УДК 004.415.3 DOI: https://doi.org/10.30837/IYF.IIS.2024.487

СУЧАСНІ ПІДХОДИ ДО ТЕСТУВАННЯ ТА ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Дем'яненко М. С.
Науковий керівник – к.т.н., доц. Чуприна А. С.
Харківський національний університет радіоелектроніки, каф. МЕЕПП
м. Харків, Україна
e-mail: maryna.demianenko.cpe@nure.ua

In the contemporary tech environment, the critical role of software testing cannot be overstated. Its core purpose is to evaluate and enhance software quality, ensuring functionality, security, and ultimately user satisfaction, which in turn builds customer trust. The emergence of Artificial Intelligence (AI) in software development heralds a new era, significantly boosting the efficiency and effectiveness of these processes. The integration of AI innovates the various phases of software testing, markedly improving the testing methodology and leading to the creation of superior software products.

У сучасному технологічному середовищі важливість тестування програмного забезпечення та забезпечення якості неможливо переоцінити. Тестування програмного забезпечення відіграє ключову роль у розробці додатків. Цей процес гарантує, що програмні системи функціонують належним чином, є безпечними, задовольняють потреби зацікавлених сторін і, зрештою, приносять користь кінцевим користувачам. Виявляючи дефекти на ранніх стадіях процесу розробки, тестування програмного забезпечення не тільки покращує якість продукту, але й сприяє довірі клієнтів і задоволенню, надаючи продукт, який був ретельно оцінений. Крім того, тестування програмного забезпечення відіграє важливу роль у виявленні вразливостей системи безпеки, що має першочергове значення, враховуючи зростаючу поширеність кіберзагроз [1].

Проблеми, з якими стикається сучасне тестування програмного забезпечення, і відповідна потреба в інноваціях є багатогранними. Технологічний ландшафт, що постійно розвивається, вимагає постійної адаптації та вдосконалення методологій тестування, щоб йти в ногу зі складнощами сучасних програмних додатків. Середовища гнучкого тестування та DevOps запровадили більш спільну та швидшу стратегію тестування, що підкреслює потребу в інноваціях у методах тестування для досягнення ефективності та результативності.

Роль штучного інтелекту (Artificial intelligence) у розробці програмного забезпечення стає дедалі помітнішою, пропонуючи нову парадигму для підвищення ефективності та результативності процесу розробки. Можливості штучного інтелекту поширюються на створення коду, підтримку розробників і аналіз складних баз коду для виявлення критичних аспектів, які потребують уваги.

У свою чергу, це явище не лише спрощує процес розробки, але й відкриває нові можливості для автоматизації та оптимізації різних етапів тестування програмного забезпечення. Використовуючи штучний інтелект, команди тестувальників можуть автоматизувати повторювані завдання, покращити відстеження помилок і дефектів та запровадити безперервне тестування протягом життєвого циклу розробки. Тому інтеграція штучного інтелекту не тільки розширює можливості команд розробників, але також може значно сприяти покращенню процесу тестування задля створення програмних продуктів вищої якості [2].

Прикладом впровадження інновації може стати автоматична модульна система у вигляді веб-додатку, що використовується командами тестувальників на початкових або ранніх стадіях існування проекту, коли процес тестування все ще має ознаки невизначеності.

Для етапу аналізу вимог та планування тестування буде використовуватись модуль збору інформації про проект та середовище. Дані про проект створюються за допомогою збору інформації від користувача, у вигляді “налаштування проекту”, фактично – опитування, яке перетворюється у набір критеріїв, що будуть описувати майбутній інструмент або фреймворк для автоматизованого тестування, таких як: необхідні браузери, мови програмування для написання автоматичних тестів, необхідні емулятори та/або можливість використання вбудованих пристроїв для тестування, наявність вичерпної документації, відкритість коду, популярність. Зібрані дані оброблятимуться методом, що являтиме собою вирішення задачі оптимізації за допомогою моделі лінійної адитивної згортки з ваговими коефіцієнтами. Для цього будуть використані заздалегідь сформовані шкали для оцінки кожного критерію:

$$Z^* = max \sum_{j=1}^{n} a_j \beta_j a_{ij}, \quad (1)$$

де β_j – вагові коефіцієнти, що відображають відносний внесок окремих критеріїв до загального критерію, α_j – нормуючі множники, α_{ij} – значення критерію

Для покращення точності оцінки проекту у наступних ітераціях доного застосунку, дані про вибір кожного користувача зберігаються до внутрішньої бази даних та будуть використані для навчання регресійної моделі.

Для виконання всебічного аналізу файлів проекту наданих у формі архіву за допомогою OpenAI API створюється zip-файл для завантаження користувачем, який містить запропоновану структуру для автоматизованого тестування проекту.

Ідентифікація мови та виявлення фреймворків: використовуючи методи обробки природної мови (NLP), програма визначає основну мову

програмування та фреймворки, які використовуються в проекті, а також результат враховує критерії оптимізаційної задачі [4]. Цей крок гарантує, що подальші рекомендації адаптовані до конкретного технологічного стеку, який використовується в проекті.

Ідентифікація критичних функціональних можливостей: використовуючи механізми синтаксичного аналізу, що надає OpenAI API, програма визначає критичні функціональні можливості в рамках проекту. Для знайдених функціональних можливостей, за допомогою функції генерації коду, створюються тести, з якими команда може почати роботу [5]. Таким чином запроваджується забезпечення охоплення тестами.

Структура проекту автоматизації тестування: програма пропонує оптимальну структуру проекту для автоматизованого тестування, що сформована та упакована до zip-архіву. Також, для полегшення інтеграції генерується код для запуску створених тестів у CI/CD середовищі.

Запропонована модель з інтеграцією ШІ значно покращує ефективність впровадження автоматизації в проекти, дозволяє прийняти обгрунтоване рішення щодо вибору методів автоматизації, оптимальних для конкретного проекту, прискорюючи процес розробки та підвищуючи якість кінцевих програмних рішень через автоматизоване тестування.

Список використаних джерел:

1. Why Do We Need Software Quality Assurance and Testing?. Softvelopers. URL: https://www.softvelopers.com/blog/importance-software-testing-quality-assurance (дата звернення: 21.03.2024).

2. Testim. What Is the Software Testing Life Cycle? A Complete Guide. AI-driven E2E automation with code-like flexibility for your most resilient tests. URL: https://www.testim.io/blog/software-testing-life-cycle/ (дата звернення: 21.03.2024).

3. Радіоелектроніка та молодь у XXI столітті. Т. 6 : Конференція "Інформаційні інтелектуальні системи" : матеріали 23-го Міжнар. молодіж. форуму, 16–18 квіт. 2019 р. / М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків, 2019. 306 с.

4. AI-Assisted Test Automation: Revolutionizing Software Testing Like Never Before. Opkey: #1 Test Automation Platform for Enterprise Continuous Testing. URL: https://www.opkey.com/blog/ai-assisted-test-automation-revolutionizing-software-testing-like-never-before (дата звернення: 21.03.2024).

5. Methods of Semantic Structured Search / Pohuliaiev, Y., Smelyakov, K., Chupryna, A., Ruban, I. 2022 IEEE 9th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), м. Kharkiv, Ukraine, 10–12 жовт. 2022 р. 2022. URL: https://doi.org/10.1109/picst57299.2022.10238538 (дата звернення: 22.03.2024).