Юрий Ковалёв
инженер-программист,
АО «Аркадия»

Чистый код в коммерческой разработке. Есть ли предел совершенству?

Немного о себе

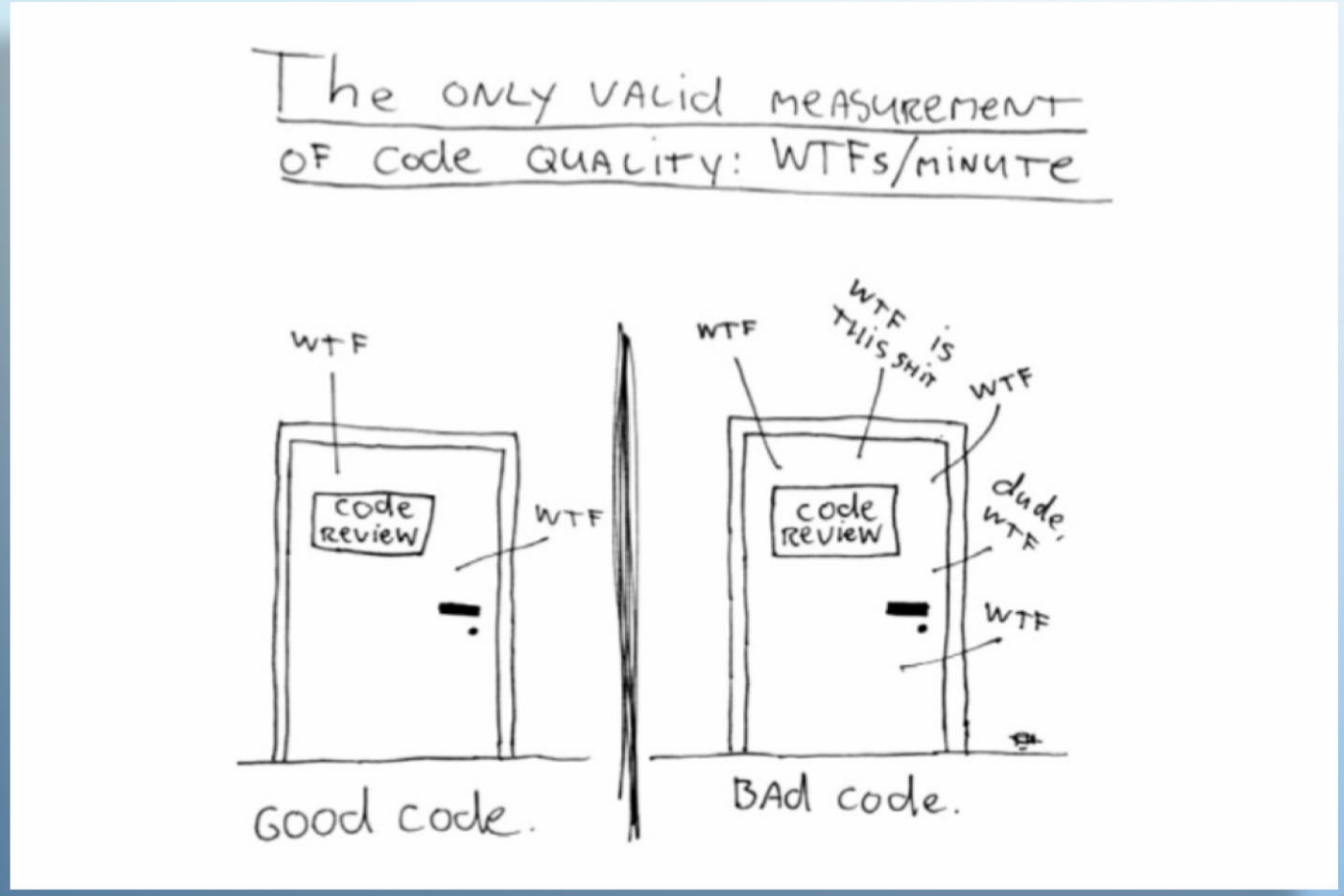
ЮРИЙ КОВАЛЁВ

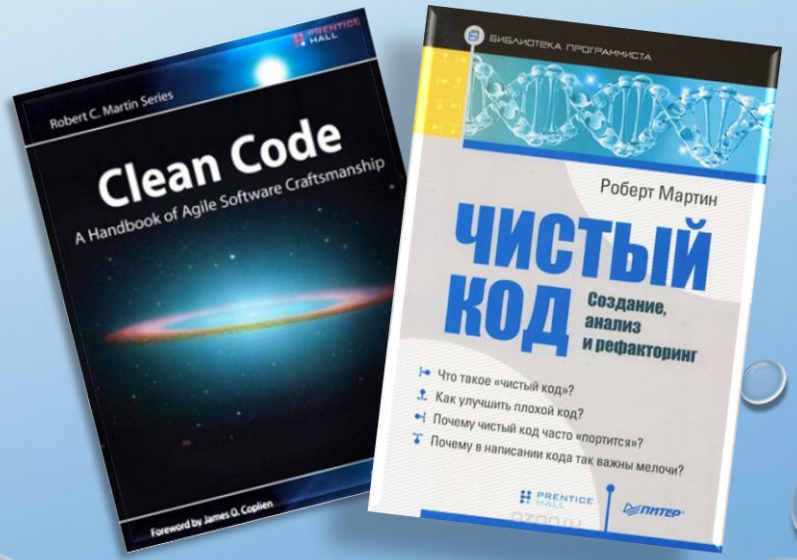
АО «АРКАДИЯ»

YURIY.KOVALEV@SOFTWARECOUNTRY.RU

- Full-stack программист
- PHP, JS, TypeScript, Angular, C++, Delphi
