<epam>

iT-34

Что не так с моим кодом или 7 причин моих страданий

Андрей Воробьёв

30.11.2019

vk.com/it34_community

Почему я здесь?

- Приходится иметь дело с плохим или нестабильным кодом

Пример номер раз

- Нет опыта разработки;
- Единственный разработчик.

Пример номер раз

Результат:

- Огромные классы и методы;
- Классы имели множество ответственностей и были сильно связаны друг с другом;
- Отсутствие юнит-тестов;
- Расширять функциональность с каждым разом становилось всё сложней.

Пример номер два

Класс-команда для вычисления:

- Класс разбит на 7 файлов;
- Общий объем почти 10к строк кода;
- Слишком много умеет и знает.

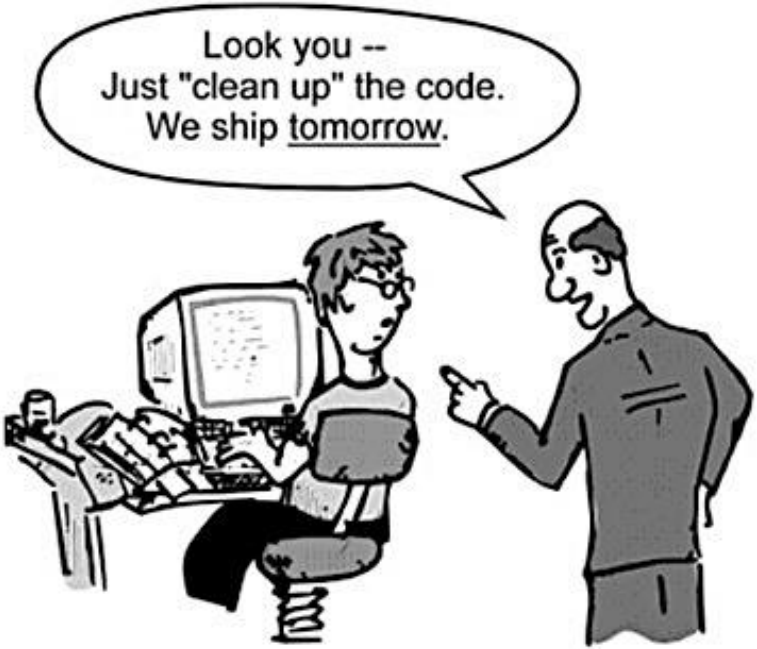
Пример номер два

Результат:

- Запутанный и нестабильный класс;
- Невозможно покрыть тестами;
- Страшно изменять.

