ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

INFORMATION TECHNOLOGY

DOI: 10.20998/2079-0023.2024.01.09
УДК 004.41+004.8

О. Г. ВОРОЧЕК, кандидат технічних наук (PhD), доцент кафедри Програмної Інженерії, Харківський національний університет радіоелектроніки, м. Харків, Україна; e-mail: olga.vorochek@nure.ua; ORCID: https://orcid.org/0000-0002-9054-9894
І. В. СОЛОВЕЙ, студент, Харківський національний університет радіоелектроніки, м. Харків, Україна; e-mail: illia.solovei@nure.ua; ORCID: https://orcid.org/0009-0005-5715-2755

ДОСЛІДЖЕННЯ ЗАСОБІВ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Предметом дослідження є засоби штучного інтелекту (ШІ) для автоматизації процесу тестування програмного забезпечення. Швидкий розвиток індустрії розробки програмного забезпечення останніми десятиріччями призвів до значного зростання конкуренції на ринку IT технологій і, як наслідок, жорсткіших вимог до відповідних продуктів та послуг. Автоматизація тестування за допомогою ШІ стає все більш актуальною завдяки своїй здатності вирішувати складні задачі, які раніше вимагали значних людських ресурсів.
Мета роботи – дослідження можливостей використання технологій ШІ для автоматизації процесів мануального тестування, що дозволить підвищити ефективність тестування, знизити витрати та покращити якість програмного забезпечення.
У статті вирішуються такі завдання: аналіз існуючих інструментів та підходів до автоматизації тестування за допомогою ШІ; розробка концептуальної моделі системи, що інтегрує ШІ у процес тестування; вивчення потенціалу ШІ для автоматизації різних аспектів тестування програмного забезпечення, таких як генерація тестових сценаріїв, виявлення дефектів, прогнозування помилок та автоматичний аналіз результатів тестування.
Використовуються такі методи: теоретичний аналіз літератури та існуючих рішень у галузі автоматизації тестування, експериментальне дослідження ефективності запропонованих методів автоматизації тестування.
Здобуто такі результати: представлено ідею системи, що інтегрує технології ШІ для автоматизації тестування програмного забезпечення. Виявлено, що використання ШІ дозволяє автоматизувати рутинні завдання тестування, значно знизити кількість людських помилок та покращити якість програмних продуктів та показники ефективності процесів верифікації та валідації.
Висновки: Розробка та впровадження систем автоматизації тестування на основі ШІ є надзвичайно актуальними і перспективними. Використання технологій ШІ дозволяє значно підвищити ефективність тестування, знизити витрати на його проведення та покращити якість програмного забезпечення. Запропонований підхід до розробки системи автоматизації тестування на основі ШІ може бути використаний як основа для подальших досліджень та розробок у цій галузі.

Ключові слова: штучний інтелект, автоматизація тестування, мануальне тестування, якість програмного забезпечення**.**

Вступ. Швидкий розвиток індустрії розробки програмного забезпечення останніми десятиріччями призвів до значного зростання конкуренції на ринку IT продуктів та послуг. У наслідок того, що кількість компаній, які готові надати замовнику якісну та професійну команду, або навпаки – команді професіоналів якісні та цікаві проєкти, суттєво збільшилась виникає проблема забезпечення високого рівня якості розроблених систем.

Одним з процесів інженерії якості є тестування, до якого, зазвичай, залучається уся команда розробки – розробники займаються написанням unit-тестів [1], їх запуском у спеціалізованих програмах, щоб не допустити потрапляння неякісного коду до кінцевої версії продукту; тестувальники на своєму боці розробляють тестові стратегії, забезпечують тестове покриття, оцінюють метрики щодо забезпечення якості, запускають тест-кейси різними методами, надають тестовий звіт та переслідують ту ж саму мету – надати якісний продукт кінцевому користувачу.

У свою чергу фахівці з тестування можуть бути двох типів: ті, хто займається мануальним тестуванням і ті, хто автоматизує процеси перевірки за допомогою написання програмного коду.

Треба розуміти, що автоматизація потребує значних фінансових витрат на налагодження і реалізацію відповідних процесів. Проте для деяких невеликих компаній, які вже мають команди ручного тестування, це не є економічно доцільним.

