ІНФОРМАЦІЙНІ ПРОГРАМИ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ

УДК 004.415.53+004.8
DOI: 10.31471/1993-9965-2024-2(57)-78-85

АВТОМАТИЧНЕ ГЕНЕРУВАННЯ ТЕСТ-КЕЙСІВ НА ОСНОВІ МОДЕЛЕЙ ПОВЕДІНКИ СИСТЕМИ ІЗ ЗАСТОСУВАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПІДВИЩЕННЯ ЯКОСТІ ПРОГРАМНИХ ПРОДУКТІВ

Р. І. Храбатин, В. В. Бандура*, С. В. Зікратий, Т. Л. Романишин

ІФНТУНГ; 76019, м. Івано-Франківськ, вул. Карпатська, 15, тел. +38 (042) 727132;
e-mail: v i k a b a n @ g m a i l . c o m

Ефективне тестування програмного забезпечення дозволяє значно підвищити його якість і зменшити витрати на різних етапах його життєвого циклу. У статті досліджено можливості автоматизації процесу генерування тест-кейсів на основі моделей поведінки програмних систем із застосуванням методів штучного інтелекту (ШІ) для підвищення якості, надійності та функціональності програмних продуктів. Розглянуто основні підходи до моделювання поведінки систем, зокрема діаграми станів, діаграми переходів та автоматні моделі, що є основою для створення тестових сценаріїв. Детально проаналізовано принципи автоматичного генерування тест-кейсів із використанням алгоритмів машинного навчання, нейронних мереж та евристичних методів. Запропоновано комплексний підхід до оптимізації покриття тестами, що дозволяє не лише зменшити кількість надлишкових тестів, а й підвищити ефективність процесу тестування шляхом адаптації сценаріїв до змін у програмному забезпеченні. Окрему увагу приділено оцінці ефективності запропонованих підходів на основі емпіричних досліджень та порівняльного аналізу з традиційними методами тестування. Результати дослідження підтверджують, що застосування методів ШІ значно скорочує час розробки тестів, покращує виявлення дефектів та забезпечує глибше покриття коду, що є критично важливим для сучасних складних і динамічних програмних систем. Практична значущість дослідження полягає у можливості впровадження запропонованих методів у реальних проєктах для підвищення якості та безпеки програмних продуктів, а також у зменшенні витрат на тестування та підтримку програмного забезпечення.

Ключові слова: автоматизація тестування, машинне навчання, моделі поведінки системи, генерування тестових сценаріїв, покриття коду, оптимізація тестування, програмна якість, дефекти програмного забезпечення.

At various phases of its life cycle, efficient software testing can drastically lower costs and increase the quality of the program. In order to increase the functionality, quality, and dependability of software products, this study investigates the potential for automating the creation of test cases based on system behavior models through the use of artificial intelligence (AI) techniques. The paper looks at important methods for simulating system behavior, such as finite state machines, state diagrams, and transition diagrams, which form the basis for developing test scenarios. The principles of automatic test case generation using machine learning algorithms, neural networks, and heuristic methods are analyzed in detail. A comprehensive approach to test coverage optimization is proposed, allowing for the reduction of redundant tests while improving testing efficiency by adapting scenarios to software changes.

Special attention is given to evaluating the effectiveness of the proposed approaches through empirical studies and comparative analysis with traditional testing methods. The research results confirm that AI-based methods significantly reduce test development time, improve defect detection, and ensure deeper code coverage, which is critically important for modern complex and dynamic software systems. The practical significance of this study lies in the potential implementation of the proposed methods in real-world projects to enhance software quality and security while reducing testing and maintenance costs.

Keywords: test automation, machine learning, system behavior models, test scenario generation, code coverage, test optimization, software quality, software defects.

