https://www.sworldjournal.com/index.php/swj/article/view/swj28-00-042
DOI: 10.30888/2663-5712.2024-28-00-042

УДК 65.012.54

APPLICATION OF ARTIFICIAL INTELLIGENCE FOR SOFTWARE TESTING AUTOMATION

ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Huralnyk F. B./ Гуральник Ф. Б.
PhD student/аспірант
ORCID: 0009-0009-3832-0206
Vinnitsa National Technical University,
95 Khmelnytske Shose, Vinnytsia, Ukraine, 21021
Вінницький національний технічний університет
Хмельницьке шосе, 95, Вінниця, Україна, 21021

Анотація. *В роботі розглядаються теоретичні аспекти застосування штучного інтелекту (ШІ) для автоматизації тестування програмного забезпечення (ПЗ). В умовах швидкозростаючої складності програмних систем і необхідності прискорення циклів розробки автоматизація тестування стає критично важливою. У статті проводиться огляд поточних технологій і підходів, таких як використання машинного навчання для прогнозування помилок, застосування комп'ютерного зору для тестування користувацького інтерфейсу, а також генерація тест-кейсів за допомогою нейронних мереж. Особлива увага приділяється аналізу переваг і недоліків кожного методу, а також теоретичним можливостям інтеграції цих технологій у процеси тестування ПЗ. Нарешті, обговорюються перспективи та майбутні напрями розвитку ШІ в галузі автоматизації тестування ПЗ. Отримані результати демонструють значний потенціал використання ШІ для підвищення якості та продуктивності процесів тестування, що робить цей напрям перспективним для подальших досліджень і впровадження.*

Ключові слова: *тестування, автоматизація тестування, ШІ, тестова документація, оптимізація тестування*

Вступ.

Автоматизація тестування програмного забезпечення (ПЗ) є однією з ключових складових процесу розробки, яка сприяє підвищенню якості продуктів та зменшенню часу їх випуску на ринок. Зі зростанням складності програмних систем і вимог до їхньої надійності, традиційні методи автоматизації тестування стають все менш ефективними та вимагають значних людських і часових ресурсів. У зв'язку з цим, застосування штучного інтелекту (ШІ) відкриває нові перспективи для оптимізації цих процесів.

ШІ, зокрема машинне навчання, комп'ютерний зір та обробка природної мови, може значно підвищити ефективність і точність тестування ПЗ. Здатність ШІ аналізувати великі обсяги даних, виявляти патерни та передбачати потенційні помилки відкриває нові можливості для створення інтелектуальних систем тестування. Використання нейронних мереж для автоматичного створення тест-кейсів та аналізу звітів про помилки може значно зменшити навантаження на тестувальників і підвищити продуктивність тестування.

Актуальність

У сучасному світі програмне забезпечення стає все більш складним і багатофункціональним, вимагаючи від розробників високої точності та надійності в процесі створення. Зростання обсягів і складності ПЗ

супроводжується підвищенням вимог до його якості та швидкості випуску на ринок. Традиційні методи тестування, засновані на ручному або навіть частково автоматизованому підході, часто не справляються з такими викликами через обмежену масштабованість і значні витрати часу та ресурсів.

В умовах швидкого розвитку інформаційних технологій та появи нових методів розробки ПЗ, таких як Agile та DevOps, тестування повинно бути інтегроване у всі етапи життєвого циклу розробки. Це вимагає нових підходів до автоматизації тестування, які можуть забезпечити високу продуктивність і точність. Штучний інтелект (ШІ) та машинне навчання пропонують інноваційні рішення для автоматизації тестування, здатні адаптуватися до змін і ефективно працювати з великими обсягами даних.

Застосування ШІ в тестуванні ПЗ має значний потенціал для підвищення якості і зниження витрат на тестування. Наприклад, алгоритми машинного навчання можуть автоматично виявляти і передбачати помилки, що значно скорочує час на пошук і виправлення багів. Використання комп'ютерного зору дозволяє автоматизувати тестування графічних інтерфейсів користувача, що забезпечує більш точну і ефективну перевірку функціональності.

Отже, актуальність даного дослідження обумовлена необхідністю вдосконалення методів автоматизації тестування програмного забезпечення в умовах зростаючих вимог до його якості та швидкості розробки. Вивчення теоретичних аспектів застосування ШІ для автоматизації тестування відкриває нові перспективи для розвитку цієї галузі, сприяючи підвищенню ефективності та надійності процесів розробки ПЗ.

Постановка задачі

Метою даного дослідження є всебічний розгляд процесу тестування програмного забезпечення з акцентом на автоматизовані методи та впровадження штучного інтелекту (ШІ) для оптимізації тестувальних процесів. В рамках цього дослідження передбачено вирішення кількох важливих завдань:

Розгляд видів тестування: Проаналізувати різні підходи та види тестування, що використовуються в індустрії розробки програмного забезпечення, приділяючи особливу увагу автоматизованому тестуванню як одному з ключових напрямів сучасного забезпечення якості.

Огляд інструментів для автоматизованого тестування: Здійснити огляд основних інструментів, які застосовуються в автоматизованому тестуванні, визначити їх переваги та сфери застосування, а також виявити інноваційні можливості, які вони надають.

Опис методів використання штучного інтелекту в автоматизованому тестуванні: Проаналізувати та описати способи застосування ШІ в автоматизації процесу тестування, зокрема в таких напрямках, як генерація тест-кейсів, аналіз результатів тестування та прогнозування дефектів.

Огляд інструментів на основі ШІ: Розглянути специфічні інструменти, що використовують ШІ для автоматизації тестування, та проаналізувати їх функціональність і вплив на якість процесу тестування.

Приклад використання ШІ в автоматизації регресійного тестування: Розглянути приклад впровадження ШІ в автоматизоване регресійне тестування,

продемонструвавши, як інтелектуальні алгоритми можуть ефективно оновлювати тест-кейси та забезпечувати стабільність продукту під час регулярних оновлень та ітерацій.

Задачі дослідження:

- Розглянути загальні принципи, види та завдання тестування ПЗ;
- Здійснити огляд сучасних інструментів автоматизованого тестування;
- Вивчити та описати методи та підходи застосування ШІ в автоматизації тестування;
- Навести приклад впровадження ШІ в регресійне тестування для оновлення тест-кейсів і демонстрації переваг цього підходу.

Це дослідження спрямоване на те, щоб надати практичний та теоретичний огляд сучасних технологій у тестуванні, підкреслити важливість інноваційного підходу з використанням ШІ та продемонструвати його вигоду для автоматизації регресійного тестування в умовах швидких змін у розробці програмного забезпечення.