7 смертных грехов программирования

1. Спешка

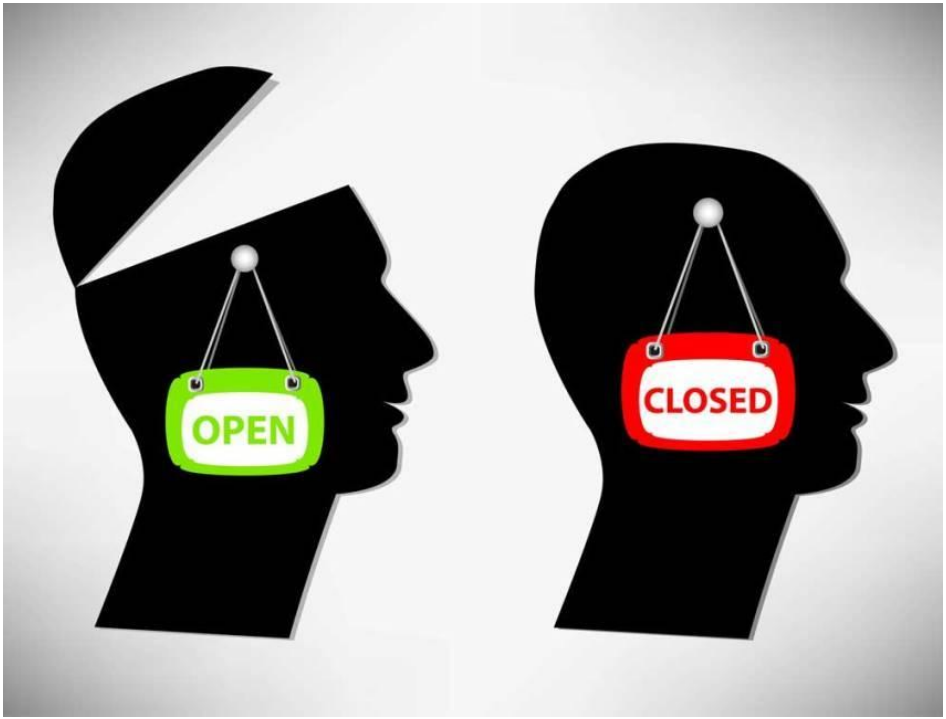


2. Апатия

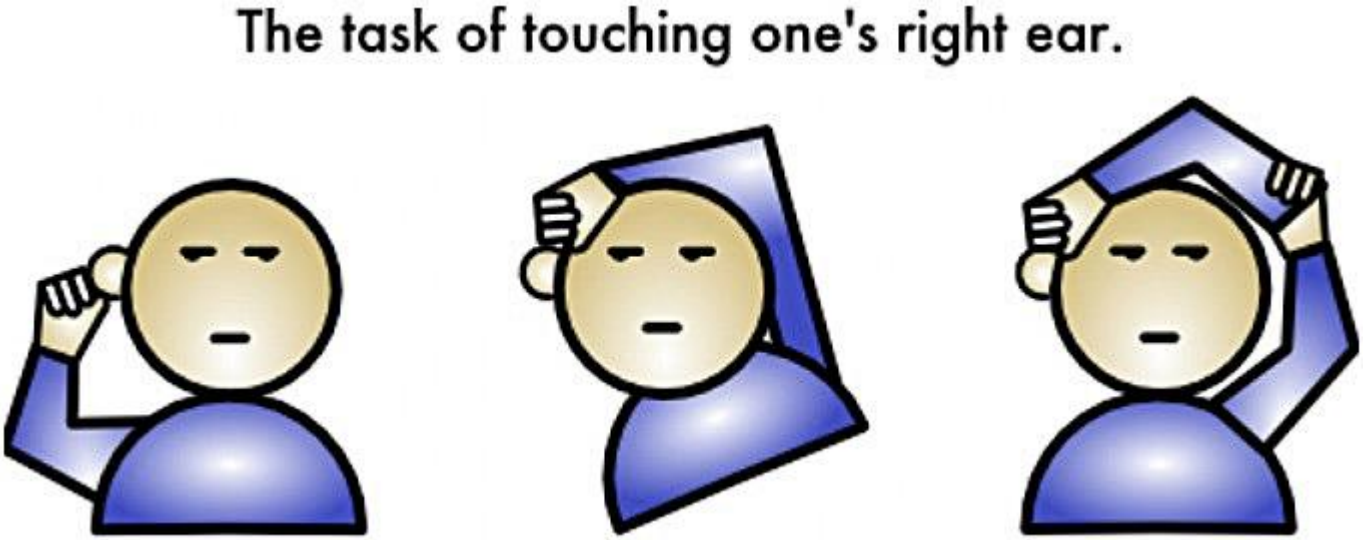

Reuse? Reuse!!
Who's ever gonna reuse
this crappy code?
NO ONE! That's who.

3. Леность

4. Ограниченность мышления

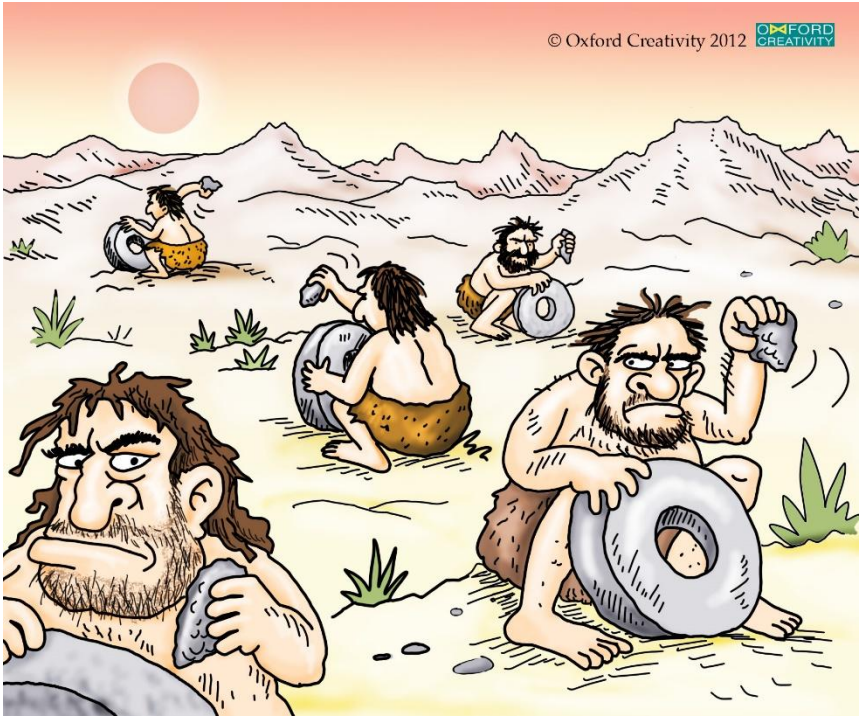


5. Алчность



6. Невежество

7. Гордыня



Наиболее частые ошибки. Антипаттерны

Антипаттерны разработки

God Object

— *Мне нужен такой-то функционал*

— *Используй MegaCoreObject!*

— *А ещё, мне нужен …*

— *Я же сказа*

God Object

Признаки:

- Большое число несвязных и несогласованных свойств и методов;
- Один класс содержит всю основную логику приложения;
- Внесение нового кода в уже существующие классы, вместо создания новых и пересмотра иерархии классов для лучшего распределения обязанностей.

