

Лекция 9: Качество программного обеспечения

Юрий Литвинов
y.litvinov@spbu.ru

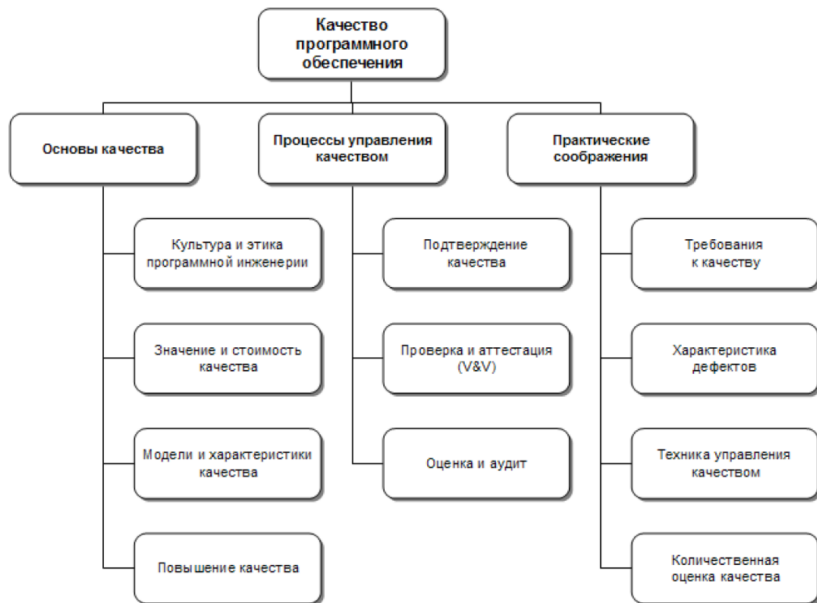
18.04.2023

Что такое качество ПО?

- ▶ Обывательский подход
 - ▶ Лёгкость в использовании, производительность, отсутствие ошибок, документация, кроссплатформенность и т.п.
- ▶ Профессиональный подход
 - ▶ Соответствие требованиям (Crosby, 1979)
 - ▶ Пригодность к использованию (Juran, Gryna, 1970)
- ▶ Жизненный подход
 - ▶ Соответствие всем требованиям, явным и неявным

Стоимость качества

- ▶ Решения о качестве принимаются на этапе работы с требованиями
- ▶ Обычно заказчик полагает качество максимальным
- ▶ Стоимость:
 - ▶ стоимость предупреждения дефектов (prevention cost)
 - ▶ стоимость оценки (appraisal cost)
 - ▶ стоимость внутренних сбоев (internal failure cost)
 - ▶ стоимость внешних сбоев (external failure cost)



Модель качества ПО

- ▶ Характеристики качества — отдельные точки зрения пользователя на качество
- ▶ Атрибуты характеристик качества — детализация разных аспектов характеристики
- ▶ Метрики качества
 - ▶ Метод измерения атрибута
 - ▶ Шкала измерения значений атрибута
 - ▶ Вес (иногда)

Характеристики качества ПО (ISO 25010:2011)

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

Функциональность

- ▶ Функциональная полнота (suitability)
- ▶ Правильность (точность) (accuracy)
- ▶ Функциональная совместимость (интероперабельность) (interoperability)
- ▶ Защищенность (security)
- ▶ Соответствие стандартам и правилам (compliance)

Надежность

- ▶ Безотказность (maturity)
- ▶ Устойчивость к отказам (fault tolerance)
- ▶ Восстанавливаемость (recoverability)
- ▶ Пригодность (dependability)
 - ▶ Готовность к использованию (availability)
 - ▶ Готовностью к непрерывному функционированию (reliability)
 - ▶ Безопасность для окружающей среды (safety)
 - ▶ Секретность и сохранность информации (confidentiality)
 - ▶ Устойчивость к самопроизвольному изменению (integrity)
 - ▶ Простота выполнения операций обслуживания (maintainability)

Удобство использования

- ▶ Понимаемость (understandability)
- ▶ Легкость изучения (learnability)
- ▶ Удобство работы (operability)
 - ▶ Оперативность
 - ▶ Согласованность
- ▶ Привлекательность (attractiveness)

Эффективность

- ▶ Временная эффективность, реактивность (time behaviour)
- ▶ Эффективность ресурсов (resource utilisation)

Сопровождаемость

- ▶ Анализируемость (analyzability)
- ▶ Изменяемость (changeability)
- ▶ Стабильность (stability)
- ▶ Тестируемость (testability)

Переносимость

- ▶ Адаптивность (adaptability)
- ▶ Настраиваемость, простота инсталляции (installability)
- ▶ Сосуществование (coexistence)
- ▶ Заменяемость (replaceability)

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

Классификация метрик

- ▶ Метрики программного продукта
- ▶ Метрики процесса
 - ▶ Общее время разработки и отдельно время для каждой стадии
 - ▶ Время модификации моделей
 - ▶ Время выполнения работ на процессе
 - ▶ Число найденных ошибок при инспектировании
 - ▶ Стоимость проверки качества
 - ▶ Стоимость процесса разработки
- ▶ Метрики использования