У свою чергу, великі компанії, що розробляють комплексні системи, наймають не одну, а навіть декілька команд тестування задля забезпечення ще більшого рівня якості продукту. Жодний продукт не може відмовитись від мануального тестування програмного забезпечення, де тести виконуються вручну тестувальниками.

У будь-якому випадку постає питання підвищення продуктивності праці та оптимізації витрат при використанні мануального тестування. Поточні тенденції програмної інженерії характеризуються широким залученням засобів і технологій штучного інтелекту для вирішення такого типу задач завдяки своїм широким можливостям і здатності вирішувати складні проблеми, які раніше вимагали значних людських ресурсів. Він дозволяє автоматизувати процеси, які потребують аналізу великих обсягів даних, прийняття швидких і точних рішень, а також адаптації до нових реалій.

ШІ також використовується для розпізнавання образів та мови, машинного навчання, прогнозування, оптимізації процесів, аналізу даних та підтримки прийняття рішень. Він знайшов своє застосування в різних галузях, таких як медицина (діагностика захворювань, аналіз медичних зображень), фінанси (передбачення ринкових трендів, управління ризиками), транспорт (автономні транспортні засоби, оптимізація логістики), виробництво (контроль якості, автоматизація виробничих процесів), маркетинг (персоналізована реклама, аналіз поведінки споживачів) та освіта (адаптивне навчання, автоматичне оцінювання знань).

Системи тестування програмного забезпечення також починають використовувати можливості штучного інтелекту для підвищення ефективності та точності тестування. ШІ допомагає автоматизувати рутинні тестові сценарії, виявляти аномалії, генерувати тестові дані, прогнозувати потенційні дефекти та аналізувати результати тестів. Наприклад, інструменти на основі ШІ можуть автоматично створювати тест-кейси, адаптувати їх до змін у коді, ідентифікувати найбільш ризиковані ділянки коду та пропонувати їх для тестування.

Однак, на сьогоднішній день таких засобів небагато, і вони ще знаходяться на етапі розвитку. Більшість існуючих рішень вимагають значних ресурсів для впровадження та підтримки. Тому дослідження та розробка нових інструментів, що інтегрують ШІ у процес тестування, є дуже актуальними та необхідними. Це дозволить значно знизити витрати на тестування, підвищити якість програмного забезпечення та скоротити час виходу продукту на ринок. Наше дослідження зосереджено на аналізі можливості впровадження інструментарію штучного інтелекту у процес ручного тестування, що робить його актуальним та перспективним для подальшого розвитку галузі.

Мета статті. Метою даної статті є дослідження можливості використання засобів та технологій штучного інтелекту задля автоматизації процесів мануального тестування.

Виклад основного матеріалу. Тестування – це діяльність, яка відбувається протягом життєвого циклу розробки програмного забезпечення, основною метою якої є виявлення помилок і переконання, що розроблена система задовольняє вимоги клієнтів. Тестування програмного забезпечення – спосіб оцінки системи шляхом виявлення відмінностей між ідентифікованими вимогами під час розробки вимог і отриманими результатами. Тестування можна проводити вручну або автоматизовано за допомогою спеціальних інструментів.

Під час мануального тестування перевіряються всі основні характеристики наданого програмного забезпечення – тестувальники запускають тест-кейси та генерують звіти без допомоги будь-яких автоматизованих інструментів тестування програмного забезпечення.

Серед переваг можна виділити швидкість та точність зворотного зв'язку, менші фінансові витрати через відсутність необхідності витрат бюджету на засоби автоматизації та окрему команду, непотрібність досвіду кодування для написання тестів.

До недоліків можна віднести меншу надійність результатів тестування через людський фактор, відсутність автоматичного створення логів (звітів), важкість виконання певних задач вручну та потреба у додатковому часі.

Автоматизоване тестування повністю покладається на попередньо розроблений тест, який запускається автоматично для порівняння фактичного результату з очікуваним [2]. Це допомагає тестувальнику визначити, чи працює програма у відповідності зі своїм функціоналом. Автоматизоване тестування дозволяє виконати повторювані завдання та тести регресії без додаткового втручання. Хоча всі процеси і виконуються автоматично, автоматизація вимагає певних ручних зусиль для створення початкових сценаріїв тестування.