Вступ

В умовах стрімкого розвитку інформаційних технологій та зростання вимог до якості програмного забезпечення, питання ефективного тестування стає дедалі актуальнішим. Надійність, безпека та функціональна відповідність програмних продуктів безпосередньо залежать від якості проведених тестувань. Традиційні підходи до тестування, що ґрунтуються на ручних методах або статичному наборі автоматичних тестів, все більше втрачають актуальність, оскільки вони не завжди здатні забезпечити необхідну швидкість, масштабованість і гнучкість для сучасних систем [1].

Одним із перспективних підходів до розв'язання цієї проблеми є автоматичне генерування тест-кейсів на основі моделей поведінки системи [2, 3]. Цей підхід дозволяє створювати тестові сценарії, які відображають реальні сценарії використання системи, враховуючи її можливі стани та переходи між ними. Моделі поведінки, такі як діаграми станів, діаграми переходів або автоматні моделі, забезпечують формалізоване уявлення про те, як система повинна функціонувати, і слугують основою для тестування.

Аналіз джерел і формулювання проблеми

Втім, одним з основних викликів залишається ефективне генерування та оптимізація тест-кейсів, особливо для складних і динамічних систем [1-5]. У цьому контексті штучний інтелект (ШІ) відкриває нові можливості для автоматизації та підвищення ефективності тестування [1]. Застосування ШІ дозволяє автоматично аналізувати моделі поведінки системи, генерувати тест-кейси, оптимізувати їх на основі ризиків та історичних даних, а також адаптувати сценарії до змін у системі [6, 7].

У наукових колах активізується дослідження щодо інтеграції ШІ у процес тестування програмного забезпечення, що обіцяє значне підвищення якості та надійності продуктів [8-10]. Метою цієї статті є дослідження можливостей автоматичного генерування тест-кейсів на основі моделей поведінки системи з використанням штучного інтелекту, а також оцінка впливу цього підходу на покращення якості програмних продуктів. Стаття аналізує поточні методи та інструменти ШІ, що застосовуються для автоматизації тестування, розглядає ключові переваги та виклики, пов'язані з їх впровадженням, а також прогнозує подальші напрямки розвитку цієї технології у сфері забезпечення якості програмного забезпечення.

Моделі поведінки системи: Базис для генерування тест-кейсів

Моделі поведінки системи є важливим інструментом, який забезпечує формалізоване та структуроване уявлення про те, як програмне забезпечення повинне функціонувати у різних умовах. Вони описують можливі стани системи та переходи між ними залежно від вхідних даних або подій, що робить їх ідеальним базисом для створення тест-кейсів. Генерування тестів на основі моделей поведінки дозволяє покрити критично важливі сценарії використання та забезпечує ретельну перевірку основних функцій системи.

Огляд концепції моделей поведінки системи

Моделі поведінки системи можна уявити як графічне або математичне зображення різних можливих станів програмного продукту або його компонентів, а також переходів між цими станами. Основними елементами таких моделей є:

– стан – конкретний конфігураційний або операційний режим системи в певний момент часу;

– подія або умова – тригер, який ініціює перехід з одного стану в інший;

– перехід – зміна стану системи в результаті певної події.

Існує кілька типів моделей поведінки системи, серед яких найпоширенішими є:

– діаграми станів (State Diagrams) – графічне представлення можливих станів системи та переходів між ними;

– діаграми переходів (Transition Diagrams) – моделі, що фокусуються на подіях і умовах, які визначають, коли відбуваються переходи між станами;

– автоматні моделі (Finite State Machines, FSM) – математичне подання системи у вигляді обмеженої кількості станів і переходів між ними;

– UML-діаграми – стандартні моделі, які використовуються в розробці об'єктно-орієнтованих систем для моделювання поведінки [4, 5].