Мета

Метою статті є дослідження використання штучного інтелекту (ШІ) в автоматизації регресійного тестування для оптимізації процесів тестування програмного забезпечення. Основна увага приділяється скороченню часу, необхідного для виконання регресійного тестування, а також автоматичному оновленню тестових сценаріїв і тестових даних за допомогою ШІ.

У рамках дослідження передбачається:

- Проведення огляду теоретичних основ регресійного тестування та автоматизації процесів тестування.
- Аналіз сучасних інструментів, що застосовуються для автоматизації регресійного тестування.
- Опис методів використання ШІ для оптимізації тестування, зокрема в аспектах генерації та оновлення тестових даних.
- Надання практичного прикладу використання ШІ для автоматизації регресійного тестування, демонструючи переваги впровадження інтелектуальних алгоритмів у цей процес.
- Надати математичне підтвердження переваг використання ШІ шляхом аналізу статистичних даних і візуалізації через графіки.

Огляд літератури

Сучасні дослідження підтверджують, що автоматизація тестування за допомогою штучного інтелекту (ШІ) значно прискорює процес тестування та підвищує його точність. Зі зростанням складності програмного забезпечення традиційне ручне тестування стає менш ефективним, оскільки потребує багато часу і ресурсів. Автоматизоване тестування з використанням ШІ здатне вирішувати цю проблему, дозволяючи виконувати тести швидше і з більшою надійністю.

Застосування ШІ в тестуванні охоплює кілька ключових напрямків, які описуються у літературі:

1. ***Прискорення тестування***: Алгоритми ШІ дозволяють виконувати тести автоматично і безперервно, значно скорочуючи час, потрібний для тестування на

кожному етапі розробки.

2. ***Автоматичне створення тест-кейсів***: ШІ може автоматично генерувати тест-кейси на основі аналізу вимог та історичних даних. Це не лише економить час тестувальників, але й знижує ймовірність пропущених помилок.

3. ***Точність виявлення помилок***: Завдяки ШІ тести стають більш надійними. Алгоритми допомагають виявляти баги, які можуть бути пропущені вручну, особливо у великих системах із складними залежностями.

Таким чином, використання ШІ для автоматизації тестування дозволяє зробити процес тестування не тільки швидшим, але й надійнішим, що підтверджується дослідженнями в цій галузі.

Тестування ПЗ

Тестування програмного забезпечення є одним із ключових етапів життєвого циклу розробки програмного забезпечення та важливим методом для оцінки розробленого продукту з метою забезпечення його якості. Перед випуском фінального продукту важливо переконатися, що вимоги, визначені на етапі аналізу, були виконані, а сам продукт не містить дефектів. У контексті розробки програмного забезпечення "баг" є назвою для дефекту, який означає, що розроблена система не виробляє точних результатів відповідно до вимог. Тестування відіграє важливу роль у процесі розробки програмного забезпечення. У процесі розробки програмного забезпечення можуть використовуватися різні техніки для полегшення тестування. Прототипування є одним із підходів, при якому розробляється експериментальний програмний артефакт, який буде відкинутий після оцінки. На відміну від цього, у ітеративному підході ранні програмні артефакти не відкидаються, а використовуються для подальшого розвитку. Інший підхід полягає у використанні фреймворків, таких як метод динамічних систем розробки (DSDM), в якому документуються найкращі практики для ітеративної та інкрементальної розробки. Розробникам програмного забезпечення необхідно знати та вміти використовувати різноманітні методи, концепції та практики тестування. Чотири основні категорії тестування підсумовані на рис. 1:

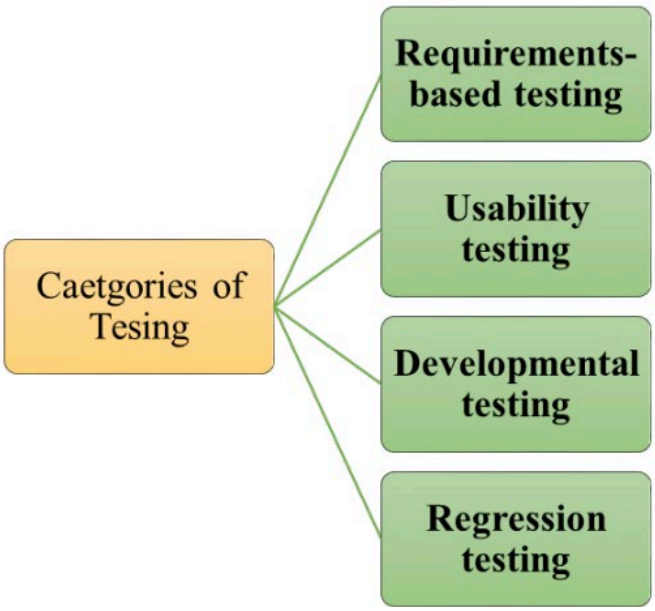

Рисунок 1 категорії тестування

В табличці 1 надано деталі по усім 4 категоріям

Requirements-based testing	- Перевірка того, що система відповідає вимогам замовника, здійснюється за допомогою раніше зібраних або сформульованих тестованих вимог. - Приймальне тестування є фінальним етапом, на якому перевіряється, чи були задоволені вимоги користувача. - Замовник формально приймає програмне забезпечення, переконавшись у правильному функціонуванні приймальних тестів.
Usability testing	Інтерфейс користувача (UI) є невід'ємною частиною кожної програмної системи. - Різні користувачі, окрім розробників, використовують систему, тому тестування зручності використання є необхідним. - Через проблеми з інтерфейсом користувача система може зазнати невдачі. - Тестування зручності використання включає систематичну перевірку інтерфейсу користувача із залученням цільових користувачів.
Development testing	Зазначає тестування, яке проводиться всією командою розробки програмного забезпечення. - Розробницьке тестування здійснюється на трьох різних рівнях: тестування одиниць (unit testing), інтеграційне тестування (integration testing) та системне тестування (system testing). - Інтеграційне тестування включає перевірку того, що всі протестовані одиниці правильно взаємодіють між собою.
Regression testing	Будь-яка форма тестування, що проводиться під час розробки або обслуговування системи, щоб забезпечити, що виправлення одного дефекту не призводить до появи нових. - Регресійні тести потрібні на рівнях одиниць, інтеграції та системи, залежно від конкретного методу розробки, що застосовується. - Регресійні тести є критично важливими під час розробницького тестування та в обслуговуванні системи.

Техніки тест дизайну

Стратегії створення тест-кейсів є важливою частиною валідації та верифікації. Основні стратегії включають тестування чорної скриньки (black-box testing) та тестування білої скриньки (white-box testing). Основна мета тестування чорної скриньки – забезпечити, що кожен аспект вимог замовника правильно

обробляється реалізацією. Тест-кейси розробляються на основі специфікацій системи, які включають деталі передбачених функцій системи.