God Object

Следствие:

- Сложно вносить изменения внутри God Object;
- Сложно использовать повторно;
- Сложно тестировать.

Golden Hammer (Silver Bullet)

— Когда у тебя в руках есть только молоток, тогда всё вокруг превращается в гвозди.

А. Маслоу

Golden Hammer (Silver Bullet)

Предпосылки:

- Стремление использовать знакомую технологию;
- Отсутствие опыта с другими технологиями;
- Риск при переходе на другое решение;
- Политические причины;
- Уверенность в преимуществах собственного решения.

Golden Hammer (Silver Bullet)

Следствие:

- неоптимальное решение;
- ненужное усложнение или недопустимое упрощение системы.

Lava Flow

— Не знаю, как это работает, но оно работает. Не удалять и не менять!

Lava Flow

Предпосылки

- Отсутствие Code Review;
- Отсутствие проектирования разработки;
- Недостаток опыта работы с технологией;
- Нет времени на рефакторинг и технический долг.

Lava Flow

Следствие:

- Увеличивается сложность проекта;
- Замедляется скорость разработки проекта;
- Сложно провести рефакторинг или внести новую функциональность.

Кулинарные антипаттерны

SPAGHETTI-ORIENTED
ARCHITECTURE
(aka Copy & Paste)

LASAGNA-ORIENTED
ARCHITECTURE
(aka Layered Monolith)

RAVIOLI-ORIENTED
ARCHITECTURE
(aka Microservices)

WHAT'S NEXT?

PROBABLY PIZZA-ORIENTED ARCHITECTURE

Hardcoding & Magic Number

```
var path = "C:/Projects/MyProject/Data/FirstTest";

var connection = Service.Connect("localhost/api");

var peopleAmount = GetEuropePopulation() + 602005;
```

Hardcoding & Magic Number

Предпосылки:

- Хардкод во время разработки или отладки;
- Спешка.

Hardcoding & Magic Number

Следствие:

- Исправная работа только в окружении, в котором ведётся разработка;
- Неустойчивость к изменениям;
- Требуется повторный деплой.

Hardcoding & Magic Number

```
var path = "C:/Projects/MyProject/Data/FirstTest.xml";
```

```
var connection = Service.Connect("localhost/api");
```

```
var peopleAmount = GetEuropePopulation() + 602005;
```

Hardcoding & Magic Number

```
var path = Path.Combine(Directory.GetCurrentDirectory(), dataPath);
```

```
var connection = Service.Connect(Settings.OperationServiceAddress);
```

```
var peopleAmount = GetEuropePopulation() + LuxembourgPopulation2018;
```

Programming by permutation

Programming by permutation

Предпосылки:

- Отсутствие желания понять как работает код;
- Отсутствие документации;
- Низкая компетенция разработчика.

Programming by permutation

Следствие:

- Невозможно предусмотреть все сценарии;
- Будет потрачено время на решение задачи перебором, а после, повторно потратится время на переделку решения;
- Приучает разработчика к тому, что написание кода — это магия, а не инженерная работа.

Архитектурные антипаттерны

Over-Engineering

Over-Engineering

Предпосылки:

- Необходимость покрыть потребности небольшой части пользователей;
- Преждевременное усложнение системы;
- Желание продемонстрировать свои способности.

Over-Engineering

Следствие:

- Потрачено слишком много сил/времени/денег на функционал, который не нужен большинству пользователей;
- Продукт стал слишком дорогим и/или долгим, из-за чего он так и не был выпущен.

Not Invented Here & Reinventing the Wheel

Not Invented Here & Reinventing the Wheel

Предпосылки:

- Высокая цена решения;
- Недоверие к чужой разработке;
- Желание поддержать свой продукт/компанию;
- Гордыня;
- Вера в то, что свои программисты лучше всех.

Not Invented Here & Reinventing the Wheel

Следствие:

- Новый велосипед может оказаться недостаточно функциональным или надёжным;
- Нужно поддерживать собственными силами;
- Потеря денег/времени.

Заключение

Вопросы?
Истории?

КОНТАКТЫ

E-mail:
andrei_vorobev@epam.com