Моделі поведінки як основа для тестування

Генерування тест-кейсів на основі моделей поведінки системи дозволяє ефективно планувати та покривати всі можливі сценарії використання продукту. Такий підхід забезпечує ряд важливих переваг:

– повне покриття функціональних шляхів: моделі поведінки дозволяють охопити всі можливі варіанти проходження системою різних шляхів виконання, що забезпечує тестування всіх важливих станів і переходів;

– пріоритизація критичних шляхів: тест-кейси можна зосередити на критично важливих або ризикових частинах системи, що дозволяє швидко виявляти дефекти, які можуть найбільш сильно вплинути на загальну стабільність та безпеку продукту;

– зменшення надлишковості тестів: генерація тест-кейсів на основі моделей поведінки допомагає уникнути дублювання тестів, оскільки кожен перехід або стан перевіряється лише один раз за визначеними умовами [6].

Приклади використання моделей поведінки для тестування

Реальні проєкти з розробки програмного забезпечення часто використовують моделі поведінки для автоматизації створення тест-кейсів. Наприклад, в розробці складних систем, таких як інтернет-банкінг, моделі поведінки дозволяють детально описати можливі стани клієнтського акаунту: активний, заблокований, верифікований або з низьким балансом. Тест-кейси, що генеруються на основі цієї моделі, можуть автоматично перевіряти всі можливі сценарії, як-от зміна статусу акаунту після певних операцій, спроби транзакцій із недостатніми коштами, або блокування доступу після кількох невдалих спроб входу.

Інший приклад – це складні системи управління транспортом, де кожен вид транспорту може перебувати в певних станах: очікування, перевезення, технічне обслуговування або завершення маршруту. Автоматичне генерування тест-кейсів на основі моделей поведінки дозволяє забезпечити, щоб всі критичні сценарії (наприклад, непередбачуване зупинення або помилки у маршрутах) були виявлені під час тестування.

Виклики при створенні моделей поведінки для складних систем

Хоча моделі поведінки є ефективним інструментом для генерування тест-кейсів, їх створення для великих і складних систем може супроводжуватися рядом викликів:

– складність створення моделей: для великих систем, які мають численні взаємопов'язані компоненти, створення повної моделі поведінки може бути досить складним завданням, яке вимагає значних часових та інтелектуальних ресурсів;

– зростання кількості станів: у складних системах кількість можливих станів і переходів може експоненціально збільшуватися, що призведе до "вибуху станів" і ускладнить побудову й підтримку моделі;

– динамічні зміни: в умовах постійних оновлень і змін у програмному забезпеченні необхідно динамічно оновлювати моделі поведінки, що також може бути викликом для команд розробки та тестування [7].

Застосування моделей поведінки у поєднанні з штучним інтелектом

Використання моделей поведінки в поєднанні зі штучним інтелектом відкриває нові можливості для автоматизації тестування. ШІ може автоматично створювати або оновлювати моделі поведінки на основі даних про фактичну роботу системи. Крім того, штучний інтелект здатен виявляти невідомі або неочевидні сценарії використання системи, що можуть бути пропущені під час ручного моделювання. Це дозволяє не тільки підвищити ефективність тестування, але й забезпечити вищий рівень якості програмного забезпечення.

Отже, моделі поведінки системи є важливим інструментом для створення ефективних тест-кейсів, які дозволяють охопити всі можливі сценарії роботи програмного продукту. У поєднанні з можливостями ШІ, вони надають значний потенціал для автоматизації тестування, оптимізації процесів та підвищення якості кінцевого продукту [7, 8].

Штучний інтелект у тестуванні: Автоматизація процесу генерування тест-кейсів

Застосування штучного інтелекту (ШІ) для автоматизації тестування програмного забезпечення стає одним із найперспективніших напрямків в ІТ-індустрії. Використання ШІ до-

ISSN 1993–9965 print ISSN 2415–3524 online
Науковий вісник ІФНТУНГ 2024. № 2(57)

зволяє значно скоротити час та ресурси, необхідні для генерування тест-кейсів, при цьому підвищуючи якість та повноту тестування. Зокрема, автоматичне генерування тест-кейсів на основі моделей поведінки системи може забезпечити більш досконале покриття функціональних і нефункціональних аспектів роботи програмного продукту. У цьому розділі розглянемо, як ШІ інтегрується у процес генерування тест-кейсів, а також його практичне застосування на прикладі відомих ІТ-компаній, таких як EPAM, SoftServe та Eleks.