Тестування сірої скриньки (grey-box testing) є іншим методом, при якому додаток тестується з повним знанням загальних аспектів системи і обмеженим знанням її внутрішнього функціонування. Основна мета тестування білої скриньки – перевірити, чи деталі реалізації системи правильні. Тест-кейси розробляються на основі деталей реалізації системи для перевірки, чи система виконує свої передбачені функції коректно.

При порівнянні можливостей автоматизації технік тестування чорної та білої скриньки, автоматизація тестування білої скриньки є легшою, ніж тестування чорної скриньки. Причина цього полягає в тому, що у тестуванні чорної скриньки програміст і тести є залежними, що може вплинути на автоматизацію. Оскільки тестування сірої скриньки поєднує риси обох технік чорної та білої скриньки, його особливості і техніки не розглядаються детально в цій статті.

На рисунку 2 зображені види методом чорної скриньки і білої.

Автоматизоване тестування — це процес тестування програмного забезпечення, який виконується за допомогою спеціальних інструментів і програм, що автоматично виконують тестові сценарії без ручного втручання. Це дозволяє ефективно перевіряти функціональність, продуктивність та інші аспекти програмного забезпечення, використовуючи заздалегідь визначені тест-кейси та сценарії.

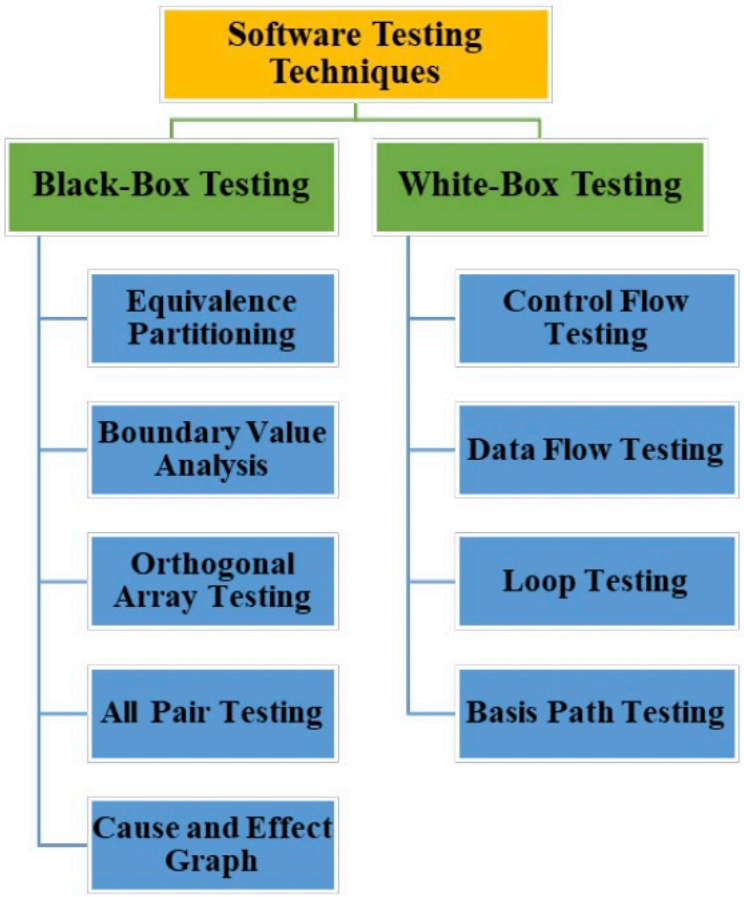

Рисунок 2 види білої і чорної скриньки

Автоматизоване тестування

З метою забезпечення того, що користувачі дійсно отримують ту цінність, яку обіцяє програмне забезпечення, з'явилося багато потужних інструментів тестування, які спрощують автоматизоване і повторюване тестування чорної скриньки. Автоматизація є важливою, оскільки процес тестування є повторювальним, і рекомендується перевіряти всі можливі сценарії. Автоматизація тестів збільшує охоплення тестування і покращує ефективність. Коли тестування автоматизоване, тести можуть виконуватися повторно, перевіряти різні вхідні значення та умови. Це дозволяє зменшити ресурси та час, необхідні для тестування. Для автоматизації функціональних, системних та приймальних тестів доступні численні інструменти. Серед них: Selenium, Watir і JMeter. Selenium і Watir використовуються для тестування локальних файлів за допомогою протоколу file://, підтримуваного веб-браузерами, в той час як JMeter потребує, щоб цільові файли були розміщені на веб-сервері.

Застосування штучного інтелекту для тестування програмного забезпечення

З огляду на зростаючу складність і функціональність сучасних додатків, для досягнення функціональних і нефункціональних вимог додатків необхідно реалізувати різні функції. Вимоги — це інформація про те, яким буде і що робитиме система, яку потрібно знати до початку розробки. Після того як вимоги будуть визначені і погоджені розробником і замовником, вони будуть реалізовані в програмному забезпеченні. Перетворення вимог у додаток за допомогою відповідних інструментів і технік є роллю системного розробника. Додаток має бути розроблений без помилок, і під час етапу кодування розробник повинен написати і виконати тест-кейси.

Тестувальник програмного забезпечення — це особа, яка відповідає за перевірку додатка і забезпечення його правильного функціонування. Тестувальники відіграють важливу роль, так само як і розробники, у діяльності з розробки програмного забезпечення, оскільки вони є частиною забезпечення якості програмного продукту. Тестувальник програмного забезпечення надсилає звіт команді розробки, в якому детально описуються виявлені баги та послідовність подій, що сприяли помилці. Як показано на рис. 3, відповідальність розробника полягає в написанні коду та тест-кейсів, а тестувальники слідують тестовим планам і виконують завдання тестування.

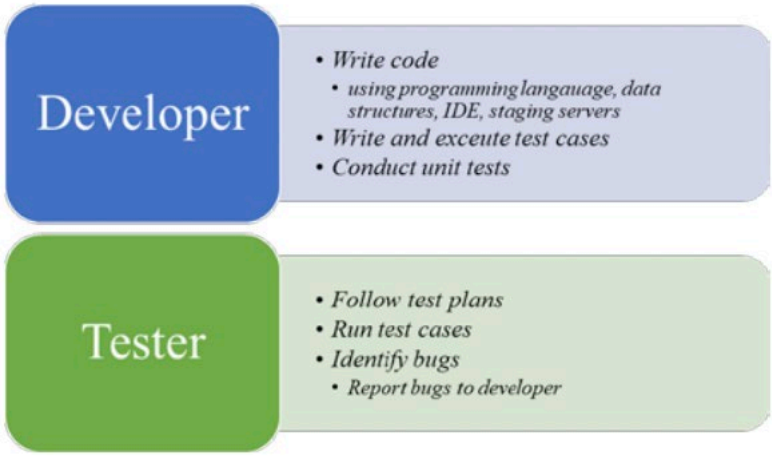

Рисунок 3 Відповідальність розробника і тестувальника на проекті

Тестування програмного забезпечення є важливим процесом, який забезпечує відповідність розробленої системи очікуванням клієнтів. Тестування допомагає захистити систему від можливих відмов, які можуть негативно вплинути на ефективність організації під час експлуатації. У ручному тестуванні тести виконуються тестувальником програмного забезпечення для виявлення помилок у системі. Хоча у ручному тестуванні можливо проводити експлораційне тестування, цей процес є трудомістким і потребує залучення людських ресурсів, а тести виконуються людиною та програмним забезпеченням.