- Опыт работы в IT сфере – более 10 лет, в том чисте разработка высоконагруженных коммерческих систем.

Что такое чистый код?





Главные признаки чистого кода

УДОБОЧИТАЕМОСТЬ

- ✓ Единые и логичные правила форматирования
- ✓ Ясные и ёмкие имена
- ✓ Лаконичность

ЛЁГКОСТЬ ВНЕСЕНИЯ ИЗМЕНЕНИЙ

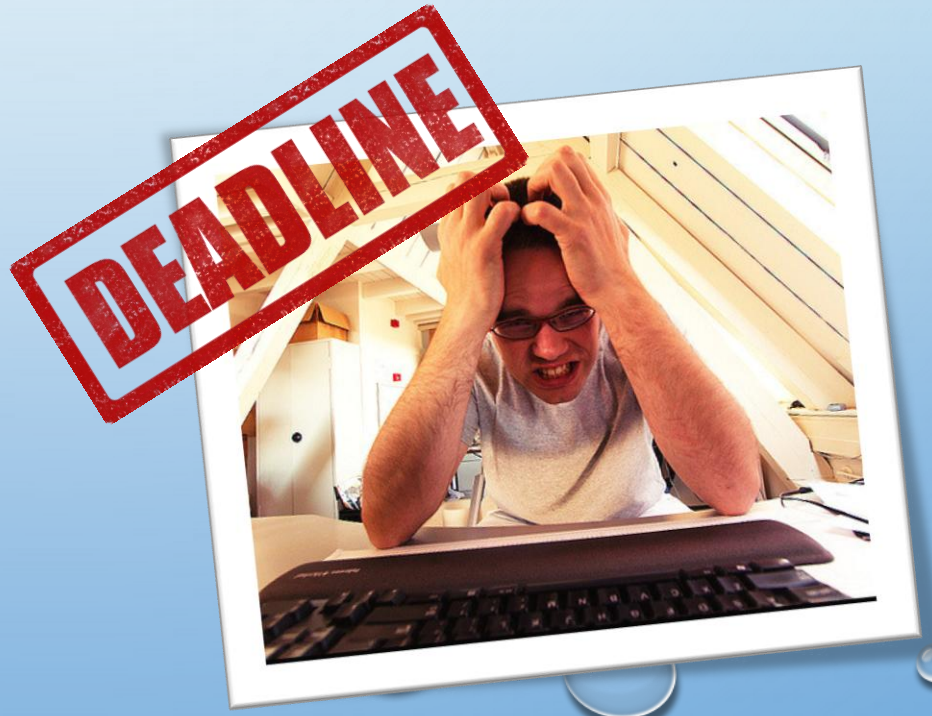
- ✓ SOLID
- ✓ Паттерны проектирования
- ✓ *Удобочитаемость*
- ✓ Тестируемость