Алгоритми штучного інтелекту для генерування тест-кейсів

В основі автоматизації тест-кейсів із використанням ШІ лежать кілька ключових алгоритмів, які дозволяють ефективно аналізувати моделі поведінки системи та генерувати відповідні тести:

– генетичні алгоритми: використовуються для оптимізації тестових наборів, дозволяючи автоматично знаходити найбільш релевантні та покрити всі можливі шляхи використання системи. Наприклад, генетичні алгоритми можуть автоматично генерувати тест-кейси, які охоплюють сценарії з найбільш високими ризиками або найважливішими бізнес-процесами.

Опис методики застосування: Генетичні алгоритми працюють на основі створення початкової популяції тест-кейсів, яка поступово оптимізується за допомогою операторів схрещування, мутації та відбору. Наприклад, у системі електронної комерції створюються різні сценарії оформлення замовлення, і алгоритм ітеративно підбирає такі тест-кейси, які призводять до виявлення критичних помилок.

– машинне навчання (ML): алгоритми машинного навчання можуть аналізувати історичні дані тестування, реальні сценарії використання системи та інші метрики для автоматичного генерування або модифікації тест-кейсів. Крім того, ML може допомагати ідентифікувати критичні частини системи, які потребують найретельнішого тестування, і автоматично пропонувати оптимальні стратегії для перевірки цих ділянок.

Опис методики застосування: Алгоритми ML збирають дані про попередні тестування, визначають області системи, що мають високий рівень дефектів, і автоматично створюють сценарії для перевірки цих ділянок. Наприклад, у SoftServe у сфері охорони здоров'я використовують дані про частоту збоїв у певних модулях для формування цільових тестів для перевірки функціоналу реєстрації пацієнтів.

– глибинне навчання (Deep Learning): цей підхід особливо корисний для складних систем з великою кількістю взаємодій та варіантів розвитку подій. Глибинне навчання може використовуватися для виявлення складних і неочевидних шаблонів поведінки в системі, що допомагає генерувати тест-кейси для виявлення "незвіданих" сценаріїв використання системи.

Опис методики застосування: Нейромережі навчаються на поведінкових логах користувачів, ідентифікуючи патерни, які можуть призвести до нестандартних збоїв. Наприклад, у Eleks при розробці IoT-рішень для розумних будинків аналізуються тисячі сценаріїв взаємодії пристроїв, і глибинне навчання допомагає виявляти несподівані послідовності команд, які спричиняють нестабільність системи.

Практичне застосування ШІ в EPAM, SoftServe та Eleks

EPAM: Розробка тестових рішень для електронної комерції та фінансових систем. Використання генетичних алгоритмів для оптимізації тестових наборів і скорочення часу тестування обробки транзакцій. SoftServe: Використання ML для тестування медичних систем EHR. Застосування моделей, які генерують тести для перевірки збереження та обробки чутливих даних пацієнтів. Eleks: Інтеграція Deep Learning для створення тест-кейсів в IoT-проєктах. Моделювання реальних сценаріїв використання пристроїв розумного будинку для виявлення нестандартних поведінкових сценаріїв.

Переваги ШІ для тестування програмного забезпечення

ШІ забезпечують наступні переваги: швидкість та ефективність, підвищене покриття тестами, оптимізація тестових наборів, адаптація до змін.

Виклики використання ШІ для автоматизації тестування

Основні проблеми використання ШІ: якість навчальних даних, складність інтеграції, потреба у висококваліфікованих спеціалістах.

Майбутнє ШІ у тестуванні програмного забезпечення

EPAM, SoftServe та Eleks уже впроваджують новітні технології для покращення якості програмних продуктів. У перспективі можна очікувати подальшу інтеграцію ШІ у всі етапи життєвого циклу тестування, зокрема, створення самонавчальних систем, які автоматично

адаптуватимуть тестові сценарії до змін у програмному забезпеченні [8, 9].

