УДК 004.4, 004.05

Василевський В. О.
Романюк О. В.

ОСОБЛИВОСТІ НАПИСАННЯ ЮНІТ-ТЕСТІВ

Вінницький національний технічний університет

Анотація

Проаналізовано основні практики написання юніт-тестів. Проведено аналіз характерних особливостей застосування модульного тестування.

Ключові слова: тестування, юніт-тестування, TDD, юніт-тест, arrange, act, asset.

Abstract

Most common practices of writing unit-tests were analyzed. Distinctive features of unit-testing using were analyzed.

Keywords: testing, unit-testing, TDD, modular test, arrange, act, asset.

Вступ

Процес тестування є одним з головних етапів усієї розробки програмних продуктів. Сьогодні звичною практикою являється проводити декілька видів тестування під час розробки. Часто використовується інтеграційне тестування одночасно з автоматичним для більшої впевненості в коректності роботи програмного та апаратного забезпечення. Хоча це використовує більше ресурсів, доведено, що більше половини проектів завершуються з перевитратами, що свідчить про доцільність використання кількох методів тестування на етапі розробки [1].

Інтеграційне тестування – це процес, коли продукт безпосередньо перевіряється людиною функціонально та візуально на правильність виконання. Серед автоматичних методів варто виділити модульне тестування (юніт-тестування), реалізацію якого можна зустріти в розробках будь-якого розміру та формату. Використання юніт-тестування дозволяє не лише виявити помилки ПЗ на ранній стадії, а й в подальшому значно прискорити процес регресійного тестування при внесенні змін в існуючий код, оскільки великі обсяги програмного коду тестуються автоматично [2].

Проте потрібно вміти обирати правильний набір та комбінацію технік тестування для програмного забезпечення, адже кожен підхід має свої недоліки та слабкі сторони, на які потрібно зважати [3].

Модульне тестування в проекті реалізується у вигляді коду невеликих юніт-тестів, які пишуть розробники програмного забезпечення. Цей вихідний код має ряд специфічних особливостей проектування та написання.

Мета дослідження – визначення основних особливостей та практик написання модульних тестів.

Об’єктом дослідження є процес написання юніт тестів.

Предмет дослідження – характерні особливості написання та використання модульних тестів.

Результати дослідження

Кожен проект намагається зробити процес написання юніт-тестів стандартизованим. В такому випадку кожна частина окремого тесту буде краще читатись і сам тест буде більш доступним для розуміння і за необхідності виправлення будь-якою людиною, що знайома із загальноприйнятою технікою написання модульних тестів.

Сьогодні техніка AAA (Arrange-Act-Assert), або ж Організувати-Діяти-Стверджувати, стала майже стандартом для усієї індустрії. В її основі лежить ідея розділити метод тесту на три секції, кожна з яких відповідальна лише за частину тесту, за якою й була названа (рисунок 1.4).

Секція Організації (Arrange) містить в собі лише код, необхідний для підготовки до виконання тесту. Тут створюються об’єкти класів, об’єкти-заглушки, якщо вони необхідні, та визначається очікуваний результат. Згодом у секції Діяти (Act) виконується метод, що тестується. В кінці, в зоні Стверджувати (Assert), відбувається перевірка чи виконана дія відповідає очікуваному результату [4].

При слідуванні цьому шаблону написання тестів, їх код стає високо структурованим та легким для розуміння. Саме тому багато бібліотек юніт-тестування надають такий програмний інтерфейс, який дозволив би реалізацію такого стандарту.

```
[Test]
0 references
public static void ContainsInt()
{
    // arrange
    var list = new List<int>(5);
    var expected = 3;

    // act
    list.Add(expected);

    // assert
    Assert.Contains(expected, list);
}
```

Рисунок 1.4 – Приклад використання техніки (Arrange-Act-Assert)

Великі бізнес-додатки мають велику кількість залежностей в собі. Для того, щоб разом із конкретним компонентом не тестувати залежності, які в нього є, необхідно використати фальшиві реалізації цих залежностей. Одним із розповсюджених прикладів розв'язання цієї проблеми є використання об'єктів-заглушок. З їх допомогою можна надавати компоненту неповні реалізації залежностей, що дозволить тестувати лише потрібний компонент, не зважаючи на правильність виконання залежних компонентів.

Розрізняють два типи фальшивих реалізацій: стаби (stabs) та моки (mocks).

Їх різниця в тому, що стаб нічого не перевіряє, а лише імітує заданий йому стан. А мок – це об'єкт, у якого є очікування. Наприклад, що певний метод буде викликаний певну кількість разів.

З технічної точки зору це означає, що використовуючи стаби в секції Assert ми перевіряємо стан класу, що тестується або результат виконаного методу. Використовуючи мок, перевіряється, чи відповідає поведінка класу до очікування мока.

Ще однією технікою, що дозволяє полегшити тестування та використовується при написанні юніт-тестів, є винесення логіки тесту в чисті функції. Такі функції не взаємодіють із зовнішнім середовищем, ніяк його не змінюють, а їх результат залежить лише від вхідних даних і не змінюється при повторному виконанні з однаковими вхідними даними. Використання чистих функцій дозволяє реалізувати принцип ізольованості тесту.

Висновки

Таким чином, виявлено, що існують загальні практики написання коду юніт-тестів, яких потрібно притримуватись для підтримки загального стилю реалізації модульного тестування в програмному продукті. Знаючи такі особливості, та застосовуючи їх, вихідний код модулю тестування можна зробити більш зрозумілим, коректним та логічно правильним.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Василевський В. О. Порівняльний аналіз фреймворків юніт-тестування в середовищі .Net [Електронний ресурс] // В.О.Василевський, О.В.Романюк / XLIX Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2020): тези доповідей. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/9024
2. Murphy Craig. Improving Application Quality Using Test-Driven Development / Craig Murphy. [Electronic resource]. – Mode of access http://www.Methodsandtools.coin/archive/ archive.php?id = 20.
3. Василевський В. О. Недоліки використання модульного тестування як основної технології тестування [Електронний ресурс] // В.О.Василевський, О.В.Романюк / Тези доповідей XI Міжнародної науково-технічної конференції

«Інформаційно-комп'ютерні технології – 2020 (ІКТ-2020)», м. Житомир, 09 - 11 квітня 2020 р. – Житомир: Житомирська політехніка, 2020. – С.
4. Ануфриев Д. Unit тесты на практике [Електронний ресурс] / Дмитрий Ануфриев // Хабр. – 2013. – Режим доступу до ресурсу: https://habr.com/ru/post/191986/.

Василевський Володимир Олегович, студент групи 1ПІ-16б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: vovavasilevskyi@gmail.com

Романюк Оксана Володимирівна, к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: romaniukoksanav@gmail.com

Vasylevskyi Volodymyr, student of group 1PI-16b, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, email: vovavasilevskyi@gmail.com

Oksana Romaniuk, Candidate of Technical Sciences, Associate Professor of the Software Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: romaniukoksanav@gmail.com