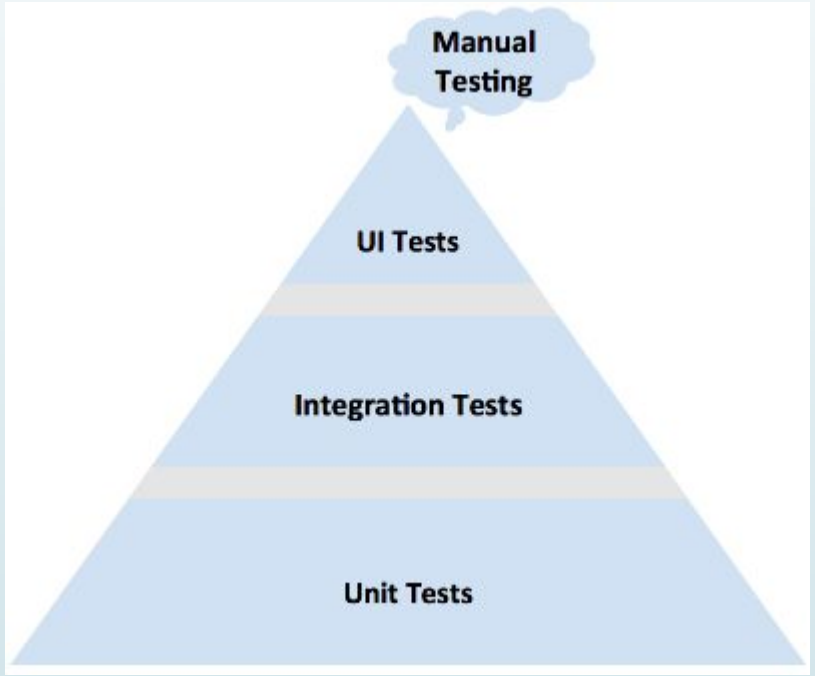
Unit testing

Пирамида тестирования



- 70-80% юнит-тестов
- 10% интеграционных тестов
- 5% системных тестов
- 5% GUI тестов.

Unit testing

- **Unit testing** — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Применение

- Быстрая проверка на ошибки в коде
- Спецификация приложения

- **Система надежно протестирована**
- **Система рассказывает о себе путем тестов**

Когда применять

- Простой код без зависимостей
- Сложный код с большим количеством зависимостей
- Сложный код без зависимостей
- Не очень сложный код с зависимостями

Тесты должны быть:

- Достоверными
- Не зависеть от окружения, на котором они выполняются
- Легко поддерживаться
- Легко читаться и быть простыми для понимания
- Соблюдать единую конвенцию именования
- Запускаться регулярно в автоматическом режиме

Фрейморки тестирования

- **MS Test**: фреймворк юнит-тестирования от компании Microsoft, который по умолчанию включен в Visual Studio
- **NUnit**: портированный фреймворк с JUnit для платформы .NET
- **xUnit.net**: фреймворк тестирования от создателей NUnit для платформы .NET

Правила именования тестов

- Выберите способ именования проектов с тестами <PROJECT_NAME>.Core.**Tests**
- Используйте такой же способ именования для тестовых классов
- Выберите «говорящий» способ именования методов тестирующих классов **[Тестируемый метод]_[Сценарий]_[Ожидаемое поведение]** *Sum_2Plus5_7Returned*

Правила написания unit-тестов

- Выберите логическое расположение тестов в вашей VCS
- Придерживайтесь единого стиля написания тела теста
- Тестирование одной вещи за один раз
- Борьба с зависимостями

Arrange-Act-Assert (AAA)

- **Arrange**: подготовка среды, в которой выполняется код
- **Act**: тестирование кода (обычно представляет одну строку кода)
- **Assert**: убеждаемся, что результат теста именно тот, что мы и ожидали

```
class CalculatorTests
{
        public void Sum_2Plus5_7Returned()
        {
                var calc = new Calculator(); // arrange
                var res = calc.Sum(2,5); // act
                Assert.AreEqual(7, res); // assert
        }
}
```

```
class CalculatorTests
{
        public void Sum_2Plus5_7Returned()
        {
                Assert.AreEqual(7, new Calculator().sum(2,5));
        }
}
```

Test Double (дублер)

- Dummy
- Fake
- Stubs
- Mocks

```
public class Order
{
    public string ProductName { get; private set; }
    public int Quantity { get; private set; }
    public bool IsFilled { get; private set; }

    public Order(string productName, int quantity)
    {
        ProductName = productName;
        Quantity = quantity;
    }

    public void Fill(IWarehouse warehouse)
    {
        if (warehouse.HasInventory(ProductName, Quantity))
        {
            warehouse.Remove(ProductName, Quantity);
            IsFilled = true;
        }
    }
}

public class Warehouse
{
    private DataAccess db;

    public Warehouse()
    {
        db = new DataAccess();
    }

    public virtual bool HasInventory(string productName, int quantity)
    {
        return db.HasInventory(productName, quantity);
    }

    public virtual void Remove(string productName, int quantity)
    {
        db.Remove(productName, quantity);
    }
}
```

Stub

```
[Test]
public void TestFillingOrderWithRhinoStub()
{
    Order order = new Order(Talisker, 50);
    var stubUserRepository = MockRepository.GenerateStub<Warehouse>();

    stubUserRepository.Stub(x => x.HasInventory(Talisker, 50)).Return(true);
    stubUserRepository.Stub(x => x.Remove(Talisker, 50));

    order.Fill(stubUserRepository);
    Assert.IsTrue(order.IsFilled);
}
```

Mock

```
[Test]
public void TestFillingOrderWithRhino()
{
    Order order = new Order(Talisker, 50);
    var mockUserRepository = MockRepository.GenerateMock<Warehouse>();

    mockUserRepository.Expect(x => x.HasInventory(Talisker, 50)).Return(true);
    mockUserRepository.Expect(x => x.Remove(Talisker, 50));
    mockUserRepository.Replay();

    order.Fill(mockUserRepository);
    Assert.IsTrue(order.IsFilled);
    mockUserRepository.VerifyAllExpectations();
}
```

The end