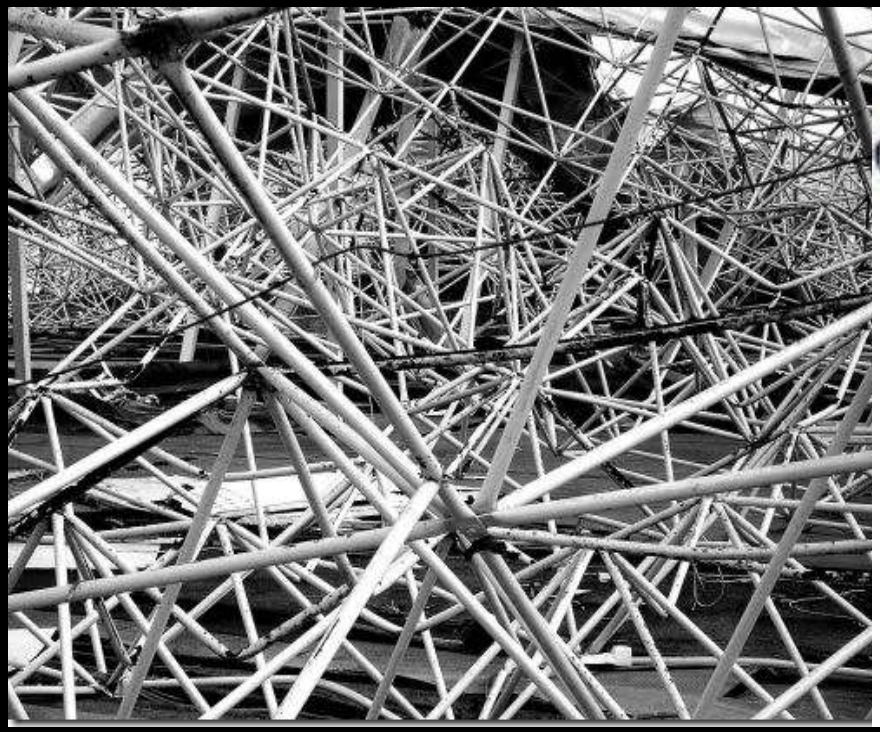
Простота Дизайна
Не раскрученная XP практика

Вячеслав Москаленко

Agile Coach, «Luxoft»

