Олег Блистів, Володимир Карашецький*

[1] НЛТУ України, Львів, Україна, ORCID: 0009-0001-4802-1164,
blystiv.o@nltu.lviv.ua
[2] НЛТУ України, Львів, Україна, ORCID: 0009-0008-1460-2674,
karashetskyy@nltu.edu.ua

Розробка концепції інтелектуального асистента для тестування програмного забезпечення на основі мовної моделі GPT

Анотація. *Розглянуто сучасні можливості застосування великих мовних моделей, зокрема GPT, у процесі тестування програмного забезпечення. Проаналізовано функціональність чотирьох популярних інструментів, що вже використовують елементи штучного інтелекту: Testim, Mabl, Functionize та Qase. Розглянуто обмеження у їхній функціональності. Запропоновано концепцію інтелектуального асистента для тестувальників ПЗ з інтеграцією мовної моделі GPT, яка охоплює аналіз вимог, генерацію тест-кейсів, інтерпретацію результатів тестування, автоматичне створення звітів і аналіз логів помилок та має на меті підвищити ефективність тестування, зменшити вплив людського чинника та автоматизувати рутинні процеси, що є актуальним у контексті сучасної практики розробки програмного забезпечення.*

Ключові слова *– тестування програмного забезпечення, GPT, штучний інтелект, автоматизація тестування, інтелектуальний асистент, аналіз вимог, генерація тест-кейсів.*

Вступ. У сучасних умовах стрімкого розвитку програмного забезпечення, його тестування відіграє ключову роль, оскільки саме від якості ПЗ залежать його конкурентоспроможність, безпека та користувацький досвід. Разом із цим зростають вимоги до ефективності роботи тестувальників, що зумовлює необхідність у використанні інструментів, здатних оптимізувати процеси контролю якості.

Незважаючи на значну кількість доступних інструментів для тестування, більшість із них або зовсім не використовують сучасні мовні моделі, або інтегрують їх лише частково. Це створює нішу для розробки нових підходів та ін-

* науковий керівник

струментів, які б поєднували гнучкість GPT-моделей із практичними потребами тестувальників.

Метою роботи є аналіз і порівняння існуючих інструментів підтримки тестування програмного забезпечення та розробка концепції інтелектуального асистента для тестування ПЗ на основі мовної моделі GPT, що дозволить підвищити ефективність тестування шляхом часткової автоматизації рутинних задач і надання рекомендацій у режимі реального часу.

Останні досягнення в галузі штучного інтелекту, зокрема розвиток великих мовних моделей (LLM), таких як GPT, відкривають нові можливості для створення інтелектуальних асистентів. Ці моделі здатні генерувати тест-кейси, аналізувати логіку коду, інтерпретувати документацію та автоматизувати частину рутинних завдань тестувальника.

Серед сучасних інструментів із використанням штучного інтелекту (ШІ) вирізняється Testim (розроблений компанією Tricentis), основним призначенням якого є автоматизація UI-тестування. Його функціональність включає автоматичне створення тест-кейсів на основі дій користувача, стабілізацію тестів завдяки автоматичному виявленню змін у DOM, наявність зручного візуального редактора тестів, а також підтримку інтеграції з CI/CD-середовищами на зразок Jenkins чи GitHub Actions [6]. До основних переваг цього інструменту можна віднести зменшення фрагментації тестів, автоматичне оновлення після змін в інтерфейсі користувача та низький поріг входу для початківців.

Інший інструментарій – Mabl, що орієнтований на автоматизоване тестування вебзастосунків та API [1]. Інструмент підтримує автоматичне тестування з використанням ШІ-аналітики, вбудоване розпізнавання змін у UI та інші функції, які сприяють підвищенню ефективності тестування. Серед його переваг можна виокремити потужну візуалізацію тестових процесів і підтримку API-тестів.

Третій інструмент – Functionize, представляє собою ML-орієнтоване рішення для тестування вебзастосунків у середовищах CI/CD. Його функціональні можливості охоплюють використання ШІ та машинного навчання для ство-

* науковий керівник

рення, виконання та підтримки тестів, а також застосування технології Smart Object Recognition, яка забезпечує стабільність тестів навіть за наявності змін у структурі додатка. Інструмент також підтримує написання тестів звичайною англійською мовою (natural language) та пропонує розширені аналітичні звіти. До переваг можна віднести високий рівень автоматизації, зниження кількості нестабільних (flake) тестів і можливість масштабування на великі проєкти.

Окрему увагу заслуговує Qase – повнофункціональна система для керування тест-кейсами, тест-планами та звітністю, яка особливо добре підходить для команд, що працюють за методологією Agile/Scrum. Цей інструмент інтегрується з популярними сервісами, такими як Jira, GitHub, GitLab і Slack, а також має API для автоматизації тест-менеджменту. Важливою інновацією є поява ШІ-асистента, здатного генерувати тест-кейси на базі вимог, збережених в системі [4]. Додатково розроблено функціональність AIDEN, яка забезпечує конвертацію ручних тестів в автоматизовані [2]. Серед переваг Qase – інтуїтивно зрозумілий інтерфейс, можливість автоматизувати існуючі тест-кейси та потенціал до повноцінного використання ШІ для генерації тестів на основі вимог.

