Министерство науки и высшего образования Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании

Отчет

Лабораторная работа по курсу новых технологий в программировании

«Юнит-тестирование»

Выполнил:

студент гр. 587-1

\_\_\_\_\_\_\_\_ Шаламов А.О.

"\_\_" \_\_\_\_\_\_\_\_\_ 2021 г.

Проверил:

Доцент кафедры КСУП:

\_\_\_\_\_\_\_\_ Горяинов А.Е.

"\_\_" \_\_\_\_\_\_\_\_\_ 2021 г.

Томск 2021 г.

2

**Содержание** Введение.......................................................................................................................3

Описание и выполнение задания...............................................................................3

Выводы.........................................................................................................................9

3

**Введение**

Целью данной лабораторной работы является изучение организации тестирования в разработке ПО и получение навыков в написании юнит-тестов.

**1** **Описание** **и** **выполнение** **задания**

Юнит-тестирование – тестирование минимальных модулей архитектуры, изолированных друг от друга. В ООП таковыми являются классы. Юнит-тесты составляются обычно на основе разработанных алгоритмов, а не на основе технического задания.

В рамках данной лабораторной работы требуется создать проект юнит-тестов, рассчитать цикломатическую сложность классов бизнес-логики и покрыть классы юнит-тестами.

UML-диаграмма классов с указанными цикломатическими сложностями выглядит следующим образом.

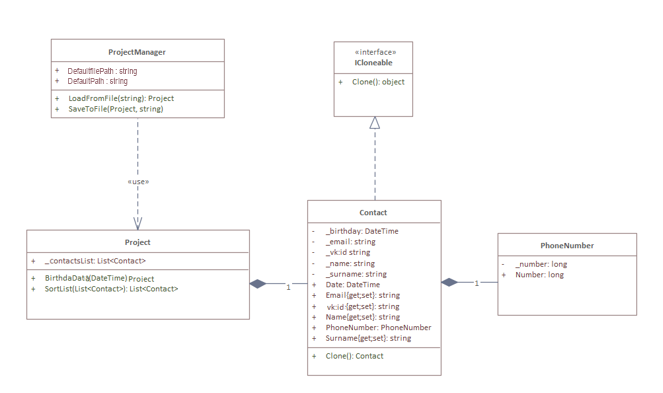


Рисунок 1 UML-диаграмма классов бизнес-логики

4

В качестве примера юнит-теста рассмотрим класс PhoneNumberTest.cs. Класс теста должен иметь атрибут [TestFixture].

[TestFixture]

class PhoneNumberTests

Перед непосредственным началом тестов выполняются методы, помеченные атрибутом SetUp. В данном случае инициализируется объект

PhoneNumber в поле класса.

[SetUp]

public void InitNumber()

{

\_number = new PhoneNumber();

}

Сами тестовые методы должны быть помечены атрибутами Test или

TestCase. Позитивный тест геттера свойства Number имеет один тестовый случай и поэтому помечен атрибутом Test. Цель: установить значение свойства, считать его и убедиться, что оно соответствует ожидаемому. Сам тест выглядит следующим образом

[Test]

public void Name\_Get\_CorrectValue()

{

//setup

var expected = 79999999999;

//act

\_number.Number = expected;

var actual = \_number.Number;

//assert

Assert.AreEqual(expected, actual, "Геттер Number возвращает неправильное имя");}

5

Методы класса Assert проверяют результат теста и используются для обозначения успешного или неудачного прохождения тестового случая. В данном случае использован метод AreEqual, сравнивающий два объекта.

В случае присвоения некорректного значения номера телефона будет

брошено исключение ArgumentException, и тест будет считаться пройденным только в этом случае.

[Test]

public void Number\_Set\_LongNumber()

{

var wrongNumber = 7777777777777;

var message = "Номер длинее 11";

Assert.Throws<ArgumentException>(

() => { \_number.Number = wrongNumber; },

message);

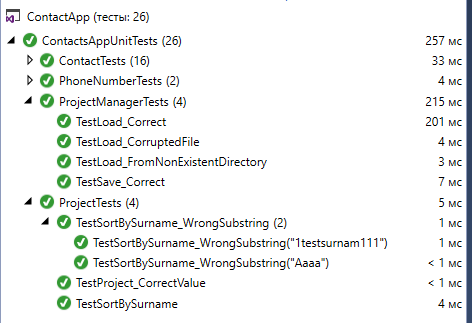
}

6

Данный тест будет считаться успешным, если переданное в метод

класса Assert булево значение будет истинным. Конкретно здесь проверяется правильность работы метода проверки номера телефона на корректность.

Всего написан 31 юнит-тест. Время работы каждого из них приведено ниже.



Все тесты пройдены успешно. Общее время составило 257 мс.

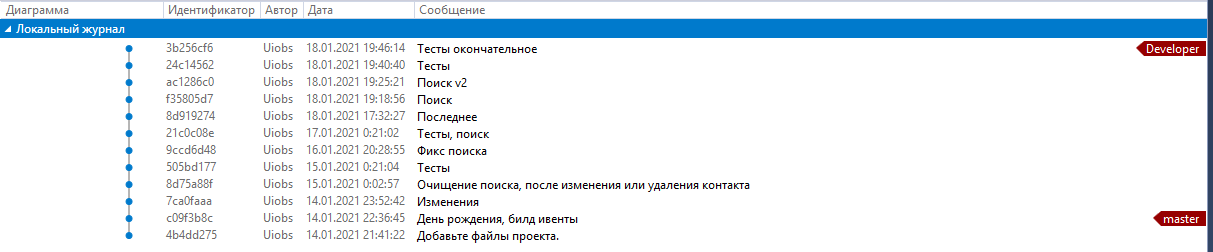


Рисунок 4 История фиксаций