До переваг автоматизованого тестування можна віднести більшу ефективність у пошуку помилок, порівняно з мануальним, підтримку різних технологій, поетапний запис зі збереженням процесу тестування, відсутність людського фактора та набагато більше покриття тестуванням, що дозволяє не пропустити навіть дрібні деталі.

Зазвичай впровадження автоматизованого тестування відбувається під час інтеграційного – фази перевірки програмного забезпечення, в якій окремі програмні модулі поєднуються та тестуються як група замість того, щоб розглядати кожний клас в окремості [3]. Цього можна легко досягти, використовуючи Junit-бібліотеки для модульного тестування програмного забезпечення на мові Java для резервного коду та Selenium-інструменту для автоматизації взаємодії веббраузера із інтерфейсом користувача.

Проте для використання цих інструментів, тестувальник потребує певних знань та навичок. До цього, автоматизоване тестування також має ряд недоліків: складність у створенні уявлення про візуальні аспекти інтерфейсу (кольори, шрифт, масштаб, контраст або розміри кнопок) без залучення людського елементу, вартість наявних інструментів та налагодження тестового сценарію, створення якого не є швидким процесом.

Спираючись на усі недоліки, доцільно було б розробити програмну систему, яка б автоматизувала

процес тестування без або з невеликою кількістю спеціальних навичок, яким потрібно вчитись.

Наразі існує велика кількість програм для автоматизації задач ручного тестування. Селеніум (Selenium) є прикладом одного з найбільш популярних інструментів. Саме він вважається галузевим стандартом для автоматизації тестування користувальницького інтерфейсу вебдодатків. Майже дев'ять з десяти тестувальників використовують або колинебудь використовували Селеніум у своїх проєктах, згідно з опитуванням щодо проблем автоматизації тестів [4].

Для розробників і тестувальників, які мають досвід та навички програмування та створення сценаріїв, Selenium пропонує гнучкість, яку не можна побачити у багатьох інших інструментах та фреймворках автоматизації тестів.

Щоб ефективно використовувати Selenium, користувачі повинні володіти передовими навичками програмування та мати запас часу, щоб створити фреймворки та бібліотеки, необхідні для автоматизації. Це головний недолік Selenium, який вирішується в інших інструментах, створених для безконтактної автоматизації тестів, як-от Katalon Studio.

Katalon Studio – це потужне і всебічне рішення для автоматизації тестування API, веб, мобільних та настільних додатків [5]. Він також має багатий набір функцій для багатьох типів тестування та підтримує декілька платформ, включаючи Windows, macOS та Linux.

Використовуючи двигуни Selenium та Appium, Katalon Studio забезпечує унікально інтегроване середовище для тестувальників, які мають труднощі в інтеграції та розгортанні різних фреймворків та бібліотек для використання Selenium та Appium, а також тих, хто вже знайомий з цими двигунами.

Проте він теж має ряд недоліків, такі як: створення сценарію обмежується лише Java та Groovy; неможливість автоматизації настільних додатків; відсутність підтримки розподіленого тестування.

Postman – ще один інструмент автоматизації, призначений для тестування API. Користувачі можуть встановити цей інструмент як розширення для браузера або настільний додаток на Mac, Linux та Windows. Він популярний не лише серед тестерів для автоматизації тестування API, але і серед розробників. За допомогою нього можна об'єднати створені тести та запити в одну автоматизовану послідовність. Postman має наступні переваги: комплексний набір функцій для проєктування, налагодження та тестування API; простий у користуванні інтерфейс; підтримка як автоматизованого, так і дослідницького тестування; використання форматів API Swagger [6] і RAML; доступна можливість упаковки запитів та надсилання іншим членам команди.

Проте також має і ряд суттєвих недоліків: обмежений варіант спільного доступу між командою; неможливо тестувати більше одного API одночасно; складний інтерфейс; відсутність журналу логів; важке перенесення папки з однієї колекції в іншу.