Наш мозг на грани выживания

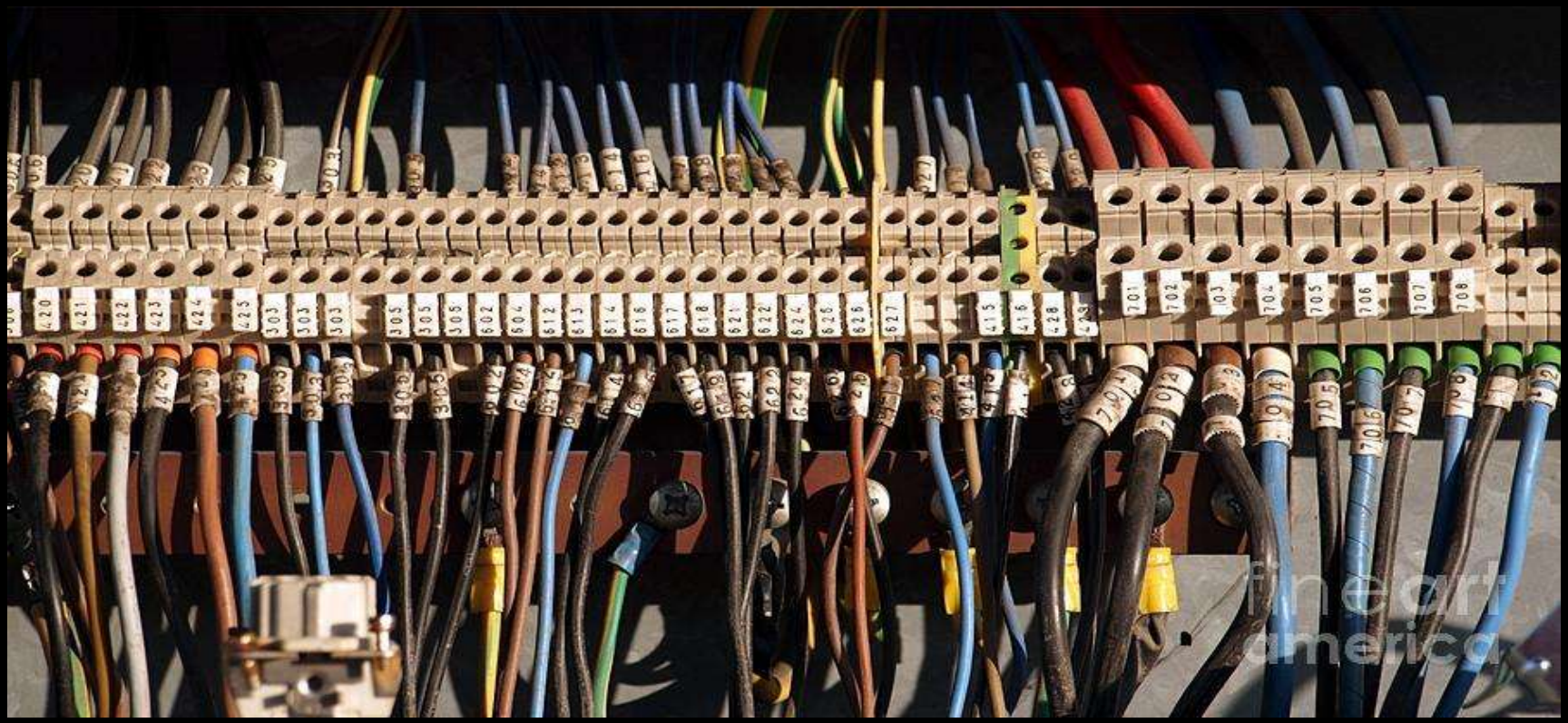


Все кажется запутанным и сложным

Сложные задачи

Сложная информация

Сложные связи



Сложный дизайн

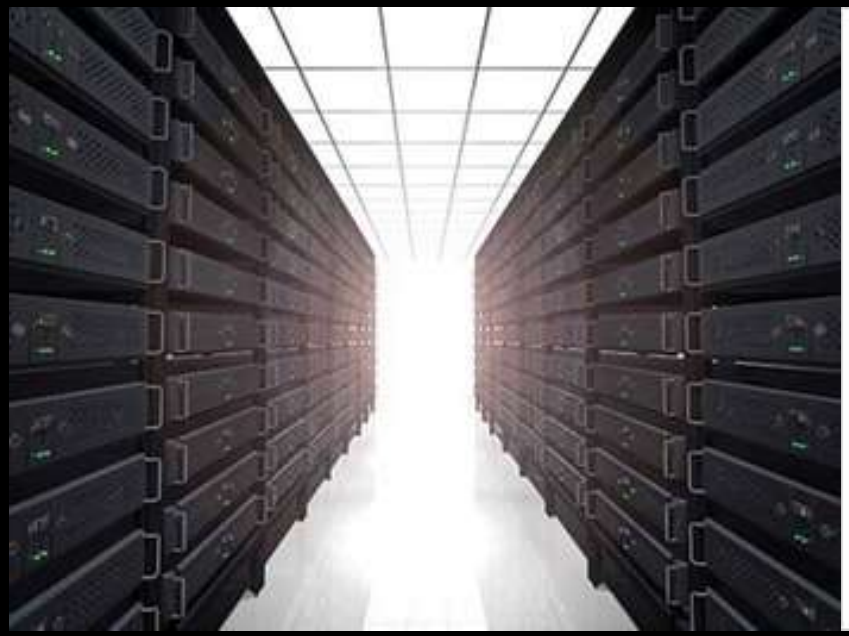
Простота встречается редко

Но что это означает ?