Автоматизоване тестування значно швидше, ніж ручне тестування, і використовує інструменти автоматизації для виконання тест-кейсів. Оскільки тестування все більше переходить до автоматизації, техніки ШІ (штучного інтелекту) знаходять своє застосування в тестуванні програмного забезпечення. Техніки ШІ підтримують автоматизацію процесів тестування ефективним способом. Алгоритми ШІ допомагають тестовому середовищу бути більш продуктивним. Алгоритми ШІ сприяють процесу і допомагають тестувальникам програмного забезпечення знаходити максимальну кількість помилок за менший проміжок часу. Система може бути виведена на ринок як надійна і точна.

Рис. 4 і Рис. 5 показують функції тестувальника і розробника у ручному тестуванні та в середовищах тестування, що підтримуються ШІ.

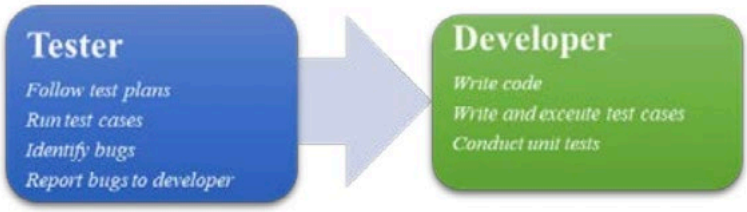

Fig. 4. Functions in a Manual Testing Environment.

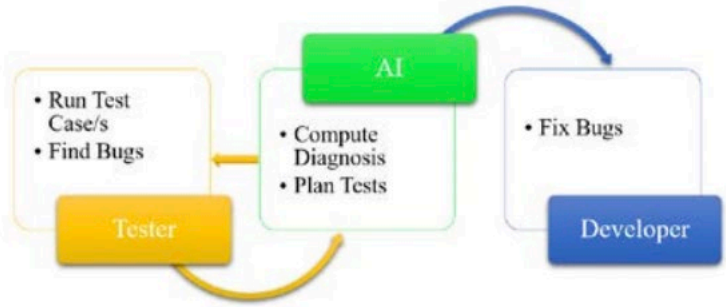

Рисунок 4 Функції в менуальному середовищі

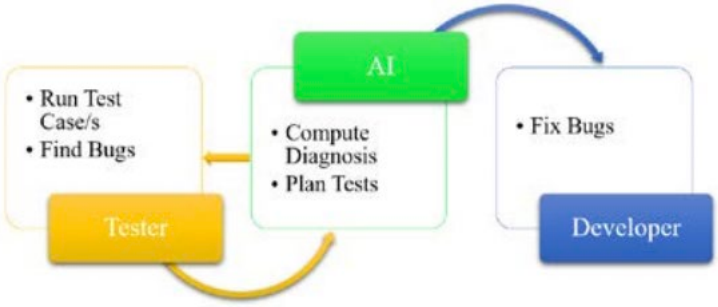

Рисунок 5 Функції в середовищі тестування, що підтримується штучним інтелектом

У середовищі ручного тестування тестувальник запускає тест-кейси відповідно до тестових планів, знаходить помилки та повідомляє про них розробникам.

У середовищі тестування на основі штучного інтелекту тестувальник також запускає тест-кейси і виявляє помилки. Однак, інструмент ШІ виконує діагностику помилок і надсилає сповіщення розробникам для їх виправлення. Інструмент ШІ автоматично планує подальші тестування, а тестувальники продовжують запускати тест-кейси та виявляти дефекти.

Функціональні та нефункціональні вимоги продукту повинні бути протестовані з забезпеченням повного покриття тестуванням. У ручному тестуванні це займе багато часу та ресурсів, тоді як застосування технік ШІ зменшує час і допомагає виявити погано спроектовані вимоги. Вимоги будуть ретельно перевірені за допомогою ШІ, і вони перейдуть на наступний етап розробки програмного забезпечення — етап проектування.

Виходячи з досвіду тестувальників, вибір і якість тестів можуть змінюватися, і тести створюються на основі вимог, випадків використання та історій користувачів. Застосування ШІ допомагає автоматично генерувати тест-кейси та ідентифікувати тести для верифікації та валідації системи. ШІ також допомагає зрозуміти покриття тестування. Неповні або невідповідні тести можуть бути виявлені на основі моделей. Крім того, автоматичне генерування тест-кейсів також зменшує ймовірність пропущених помилок і дефектів.

Застосування ШІ в автоматизації тестування

Бізнес-організації в різних секторах впроваджують методи штучного інтелекту (ШІ), щоб зробити свої операції ефективнішими. У процесі розробки програмного забезпечення автоматизоване тестування широко використовується для підвищення ефективності тестування. Існують різні способи, як техніки ШІ можуть підтримувати процес тестування програмного забезпечення. Найбільшу користь від ШІ отримують під час тестування "чорного ящика". Застосування технік ШІ у тестуванні програмного забезпечення подано в переліку. На рисунку 6 зображено автоматизацію тестування у формі піраміди згідно з типами тестів. Тестування інтерфейсу користувача (UI) розміщене на найвищому рівні, оскільки це найдорожчий вид тестування. Автоматизація тестування може проводитися на нижчих рівнях, таких як API і Unit тестування, щоб досягти найкращого покриття тестів та зменшити час і вартість.

Тестування інтерфейсу користувача (UI): Тести на зручність використовуються для виявлення будь-яких невідповідностей у дизайні або проблем з використанням інтерфейсу користувача. Для створення тестів UI техніки штучного інтелекту використовують технології розпізнавання зображень для навігації по додатку та візуальної перевірки об'єктів і компонентів інтерфейсу користувача. Інструменти автоматизації тестування декодують Document Object Model (DOM) та пов'язаний код, щоб визначити властивості об'єктів під час тестування UI на основі ШІ. Тести UI проводяться після етапів проєктування та розробки програмної системи. У піраміді тестування UI-тести знаходяться на найвищому рівні, їх автоматизація є складною, і вони є одними з найважчих для автоматизації.