JMeter – ще один інструмент з відкритим кодом, призначений для тестового завантаження та вимірювання продуктивності. Даний інструмент також використовується для тестування API та сервісів, особливо для тестування продуктивності API. JMeter є третім найпопулярнішим інструментом для автоматизації тестів, на який посилаються 25% респондентів в опитуванні викликів автоматизації тестів [7].

Цей список інструментів не є повним, але він представляє найкращі інструменти, які є зрілими, популярними та надають можливості для вирішення тих проблем, з якими зараз стикаються організації, коли справа доходить до якості та швидкості.

Проте усі вони не є ідеальними. Використання багатьох цих інструментів потребують певних знань та навичок у програмуванні, що вимагає від компаній наймання окремої команди автоматизованого тестування. На сучасному ринку ще не існує програмної системи, якою могли б користуватися не тільки автоматизатори, а й мануальні тестувальники.

ШІ – інноваційна технологія, яка може бути застосована для забезпечення якості (QA) [8-9], і використання якої при валідації та верифікації програмного забезпечення, так само обґрунтоване, як і застосування методів штучного інтелекту у інших сферах, як-от охорона здоров'я, виробництво тощо. Поява великих даних та керування ними вимагає потужних обчислювальних інструментів, а штучний інтелект допомагає підвищити ефективність цієї діяльності. Методи ШІ застосовуються в різних областях розробки програмного забезпечення, таких як розробка вимог, проєктування та тестування. Штучний інтелект у тестуванні програмного забезпечення має на меті зробити тестування розумнішим і ефективнішим, він допомагає скоротити трудомістке ручне тестування, тож команди можуть зосередитися на складніших завданнях, як-от створення нових інноваційних функцій.

2023 Gartner Hype Cycle™ for Artificial Intelligence (AI) (див. рис. 1) визначає інновації та методи, які пропонують значні та навіть трансформаційні переваги.

Рис 1. Цикл ажіотажу Штучного Інтелекту

Генеративний штучний інтелект домінує в дискусіях щодо ШІ, підвищуючи продуктивність розробників і спеціалістів у дуже реальні способи, використовуючи такі системи, як ChatGPT. Це змусило організації та галузі переглянути свої бізнес-процеси та цінність людських ресурсів, підштовхнувши GenAI до піку завищених очікувань у циклі ажіотажу.

Gartner бачить дві сторони генеративного руху ШІ на шляху до більш потужних систем:

- інновації, які будуть підтримуватися GenAI;
- інновації, які сприятимуть розвитку GenAI.

Генеративний штучний інтелект впливає на бізнес, оскільки він пов'язаний із виявленням, створенням, автентичністю та регулюванням інформації і, що суттєво, він також має можливість автоматизувати роботу людини та досвід клієнтів і співробітників [10].

Методи ШІ можуть підтримувати процес тестування програмного забезпечення різними способами.

Типовими задачами, що виконуються у процесі тестування є тестування інтерфейсу користувача (UI) та тестування інтерфейсу прикладних програм (API).

Особливістю таких перевірок при ручному тестуванні є те, що більшість їх виконується на підставі інформації про бізнес-логіку системи.

Тестування UI передбачає перевірку таких аспектів як:

- зовнішній вигляд інтерфейсу. Перевірка відповідності дизайну макетам, правильність відображення на різних пристроях, відповідність стилів тощо;
- елементи керування. Коректність їхньої роботи, доступність для користувачів;
- навігація. Перевірка простоти переміщення між частинами системи, працездатність меню та інше;
- адаптивність;
- повідомлення про помилки та підказки.

Для побудови таких тестів методи ШІ використовують розпізнавання зображень для перевірки навігації та візуального тестування об'єктів і компонентів інтерфейсу користувача. Інструменти автоматизації декодують об'єктну модель документа і пов'язаний код для визначення властивостей об'єктів у тестуванні інтерфейсу користувача на основі ШІ. Тести інтерфейсу користувача виконуються після проєктування та розробки системи програмного забезпечення. Автоматизація цих перевірок є складною, і їх найважче автоматизувати.