Умение достигать максимального эффекта

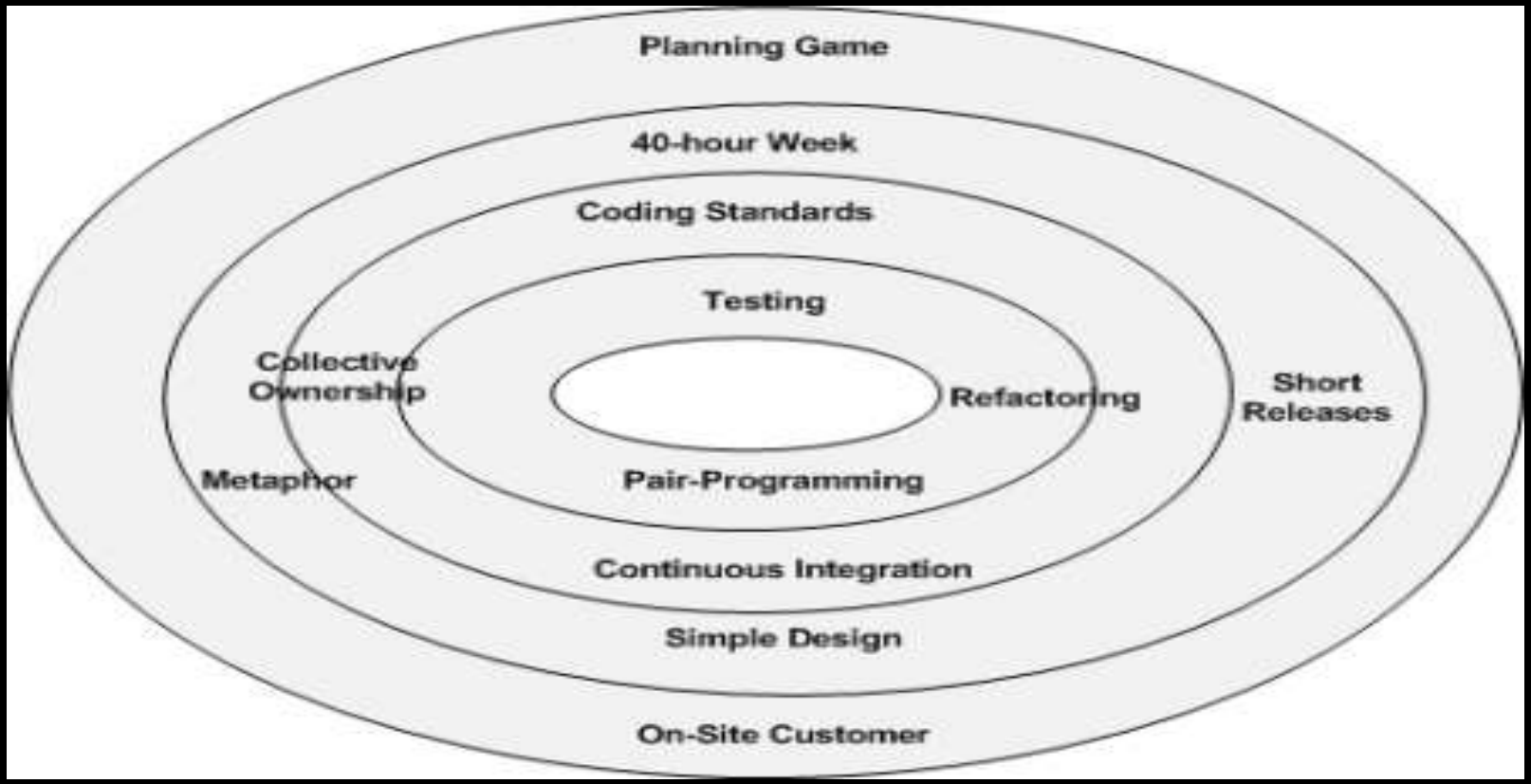
минимальными затратами

Приведение сложных вещей



К простым формам

Как уменьшить сложность дизайна ?

