LUXOFT TRAINING

Разработка через тестирование Anti-Patterns

Test Driven Development

Ivan Dyachenko <IDyachenko@luxoft.com>

Module 12

Анти-паттерны (anti-patterns)

Анти-паттерны (anti-patterns), также известные как ловушки (pitfalls) — это классы наиболее часто внедряемых плохих решений проблем. Они изучаются как категория, в случае, когда их хотят избежать в будущем, и некоторые отдельные случаи их могут быть распознаны при изучении неработающих систем.

Анти-паттерны (anti-patterns)

Анти-паттерны (anti-patterns), также известные как ловушки (pitfalls) — это классы наиболее часто внедряемых плохих решений проблем. Они изучаются, как категория, в случае когда их хотят избежать в будущем, и некоторые отдельные случаи их могут быть распознаны при изучении неработающих систем.

Определение антишаблонов

LUXOFT Training Center

- Антишаблоны выделяют и документируют приемы разработки, которые негативно влияют на систему
- Изучение антишаблонов помогает вам избегать их

Multiple Assertions

Тест содержит множество Asserts

```
public class MyTestCase extends TestCase {
    public void testSomething() {
        // Set up for the test, manipulating local variables
        assertTrue(condition1);
        assertTrue(condition2);
        assertTrue(condition3);
    }
}
```

Multiple Assertions

Решение – разбить тесть на три отдельных теста

```
public class MyTestCase extends TestCase {
    // Local variables become instance variables

    protected void setUp() {
        // Set up for the test, manipulating instance variables
    }

    public void testCondition1() {
        assertTrue(condition1);
    }

    public void testCondition2() {
        assertTrue(condition2);
    }

    public void testCondition3() {
        assertTrue(condition3);
    }
}
```

Using the Wrong Assert

Использование не подходящих assert-ов

```
assertTrue("Objects must be the same", expected == actual);
assertTrue("Objects must be equal", expected.equals(actual));
assertTrue("Object must be null", actual == null);
assertTrue("Object must not be null", actual != null);
```

Решение – использовать специализированные asserts

```
assertSame("Objects must be the same", expected, actual);
assertEquals("Objects must be equal", expected, actual);
assertNull("Object must be null", actual);
assertNotNull("Object must not be null", actual);
```

Catching Unexpected Exceptions

LUXOFT Training Center

Попытка обработать Unexpected Exceptions

```
public void testCalculation() {
    try {
        deepThought.calculate();
        assertEquals("Calculation wrong", 42, deepThought.getResult());
    } catch (CalculationException ex) {
        Log.error("Calculation caused exception", ex);
    }
}
```

Решение – не делать этого

```
public void testCalculation() throw CalculationException {
    deepThought.calculate();
    assertEquals("Calculation wrong", 42, deepThought.getResult());
}
```

The Liar (Лжец)

LUXOFT Training Center

- Тест, который успешно выполняет все кейсы и выглядит работающим правильно
- При детальном рассмотрении оказыается, что он тестирует не то, что нужно, либо вообще ничего не тестирует

Excessive setup (Попробуй запусти!)

LUXOFT Training Center

- Требует сложной и длительной настройки тестового окружения (сотни строк кода, подготавливающие к запуску одного теста)
- Если тест проваливается, требуется много времени чтобы разобраться, дефект ли это приложения или мы что-то упустили при подготовке
- Появление такого теста, как правило, сигнализирует о проблемах в архитектуре

The Giant (Гигант)

LUXOFT Training Center

- Огромный тест, выполняющий сотни проверок и содержащий множество тест-кейсов
- Обычно говорит о том, что мы имеем дело с **god-object** (класс, который слишком много знает или делает)

The Mockery (Подделка)

LUXOFT Training Center

- Тест содержит огромное количество mock- и stub-объектов
- В результате тестирует не столько поведение модуля, сколько поведение самих моков, стабов и результатов их работы

The Inspector (Инспектор)

LUXOFT Training Center

- Модульный тест использует самые изощренные методы для того, чтобы добраться до «самых приватных» членов класса
- Тест нарушает инкапсуляцию класса
- Любая попытка рефакторинга тестируемого класса приводит к переделке теста

Generous Leftovers (Щедрые Остатки)

LUXOFT Training Center

- Вариант, когда один unit-тест создаёт данные, которые другой тест потом переиспользует
- Если «генератор данных» будет по какой-то причине вызван позже или пропущен, то тест, использующий его данные, не пройдёт