У тестуванні API перевірка інтерфейсів прикладних програм (API) здійснюється шляхом тестування різних параметрів якості інтерфейсів, таких як функціональність, продуктивність, надійність і безпека. Автоматизація дозволяє почати перевірки на ранній стадії, що допомагає виявити та виправити помилки та зменшити зусилля під час тестування.

Типовий розподіл задач з верифікації та валідації з використанням ШІ можна представити схематично наступним чином (див. рис. 2) [11].

Рис. 2. Верифікація та валідація з використанням ШІ

На додаток до вищезгаданих стратегій автоматизації тестування ШІ також можна застосовувати як адаптивний метод для виявлення змін на рівні елементів, засіб передбачення неправильних тестових випадків та надання рекомендацій щодо вирішення цих проблем, як інструмент побудови моделей поведінки, для імітування поведінки користувачів під час використання системи за географічним положенням, пристроями та демографічними показниками, як вхідні дані для створення наборів тестів, як засіб прогнозування та автоматизації наборів тестів, для автоматизації сценаріїв.

Методи штучного інтелекту застосовуються для обслуговування тестів автоматизації, генерації тестових даних, раннього зворотного зв'язку під час тестування тощо (див. рис.3).

Рис. 3. Типові приклади використання ШІ

Приклади ШІ-інструментів автоматизації тестування, які представлені на ринку:

- Testim (https://www.testim.io/). Використовує машинне навчання для розробки, виконання та обслуговування автоматизованих тестів. Перевага – швидке створення стабільних тестів з навчанням, що нівелює

необхідність постійно підтримувати тести з кожною зміною коду;

- Functionize (https://www.functionize.com/). Інструмент тестування із застосуванням ШІ та алгоритмів машинного навчання для автоматизації тестування. Аналізує та вивчає поведінку користувачів, а потім генерує тести, що повторюють ці дії. Він також використовує технологію самовідновлення для автоматичного виявлення та вирішення проблем;
- Appvance (https://appvance.ai/). Використовує автоматизацію тестування на основі ШІ. Навчається на попередніх тестах і постійно вдосконалює свої методи тестування. Може створювати складні сценарії з мінімальними витратами часу та значно покращувати традиційне написання сценаріїв;
- Testcraft (https://home.testcraft.app/). Ця платформа автоматизації тестування на основі ШІ підтримує регресійне тестування, а також використовується для моніторингу вебдодатків;
- Watir (https://github.com/watir/watir). ШІ-інструмент з відкритим кодом для автоматизації тестування на основі Ruby. Він імітує взаємодію користувача з програмами, що підтримують тестування кількох браузерів;
- ACCELQ (https://www.accelq.com/). Потужний безкодовий інструмент на базі штучного інтелекту, який дозволяє автоматизувати багатоканальне тестування в Інтернеті, на мобільних пристроях, настільних комп'ютерах, API та серверній частині. Підтримує безперервну автоматизацію тестування;
- Katalon (https://katalon.com/). ШІ-інструмент для автоматизації тестування, який надає комплексне рішення для тестування мобільних додатків і вебсайтів. Він забезпечує вбудований інтерфейс для тестування. Він має надійне сховище об'єктів, багатомовну підтримку тощо;
- Selenium (https://www.selenium.dev/). Інструмент з використанням ШІ для автоматизації тестування, який в основному використовується для регресійного тестування, функціонального тестування та тестування продуктивності;
- Lamba Test (https://www.lambdatest.com/). ШІ-інструмент, який може запускати взаємне тестування та автоматизувати тестування понад 3000 пристроїв. Хмарна система безперервного тестування, яка підтримує 120 інтеграцій, а також тестування геолокації.

Проведений аналіз існуючих інструментів довів гіпотезу, що незважаючи на широке різноманіття продуктів автоматизації з ШІ, впровадження кожного з них у процес розробки передбачає певних витрат зусиль, часу та ресурсів, при цьому забезпечуючи надлишковий функціонал для невеликих команд, які як правило, складаються з розробників та, у найкращому випадку, мануальних тестувальників.

Все це зумовило спробу визначити концепт такої системи автоматизації процесу тестування, яка б дозволяла швидко і гнучко побудувати екосистему для контролю якості розробки програмних продуктів в умовах браку фахівців відповідної кваліфікації, тим самим забезпечивши можливість гарантування базової якості продуктів будь-якого виробника.

