УДК 004.4, 004.05

Василевський В. О., студент,
Романюк О. В. канд. техн. наук, доцент
Вінницький національний технічний університет

НЕДОЛІКИ ВИКОРИСТАННЯ МОДУЛЬНОГО ТЕСТУВАННЯ ЯК ОСНОВНОЇ ТЕХНОЛОГІЇ ТЕСТУВАННЯ

Незалежно від розміру, цілі та складності, кожен програмний проєкт підлягає тестуванню. Модульне тестування, або юніт-тестування – процес програмування, який дозволяє ізольовано перевірити кожен окремий модуль вихідного коду. Окремий тест пишеться для кожної нетривіальної функції або методу. Таке рішення дозволяє відносно швидко перевірити чи не призвела чергова зміна коду до регресії, тобто появі нових помилок у відтестованих модулях додатку, а також дозволяє виявляти і, як наслідок, усувати такі помилки [1].

Модульне тестування має ряд переваг перед іншими методами тестування, через що сьогодні більшість розробників прагнуть покрити юніт-тестами якомога більшу частину модулів своїх програмних проєктів. Проте не на кожному проєкті та не для кожного модуля можливе використання юніт-тестування. Серед основних причин відмовитись від модульного тестування в процесі розробки є:

1. *Необхідність тестування складного коду.* Багато задач в програмуванні мають комбінаторний характер, тобто вихідний результат може набувати різних значень. У випадку з булевою змінною це лише два варіанти, але за необхідності опрацювати дерево, або іншу складну струк-туру даних, варіантів кінцевого результату може бути незліченно багато. Кожен такий варіант необхідно опрацювати та перевірити за допомогою тесту, кожен з яких займає мінімум 3-5 стрічок коду. Виходячи з того, що на практиці неможливо трасувати усі можливі шляхи виконання програми, модульне тестування не дозволяє відловити усі помил-ки, окрім найпростіших випадків.

2. Очікуваний результат є лише приблизним. В модульному тестуванні кожен окремий тест повинен повертати твердження про відповідність очікуваного результату до отриманого. Це стає неможливим, коли не можна завчасно знати який результат є очікуваним за певних вхідних даних, як наприклад у математичному моделюванні, або за використання машинного навчання та добування даних [2].

3. Використання коду, що працює зі системою. Код, що потребує взаємодії із апаратними портами, таймерами або іншими нестабільними частинами системи дуже складно перевірити в ізольованому стані. В цьому випадку виникає необхідність переходити до більш абстрактних засобів та імітувати використання апаратних та елементів системи, хоча й це не дає можливості повністю повторити часто невизначений стан системи.

4. Помилки в інтеграції та швидкості виконання. Модульне тестування не підходить для перевірки системи в цілому, оскільки відбувається тест кожного модулю окремо. Це означає, що помилки інтеграції, системного рівня, функцій, які виконуються в різних модулях, не будуть визначені. Окрім цього, юніт-тести не придатні для тестування швид-кості виконання та загальної оптимізації системи. Для цього модульне тестування застосовують із іншими методиками [3].

5. При загально-низькій культурі програмування. Для отримання максимальної користі від модульного тестування необхідно чітко слідкувати за технологією тестування протягом усього життєвого циклу розробки. Необхідно зберігати записи про всі проведенні тести та про всі зміни вихідного коду. В результаті, якщо більш пізня версія прог-рами не проходить конкретний тест, можна звернутись до записів і перевірити на якій версії цей тест виконувався і які зміни були внесені, тобто виявити зміни, що привели до помилки і легко усунути її. Якщо ігнорувати ці вимоги, це призведе до накопичення інформації, яку вже неможливо перевірити, а тому й виправити. Кожен учасник, задіяний в розробці програмного забезпечення повинен нести відповідальність за загальну технологію тестування та її підтримку.

Проведене дослідження дозволить краще оцінити доцільність використання модульного тестування як єдиної технології тестування проєк-ту.

Список використаних джерел

1. Василевський В. О., Романюк О. В. Порівняльний аналіз фреймворків юніт-тестування в середовищі .Net [Електронний ресурс]// 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. Юнит-тестирование. [Електронний ресурс] // Режим доступу: https://habr.com/ru/post/169381

3. Unit-testing. [Electronic resource] // Режим доступу: https://en.wikipedia.org/wiki/Unit_testing