Проведений аналіз показав, що хоча більшість інструментів реалізують окремі функції (генерація тестів, створення звітів), жоден із них не охоплює повний цикл, включно з аналізом вимог та інтелектуальним аналізом логів. Найближчими за функціоналом є Qase та Functionize, проте навіть вони не забезпечують глибокої обробки специфікацій або складного лог-аналізу [3].

Пропонується концепція, яка передбачає створення інтелектуального інструмента для підтримки повного циклу тестування ПЗ, який інтегрує мовну модель GPT або її модифікації. Такий інструмент орієнтований на підвищення ефективності тестування завдяки автоматизації рутинних процесів і наданню рекомендацій у режимі реального часу.

Одним із ключових функціональних блоків є модуль аналізу вимог до ПЗ. Його завдання полягає у виявленні суперечностей, прогалин і неузгодженостей у специфікаціях, перевірці відповідності вимог загальновизнаним стандартам

* науковий керівник

якості (зокрема, SMART-критеріям), а також у семантичному групуванні вимог для подальшого формування відповідних тест-кейсів.

Наступний блок відповідає за автоматичну генерацію тест-кейсів на основі текстових вимог, user stories або acceptance criteria. Крім цього, система має пропонувати оптимізацію тестових наборів, зокрема шляхом виявлення надлишкових або дубльованих кейсів.

Функціональність аналізу результатів тестування охоплює як інтерпретацію результатів автоматизованих і ручних тестів, так і виявлення характерних тенденцій, зокрема повторюваних збоїв, нестабільних тестів або ділянок із низьким рівнем покриття [5].

Окремий блок відповідає за автоматичне створення звітів. Ця частина системи забезпечує формування як узагальнених, так і деталізованих звітів, призначених для різних категорій користувачів: команди тестувальників, менеджменту або замовника. На основі отриманих результатів також можуть автоматично генеруватися рекомендації, наприклад, щодо доцільності випуску релізу або необхідності додаткових перевірок.

Ще одним важливим елементом є інтелектуальний аналіз логів помилок. У рамках цього блоку здійснюється обробка і кластеризація лог-файлів із виокремленням основних проблемних зон, визначенням найпоширеніших помилок і виявленням зв’язків між різними інцидентами. Крім того, система має виявляти потенційно вразливі або нестабільні частини програмного продукту.

У сукупності всі ці функціональні блоки утворюють цілісну архітектуру, здатну підтримувати процес тестування на всіх його етапах – від аналізу вимог до формування підсумкової аналітики.

Висновок. Аналіз сучасних інструментів підтримки тестування ПЗ підтверджує наявність значного потенціалу для вдосконалення процесу тестування за допомогою GPT-моделей. Жоден із розглянутих продуктів не забезпечує комплексного підходу, який включає аналіз вимог, генерацію тест-кейсів, інтерпретацію логів і повноцінну автоматизацію звітності. Розробка інтелектуального асистента тестувальника на основі GPT, який підтримує всі етапи тесту-

* науковий керівник

вання, є актуальним і перспективним напрямом досліджень у галузі комп'ютерних наук.

Список використаних літературних джерел

[1] Ricca F., Marchetto A., Stocco A. A Multi-Year Grey Literature Review on AI-assisted Test Automation [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://arxiv.org/abs/2408.06224? (дата звернення 20.10.2025). – Назва з екрана.

[2] Garousi V. et al. AI-powered software testing tools: A systematic review and empirical assessment of their features and limitations [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://arxiv.org/abs/2409.00411? (дата звернення 20.10.2025). – Назва з екрана.

[3] Wu Xudong. Review of Anomaly Detection Based on Log Analysis [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://reference-global.com/article/10.21307/ijanmc-2020-036? (дата звернення 20.10.2025). – Назва з екрана.

[4] Schäfer M. et al. An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://arxiv.org/abs/2302.06527? (дата звернення 20.10.2025). – Назва з екрана.

[5] Boukhlif M., Hanine M., Kharmoum N. A Decade of Intelligent Software Testing Research: A Bibliometric Analysis [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.mdpi.com/2079-9292/12/9/2109?
(дата звернення 20.10.2025). – Назва з екрана.

[6] Blystiv O., Karashetskyy V. Аналіз та оптимізація технік попарного тестування та ортогональних масивів для ефективного тестування програмного забезпечення з використанням методології пріоритизації тест-кейсів [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://conf.nltu.edu.ua/index.php/nltu150/article/view/208 (дата звернення 20.10.2025). – Назва з екрана.

* науковий керівник

Development of the concept of an intelligent assistant for software testing based on the GPT language model

Oleh Blystiv, Volodymyr Karashetskyy

The current possibilities of using large language models, in particular GPT, in the software testing process are considered. The functionality of four popular tools that already use elements of artificial intelligence is analyzed: Testim, Mabl, Functionize and Qase. The limitations of their functionality are considered. The concept of an intelligent assistant for software testers with the integration of the GPT language model is proposed, which covers requirements analysis, test case generation, test result interpretation, automatic report generation and error log analysis and aims to increase testing efficiency, reduce the impact of the human factor and automate routine processes, which is relevant in the context of modern software development practice.

* науковий керівник