Основні проблеми, які повинна вирішувати така система: забезпечити зрозумілий фреймворк тестування; запропонувати можливість генерації адаптивної маршрутної карти процесу з виходом на вбудовані засоби сторонні інструменти, що реалізують відповідні типи тестування. Ця система має використовувати можливості ШІ, особливо генеративного, для реалізації тих дій, для яких потрібна кваліфікація розробника (наприклад, за необхідністю, для створення юніт-тестів).

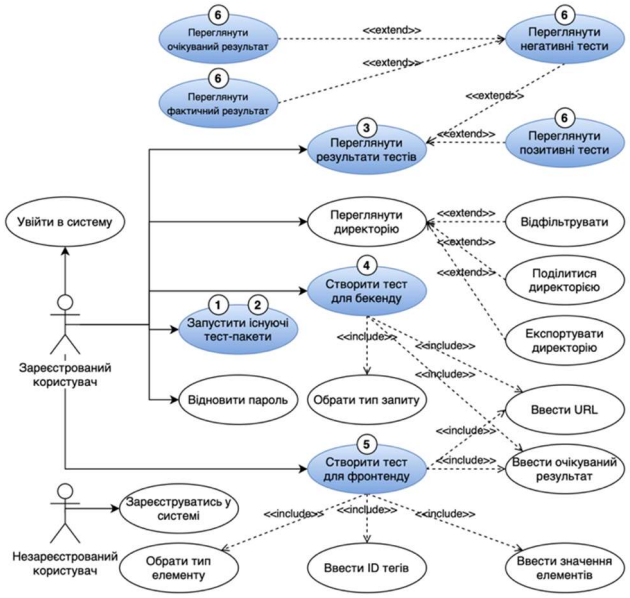

Рис. 4. UML діаграма прецедентів

Проілюструємо прикладом use-case діаграми прототипу такої системи, позначаючи елементи, для яких доцільно використання засобів ШІ.

Припустимо, що після входу в систему користувачу доступний ряд функцій. Він може переглянути вже збережені тестові набори та маніпулювати ними, тобто поділитись папкою з тестами, перейменувати, видалити, додати новий тест, експортувати або копіювати вже існуючий. Також є можливість шукати певний тестовий набір (папку) за допомогою пошукового поля. Також доступна функція перегляду та аналізу вже запущених раніше тестових компонентів, користувач може переглянути позитивні (passed) та негативні (failed) тести. У випадку якщо є негативні тести, користувач може переглянути очікуваний та фактичний результат, а також визначити розбіжності між ними.

Останні та найголовніші функції системи — це створення тестів для фронтенду та бекенду тощо. Після чого користувач матиме змогу згенерувати звіт визначеної форми (див. рис. 4).

Як видно з рисунку, інструменти штучного інтелекту будуть доречними для наступних складових системи [12].

1. Генерація маршрутної карти тестування продукту. Після входу до системи користувач вказує параметри розробки (тип продукту, архітектуру продукту, ключові функціональні та нефункціональні вимоги та інше), на підставі яких з використанням ШІ генерується Blueprint процесу тестування з посиланнями на інструментальні засоби, зовнішні чи вбудовані.

2. Розробка тестових артефактів (план тестування, тест-кейси, тестові сценарії, тестові набори даних) та формування тестових пакетів – інструменти генеративного ШІ дозволять оперувати не тільки базою знань самої системи, але й використовуючи глобальний пошук застосовувати досвід проєктів інших виробників.

3. Перегляд та аналіз тестів. ШІ дозволить проводити актуалізацію тестового пакету і підтримувати їхню тестопридатність.

4. Тестування бекенду, а також розробка юніт-тестів. Автоматизована генерація коду.

5. Тестування фронтенду. Спектр можливостей ШІ описаний вище. Також може здійснюватись не тільки перевірка, але й генерування рекомендацій.

6. Генерація звітів, так само як і генерація тестових артефактів, може виконуватись зі ШІ, враховуючи формування додаткової аналітики щодо ефективності, виявлення причин і взаємозалежностей дефектів.