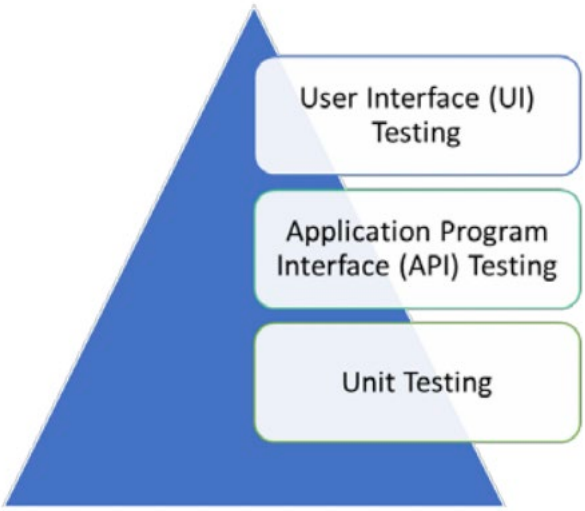

Рисунок 6 Рівні автоматизованого тестування

Тестування програмного інтерфейсу застосунків (API): Під час тестування API здійснюється перевірка програмних інтерфейсів шляхом тестування різних факторів якості, таких як функціональність, продуктивність, надійність і безпека. Нефункціональні вимоги, як-от зовнішній вигляд, не є релевантними для тестування API, оскільки воно фокусується на бізнес-логіці. Без врахування користувацького інтерфейсу автоматизація API може використовуватися для перевірки бізнес-логіки. Автоматизація API дозволяє команді розробників програмного забезпечення почати тестування на ранньому етапі, що допомагає швидше виявляти та виправляти помилки, знижуючи витрати на тестування за допомогою ШІ.

Створення та оновлення модульних тестів: Модульні тести використовуються для перевірки правильного виконання невеликих частин коду, таких як функції або методи об'єктів, перед тим як вони будуть об'єднані з іншими модулями для формування більшої функціональної особливості. Розробники витрачають значну кількість часу на написання та підтримку модульних тестів, що зазвичай менш цікаве, ніж написання коду програми. Продукти на основі штучного інтелекту для автоматизованої розробки модульних тестів можуть бути корисними, оскільки їх можна швидко згенерувати. Стратегія автоматизації починається з модульних тестів. Модульні тести пишуться розробниками, використовуючи ту ж мову програмування, що й для розробки додатків. Раннє виявлення та виправлення помилок під час модульного тестування запобігає виникненню дефектів регресії.

Окрім вищезазначених стратегій автоматизації тестування, штучний інтелект також можна застосовувати в таких сферах тестування програмного забезпечення:

- ✓ ШІ використовується як адаптивний метод для виявлення змін на рівні елементів, що підвищує надійність тестових наборів.
- ✓ Допомагає у прогнозуванні некоректних тестових випадків, що призводять до повних збоїв тестування, а також надає рекомендації щодо вирішення цих проблем.

- ✓ Використовується для симуляції поведінкових моделей.
- ✓ Техніки ШІ допомагають симулювати поведінку людей під час використання системи за географічними даними, пристроями та демографією, які використовуються для створення тестових наборів.
- ✓ Алгоритми ШІ ефективні у прогнозуванні та автоматизації тестових наборів.
- ✓ Застосовується в автоматизації сценаріїв.
- ✓ Автоматизація тестового сценарію не потрібна, оскільки він виконується автоматично за допомогою алгоритму ШІ.
- ✓ Допомагає у виявленні дефектів, на основі яких можна визначати тестові набори, що прискорює процес прийняття рішень щодо охоплення тестування та оптимізації тестових наборів.
- ✓ Збільшує кількість тестів та їхній масштаб.
- ✓ Техніки ШІ застосовуються для підтримки автоматизованих тестів, генерації тестових даних, надання раннього зворотного зв'язку у процесі тестування тощо.

Інструменти для автоматизації тестування на основі AI

В таблиці 2 наведені приклади інструментів автоматизації.

Назва інструменту	Опис
Testim	використовує машинне навчання для створення, виконання та підтримки тестів. Цей інструмент покращує тестові набори шляхом аналізу поведінки тестів і автоматичного виправлення помилок.
Mabl	застосовує AI для самонавчання та оптимізації автоматизованих тестів, що допомагає швидше виявляти баги й покращувати тестовий процес.
Functionize	використовує AI для створення тестів без написання коду, дозволяючи автоматизувати функціональні та регресійні тести.
Applitools	спеціалізується на AI-візуальному тестуванні, допомагаючи автоматично перевіряти інтерфейси користувача на наявність візуальних помилок.
Watir	це інструмент з відкритим кодом, який використовує бібліотеки Ruby для автоматизації тестів. Watir використовується для тестування вебсайтів і використовує Selenium для автоматизації браузера. Крім того, Watir також підтримує написання стабільних і зручних для підтримки тестових сценаріїв.

Функціональні тести є частиною інтеграційних тестів і призначені для оцінки конкретних завдань. Наприклад, реєстрація користувача в онлайн-системі: функція реєстрації повинна перевіряти правильність введених даних. Інші приклади включають пошук в каталозі та онлайн-оплату в додатку електронної комерції тощо. Для виконання цих тестів потрібен широкий діапазон значень тестових даних.

Тестування продуктивності використовується для вимірювання здатності серверів реагувати на запити користувачів в мережевому середовищі. Навантажувальне тестування є методом оцінки продуктивності сервера шляхом застосування певного навантаження до додатку з метою оцінки часу відгуку для окремих функцій.

Метою тестування безпеки є забезпечення того, що додаток має контролі аутентифікації та авторизації та не є вразливим до атак. Інтерфейси користувача зазвичай перевіряються на помилки навігації, помилки відображення та проблеми з використанням елементів управління. Регресійне тестування може виявити дефекти, що знову з'являються в результаті змін у програмному забезпеченні, які не були заплановані.

Практичне застосування: Використання Watir

Watir (Web Application Testing in Ruby) було обрано як інструмент для експериментування з програмним забезпеченням для підтримки технічних аспектів дослідження. Watir підтримує майже всі веб-браузери. Вебсайти та інтерфейси користувача можуть бути протестовані за допомогою Watir. Тестовий фреймворк, що використовує Watir, дотримується наступного порядку, як показано на рисунку 7.

