Лабораторные работы «Командная разработка программных средств»

к.т.н. Курганская О.В., доц. кафедры ИСиЗИ,
ИрГУПС-2017

Лабораторная 1.3. Кодирование, ревью кода

- *Разбор предыдущей лабораторной.*
- *Задание 1*. Кодирование, написание тестов.
- *Задание 2.* Ревью кода.
  - В ревью должны быть задействованы все участники группы (как авторы или как рецензенты).
  - Контрольные списки должны быть разработаны самостоятельно (минимум 5 пунктов списка).
  - В ревью могут участвовать: код, архитектура, юнит-тесты.
- *Отчет по лабораторной*: результат лабораторной, описание собственного вклада в решение задачи.

Разбор предыдущей лабораторной

« … люди постоянно должны изучать что-то новое, что бы не происходило стагнации личности, а позже и деградации. Это может вызвать потерю ценностей и бесцельное проведение жизни.»

Коллектив авторов

Основные замечания:

- Отсутствие или низкое качество псевдокода
- Своеобразное представление результатов проектирования (одна диаграмма классов, миллион непонятых блок-схем)
- Отсутствие соглашений по защитному программированию
- Отсутствие соглашений по структуре ini-файлов (не у всех).
- Отсутствие соглашение о получении обрабатываемого файла (не у всех)
- Отсутствие документирования, тестирования и отладки как задачи
- Необоснованные оценки трудозатрат (29 часов на весь проект).
- Загадочные имена функций и переменных (OrigINItoOutINI, type Data struct)
- Наличие немотивированных отступлений.

Контрольные списки для ревью кода

Ревью кода: пример контрольного списка

Общие вопросы:

- Работает ли код? Выполняет ли он свои прямые обязанности, корректна ли логика, и т. д.
- Легок ли код для понимания?
- Соответствует ли код вашему стилю написания кода? Обычно это относится к расположению скобок, названиям переменных и функций, длинам строк, отступам, форматированию и комментариям.
- Есть ли в ревью избыточный или повторяющийся код?
- Является ли код связным насколько это возможно?
- Можно ли избавиться от глобальных переменных или переместить их?
- Есть ли закомментированный код?
- У циклов есть установленная длина и корректные условия завершения?
- Может ли что-то в коде быть заменено библиотечными функциями?
- Может ли быть удалена часть кода, предназначенного для логирования или отладки?

Ревью кода: пример контрольного списка

- Безопасность и защитное программирование
  - Все ли входные данные проверяются (на корректный тип, длину, формат, диапазон)?
  - Обрабатываются ли ошибки при использовании сторонних утилит?
  - Выходные данные проверяются?
  - Обрабатываются ли неверные значения параметров?
  - Ведется ли журнал ошибок?

Ревью кода: пример контрольного списка

Документация

- Есть ли комментарии? Раскрывают ли они смысл кода?
- Все ли функции прокомментированы?
- Есть ли какое-то необычное поведение или описание пограничных случаев?
- Использование и функционирование сторонних библиотек документировано?
- Все ли структуры данных и единицы измерения описаны?
- Есть ли незавершенный код? Если есть, должен ли он быть удален или помечен маркером типа «TODO»

Ревью кода: пример контрольного списка

Тестирование

- Является ли код тестируемым? Например, он не должен содержать слишком много зависимостей или скрывать их, тестовые фреймворки должны иметь возможность использовать методы кода, и т. д.
- Есть ли тесты и если есть, то достаточны ли они? Например, они покрывают код в нужной мере.
- Юнит-тесты на самом деле проверяют, что код предоставляет требуемую функциональность?
- Все ли массивы проверяются на «выход за границы»?
- Может ли любой тестирующий код быть заменен с использованием существующего API?

Тестирование в Go

Юнит-тесты

- Пакет testing
- Файл ИмяТестируемойСущности_test.go
- В этом файле:

func TestИмяТестируемойФункции(t *testing.T)

- Команда go test

Пример

```go
// adder.go
package adder

import "errors"

func AddPositive(a, b int) (int, error) {
    if a < 0 || b < 0 {
        return 0, errors.New("only positive integers allowed")
    }
    return a + b, nil
}
```

Пример

```go
// adder_test.go
package adder

import "testing"

func TestAddPositiveSuccess(t *testing.T) {
    expected := 5
    res, err := AddPositive(2, 3)
    if err != nil {
        t.Fatal(err)
    }
    if res != expected {
        t.Fatal("Achtung! value != expected")
    }
}

func TestAddPositiveFail(t *testing.T) {
    _, err := AddPositive(2, -3)
    if err == nil {
        t.Fatal("must return error")
```

Пример

```
bash-3.2$ go test
--- FAIL: TestAddPositiveFail (0.00s)
    adder_test.go:19: must return error
FAIL
exit status 1
FAIL    _/tmp/adder 0.009s
```

Отчет по лабораторной

Состав отчета

По заданию 1:

- архитектура проекта
- написанный автором код

По заданию 2:

- Роль автора отчета (рецензент или рецензируемый),
- код, предоставленный на ревью
- контрольный список
- замечания по коду
- ответы на замечания.