Главные признаки грязного кода

- ✓ На понимание кода тратится очень много времени
- ✓ Для анализа тела функции требуется вертикальный скроллинг
- ✓ Строки кода широки и требуют горизонтального скроллинга
- ✓ Код требует обильных комментариев
- ✓ Высокая связность кода между компонентами системы, затрудняющая понимание происходящего
- ✓ Дублирование фрагментов кода

Причины грязного кода

- **Спешка, горящие сроки**
- Нежелание тратить время «впустую», когда задача уже «выполнена»
- Усталость после тяжёлого дебаггинга
- По неопытности и незнанию
- Отсутствие чёткого ТЗ
- Плохая архитектура



Яркие примеры из жизни

```php
function resolve_action($action) {
    return $action;
}
```

Done!

```php
if (indeterminate === true) {
    indeterminate = true;
} else {
    indeterminate = false;
}
```

Really?

```php
$TEST->id = $tempSurveyId;
$TEST->qt_load_db();

if ($TEST->show_all_replies) {
    $TEST->id = $surveyid;
    $TEST->qt_load_db();
}
```

Once again

```php
function dummy_users_callback($a,$b) {
    $a = $a;
    $b = $b;
    return array();
}
```

Indeed

Lifecycle чистого и грязного кода

- *См. «Главный Закон Контроля Качества ПО», С. Макконнелл – «Совершенный код»: гл. 20.5*

Практическая сторона

```
/**
 * Verify
 * @access public
 */
function verify_answer($answer) {
    if ($this->type == MATCH ) {
        return $this->question_object->validateAnswer( $answer );
    } else if ($this->mandatory) {
        if (is_array($answer)) {
            if (count($answer)==0) {
                return 1;
            }
        } elseif ($this->type == DROPDOWN ) {
            if($answer==-1) {  //not selected
                return 1;
            }
        } else {
            if (trim($answer) == '' ) return 1;
        }
    }
    return 0;
}
```

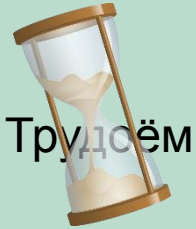
Основное правило IT-бойскаута

Всегда оставляй код чище, чем он был до твоего прихода

Бойскаут за работой

Шаг 1. Приведение к единому стилю форматирования

```
public function verify_answer($answer) {
    if ($this->type == MATCH) {
        return $this->question_object->validateAnswer($answer);
    } else if ($this->mandatory) {
        if (is_array($answer)) {
            if (count($answer) == 0) {
                return 1;
            }
        } else if ($this->type == DROPDOWN) {
            if ($answer == -1) { //not selected
                return 1;
            }
        } else if (trim($answer) == '') {
            return 1;
        }
    }
    return 0;
}
```



Трудоёмкость
1-5 мин.

Эффект
5-10%

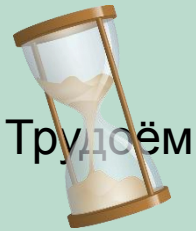
Применимо
всегда

Бойскаут за работой

Шаг 2. Повышение выразительности кода

```
public function verify_answer($answer) {
    if ($this->type == MATCH) {
        return $this->question_object->validateAnswer($answer);
    }

    if ($this->mandatory && (
        (is_array($answer) && (count($answer) == 0)) ||
        (is_string($answer) && (trim($answer) == '')) ||
        (($this->type == DROPDOWN) && ($answer == -1))
    )) {
        return 1;
    }
    return 0;
}
```



Трудоёмкость
2-10 мин.