Классификация метрик

- ▶ Метрики программного продукта
- ▶ Метрики процесса
- ▶ Метрики использования
 - ▶ Точность и полнота реализации задач пользователя
 - ▶ Затраченные ресурсы на эффективное решение задач пользователя

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 = $\text{LOC} / \text{Затраты}$
- ▶ Удельная стоимость = $\text{Затраты} / \text{LOC}$
- ▶ Качество кода = $\text{Число ошибок} / \text{LOC}$
- ▶ Документированность = $\text{Число страниц документации} / \text{LOC}$

Ещё метрики

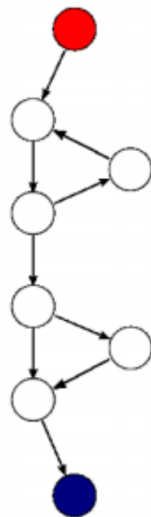
- ▶ Метрики Холстеда
- ▶ Метрики С. Чидамбера и К. Кемерера
- ▶ Метрики Ф. Абреу
- ▶ Метрики Л. Константейна и Э. Йордана
- ▶ Метрики Л. Отта и Б. Мехра
- ▶ Метрики Д. Биемена и Б. Кенга
- ▶ Метрики М. Лоренца и Д. Кидда
- ▶ Метрики Р. Байндера
- ▶ ...

Метрики Холстеда

- ▶ Number of Unique Operators (NUOprtr)
- ▶ Number of Unique Operands (NUOprnd)
- ▶ Number of Operators (Noprtr)
- ▶ Number of Operands (Noprnd)
- ▶ Halstead Program Volume (HPVol)
 $= (Noprtr + Noprnd) \times \log_2(NUOprtr + NUOprnd)$
- ▶ Halstead Difficulty (HDiff) $= \left(\frac{NUOprtr}{2}\right) \times \left(\frac{Noprnd}{NUOprnd}\right)$
- ▶ Halstead Effort (HEff) $= HDiff \times HPVol$

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)
 - ▶ $WMC = \sum_{i=1}^n C_i$, где C_i — как-то посчитанная сложность метода i
- ▶ Depth of Inheritance Tree (DIT)
- ▶ Number of children (NOC)
- ▶ Coupling between object classes (CBO)
 - ▶ Количество вызовов методов или полей
- ▶ Response For a Class (RFC) = $|\{M\} \cup_i \{R_i\}|$
 - ▶ $\{R_i\}$ — множество методов, вызываемых методом i
 - ▶ $\{M\}$ — множество всех методов в классе
- ▶ Lack of Cohesion in Methods (LCOM)
 - ▶ NotRelated — количество пар методов без общих полей/свойств
 - ▶ Related — количество пар методов с общими полями/свойствами

$$LCOM = \begin{cases} NotRelated - Related, & \text{если } NotRelated > Related. \\ 0, & \text{в противном случае.} \end{cases}$$

Полезные модификации WMC

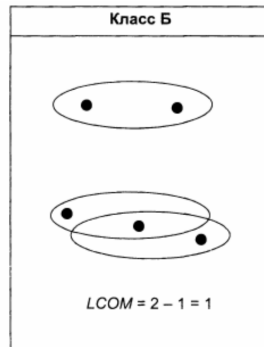
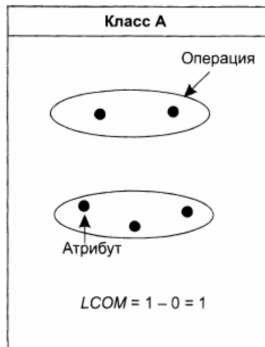
- ▶ $WMC2 = \sum_{i=1}^n$ количество параметров i-го метода
- ▶ ANAM (Average Number of Arguments per Method) = $\frac{WMC2}{WMC}$

```
SetInterval(min, max),  
SetMethod(method),  
SetPrecision(precision),  
SetFunctionToIntegrate(function),  
Integrate();
```

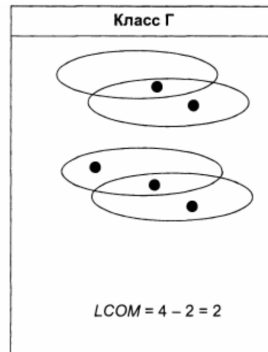
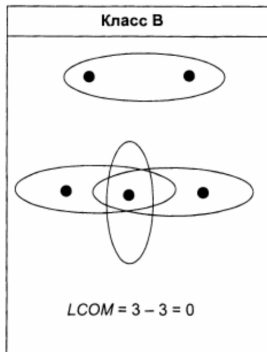
vs

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

LCOM: недостатки (1)



LCOM: недостатки (2)