The Local Hero (Местный герой)

LUXOFT Training Center

- Тест написан так, что исполняется только в конкретном окружении (например, только на машине разработчика)
- Попытка выполнить тест в другом окружении (например, на сервере сборки) ведет к неудаче

The Nitpicker (Крохобор)

LUXOFT Training Center

- Тест проверяет по шаблону весь поток выходных данных и валится при малейших изменениях в них, в то время как значимой для тестирования является лишь небольшая часть этих данных
- Часто встречается при тестировании web-приложений

The Secret Catcher (Тайнос Агентос)

LUXOFT Training Center

- На первый взгляд, тест ничего не проверяет (отсутствуют методы assert)
- Тест вызывает различные методы системы и рассчитывает на то, что при возникновении исключения, тестовый фреймворк самостоятельно обработает его и отрапортует

```
@Test(expected = Exception.class)
public void itShouldThrowDivideByZeroException() {
    // some code that throws another exception yet passes the test
}
```

The Dodger (Лентяй)

LUXOFT Training Center

- Тестирует множество побочных эффектов тестового случая вместо того, чтобы тестировать требуемое поведение
- Обычно возникает из-за того, что побочные эффекты протестировать легче

The Loudmouth (Болтун)

LUXOFT Training Center

- Модульный тест заваливает консоль диагностическими сообщениями, логами и прочей ненужной информацией
- Обычно возникают из-за того, что тест использовался разработчиком для отладки своего кода, после чего отладочные сообщения не были удалены

The Greedy Catcher (Жадный Ловец)

LUXOFT Training Center

- «Проглатывает» исключения в исходном коде, иногда вместе с трассировкой стека
- Заменяет их своими, менее информативными сообщениями, либо выбрасывает диагностическое сообщение и успешно завершается

The Sequencer (Любитель Порядка)

- Тест, который проявляет чувствительность, например, к последовательности элементов несортированного списка
- Тест, который зависит от последовательности выполнения тестов
- В первом случае появляются плавающие ошибки, во втором - нарушается изолированность юнит-тестов

Hidden Dependency (Скрытая зависимость)

LUXOFT Training Center

- Проявляет зависимость от данных, которые должны быть созданы где-то и кем-то до того, как тест будет исполнен
- Если нужных данных нет, тест обычно валится с невнятным сообщением, без намека на то, что ему необходимо для успешного завершения
- Нарушает изолированность юнит-тестов

The Enumerator (Счетчик)

LUXOFT Training Center

- Тест, каждый метод которого именуется просто: test1, test2, test3 и т.д.
- В результате абсолютно не понятно, что и где тестируется и единственный вариант узнать это – вникать в код

The Stranger (Чужак)

LUXOFT Training Center

- Тестовый метод, который тестирует что-то, не имеющее никакого отношения к тому, что тестирует остальной модульный тест
- Обычно случается из-за того, что нужно протестировать объект, связанный с тестируемым, но делается это в отрыве от назначенной связи

OS Evangelist (ОС Евангелист)

LUXOFT Training Center

- Тест, который предполагает, что будет исполняться под конкретной операционной системой
- Простой пример – использование в проверках символа перевода строки для Windows, при запуске под unix-системами такой тест провалится

Success Against All Odds (Успех Любой Ценой)

LUXOFT Training Center

- Тест, который изначально был написан для успешного прохождения, а не тестирования чего-либо
- Побочным эффектом является недостаточно глубокое тестирование и успешное прохождение там, где правильный тест должен провалиться

The Free Ride («Заяц»)

LUXOFT Training Center

- Вместо написания нового тестового метода добавляется еще одна проверка в уже существующий
- Увеличивает время на локализацию ошибки
- Ведет к появлению «Гигантов» и «Избранных»

The One (Избранный)

LUXOFT Training Center

- Комбинация нескольких анти-паттернов, особенно Гиганта и множества Зайцев
- Тест содержит всего один метод, который тестирует весь функционал класса
- Как правило, тестовый метод имеет такое же название, как и тестовый класс
- В самом методе – долгая настройка, после чего множество проверок

The Piping Tom (Любопытная Варвара)

- Благодаря разделяемым ресурсам, тест может «подсмотреть» результаты других тестов, в результате чего валится, несмотря на то что система находится в валидном состоянии
- Обычно возникает из-за использования статических переменных, которые не были очищены предыдущими тестами