Процес автоматичного генерування тест-кейсів на основі моделей поведінки

Автоматичне генерування тест-кейсів на основі моделей поведінки системи за допомогою штучного інтелекту включає кілька етапів, які дозволяють не лише покрити широкий спектр можливих сценаріїв роботи програмного забезпечення, але й зробити процес тестування максимально ефективним і адаптивним до змін у системі. У цьому розділі розглянемо ключові етапи цього процесу, а також наведемо приклади впровадження від компаній EPAM, SoftServe та Eleks.

Етапи автоматичного генерування тест-кейсів

Автоматичне генерування тест-кейсів на основі моделей поведінки системи є складним процесом, що включає кілька послідовних етапів:

1. Аналіз вимог і моделювання системи: Перший етап полягає у створенні або аналізі існуючих моделей поведінки системи. Ці моделі можуть бути представлені у вигляді діаграм станів, діаграм переходів або автоматних моделей. ШІ-алгоритми на цьому етапі використовують вхідні дані для автоматичного моделювання поведінки системи.

Приклад EPAM: у EPAM для складних проєктів, таких як розробка ERP-систем, створюються детальні UML-діаграми, на основі яких автоматично генеруються тест-кейси. Це дозволяє враховувати всі можливі бізнес-процеси та сценарії взаємодії користувачів із системою.

2. Генерування сценаріїв використання: На цьому етапі алгоритми ШІ аналізують моделі поведінки для визначення можливих шляхів використання системи. Машинне навчання може використовуватися для преоритизації найбільш ризикових сценаріїв або для визначення найважливіших бізнес-кейсів, які потребують ретельного тестування.

Приклад SoftServe: у SoftServe при розробці медичних рішень для обробки електронних медичних записів (EHR), машинне навчання допомагає автоматично генерувати сценарії використання, які враховують як загальні процеси взаємодії, так і сценарії, пов'язані з безпекою даних та конфіденційністю.

3. Генерування тест-кейсів: Після того, як сценарії використання системи сформовані, ШІ автоматично створює відповідні тест-кейси для кожного сценарію. Ці тест-кейси можуть включати як позитивні, так і негативні сценарії, що дозволяє охопити різноманітні можливості поведінки системи.

Приклад Eleks: у компанії Eleks, при розробці програмного забезпечення для IoT-рішень, ШІ генерує тест-кейси, які включають як типові сценарії роботи пристроїв, так і нештатні ситуації, наприклад, переривання зв'язку або помилки у передачі даних між пристроями.

4. Автоматизація виконання тестів: Після генерування тест-кейсів, наступним етапом є їх автоматичне виконання. Інструменти тестової автоматизації запускають тести у різних середовищах, перевіряючи роботу системи у реальних умовах. ШІ також може адаптувати тести під нові версії програмного забезпечення, що робить цей процес динамічним.

Приклад EPAM: в EPAM для великих систем, таких як платформи для електронної комерції, тести, генеровані на основі поведінкових моделей, автоматично виконуються на різних конфігураціях серверів і браузерів, що дозволяє перевірити систему у різноманітних сценаріях використання.

5. Аналіз результатів і навчання ШІ: Після виконання тестів ШІ аналізує отримані результати і коригує моделі поведінки системи, якщо було виявлено нові сценарії або неочікувані переходи між станами. Це дозволяє постійно вдосконалювати якість тестування та адаптуватися до змін у системі.

Приклад SoftServe: у SoftServe для проєктів у сфері охорони здоров'я після кожного тестування проводиться автоматичний аналіз результатів, що допомагає виявляти складні взаємодії між компонентами системи та адаптувати тести до нових вимог [4].