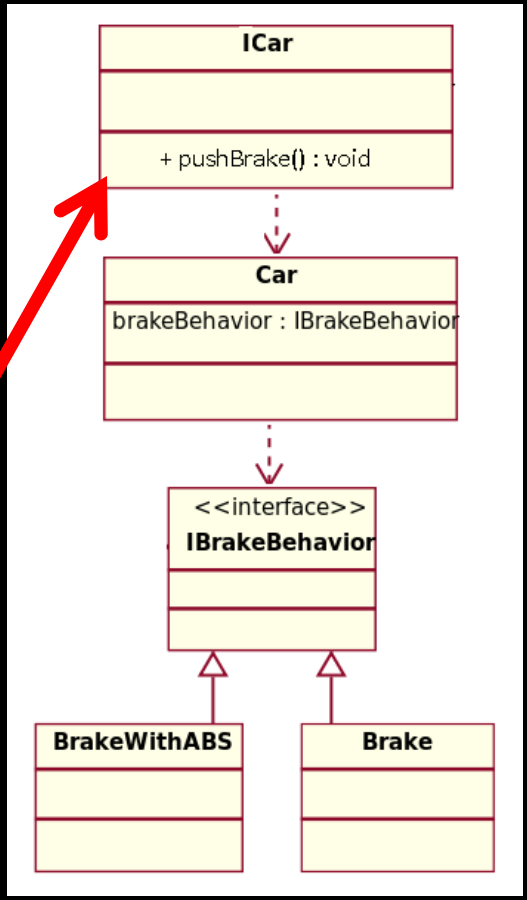


Путь экстремального программирования

- все тесты корректно срабатывают;
- дизайн не содержит дублирующегося кода;
- дизайн четко выражает намерения программиста;
- дизайн содержит наименьшее возможное количество классов и методов.

1. Фокусироваться на тестах

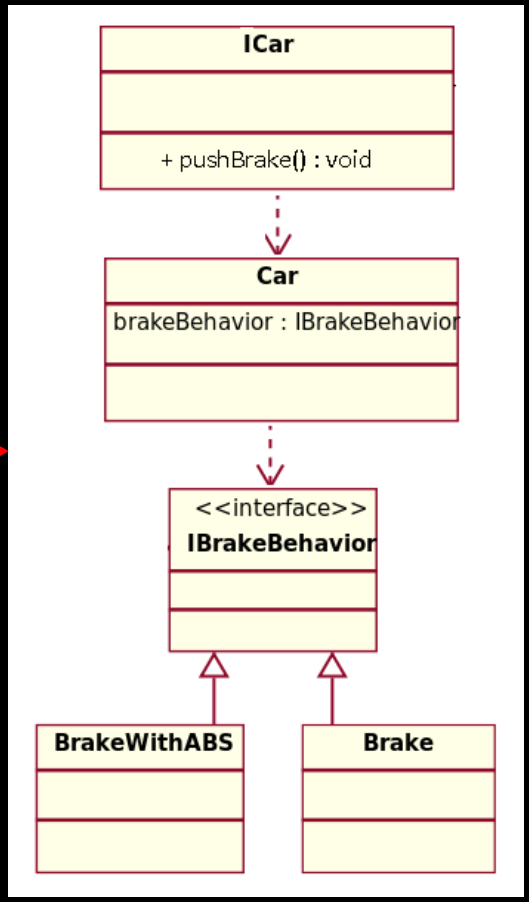
```
@Test() void
given_push_pressure_100_and_car_with_ABS_verify_br
aking_distance_30() {
    ICar car = prepareCarWithABS();
    car.pushBrake(100);
    assertThat(car.getBrakingDistance(), is(30));
}

@Test() void
given_push_pressure_50_and_car_without_ABS_verify_
braking_distance_25() {
     ICar car = prepareCarWithoutABS();
     car.pushBrake(100);
     assertThat(car.getBrakingDistance(), is(25));
}
```