Эффект
1-15%

Применимо
часто

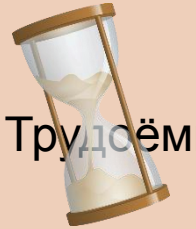
Бойскаут за работой

Шаг 3. Присвоение хороших имён

```
public function isAnswerInvalid($answer) {
    if ($this->type == MATCH) {
        return $this->question_object->isAnswerInvalid($answer);
    }

    $hasNoAnswer =
        (is_array($answer) && (count($answer) == 0)) ||
        (is_string($answer) && (trim($answer) == '')) ||
        (($this->type == DROPDOWN) && ($answer == NO_ANSWER));

    return ($this->mandatory && $hasNoAnswer) ? 1 : 0;
}
```



Трудоёмкость
5-20 мин.

Эффект
15-25%

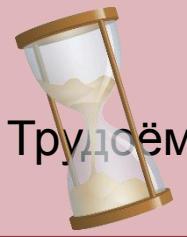
Применимо
всегда

Бойскаут за работой

Шаг 4. Улучшение архитектуры

```
class DropdownQuestion {
    ...
    public function isAnswerInvalid($answer) {
        $hasNoAnswer = ($answer == NO_ANSWER);
        return ($this->mandatory && $hasNoAnswer);
    }
    ...
}

class MatchQuestion extends Question {
    ...
    public function isAnswerInvalid($answer) {
        return $this->question_object->isAnswerInvalid($answer);
    }
    ...
}
```



Трудоёмкость
15-30 мин.

Эффект
20-50%

Применимо
редко

Бойскаут за работой

Тесты

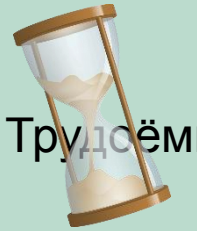
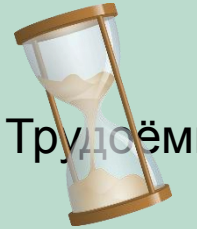
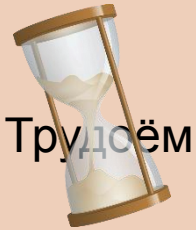
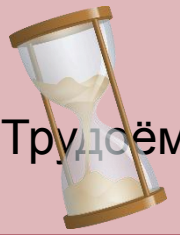
ПРЕИМУЩЕСТВА

- ✓ Упрощают проведение масштабного рефакторинга
- ✓ Позволяют глубже понять требования к системе
- ✓ Защищают код от ошибок

НЕДОСТАТКИ

- ✓ Трудозатратно
- ✓ Заметно замедляется процесс разработки
- ✓ Полное покрытие тестами не всегда достижимо

Итоги

Приведение кода к единому стилю форматирования	 Трудоёмкость **низкая**	 Эффект **5-10%**	 Применимо **всегда**
Повышение выразительности кода	 Трудоёмкость **низкая**	 Эффект **1-15%**	 Применимо **часто**
Присвоение хороших имён	 Трудоёмкость **средняя**	 Эффект **15-25%**	 Применимо **всегда**
Улучшение архитектуры	 Трудоёмкость **высокая**	 Эффект **20-50%**	 Применимо **редко**

Список полезных источников

- **Р. Мартин – «Чистый код. Создание, Анализ и рефакторинг», 2010**
- М. Фаулер – «Рефакторинг. Улучшение существующего кода», 2008
- М. К. Физерс – «Эффективная работа с унаследованным кодом», 2016
- Дж. Кериевски – «Рефакторинг с использованием шаблонов», 2006
- С. Макконнелл – «Совершенный код», 2010