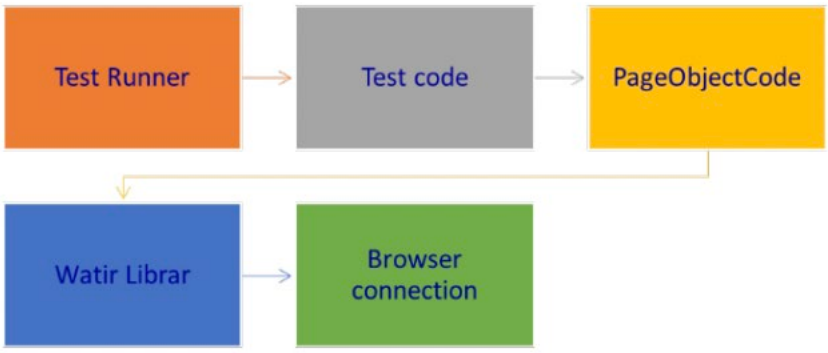

Рисунок 7 Порядок тестового фреймворку в Watir

Після встановлення необхідних драйверів були написані тестові скрипти для відкриття браузерів у RubyMine IDE. Тестові скрипти написані на мові Ruby. Watir обрано для практичного застосування, оскільки це дуже простий у використанні відкритий інструмент. Цей інструмент розроблений з використанням Ruby.

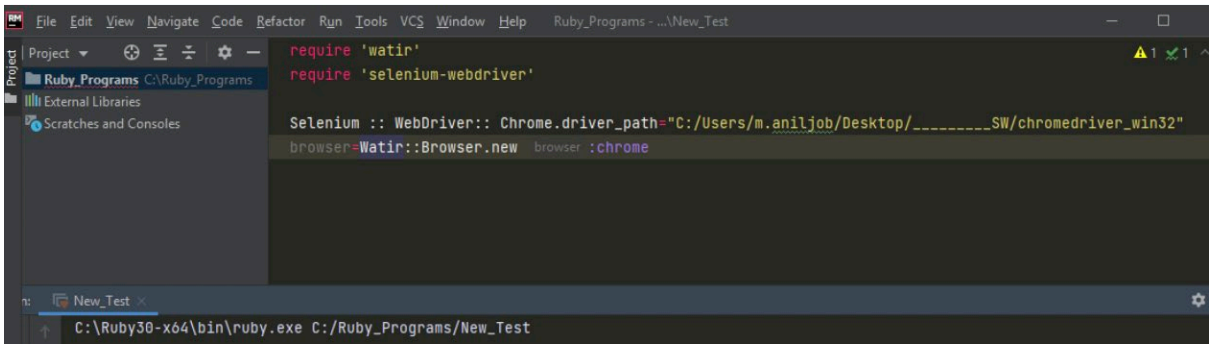

Рисунок 8 тестовий скрипт для відкриття браузера Chrome

Будь-який веб-додаток може бути автоматизований, незалежно від браузера, в якому він працює. Watir має вбудовані бібліотеки для підтримки різних видів активностей, таких як продуктивність сторінки. Рисунок 8 демонструє тестовий скрипт для відкриття браузера Chrome.

Рисунок 9 показує інший приклад створення та виконання тест-кейсу за допомогою Watir. Використано простий тест для перевірки точності коду. Тестовий скрипт успішно виконується.

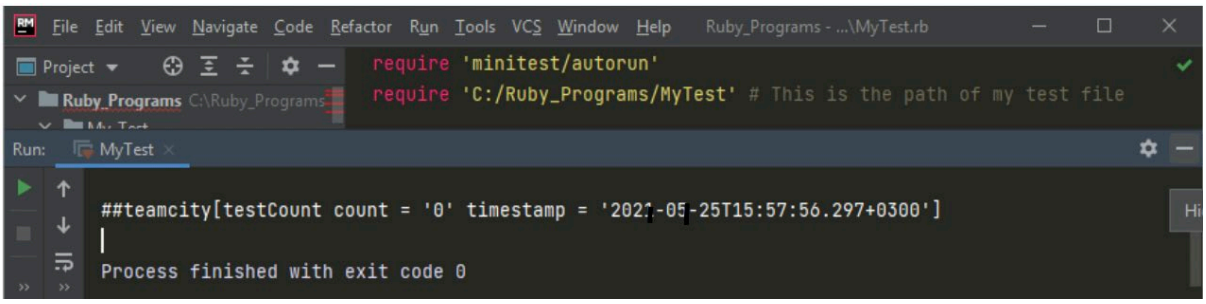

Рисунок 9 приклад тестового скрипту

Рисунок 10 показує приклад тестового скрипту для тестування інтерфейсу користувача.

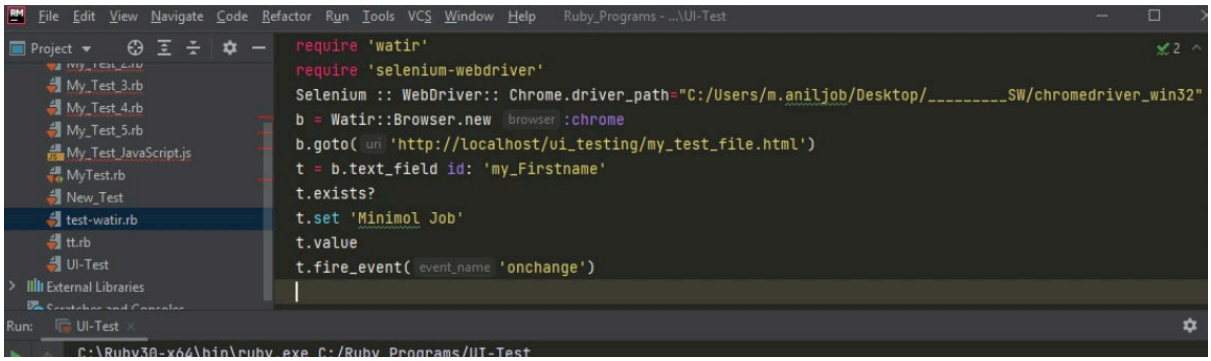

Рисунок 10. Скрипт для тестування інтерфейсу

Презентація веб-сторінки здійснюється за допомогою HTML, а функція JavaScript використовується для статичної частини коду сторінки за допомогою HTML. Рисунок 11 показує код.