Тесты простые, не отключены и падают редко. Шаблоны xUnit

2. Уменьшать количества кода

```
public void pushBrake(int pressure) {
    …
    if (carWithABS()) {
        …
    } else {
        …
    }
    …
    if (carWithABS()) {
    …
    } else {
        …
    }
    …
}
```



Шаблоны проектирования

3. Выражать намерение через код

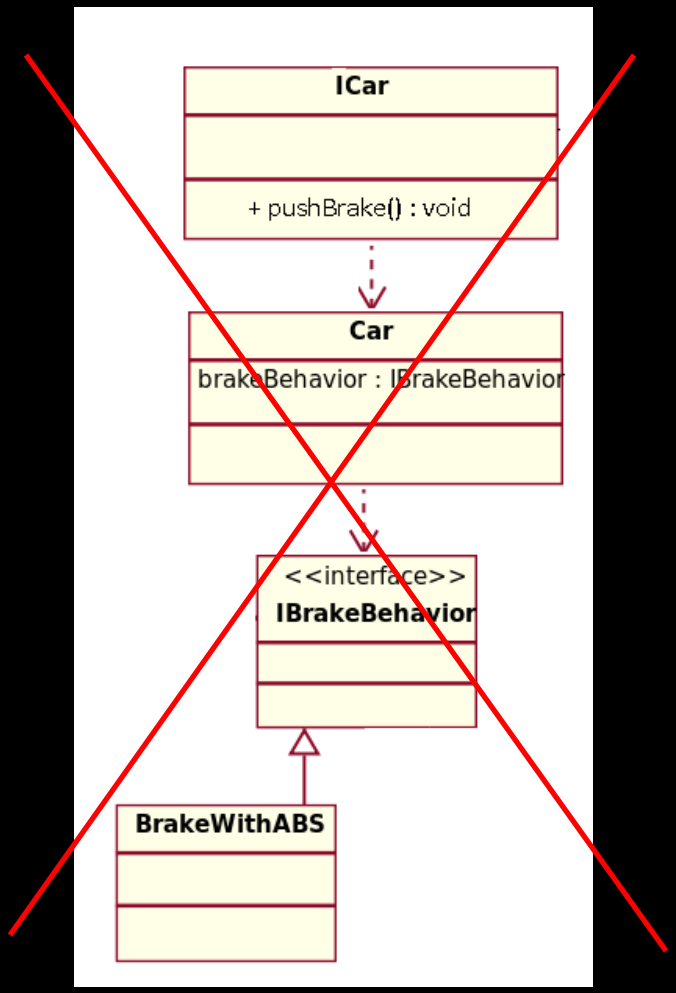
```
public List<int[]> getThem() {
   List<int[]> list1 = new  ArrayList<int[]>();
   for (int[] x : theList)
     if (x[0] == 4)
        list1.add(x);
     return list1;
}
```

```
public List<Cell> getFlaggedCells() {
   List<Cell> flaggedCells = new ArrayList<Cell>();
   for (Cell cell : gameBoard)
   if (cell.isFlagged())
      flaggedCells.add(cell);
   return flaggedCells;
}
```

Clean Code, SOLID

4. Не плодить сущности

- Фабрика, которая возвращает один тип объектов
- Библиотеки, которые никогда не используются
- Дополнительные абстракции, которые «потом» понадобятся



4. Упрощать внешние интерфейсы

Вот и все

Полезные дисциплины

- Design Patterns
- Emergent Design
- xUnit Patterns
- Refactoring Patterns
- Object Oriented Design
- Doman Driven Design

Изучаем сквозь призму «Простого Дизайна»

Послесловие

mvn clean install

Или за что я люблю Maven