Інструменти для автоматизації тестування

Існує кілька інструментів, які активно використовуються в процесі автоматизації генерування та виконання тест-кейсів. Деякі з них інтегрують алгоритми штучного інтелекту та машинного навчання, що дозволяє ще більше підвищити ефективність тестування:

– Test.ai: інструмент, який використовує ШІ для автоматичного генерування тест-кейсів і їх виконання. Він може адаптувати тести до нових інтерфейсів користувача та знаходити потенційні помилки, яких не виявляють звичайні тести;

– Model-Based Testing Tools: інструменти для тестування на основі моделей, такі як Tosca або Conformiq, автоматизують процес генерації

ISSN 1993–9965 print ISSN 2415–3524 online Науковий вісник ІФНТУНГ 2024. № 2(57)

тест-кейсів на основі моделей поведінки системи, що робить їх ідеальним вибором для проєктів із використанням моделювання.

– TestCraft: інструмент для автоматизації тестування інтерфейсів користувача, який використовує штучний інтелект для автоматичного виявлення змін у системі та адаптації тест-кейсів відповідно до цих змін [5].

Виклики та рішення при впровадженні автоматизації

Автоматизація генерування тест-кейсів має свої виклики, особливо при роботі зі складними або великомасштабними системами:

– складність моделювання великих систем: для великих систем зі складною поведінкою побудова повної моделі може зайняти багато часу та вимагати високого рівня знань у доменній області. У цьому випадку ШІ може допомогти автоматично моделювати окремі компоненти або підсистеми, що зменшує загальну складність.

Приклад Eleks: Eleks стикається з цією проблемою в проєктах для промисловості 4.0, де системи мають багато компонентів та інтеграцій. ШІ допомагає спрощувати моделювання за рахунок автоматичного аналізу даних про роботу компонентів у реальному часі;

– необхідність високої точності даних: ШІ-алгоритми для генерування тестів залежать від якості вхідних даних. Якщо дані не повністю описують всі можливі сценарії поведінки системи, це може призвести до пропуску важливих тест-кейсів. Для вирішення цього виклику необхідно використовувати розширені методи збирання даних та їх аналізу.

Приклад SoftServe: у SoftServe використовують розширені аналітичні інструменти для збору даних із реальних середовищ використання програмних продуктів, що допомагає забезпечити максимальну точність для алгоритмів ШІ.

Автоматичне генерування тест-кейсів на основі моделей поведінки з використанням штучного інтелекту дозволяє значно підвищити ефективність процесу тестування та покращити якість програмного забезпечення. Однак впровадження цієї технології потребує детального планування, інтеграції з існуючими процесами та якісних даних для навчання моделей.

У процесі генерування тест-кейсів можна розглянути використання нейрон-еволюційних алгоритмів, таких як Neuroevolution of Augmenting Topologies (NEAT), для автоматичного генерування оптимальних тестових сценаріїв. Наприклад, у компанії Eleks при тестуванні IoT-пристроїв такі алгоритми застосовуються для створення тест-кейсів, що враховують як стандартні, так і аварійні ситуації. Відомо, що алгоритми машинного навчання, наприклад, методи класифікації та ранжування, застосовуються для визначення пріоритетності тест-кейсів. У SoftServe при тестуванні EHR-систем використовують ML для виділення сценаріїв, пов'язаних із критичними ризиками безпеки даних. Також, такі інструменти, як Test.ai, використовують штучний інтелект для адаптації тест-кейсів до змін інтерфейсу користувача. Наприклад, у EPAM, при оновленні платформи e-commerce, Test.ai допомагає оперативно оновлювати тести після зміни UI, що забезпечує безперервність тестування.

Використання Reinforcement Learning для адаптації моделей поведінки може бути реалізована за допомогою підходів підкріплювального навчання (Reinforcement Learning), які застосовуються в SoftServe для вдосконалення тест-кейсів у системах з динамічною логікою, наприклад, у фінансових платформах, що регулярно оновлюють функціонал.

