УДК 004.415.538

СУЧАСНІ ПІДХОДИ ДО ТЕСТУВАННЯ ТА ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Ступнікова Є.В.
e-mail: yelyzaveta.stupnikova@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

This paper examines modern approaches to software testing and quality assurance, analyzing the use of Test-Driven Development (TDD) and Behavior-Driven Development (BDD) methodologies in software application development. The study demonstrates how these methodologies can operate within a unified development-testing process, mimicking cognitive processes and functioning effectively in complex digital ecosystems. The paper suggests that successful quality assurance strategies must depend not solely on the quantity of tests performed but rather on alignment with business objectives and adaptability of a software process development. This approach fosters a quality-oriented culture.

У сучасному світі розробки програмного забезпечення якість веб-додатків стає критичним фактором успіху проектів, що зумовлює зростання важливості ефективних підходів до тестування. Метою дослідження є проведення аналізу автоматизованого тестування в контексті парадигм Test-Driven Development (TDD) та Behavior-Driven Development (BDD), визначивши оптимальні сценарії їх застосування для різних типів веб-проектів.

Сценарії BDD пишуться простою зрозумілою мовою з використанням синтаксису Given-When-Then (Дано-Коли-Тоді). Наступним кроком є створення скелету для автоматизованих тестів за допомогою обраного BDD-інструменту [1]. Завдяки опису природною мовою (англійською) сценарії користувача BDD значно покращує комунікацію для всіх учасників процесу розробки.

Сучасні підходи до тестування додатків можна розділити на дві основні парадигми: TDD, що акцентує увагу на розробці через тестування функціональності, та BDD, який зосереджується на поведінкових аспектах програмного забезпечення з точки зору користувача і забезпечує зв’язок між усіма аспектами продукту (вимогами, розробкою та тестуванням). BDD фактично імітує дії реального користувача. Інструменти автоматизованого тестування для TDD (такі як Jest, Mocha і Jasmine) орієнтовані на технічну реалізацію компонентів програми та модульне тестування, тоді як BDD-інструменти (Cucumber, Specflow, Cypress, JBehave) більше зосереджені на бізнес-логіці. Розробник під час створення тестів за TDD підходом фактично займається проектуванням модуля, заздалегідь обмірковуючи призначення та функціональні обов'язки цього

модуля. У такий спосіб він визначає архітектуру та відповідальність програмного компонента ще до початку його безпосередньої розробки. Оптимальна архітектура бази даних, у свою чергу, забезпечує легку підтримку та розвиток системи [2].

Проблема полягає в неправильному сприйнятті цих методологій як взаємовиключних альтернатив, а не як взаємодоповнюючих інструментів. Складні системи визначаються великою розмірністю, містять значну кількість елементів та зв’язків між ними [3]. Працівники які фокусуються виключно на TDD, часто досягають технічної досконалості коду, але ризикують створити продукт, що не відповідає реальним потребам користувачів. Натомість, прихильники чистого BDD можуть забезпечити відповідність бізнес-вимогам, але страждати від нижчої технічної якості.

З однієї сторони, тестування кінцевої поведінки користувача є надважливим. Але чи можливо ефективно перевірити поведінку системи без ґрунтовного тестування її функціональних компонентів? Тож BDD і TDD ні в якому разі не мають розглядатися як конкуруючі чи взаємовиключні, а навпаки, як взаємодоповнюючі підходи. BDD фокусує увагу на критичних бізнес-функціях, а TDD дозволяє розставити пріоритети на рівні компонентів, що разом скорочує час тестування. Унікальність цього симбіозу полягає в тому, що він дозволяє одночасно вирішувати два фундаментальні питання розробки: "Чи правильно ми будуємо систему?" та "Чи правильну систему ми будуємо?"

У результаті можна зробити висновок, що ефективне тестування веб-додатків потребує інтеграції підходів TDD та BDD, оскільки кожен з них фокусується на важливих аспектах розробки: технічній якості та бізнес-логіці. TDD забезпечує стабільність коду, тоді як BDD допомагає забезпечити відповідність продукту вимогам користувача і бізнесу.

Список використаних джерел:

1. Schults C. What are TDD and BDD and how to use them?!. The Pair Programming Blog. URL: https://blog.coscreen.co/blog/tdd-and-bdd/ (дата звернення: 02.03.2025).

2. Чуприна А. С. Дослідження моделей та архітектурних рішень в базах даних Web-технологій / А. С. Чуприна, А. С. Штельма // Поліграфічні, мультимедійні та web-технології : тези доп. IX Міжнар. наук.-техн. конф., 14-18 травня 2024 р. – Т. 1. – Харків: ТОВ «Друкарня Мадрид», 2024. – С. 193-194.

3. Пономаренко О. Програмна платформа для оцінювання ефективності агрегації структурної моделі складних систем / О. Пономаренко, В. Горбачов // Сучасний стан наукових досліджень та технологій в промисловості. – 2023. – № 3(25). – С. 79–87.