Безумовно, що це не повний перелік можливостей автоматизації, але достатній для підтвердження доцільності подальших досліджень і розвитку запропонованої ідеї.

Висновки. Основною проблемою сучасних систем автоматизації процесу тестування програмного забезпечення є тенденція до їхнього надмірного ускладнення. Це призводить до втрати можливості використання цих засобів розробниками, для яких принциповими є обмеження ресурсів, доступних для процесів забезпечення якості. Дана стаття присвячена дослідженню існуючих можливостей використання технологій штучного інтелекту для організації процесів верифікації та валідації програмних продуктів, і спробі на його основі визначення необхідного і достатнього набору методів та засобів, використання яких дозволить підвищити рівень гарантування якості будь-якого проєкту та суттєво знизити витрати на реалізацію таких процесів.

Список використаної літератури

1. Osherove R. *The Art of Unit Testing. With Examples in C#.* New York: Manning, 2013. 292 p.
2. Automation Testing – Software Testing. *GeeksForGeeks*. URL: https://www.geeksforgeeks.org/automation-testing-software-testing/ (дата звернення: 09.05.2024).
3. Richardson J. A. *Automating & Testing a REST API*. Hertfordshire: Compendium Developments, 2017. 450 p.
4. Top Best 10 Automation Testing Tools in 2024. *Medium*. URL: https://medium.com/@dmautomationqa/top-best-10-automation-testing-tools-in-2024-e88b0b2bf493 (дата звернення: 10.05.2024).
5. Nayyar A. *Instant Approach to Software Testing Principles, Applications, Techniques, and Practices*. New Delhi: Bpb Publications, 2019. 370 p.
6. Ponelat J., Rosenstock L. *Designing APIs with Swagger and OpenAPI*. New York: Manning, 2022. 424 p.
7. Black R. *Advanced Software Testing - Vol. 1, 2nd Edition: Guide to the ISTQB Advanced Certification as an Advanced Test Analyst*. Los Angeles: Rocky Nook Inc., 2016. 376 p.
8. Buenen M., Natarajan S. World Quality Report 2022-23. *Campgemini*. URL: https://www.capgemini.com/wp-content/uploads/2022/10/WQR-2022-Report-Final.pdf (дата звернення: 12.05.2024).
9. Mariani L., Hao D., Subramanyan R., Zhu H. The central role of test automation in software quality assurance. *Software Quality Journal*. 2017. Vol. 25, No. 3. P. 797–802. DOI: doi.org/10.1007/s11219-017-9383-5.
10. What's New in Artificial Intelligence from the 2023 Gartner Hype Cycle. *Gartner*. URL: https://www.gartner.com/en/articles/what-s-new-in-artificial-intelligence-from-the-2023-gartner-hype-cycle (дата звернення: 12.05.2024).
11. Minimol Anil Job. Automating and Optimizing Software Testing using Artificial Intelligence Techniques. *International Journal of Advanced Computer Science and Applications (IJACSA)*. 2021. Vol. 12, No. 5. P. 594-602. DOI: http://dx.doi.org/10.14569/IJACSA.2021.0120571.
12. Wang J., Huang Y., Chen C., Liu Z., Wang S., Wang Q. Software Testing With Large Language Models: Survey, Landscape, and Vision. *IEEE Transactions on Software Engineering*. 2023. Vol. 50. P. 911-936. DOI: doi.org/10.1109/TSE.2024.3368208.

References (transliterated)

1. Osherove R. *The Art of Unit Testing. With Examples in C#.* New York, Manning, 2013. 292 p.
2. Automation Testing – Software Testing. *GeeksForGeeks*. Available at: https://www.geeksforgeeks.org/automation-testing-software-testing/ (accessed 09.05.2024).
3. Richardson J. A. *Automating & Testing a REST API*. Hertfordshire, Compendium Developments, 2017. 450 p.
4. Top Best 10 Automation Testing Tools in 2024. *Medium*. Available at: https://medium.com/best-automation-testing-tools (accessed 10.05.2024).
5. Nayyar A. *Instant Approach to Software Testing Principles, Applications, Techniques, and Practices*. New Delhi, Bpb Publications, 2019. 370 p.
6. Ponelat J., Rosenstock L. *Designing APIs with Swagger and OpenAPI*. New York, Manning, 2022. 424 p.
7. Black R. *Advanced Software Testing - Vol. 1, 2nd Edition: Guide to the ISTQB Advanced Certification as an Advanced Test Analyst*. Los Angeles, Rocky Nook Inc., 2016. 376 p.