Після виконання тестових кейсів можна зробити висновок, що Watir дозволяє легко тестувати завантаження файлів для інтерфейсу користувача веб-додатку. Іноді також надаються сповіщення про тести в спливаючих вікнах. Іншою особливістю Watir є можливість вимірювання продуктивності сторінки за допомогою об'єкта продуктивності. Навігація, час відгуку та продуктивність пам'яті можуть бути виміряні шляхом отримання цих даних, коли додаток підключений до браузера. Особливість Page object, доступна в Watir, сприяє повторному використанню коду у формі класів. Ця функція Watir допомагає автоматизації додатка без надмірності коду. Без відкриття браузера деталі отримуються в командному рядку, що підтримує виконання тестових кейсів інтерфейсу користувача в командному рядку.

```
<html>
   <head>
      <title>A sample test of UI using Watir</title>
   </head>

   <body>
      <script type = "text/JavaScript">
         function value_test() {
            console.log("value_test");
            var my_Firstname = document.getElementById("my_Firstname");
            if (my_Firstname.value!= "") {
               document.getElementById("my_Firstname").innerHTML =
               "My First Name is  : " + my_Firstname.value;
                document.getElementById("disp_my_Firstname").
                style. display = "";
                }
            }
      </script>
<p>
<div id = "div_my_Firstname">
   Please Enter Your First Name :
   <input type = "text" id = "my_Firstname" name = "my_Firstname"
    onchange = " value_test ()" />
 </div></p>
<p>
<div style = "display:none;" id = "display_my_Firstname">
</div></p>
</body>
</html>
```

Рисунок 11 HTML, а функція JavaScript

Методи використання штучного інтелекту в тестуванні програмного забезпечення

У сучасній розробці програмного забезпечення забезпечення якості є критично важливим етапом. Впровадження штучного інтелекту у процеси тестування відкриває нові можливості для підвищення ефективності та точності.

Статистичні дані про впровадження ШІ в тестування

Згідно з дослідженням IDC Software Survey 2023, 48% підприємств вважають перевірку програмного коду та тестування одними з найважливіших завдань, у яких ШІ може ефективно допомагати розробникам. Крім того, за даними Всесвітнього звіту про якість за 2023 рік, 77% організацій інвестують у рішення на основі ШІ для підтримки своїх зусиль у забезпеченні якості.

Переваги використання ШІ в тестуванні

1. Підвищення точності виявлення дефектів: ШІ аналізує великі обсяги даних, виявляючи закономірності та потенційні проблемні області, що дозволяє тестувальникам ефективніше відстежувати та виявляти дефекти.

2. Скорочення часу тестування: ШІ виконує тести за кількома сценаріями за короткий час, значно скорочуючи цикл тестування та прискорюючи вихід продукту на ринок.

3. Автоматична генерація тест-кейсів: Алгоритми машинного навчання аналізують вимоги та існуючі тестові сценарії, автоматично створюючи нові тест-кейси, що забезпечує ширше покриття та зменшує ймовірність пропуску критичних помилок.

4. Прогнозування дефектів: ШІ аналізує історичні дані про дефекти та визначає області коду, найбільш схильні до помилок, що дозволяє командам тестувальників зосередити зусилля на критичних компонентах системи.

Підходи до використання ШІ в гнучких методологіях

Гнучкі методології розробки ПЗ, такі як Agile, передбачають швидкі ітерації та часті релізи. Інтеграція ШІ в такі процеси може відбуватися наступними способами:

1. Автоматична генерація тест-кейсів: ШІ аналізує вимоги та існуючі тестові сценарії, автоматично створюючи нові тест-кейси, що забезпечує ширше покриття та зменшує ймовірність пропуску критичних помилок.

2. Прогнозування дефектів: Алгоритми машинного навчання аналізують історичні дані про дефекти та визначають області коду, найбільш схильні до помилок, що дозволяє командам тестувальників зосередити зусилля на критичних компонентах системи.

3. Тестування графічного інтерфейсу користувача (GUI): Технології комп'ютерного зору автоматизують перевірку інтерфейсу, виявляючи візуальні дефекти та перевіряючи відповідність дизайну, що підвищує якість кінцевого продукту.

Приклади використання ШІ в тестуванні

- Автоматизація регресійного тестування;
- Оптимізація тестування продуктивності.

Впровадження штучного інтелекту в регресійному тестуванні

Регресійне тестування є невід'ємною частиною процесу забезпечення якості програмного забезпечення, метою якого є перевірка функціональності системи після внесення змін до коду. Це дозволяє гарантувати, що нові зміни не викликають негативного впливу на існуючі функціональні можливості. Інтеграція штучного інтелекту (ШІ) в регресійне тестування сприяє значному підвищенню ефективності цього процесу, зменшуючи час і ресурси, необхідні для його реалізації.

Переваги використання ШІ в регресійному тестуванні

1. Автоматизація оновлення тест-кейсів

ШІ дозволяє автоматично аналізувати зміни у кодовій базі та відповідно оновлювати або створювати нові тест-кейси. Завдяки цьому знижується залежність від ручного втручання, що істотно прискорює процес тестування.

Практичний приклад: У мобільному додатку для онлайн-банкінгу впровадження ШІ дозволило створювати автоматизовані тест-кейси для перевірки нових функцій, таких як платіжні шаблони. Алгоритм генерував сценарії для перевірки відповідності формату введення даних, обробки помилкових значень і коректності транзакцій.

2. Пріоритизація тестів

Завдяки алгоритмам машинного навчання, ШІ може оцінювати зміни в системі та визначати, які тест-кейси мають бути виконані в першу чергу. Такий підхід забезпечує ефективніше використання ресурсів і швидше виявлення критичних помилок.

Практичний приклад: У веб-додатку для управління проєктами система ШІ пріоритизувала тестування функцій синхронізації з календарями, що зменшило час на перевірку модулів із високою вірогідністю виникнення дефектів.

3. Аналіз результатів і виявлення аномалій

Використовуючи потужні алгоритми аналізу даних, ШІ здатний визначати аномалії у результатах тестування, які можуть вказувати на приховані дефекти. Це забезпечує додатковий рівень захисту від помилок, які могли б залишитися непоміченими під час ручного аналізу.

Практичний приклад: У платформі для електронної комерції ШІ ідентифікував аномальні відхилення у відображенні фільтрів товарів, що дозволило вчасно виявити проблему з оновленням бази даних.

Практичне впровадження ШІ в регресійному тестуванні

Інтеграція інструментів на базі ШІ у процес регресійного тестування надає компаніям можливість значно оптимізувати свої бізнес-процеси.

Приклад впровадження: У розробці веб-платформи ШІ виконував аналіз коду після кожного релізу, визначаючи зони змін і генеруючи відповідні тест-кейси. Наприклад, якщо було додано новий модуль для обробки замовлень, ШІ створював сценарії для перевірки коректності обробки даних, інтеграції з платіжними системами та генерації квитанцій. Це скоротило час тестування на 30%, підвищивши виявлення дефектів на 20%.

Наукова новизна підходу застосування штучного інтелекту в тестуванні програмного забезпечення

Впровадження штучного інтелекту у процеси тестування програмного забезпечення представляє собою інноваційний підхід, що значно відрізняється від традиційних методів забезпечення якості. Ця новизна проявляється у кількох ключових аспектах:

Інтелектуальна автоматизація тестування: Традиційні методи автоматизації базуються на жорстко визначених сценаріях, які потребують постійного оновлення та підтримки. ШІ, зокрема методи машинного навчання, дозволяють створювати адаптивні системи тестування, які самостійно навчаються на основі попередніх результатів та можуть генерувати нові тест-кейси без втручання людини. Це забезпечує більш гнучкий та ефективний процес тестування.

