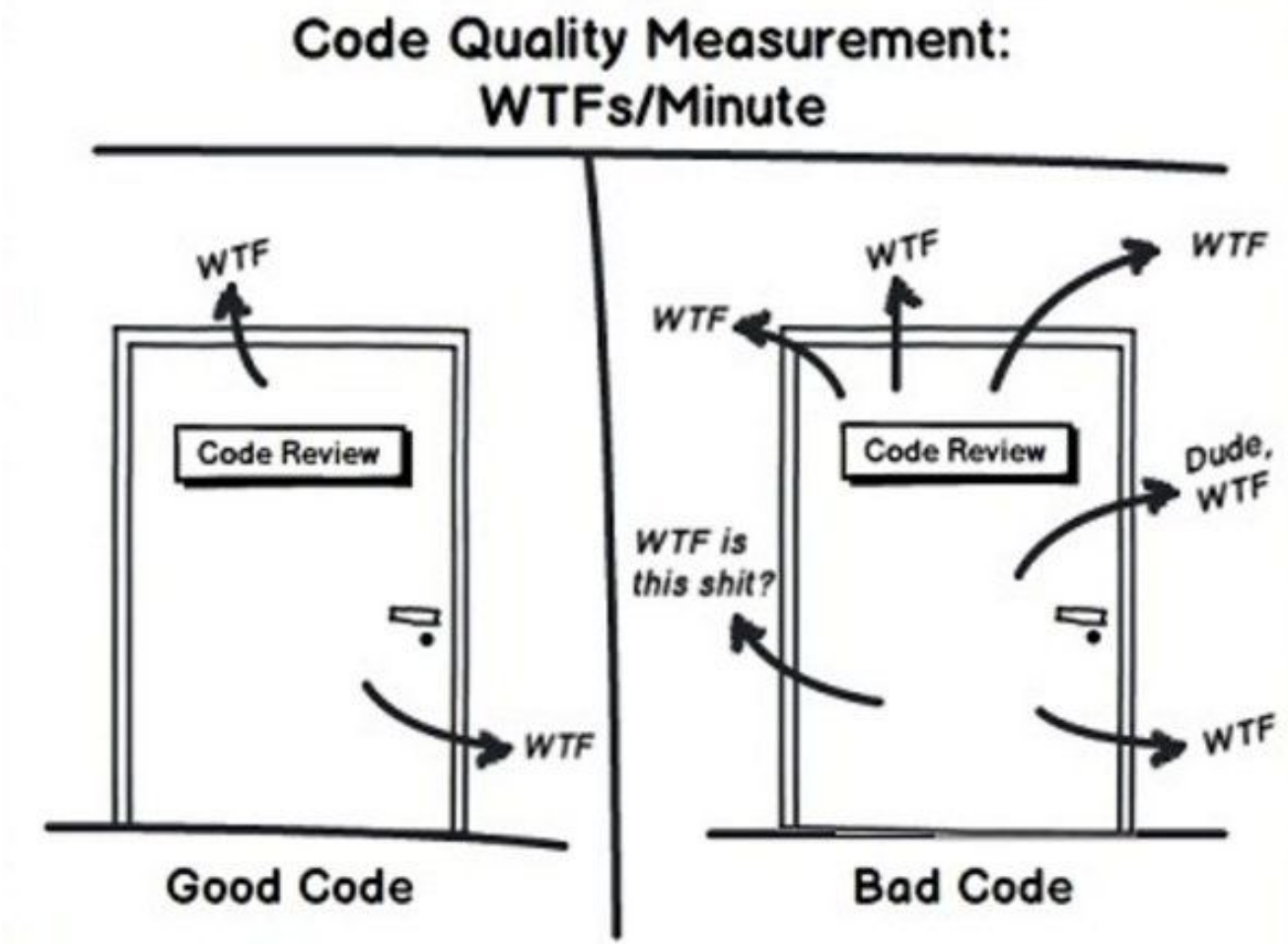
Фактор сопряжения (COF)

$$COF = \frac{\sum_{i=1}^{TC}[\sum_{j=1}^{TC} is\_client(C_i, C_j)]}{TC^2 - TC}$$

$$is\_client(C_c, C_s) = \begin{cases} 1, \text{if } C_c => C_s \cap C_c \neq C_s \\ 0, \text{else,} \end{cases}$$

Метрики для тестирования

- Недостаток связности в методах
- Процент публичных и защищенных методов
- Публичный доступ к атрибутам
- Количество корневых классов
- Количество детей, Высота дерева наследования
- Процентное количество не переопределенных запросов
- Процентное количество динамических запросов
- Скачок класса
- Скачок системы


Code Quality Measurement:
WTFs/Minute
WTF
Code Review
WTF
Good Code
WTF
WTF
WTF
WTF
Code Review
Dude,
WTF
WTF is
this shit?
WTF
Bad Code

Аудит программного кода

- Сбор информации, накопление знаний, формирование эталонов
- Ручной
  - экспертный
  - расчётный
- Автоматический
  - измерительный
  - регистрационный
- Тулы
  - https://plugins.jetbrains.com/plugin/93-metricsreloaded
  - http://metrics.sourceforge.net/
  - https://www.codacy.com/
  - …

Тестирование

- Любая программа содержит ошибки
- Если программа не содержит ошибок, их содержит алгоритм, который реализует эта программа
- Если ни программа, ни алгоритм ошибок не содержат, такая программа даром никому не нужна

Виды тестирования

- По объекту тестирования
  - функциональное
  - тестирование производительности
    - нагрузочное
    - стресс-тестирование
    - тестирование стабильности
  - тестирование удобства использования
  - тестирование интерфейса пользователя
  - тестирование безопасности
  - тестирование локализации
  - тестирование совместимости

Виды тестирования (2)

- По знанию о системе
  - white box
  - black box
- По степени автоматизации
  - ручное
  - автоматическое
- По степени подготовленности
  - тестирование по документации
  - ad-hoc тестирование

Виды тестирования (3)

- По времени проведения
  - альфа-тестирование
    - smoke testing
    - тестирование новой функциональности
    - регрессионное тестирование
    - приемочное тестирование
  - бета-тестирование
- По степени изолированности компонентов
  - компонентное/модульное
  - интеграционное
  - системное

Почему тесты полезны

- Помогают искать ошибки
- Облегчают изменение программы
  - помогают при рефакторинге
- Тесты — документация к коду
- Помогают улучшить архитектуру
  - приложение как набор библиотек
  - многоуровневая архитектура
  - управление зависимостями
    - изоляция компонент
    - dependency injection