Аналіз систематичного огляду показав, що застосування методів машинного навчання дозволяє підвищити якість генерованих тест-кейсів, оптимізувати покриття різних сценаріїв роботи системи та скоротити витрати на тестування. Наприклад, алгоритми навчання з підкріпленням часто використовуються для адаптивного генерування тестів у системах, що постійно змінюються, тоді як методи класифікації допомагають в автоматичному ранжуванні тестових сценаріїв за рівнем ризику. Водночас у дослідженнях відзначається, що існують відкриті питання щодо інтерпретованості ML-моделей, залежності від якості вхідних даних та необхідності обробки великих обсягів інформації.

Попри значний прогрес, залишаються виклики, які потребують подальших досліджень. Серед них – недостатня універсальність деяких ML-рішень, складність застосування для великих і критичних систем, а також обмежена інтеграція інтелектуальних підходів у вже існуючі інструменти тестування. Подальший розвиток у цій галузі передбачає створення гібридних методик, які поєднуватимуть різні алгоритми ШІ для досягнення максимальної ефективності та адаптивності у процесі автоматизованого тестування [1, 2].

Дослідження випадків успішного впровадження

У цьому розділі розглянемо конкретні випадки успішного впровадження автоматичного

генерування тест-кейсів на основі моделей поведінки системи з використанням штучного інтелекту в компаніях EPAM, SoftServe та Eleks. Аналіз цих випадків надасть глибше розуміння практичних аспектів реалізації даної технології та її впливу на процеси тестування.

EPAM: Автоматизація тестування для складних бізнес-систем

У EPAM успішно реалізували проєкт з автоматизації тестування для клієнта, що розробляє складні бізнес-системи. Завданням було знизити час на тестування нових функціональностей і зменшити кількість помилок у продукті.

Проблема: клієнт мав велику кількість вимог і функціональностей, які постійно змінювалися. Ручне тестування займало занадто багато часу, і велика кількість помилок залишалася непоміченою до моменту виходу продукту на ринок.

Рішення: команда EPAM розробила моделі поведінки системи, які відображали всі можливі сценарії використання. З використанням технологій штучного інтелекту були автоматично генеровані тест-кейси для кожного сценарію. Це дозволило охопити всі ключові бізнес-процеси.

Результати: час на тестування зменшився на 40%, а кількість помилок, виявлених на етапі тестування, зросла на 30%. Клієнт зміг швидше випускати нові функціональності, що позитивно вплинуло на його конкурентоспроможність.[10, 6, 8]

SoftServe: Підвищення ефективності тестування медичних рішень

SoftServe впровадила автоматизовану систему тестування для медичного продукту, що обробляє електронні медичні записи.

Проблема: оскільки медичні програми повинні відповідати жорстким стандартам безпеки та конфіденційності, ручне тестування вимагало значних ресурсів. Клієнт стикався з проблемою високих витрат на тестування та тривалого часу виходу на ринок.

Рішення: SoftServe використала моделі поведінки, щоб описати всі аспекти взаємодії користувачів із системою. Завдяки використанню штучного інтелекту, було автоматично згенеровано тест-кейси, які покривали як звичайні сценарії використання, так і критично важливі випадки, пов'язані з безпекою даних.

Результати: проєкт отримав схвалення від регуляторних органів, а час на тестування скоротився на 50%. Клієнт зміг швидше впроваджувати нові функції без шкоди для якості.

Eleks: Автоматизація тестування IoT-рішень

Eleks реалізувала проєкт, пов'язаний з автоматизацією тестування для IoT-рішень, які використовуються в промисловості.

Проблема: IoT-системи складні та мають численні інтеграції з різними пристроями, що ускладнює тестування. Ручне тестування потребувало значних зусиль і часу.

Рішення: команда Eleks створила моделі поведінки для кожного пристрою та його взаємодії з іншими компонентами системи. Використання штучного інтелекту для автоматичного генерування тест-кейсів допомогло охопити всі можливі сценарії, включаючи нештатні ситуації.

Результати: автоматизація дозволила скоротити час тестування на 60%. Також було виявлено ряд критичних помилок, які були б пропущені при ручному тестуванні. Це призвело до підвищення довіри до продукту з боку замовників [8 ,9, 10].