8. Buenen M., Natarajan S. World Quality Report 2022-23. *Campgemini*. Available at: https://www.capgemini.com/wp-content/uploads/2022/10/WQR-2022-Report-Final.pdf (accessed 12.05.2024).
9. Mariani L., Hao D., Subramanyan R., Zhu H. The central role of test automation in software quality assurance. *Software Quality Journal*. 2017. Vol. 25, No. 3. pp. 797–802. DOI: doi.org/10.1007/s11219-017-9383-5.
10. What's New in Artificial Intelligence from the 2023 Gartner Hype Cycle. *Gartner*. Available at: https://www.gartner.com/en/articles/what-s-new-in-artificial-intelligence-from-the-2023-gartner-hype-cycle (accessed 12.05.2024).
11. Minimol Anil Job. Automating and Optimizing Software Testing using Artificial Intelligence Techniques. *International Journal of Advanced Computer Science and Applications (IJACSA)*. 2021. Vol. 12, No. 5. pp. 594-602. DOI: http://dx.doi.org/10.14569/IJACSA.2021.0120571.
12. Wang J., Huang Y., Chen C., Liu Z., Wang S., Wang Q. Software Testing With Large Language Models: Survey, Landscape, and Vision. *IEEE Transactions on Software Engineering*. 2023. Vol. 50. pp. 911-936. DOI: doi.org/10.1109/TSE.2024.3368208.

Надійшла (received) 15.05.2024

UDC 004.41+004.8

O. H. VOROCHEK, Associate Professor of the Department of Software Engineering, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine; e-mail: olga.vorochek@nure.ua; ORCID: https://orcid.org/0000-0002-9054-9894
I. V. SOLOVEI, Student, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine; e-mail: illia.solovei@nure.ua; ORCID: https://orcid.org/0009-0005-5715-2755

RESEARCH ON ARTIFICIAL INTELLIGENCE TOOLS FOR AUTOMATING THE SOFTWARE TESTING PROCESS

The subject matter of the article is artificial intelligence (AI) tools for automating the software testing process. The rapid development of the software development industry in recent decades has led to a significant increase in competition in the IT technology market and, as a result, stricter requirements for corresponding products and services. AI-driven test automation is becoming increasingly relevant due to its ability to solve complex tasks that previously required significant human resources.
The goal of the work is to investigate the possibilities of using AI technologies to automate manual testing processes, which will increase testing efficiency, reduce costs, and improve software quality.
The following tasks were solved in the article: analysis of existing tools and approaches to test automation using AI; development of a conceptual model of a system that integrates AI into the testing process; exploring the potential of AI to automate various aspects of software testing, such as generating test scenarios, detecting defects, predicting errors, and automatically analyzing test results.
The following methods are used: theoretical analysis of the literature and existing solutions in the field of test automation, experimental study of the effectiveness of the proposed test automation methods.
The following results were obtained: the concept of a system that integrates AI technologies for automating software testing is presented. It has been found that the use of AI allows automating routine testing tasks, significantly reducing the number of human errors, and improving the quality of software products and the effectiveness of verification and validation processes.
Conclusions: The development and implementation of AI-based testing automation systems are extremely relevant and promising. The use of AI technologies makes it possible to significantly increase the efficiency of testing, reduce the costs of its implementation, and improve the quality of software. The proposed approach to the development of an AI-based test automation system can be used as a basis for further research and development in this field.

Keywords: Artificial Intelligence, test automation, manual testing, software quality.

Повні імена авторів / Author's full names

Автор 1 / Author 1: Ворочек Ольга Григорівна / Vorochek Olga Hryhorivna
Автор 2 / Author 2: Соловей Ілля Владиславович / Solovei Illia Vladyslavovych