УДК 004.7

Тороп М.Д., здобувач,
Чижмотря О.В., ст. викладач
Державний університет «Житомирська політехніка»

АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ ВЕБЗАСТОСУНКІВ З ВИКОРИСТАННЯМ LLM-АГЕНТІВ

Тестування є важливою частиною кожного етапу розробки програмного забезпечення, але не є рідкістю, що тестування стає ресурсоємним. Традиційні підходи автоматизованого тестування вимагають значних зусиль на написання та підтримку тестів, особливо при частих змінах інтерфейсу. Поява великих мовних моделей (LLM) відкриває нові можливості для автоматизації цього процесу: від генерації тестових сценаріїв до автономного виконання тестів та інтеграції результатів. Метою даної роботи є аналіз практичних підходів до застосування LLM-агентів у тестуванні вебзастосунків та оцінки їх ефективності.

Автоматизовані фреймворки для тестування (такі як Playwright, Selenium, Cypress) потребують написання детальних скриптів із прив'язкою до конкретних CSS-селекторів або XPath-виразів. Зміна інтерфейсу призводить до масового порушення тестів, навіть якщо функціональність залишилась незмінною. Крім того, розробники схильні пропускати нетипові сценарії та граничні випадки, що знижує загальне покриття тестами.

Найбільш зрілим застосуванням LLM у тестуванні є автоматична генерація юніт-тестів на основі аналізу вихідного коду. Інструменти GitHub Copilot, Amazon Q і їм подібні здатні аналізувати функцію або клас і генерувати набір тест-кейсів [1], включаючи граничні випадки. Порівняльні дослідження показують, що LLM-генеровані тести в окремих дослідженнях здатні забезпечити помітно вище покриття коду порівняно з ручним написанням тестів, особливо у частині негативних сценаріїв та граничних випадків.

Агентний підхід до end-to-end тестування реалізується через LLM, що взаємодіє з вебзастосунком через браузер аналогічно до реального користувача. На відміну від скриптових підходів, агент орієнтується на семантику елементів інтерфейсу, а не на конкретні селектори. Це робить тести значно стійкішими до змін UI: якщо кнопка «Надіслати» змінила клас або положення на сторінці, агент у ряді випадків дозволяє зменшити залежність від жорстко заданих селекторів.

Агент отримує user story або текстовий опис функціональності та самостійно генерує послідовність дій: клікає на елементи, заповнює

форми, перевіряє стан сторінки. Результати виконання порівнюються з очікуваними, а відхилення інтерпретуються як потенційні дефекти.

LLM-агенти показують перспективні результати в окремих дослідженнях у аналізі результатів тестування. При виявленні збою агент аналізує стек викликів, логи та знімки стану застосунку, може допомогти у локалізації причини помилки [2]. Це скорочує час діагностики регресій, які в складних системах можуть займати від кількох годин до кількох днів [3].

Попри перелічені можливості, LLM-агенти мають і суттєві обмеження. З практичної точки зору, недетермінованість моделей ускладнює відтворюваність результатів, а це критична вимога для будь-якого CI/CD пайплайну, де тест має давати однакову відповідь при кожному запуску. Не менш важливим є економічний аспект: вартість API-викликів при масштабному тестуванні може виявитись невиправданою для невеликих проєктів. Окремо стоїть проблема якості: на складній бізнес-логіці моделі схильні до галюцинацій, тому згенеровані тести потребують обов'язкової верифікації розробником. Нарешті, агент із доступом до браузера та файлової системи сам по собі є потенційним вектором атаки – зокрема, indirect prompt injection через шкідливий вміст на опрацьовуваних сторінках.

Узагальнюючи, LLM-агенти суттєво змінюють підхід до автоматизації тестування вебзастосунків, пропонуючи вищий рівень абстракції та скорочення ручних зусиль. Найефективнішим є гібридний підхід: використання LLM для генерації базових тестів і граничних випадків у поєднанні з традиційними фреймворками для стабільного CI/CD. Перспективним напрямом є розробка спеціалізованих тест-орієнтованих моделей та стандартів оцінки якості LLM-генерованих тестових наборів.

Список використаних джерел:

1. Chen B. etc. Automated Unit Test Generation using Large Language Models. Proceedings of the 46th IEEE/ACM International Conference on Software Engineering. IEEE/ACM. 2024.

2. Deng Y. etc. AdbGPT: Empowering Bug Report Reproduction based on Large Language Models. Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. New York. ACM. 2023.

3. Kang S., Yoon J., Yoo S. Large Language Models are Few-Shot Testers: Exploring LLM-Based General Bug Reproduction. Proceedings of the 45th IEEE/ACM International Conference on Software Engineering. IEEE Computer Society. 2023. Pp. 2312–2323. URL: https://doi.org/10.1109/ICSE48619.2023.00194.