The Slow Poke (Тормоз)

- Тест выполняется необычайно медленно
- Как правило, при запуске такого теста разработчик спокойно уходит на перекур, обед, или, что еще хуже, запускает его в конце рабочего дня перед уходом домой

The Cuckoo (Кукушонок)

LUXOFT Training Center

- Тест использует общие объекты, создаваемые при инициализации
- При своем выполнении уничтожает эти объекты, создавая вместо них необходимые ему реализации

The Bounty Hunter (Наемник)

- Тест покрывает как можно больше кода, с целью достичь необходимого уровня покрытия
- При этом толком ничего не тестируется

Roll The Dice (Русская рулетка)

LUXOFT Training Center

- Вместо тестирования граничных условий, генерируются случайные значения параметров
- В результате появляются плавающие ошибки - тесты иногда проходят, а иногда – нет

The Mother Hen (Наседка)

LUXOFT Training Center

- Общий setup, который делает гораздо больше, чем тестам нужно фактически
- Это может быть признаком того, что установка была написана до того, как тесты

The Wild Goose (Дикий гусь)

LUXOFT Training Center

- Модульный тест, который кажется простым, но требует все больших усилий для инициализации данных, которые необходимы, что бы тест прошел

The Homing Pigeon (Почтовый голубь)

LUXOFT Training Center

- Модульный тест, который надо запустить в определенном месте в иерархии классов

The Dodo

LUXOFT Training Center

- Модульный тест, который более не нужен.
- Тест и функциональность надо удалить

Happy Path (Счастливый путь)

Модульный тест не тестирует пограничные условия и исключения

```
public class Factorial {
    public int eval(int _num) {
        return 6;
    }
}
```

```
@Test
public void should_return_factorial() {
    Factorial factorial = new Factorial();
    Assert.assertEquals(6, factorial.eval(3));
}
```

Easy tests (Простые тесты)

LUXOFT Training Center

Модульный тест тестирует очевидные вещи

```
testEqualsReflexive()
testEqualsSymmetric()
testEqualsTransitive()
testEqualsOnNullParameter()
testEqualsWorksMoreThanOnce()
testEqualsFailsOnSubclass()
testEqualsIsStillReflexive()
```

Overly complex tests

LUXOFT Training Center

```
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;

import junit.framework.TestCase;

public class RecordTest extends TestCase {
    public void testRecordContainsCorrectCustomerData() {
        // setup
        String expectedName = "Estragon";
        int expectedId = 1001;
        String [] expectedItemNames = {"A man", "A plan", "A canal", "Suez"};
        Customer customer = new Customer(expectedId, expectedName,
                                         expectedItemNames);
        // execute
        BillingCenter.processCustomer(customer);
        // assert results
        File file = new File("customer.rec");
        assertTrue(file.exists());
        FileInputStream fis = new FileInputStream(file);
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        byte [] buffer = new byte[16];
        int numRead;
        while ((numRead = fis.read(buffer)) >= 0) {
            baos.write(buffer, 0, numRead);
        }
        byte [] record = baos.toByteArray();
        assertEquals(128, record.length); // exactly one record
        String actualName = new String(record, 0, 15).trim();
        assertEquals(expectedName, actualName);
        int [] temp = new int[4];
        temp[0] = record[15];
        temp[1] = record[16];
        temp[2] = record[17];
        temp[3] = record[18];
        int actualId = (temp[0] << 24) & (temp[1] << 16) &
                       (temp[2] << 8) & temp[3];
        assertEquals(expectedId, actualId);
        int itemFieldLength = 16;
        int itemFieldOffset = 19;
        for (int i = 0; i < 4; ++i) {
            String actualItemName = new String(record,
                                               itemFieldOffset + itemFieldLength * i, itemFieldLength);
            assertEquals(expectedItemNames[i], actualItemName.trim());
        }
    }
}
```

The Test With No Name

LUXOFT Training Center

Модульный тест без внятного имени

```
@Test
public void testForBUG123() {
    // some code
}
```

Wait and See (Подождем – увидим)

LUXOFT Training Center

- Модульный тест использует Thread.sleep() что бы “дождаться” необходимого условия

Вопросы ?

Разработка через тестирование

IDyachenko@luxoft.com

git clone git://github.com/ivan-dyachenko/Trainings.git

https://github.com/ivan-dyachenko/Trainings