УДК 004.415.3

ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ ТА МОКУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ПЛАТФОРМІ .NET

Богун В. М.
Науковий керівник – к.т.н., доц. Голян В. В.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: vladyslav.bohun@nure.ua

This work is devoted to the research of methods of creating mechanisms of testing and mocking in the context of software development on the .NET platform. It includes an analysis of the most common unit testing frameworks – NUnit, xUnit, and MSTest – focusing on their key features, attributes, and usage characteristics. The purpose of the work is to identify the optimal testing tools according to the specific conditions of the project and the environment, as well as to analyze the effectiveness of frameworks for the most productive code mocking. The main focus is on the comparative analysis of frameworks in order to find a comprehensive and effective method for improving the quality and reliability of software products.

У процесі дослідження предметної області ми можемо встановити деякі проблеми із підходом до тестування. Вибір фреймворку та підходу до побудови механізмів тестування може бути обумовлений технічними вимогами проекту.

Тип та характер проекту можуть вимагати різних підходів до тестування. Наприклад, для одиничного тестування компонентів може підходити один фреймворк, тоді як для інтеграційних тестів інший. Ця проблема актуальна через різницю у специфікаціях проектів.

Фреймворк з активною спільнотою і підтримкою може бути надійнішим вибором. Спільнота забезпечує швидше виявлення та виправлення помилок, а також забезпечує доступ до ресурсів та допомоги.

Можливість інтегрувати фреймворк з іншими інструментами, такими як системи автоматизованої збірки (CI/CD), також важлива для розробників.

Розробники можуть не розуміти всі різновиди тестування, такі як тестування одиниць, інтеграційне тестування, прийомне тестування та інше.

Саме тому наше дослідження методів створення механізмів тестування та мокування(Mocking) програмного забезпечення, на платформі .Net – є актуальним, адже вирішить вище зазначені труднощі розробників, та закриє багато питань з вибору підходу та стратегії до тестування ПЗ [1].

У ході даної роботи необхідно було дослідити предметну область побудови тестових механізмів за допомогою фреймворків та підходів до тестування на платформі .Net.

Провести порівняльний аналіз та аналіз призначення кожного фреймворку та підходу до тестування.

Провести порівняльні експерименти за певними метриками для визначення більш оптимальної бібліотеки у різних сценаріях тестування програмного забезпечення.

Ця робота спрямована на створення дорожньої карти для розробників щодо знаходження найефективнішого та комплексного підходу до тестування застосунків.

Реалізація наукового дослідження складається з наступних етапів:

– аналіз предметної області;
– аналіз методів тестування;
– розглядання особливостей NUnit, xUnit, MSTest;
– порівняння атрибутів методів побудови механізмів тестування;
– аналіз мокування;
– формування висновків.

Було розглянуто декілька бібліотек тестування, вони представляють собою комплексні методи тестування. Їх порівняння проводиться емпіричними експериментами які містять певні метрики які відповідають різним структурним аспектам порівняння, ось приклади із описом метрик за якими було порівняно бібліотеки:

– час виконання тестів: вимірювання часу, який потрібний для виконання одного або кількох наборів тестів в кожному з фреймворків;

– час запуску тестів: вимірювання часу, який потрібен для запуску фреймворка та завантаження тестових сценаріїв;

– обсяг пам'яті, використовуваний фреймворком: визначення кількості пам'яті, яку використовує фреймворк для запуску тестів та їх виконання;

– стабільність тестового середовища: вимірювання кількості помилок або збоїв, які виникають при запуску тестів у кожному з фреймворків;

– масштабованість: оцінка можливості фреймворка ефективно працювати з великим обсягом тестів та різними конфігураціями проекту;

– підтримка паралельного виконання тестів: визначення швидкості та ефективності паралельного виконання тестів у кожному з фреймворків;

– кількість доступних ресурсів та інструментів: аналіз кількості інструментів та ресурсів, які доступні для автоматизації, звітності та аналізу результатів тестування.

За результатами можна зазначити, що NUnit надає гнучкість через різноманітні атрибути, які можуть бути використані для деталізованого управління тестовими сценаріями. Наприклад, атрибути [TestCase] та [Theory] дозволяють легко впроваджувати параметризовані тести.

Також NUnit підтримує паралельне виконання тестів, що дозволяє оптимізувати час виконання тестів і підвищує гнучкість в управлінні ресурсами. xUnit використовує конструктори класів для ініціалізації та

інтерфейс IDisposable для очищення, що дозволяє більшу гнучкість в управлінні станом тестів. Це сприяє написанню чистих та ізольованих тестових сценаріїв, також він використовує [Fact] для не параметризованих тестів і [Theory] для параметризованих, що дозволяє гнучко керувати даними тестів.

MSTest, будучи інтегрованим рішенням в Visual Studio, часто вважається менш гнучким порівняно з NUnit та xUnit, оскільки він менш орієнтований на спільноту та має меншу підтримку з боку сторонніх розробників інструментів.

У висновку NUnit та xUnit пропонують більшу гнучкість у контексті мокування завдяки своїм особливим підходам до ініціалізації, управління станом тесту, та більшому вибору атрибутів та підтримці з боку спільноти.

Це робить їх більш підходящими для складних сценаріїв тестування, де важливо глибоке управління тестовими випадками та ізоляція станів. MSTest, навпаки, краще підходить для інтегрованих сценаріїв у середовищі Visual Studio, де може бути достатньо його базового функціоналу.

NUnit відзначається своєю гнучкістю та розширеними можливостями, що робить його ідеальним для складних тестових сценаріїв. Підтримка параметризації та можливість визначення порядку виконання тестів дозволяє розробникам ефективно адаптувати NUnit до своїх потреб [2].

xUnit пропонує високу ступінь ізоляції тестів та чистоту коду, що робить його відмінним вибором для проектів, де необхідна ізоляція та незалежність тестів. Ефективність у параметризації та унікальний підхід до управління тестовим середовищем роблять xUnit привабливим для багатьох розробників. MSTest, як інтегрований компонент Visual Studio, забезпечує легкість використання та зручність для користувачів Visual Studio. Цей фреймворк є гарним вибором для проектів, що вже інтегровані з екосистемою Microsoft та потребують тісної інтеграції з інструментами Visual Studio. Також він завдяки більш проробленим сценаріям тестування – більше підходить до процесу мокування.

Загалом, щоб остаточно визначити найефективніший метод – потрібно додатково провести автоматизовані тести через локальні сітки та більш детальне крос браузерне тестування.

Список використаних джерел:

1. Редакційний програмний комплекс клієнт-серверної архітектури з «товстими клієнтами», Сокорчук І, ХНУРЕ, 2018.: https://openarchive.nure.ua/entities/publication/427a0cb4-35ba-45b3-9d3c-8e79de13669a (дата звернення: 01.02.2024).

2. Steve Smith. Unit testing C# in .NET using dotnet test and xUnit. 2024: https://learn.microsoft.com/dotnet/core/testing/unit-testing-with-dotnet-test (дата звернення: 01.02.2024).