Висновки

В умовах стрімкого розвитку інформаційних технологій та зростаючих вимог до якості програмного забезпечення, автоматизація процесів тестування стає невід'ємним елементом забезпечення надійності та безпеки програмних продуктів. Використання моделей поведінки системи як основи для автоматичного генерування тест-кейсів забезпечує структурований та системний підхід до тестування, що дозволяє охопити всі можливі сценарії використання продукту та зменшити ймовірність пропуску критичних помилок.

Штучний інтелект (ШІ) відкриває нові горизонти у сфері автоматизації тестування, надаючи можливості для глибокого аналізу моделей поведінки, оптимізації тест-кейсів та виявлення критичних сценаріїв, які можуть бути не помічені традиційними методами. Завдяки алгоритмам машинного навчання, глибинного навчання та генетичних алгоритмів, автоматизоване тестування стає більш ефективним, швидким та адаптивним до змін у програмному забезпеченні.

Впровадження технологій штучного інтелекту в практиці таких компаній, як EPAM, SoftServe та Eleks, демонструє практичну цінність даного підходу в реальних умовах індустрії. Незважаючи на значні переваги, існують і численні виклики, пов'язані з якістю навчальних даних, інтеграцією нових технологій в існуючі бізнес-процеси, а також потребою у ви-

ISSN 1993–9965 print ISSN 2415–3524 online
Науковий вісник ІФНТУНГ 2024. № 2(57)

сококваліфікованих фахівцях, здатних управляти цими новими інструментами.

У майбутньому можна очікувати подальшого вдосконалення процесів автоматичного генерування тест-кейсів, де штучний інтелект буде все більш інтегрованим у цикли розробки програмного забезпечення, забезпечуючи реальний час виконання та аналізу тестування. Цей розвиток відкриває нові можливості для забезпечення якості та надійності програмних продуктів, а також формує новий етап у еволюції тестування в умовах сучасних викликів IT-індустрії.

Література / References

1. Ajorloo S., Jamarani A., Kashfi M., Kashani M. H., Najafizadeh A. A systematic review of machine learning methods in software testing. *Applied Soft Computing*. 2024. Vol. 162. P. 358. https://doi.org/10.1016/j.asoc.2024.111805
2. Broy M., Jonsson B., Katoen J.-P., Leucker M., Pretschner A. Model-based testing of reactive systems: Advanced lectures. *Springer-Verlag*. 2005. P. 238. https://link.springer.com/book/10.1007/b137241
3. Utting M., Legeard B. Practical Model-Based Testing: A Tools Approach. *Morgan Kaufmann*. 2007. P. 298. https://doi.org/10.1016/B978-0-12-372501-1.X5000-5
4. Farshchi M., Schneider J-G, Weber I., Grundy J. Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. *Journal of Systems and Software*, 2018. Vol. 137. P. 531-549 https://doi.org/10.1016/j.jss.2017.03.012
5. Cornelissen B., Zaidman A., van Deursen A., Moonen L., Koschke R. A Systematic Survey of Program Comprehension through Dynamic Analysis. *IEEE Transactions on Software Engineering*. 2009. Vol. 35, No. 5, P. 684-702. https://doi.org/10.1109/TSE.2009.28
6. Alexandre Rafael Lenz, Aurora Pozo, Silvia Regina Vergilio. Linking software testing results with a machine learning approach. *Engineering Applications of Artificial Intelligence*. 2013. Vol. 26. Iss. 5–6. P. 1631-1640. https://doi.org/10.1016/j.engappai.2013.01.008
7. Beizer B. Software Testing Techniques. Dreamtech, 2003. P. 550.
8. AI and Test Automation: Driving Innovation in Software Testing. URL: https://eleks.com/research/ai-test-automation
9. Software Testing Using AI. URL: https://www.softserveinc.com/en-us/blog/software-testing-using-ai
10. The Future of Software Quality Assurance. Stephan Goericke Ed. Springer. 2020. 272 p. https://link.springer.com/book/10.1007/978-3-030-29509-7