ООП, лекция 3-4

Качество ПО
(самые общие соображения)

Соломатин Д.И., ПиИТ, ФКН, ВГУ
solomatin@cs.vsu.ru

Факторы качества

- Качество программного обеспечения – что это?
  - Внешние факторы – видимы пользователю
    - корректность
    - устойчивость
    - расширяемость (в случае системных библиотек, API и т.д.)
    - повторное использование (в случае системных библиотек, API и т.д.)
    - совместимость
    - эффективность
    - переносимость
    - простота использования
    - функциональность
    - своевременность
    - и т.п.
  - Внутренние факторы – видимы только разработчику
    - расширяемость
    - повторное использование
    - переносимость
    - читаемость
    - модульность
    - легкость обнаружения и исправления ошибок
    - и т.п.

Внешние факторы качества ПО

- **Корректность**. Программа должна делать то, что от нее ждут. Грубо говоря, программа не должна содержать ошибок.
- **Устойчивость**. Аварийные ситуации, которые могут возникнуть во время работы программы, не должны приводить к плачевным последствиям.
- **Совместимость**. Программа должна корректно работать в окружении других программ.
- **Эффективность.** Это способность ПО как можно меньше зависеть от ресурсов оборудования. Программа должна работать за приемлемое время на как можно более широком круге аппаратных конфигураций (имеются в виду различные по производительности конфигурации одной и той же платформы).
- **Простота использования**. Освоение программы не должно представлять затруднений для пользователя.
- **Функциональность**. Система не должна уметь больше, чем необходимо, ведь это делает ее громоздкой для пользователя и осложняет разработку. Однако функциональность не должна быть недостаточной — программа должна уметь все, что нужно пользователю.
- **Своевременность**. Программа должна появляться тогда, когда она нужна. Если она появится с опозданием, в ней, вероятнее всего, уже не будет смысла для пользователя.

Связь факторов

- Внешние факторы качества ПО — конечная цель, которую необходимо достичь.

- Считается, что достичь эту цель можно, поддерживая на высоком уровне внутренние факторы качества. Т.е. внутреннее качество порождает внешнее.

Дизайн программной системы

- (Внутренний) дизайн программной системы можно определить как совокупность свойств системы, определяющих ее внутреннее качество
  - единство дизайна
  - используемые технологии
  - качество кода
  - документация
  - и т.д.

Единство дизайна

- Все части программы должны быть написаны одинаково.
- Много разных «хороших дизайнов» в одной программе — это один большой плохой дизайн.

Качество исходного кода

- Можно трактовать
  - в узком смысле (соответствие стандартам кодирования, наличие комментариев и т.д.)
  - в широком смысле (с учетом архитектуры системы)
- Качество в широком смысле важнее

Желательные характеристики проекта

- **Минимальная сложность** (интегральная характеристика)
- **Простота сопровождения** (программы давно уже пишут для людей, а не машин)

- **Слабое сопряжение** (минимальное число соединений между различными частями программы)
- **Расширяемость** (возможность расширять/улучшать систему не затрагивая ее основную структуру)
- **Повторное использование кода** (устранения любых дублирований кода)
- **Высокий коэффициент объединения по выходу** (к конкретному классу обращается большое число других классов)
- **Низкий или средний коэффициент разветвления по входу** (конкретный класс обращается с малым числом других классов)
- **Портируемость** (простота адаптирования системы к другой среде)
- **Минимальная, но полная функциональность**
- **Стратификация** (разделение уровней декомпозиции, позволяющее изучить систему на любом уровне, не потеряв при этом согласованное ее представление; проще говоря – разделение по уровням)
- **Соответствие стандартным методикам**

Плохо и очень плохо

- Дублирование кода
- Глобальные локальные переменные
- Работа с глобальными переменными вместо передачи параметров в функции (характерно для процедурного подхода)
- Неоправданно тесная связь единиц программы (методов, модулей и т.д.)
  - В программа, в которой Form2 читает свойства компонентов Form3, и меняет значения глобальных переменных в модуле, описывающем Form1, начисто лишена какой-либо архитектуры (попросту «каша»)
  - Подобную программу в большинстве случаев невозможно сопровождать (проще и дешевле выкинуть и написать новую)
- Отсутствие разделения логики и отображения
  - Например, код в обработчике события по нажатию кнопки, показывает диалог выбора файла, проверяет корректность выбранного имени файла, производит обработку данных файла, результат обработки показывает на форме, и все это без делегирования функциональности другим частям программы)
- и т.д.

Еще один типичный пример в коде студентов

- Программа, в которой надо задать свойства в диалоговом окне
  - Решение студента:

    MainForm открывает OptionsForm, в OptionsForm по нажатию кнопки OK значения параметров пишутся в свойства MainForm (хорошо, если не в глобальные переменные)

  - Нормальное решение:
    - Для параметров заведен новый тип данных (структура или класс)
    - Написана функция, которая возвращает выбранные параметры в виде описанного типа (уже внутри функции создается и отображается OptionsForm)
    - MainForm вызывает данную функцию и обрабатывает возвращаемое значение
    - В результате MainForm и OptionsForm ничего не знают друг о друге (а знают лишь о типе, описывающем параметры)