LUXOFT TRAINING

Разработка через тестирование Legacy code

Test Driven Development

Ivan Dyachenko <IDyachenko@luxoft.com>

Module 9

Содержание

LUXOFT Training Center

Legacy-код

LUXOFT Training Center

Причины появления legacy-код(а):

- Появление новых технологий
- Развитие существующих технологий
- Бурное развитие системы

Появление новых технологий

LUXOFT Training Center

Существуют огромные программные комплексы, написанные на языке C++, который считался передовым всего десять лет назад

Развитие существующих технологий

Частые изменения в языках и платформах Java/C# очень быстро приводят к появлению унаследованных решений, поскольку то, что когда-то считалось новаторским, превращается в неподдерживаемый, устаревший код

Бурное развитие системы

Зачастую, программисты сосредоточивают усилия именно на реализации новых возможностей программного обеспечения – применяются новые технологии, но существующий функционал при этом не совершенствуется

Legacy-код

Итак, нужно ли писать тесты для legacy-кода?

Legacy-код

LUXOFT Training Center

Да, нужно!

Legacy-код

LUXOFT Training Center

- Unit-тесты позволят вам понять, как работает код
- Unit-тесты позволят вам убедиться, что ваши изменения не нарушили логику работы модуля

Legacy-код

LUXOFT Training Center

- Ваша цель – получить тесты, которым можно доверять
- Если функция полностью покрыта модульными тестами, то вы не будете боятся ее изменить или даже полностью переписать

Избавление от зависимостей

- Как правило, такой код имеет множество зависимостей: подключает различные lib и dll, посылает сообщения в сеть или другим компонентам, отображает что-то в GUI и т.д.
- Итак, первое, что нам необходимо сделать – это избавиться от этих зависимостей
- Для этого можно использовать уже знакомые нам mock- и stub-объекты

Dependency Injection (DI)

LUXOFT Training Center

- Зачастую, этих средств недостаточно и необходимо проводить рефакторинг
- Здесь на помощь нам приходит паттерн “Dependency injection”
- Суть его заключается в следующем: если класс A использует класс B, то необходимо сделать так, чтобы конкретная реализация класса B передавалась классу A “извне”, а не определялась внутри него

Рассмотрим пример

LUXOFT Training Center

```
public interface ISoundProvider {
    void Read();
    void Write();
}

public class FileSoundProvider implements ISoundProvider {
    void Read() {
        //implements reading audio from file
    }

    void Write () {
        //implements writing audio to file
    }
}
```

Рассмотрим пример

```
public class ExternalSoundProvider implements ISoundProvider {
    void Read() {
        //implements reading audio from external device
    }

    void Write() {
        //implements writing audio to external device
    }
}
```

- У нас есть интерфейс и две его реализации
- Рассмотрим класс, использующий одну из реализаций этого интерфейса

Рассмотрим пример

LUXOFT Training Center

```
public class ImportantClass {
    ISoundProvider provider;

    public ImportantClass() {
        this.provider = new FileSoundProvider();
    }

    void doReallyImportantStuff() {
        this.provider.Analyze();
    }
}
```

Dependency Injection (DI)

LUXOFT Training Center

- Несмотря на то, что данный класс использует интерфейс, смысла в этом нет, поскольку конкретная реализация задается в этом же классе
- Модифицируем код этого класса для того, чтобы он получал реализацию интерфейса “извне”
- Для этого достаточно немного изменить конструктор

Dependency Injection (DI)

LUXOFT Training Center

```
public class ImportantClass {
    ISoundProvider provider;

    public ImportantClass(ISoundProvider provider) {
        this.provider = provider;
    }

    void doReallyImportantStuff() {
        this.provider.Analyze();
    }
}
```

Проблемы покрытия legacy-кода

- После того, как все связи с внешним миром у класса или функции оборваны - мы ее полностью контролируем и можем полностью проверить ее работу
- Для этого может понадобиться создать еще несколько заглушек, которые будут выдавать нужные данные для конкретных тестов

Проблемы покрытия legacy-кода

- В итоге получается, что надо написать достаточно много дополнительного кода только для того, чтобы написать первый тест
- Однако для второго теста, такого кода потребуется уже меньше
- «Десятый» тест уже пишется спокойно, с использованием ранее написанного кода

Вопросы ?

Разработка через тестирование

IDyachenko@luxoft.com

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

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