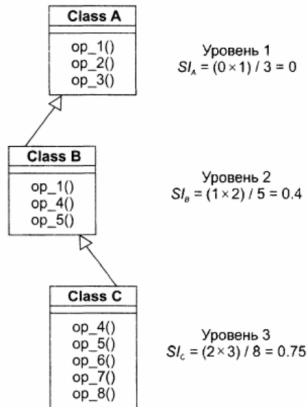
Модификация LCOM*

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

- ▶ 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)
 $SI = (NOO \times \text{уровень}) / M_{\text{общ.}}$
- ▶ 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)

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

- ▶ $M_h(C_i)$ — количество private-методов в классе C_i
- ▶ $M_a(C_i)$ — общее количество методов в классе C_i (без унаследованных)

Фактор закрытости свойства (AHF)

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

- ▶ $A_h(C_i)$ — количество private-атрибутов в классе C_i
- ▶ $A_a(C_i)$ — общее количество атрибутов в классе C_i

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 переопределенных методов в классе C_i
- ▶ $M_a(C_i)$ — общее количество методов в классе C_i

Фактор наследования свойства (AIF)

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

- ▶ $A_i(C_i)$ — количество унаследованных и не переопределенных атрибутов в классе C_i
- ▶ $A_a(C_i)$ — общее количество атрибутов в классе C_i

Фактор полиморфизма (POF)

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

- ▶ $M_o(C_i)$ — 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)

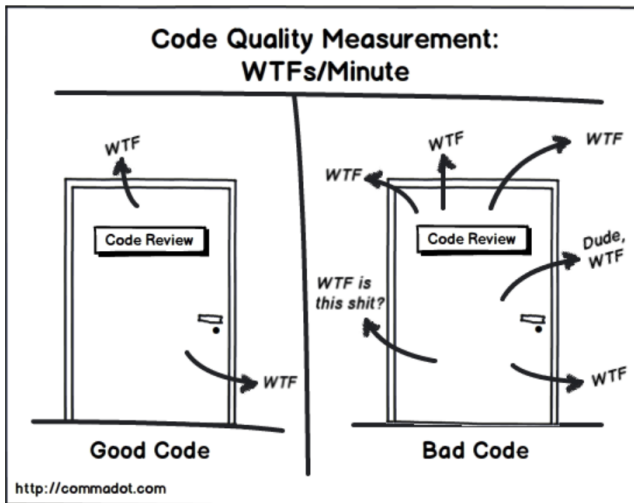
$$COF = \frac{\sum_{i=1}^{TC} \left(\sum_{j=1}^{TC} is_client(C_i, C_j) \right)}{TC^2 - TC}$$

$$is_client(C_c, C_s) = \begin{cases} 1, & \text{если } C_c \Rightarrow C_s \cap C_c \neq C_s \\ 0, & \text{в противном случае.} \end{cases}$$

- $C_c \Rightarrow C_s$ — класс-клиент содержит по меньшей мере одну не унаследованную ссылку на атрибут или метод класса-поставщика

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

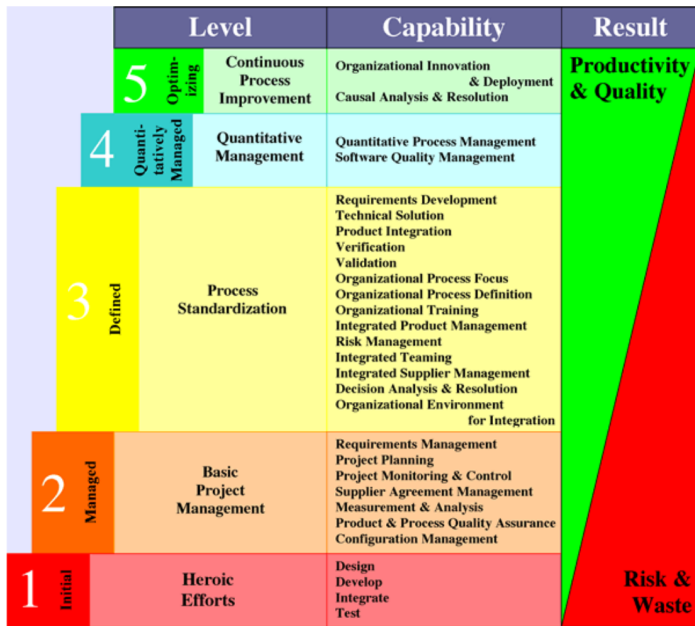


Аудит программного кода

- ▶ Сбор информации, накопление знаний, формирование эталонов
- ▶ Ручной
 - ▶ Экспертный
 - ▶ Расчётный
- ▶ Автоматический
 - ▶ <https://plugins.jetbrains.com/plugin/93-metricsreloaded>
 - ▶ <http://metrics.sourceforge.net/>
 - ▶ <https://www.codacy.com/>

Capability Maturity Model Integration (CMMI)

- ▶ Комплексная модель производительности и зрелости компании
- ▶ Пять уровней зрелости
- ▶ 22 области усовершенствования
 - ▶ Управление процессами
 - ▶ Управление проектами
 - ▶ Инженерные области
 - ▶ Служебные области
- ▶ Цели: общие и специфические
- ▶ Best Practices



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