Прогнозування та виявлення дефектів: Використання ШІ для аналізу історичних даних про дефекти дозволяє прогнозувати потенційні проблеми в коді ще до їх появи. Це дає можливість проактивно виявляти та усувати вразливості, підвищуючи надійність та безпеку ПЗ.

Адаптивність до змін: ШІ-системи можуть автоматично адаптуватися до змін у коді або вимогах, що зменшує потребу в ручному оновленні тестових сценаріїв. Це особливо важливо в умовах гнучких методологій розробки, де зміни відбуваються часто та швидко.

Оптимізація ресурсів: Використання ШІ дозволяє оптимізувати розподіл ресурсів, зменшуючи навантаження на тестувальників та скорочуючи час, необхідний для проведення тестування. Це сприяє зниженню витрат та підвищенню ефективності процесу розробки.

Висновки

Створення якісного програмного забезпечення для клієнтів у зазначений термін, враховуючи всі вимоги, є важливим завданням. Використання традиційного підходу до тестування не завжди є ефективним. Існує безліч інструментів автоматизації для виконання тестування програмного забезпечення. Техніки штучного інтелекту мають значний вплив на різні етапи розробки програмного забезпечення, включаючи тестування. Застосування ШІ в автоматизації тестування є відповідним рішенням для програмного тестування. Активності для створення програмного забезпечення без дефектів. У цій статті розглянуто роль інструментів штучного інтелекту в тестуванні програмного забезпечення.

У статті також досліджено різні типи технік тестування для валідації та перевірки програмної системи. Вибір відповідних інструментів автоматизації тестування на основі штучного інтелекту важливий залежно від типу тестування, і в статті обговорюються особливості популярних інструментів тестування на основі ШІ.

Нарешті, в статті представлені переваги застосування інструментів тестування на основі ШІ в тестуванні програмного забезпечення. Розширення цієї наукової роботи буде представлено шляхом оцінки інструментів автоматизації тестування на основі ШІ з урахуванням більш детальних технічних аспектів. Покращений практичний аспект буде охоплений реальним впровадженням автоматизації тестування білого ящика з використанням мобільного додатку.

Література

1. Introducing ChatGPT and Whisper APIs. https://openai.com/blog/introducing-chatgpt-and-whisper-apis.

2. Automating and Optimizing Software Testing using Artificial Intelligence Techniques 2021. https://thesai.org/Downloads/Volume12No5/Paper_71-Automating_and_Optimizing_Software_Testing.pdf

3. Page object models. Режим доступу: https://www.selenium.dev/documentation/test_practices/encouraged/page_object_models/

4. Watir documentation. http://watir.com/

5. Мартін Р. Чистий код: Створення, аналіз і рефакторинг. Київ: Видавництво "Оріон", 2019, с. 455

6. Річардсон А. Selenium Simplified: Посібник з автоматизації веб-тестування. London: Compendium Developments, 2012, с. 380

7. Месзарош Г. xUnit Test Patterns: Refactoring Test Code. Boston: Addison-Wesley, 2007, с. 650

8. Гомер С., Бассі Б., Брук П., Шу А. Автоматизація тестування програмного забезпечення: Методи, інструменти та підходи. Київ: Видавництво "Вільямс", 2011.

9. Деніел Сітунаяке, Джені Планкет, “AI at the age” 2023, с. 150-200.

10. Майк Конб, “Оцінювання і планування в Agile 2019, с. 336.

11. Арон Аксельрод, “Complite guide to test automation”, с. 600, 2018.

12. Ліза Кріспін, “Agile testing: A practical guide for testers and agile teams” 2008, с. 521

13. Сила штучного інтелекту в тестуванні продуктивності додатк. https://www.applicationperformancetesting.com/uk/the-power-of-ai-in-application-performance-testing.html

14. Використання штучного інтелекту для спрощеного тестування програмного забезпечення в CI/CD конвеєрах. https://peerdh.com/uk/blogs/programming-insights/harnessing-artificial-intelligence-for-streamlined-software-testing-in-ci-cd-pipelines

References:

1. Introducing ChatGPT and Whisper APIs. https://openai.com/blog/introducing-chatgpt-and-whisper-apis.

2. Automating and Optimizing Software Testing using Artificial Intelligence Techniques, 2021. https://thesai.org/Downloads/Volume12No5/Paper_71-Automating_and_Optimizing_Software_Testing.pdf.

3. Page Object Models. Access mode: https://www.selenium.dev/documentation/test_practices/encouraged/page_object_models/.

4. Watir Documentation. http://watir.com/.

5. Martin R. Clean Code: A Handbook of Agile Software Craftsmanship. Kyiv: Orion Publishing, 2019, p. 455.

6. Richardson A. Selenium Simplified: A Guide to Web Testing Automation. London: Compendium Developments, 2012, p. 380.

7. Meszaros G. xUnit Test Patterns: Refactoring Test Code. Boston: Addison-Wesley, 2007, p. 650.

8. Homer S., Bassi B., Brooke P., Hsu A. Software Testing Automation: Methods, Tools, and Approaches. Kyiv: Williams Publishing, 2011.

9. Situnayake D., Plunkett J. AI at the Age, 2023, p. 150-200.

10. Mike Cohn. Agile Estimating and Planning, 2019, p. 336.

11. Aron Axelrod. Complete Guide to Test Automation, 2018, p. 600.

12. Lisa Crispin. Agile Testing: A Practical Guide for Testers and Agile Teams, 2008, p. 521.

13. The Power of AI in Application Performance Testing. https://www.applicationperformancetesting.com/uk/the-power-of-ai-in-application-performance-testing.html.

14. Harnessing Artificial Intelligence for Streamlined Software Testing in CI/CD Pipelines. https://peerdh.com/uk/blogs/programming-insights/harnessing-artificial-intelligence-for-streamlined-software-testing-in-ci-cd-pipelines.

Abstract. *This paper examines the theoretical aspects of applying artificial intelligence (AI) to automate software testing. Given the rapidly increasing complexity of software systems and the need to accelerate development cycles, test automation has become critically important. The article reviews current technologies and approaches, such as using machine learning to predict defects, applying computer vision for user interface testing, and generating test cases through neural networks. Special attention is paid to analyzing the advantages and disadvantages of each method, as well as the theoretical possibilities of integrating these technologies into software testing processes. Finally, the prospects and future directions of AI development in the field of test automation are discussed. The findings demonstrate the significant potential of AI to enhance the quality and efficiency of testing processes, making this area promising for further research and implementation.*

Keywords: *testing, test automation, AI, test documentation, testing optimization*