УДК 004.45:004.942:[004.032.6+004.45+004.942]
DOI: https://doi.org/10.30837/IYF.CVSAMM.2024.311

ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Шкурко В.В.
Науковий керівник – канд. техн. наук, доц. Поляков А.О.
Харківський національний університет радіоелектроніки, каф. ПМ,
м. Харків, Україна
e-mail: viacheslav.shkurko@nure.ua

This scientific article explores the burgeoning field of applying artificial intelligence (AI) to software testing, aiming to enhance the efficiency and effectiveness of the software development lifecycle. The study delves into the integration of AI techniques, including machine learning, natural language processing, and automation, to streamline testing processes and address the evolving challenges in software quality assurance.

Тестування програмного забезпечення (ПЗ) – процес перевірки відповідності заявлених до продукту вимог і реалізованої функціональності, який здійснюють шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів. Це комплексний процес, який включає безліч видів активностей: аналіз вимог, створення тест-кейсів, безпосереднє проведення тестування продукту, пошук і фіксування багів. Залежно від потреб продукту, процес тестування ПЗ може включати різні види тестування, які охоплюють перевірку різного функціоналу і критично необхідних вимог: тестування зручності використання, тестування інтерфейсу, функціональне тестування і т.п.. Окремо виділяють автоматизоване тестування (що дозволяє оптимізувати процес тестування для випадків, коли необхідно обробляти великий обсяг даних і функціонал, що повторюється), так і мануальне тестування (у разі, якщо процес автоматизації стає надто складним та витратним). Залежно від різних етапів життєвого циклу існують види тестування, спрямовані на перевірку змін або впровадження нового функціоналу, так і перевірку стабільності роботи продукту у цілому (регресійне та системне тестування).

В останні роки складність програмних систем зросла, що зумовлено складною архітектурою, розгалуженою функціональністю та зростаючим попитом на швидкі випуски програмного забезпечення. У відповідь на ці виклики інтеграція штучного інтелекту (ШІ) в тестування ПЗ набула популярності: за рахунок ШІ зменшується вплив людського фактору із-за чого підвищується швидкість проходження процесу тестування та точність знаходження дефектів у автоматизованому та API тестуванні, згорткові нейронні мережі (CNN) використовуються для тестування на основі зображень або для візуальної ідентифікації дефектів, рекурентні нейронні мережі (RNN) здатні аналізувати вимоги та на їх основі генерувати тест-кейси.

Інтеграція ШІ у тестування ПЗ сприяє створенню більш спільного та гнучкого середовища розробки. Методи ШІ можуть легко інтегруватися в існуючий процес розробки ПЗ, забезпечуючи зворотний зв'язок з розробниками в режимі реального часу на етапі кодування. Така інтеграція дозволяє виявляти потенційні проблеми на ранній стадії, скорочуючи час і зусилля, необхідні для налагодження, і забезпечуючи більш ефективний процес розробки [5].

ШІ з його здатністю аналізувати величезні масиви даних, виявляти закономірності та вчитися на ітеративних процесах пропонує перспективний шлях до оптимізації та прискорення життєвого циклу тестування ПЗ. Традиційні методи тестування, хоч і ефективні, але часто забирають багато часу та ресурсів: детальний аналіз вимог, ручне створення тест-кейсів, тестування великих баз даних та об’ємні види тестування (API, регрессійне та системне). В цій роботі аналізується застосування штучного інтелекту в галузі тестування ПЗ, досліджується його значення, цілі та методології [1].

Один з ключових внесків полягає в здатності ШІ самостійно виявляти закономірності, аномалії та потенційні проблеми в коді, протиріччя у вимогах до ПЗ. Ця адаптивність спрощує взаємодію з частинами програмних додатків, що змінюються (вимоги та функціонал) і різноманітними середовищами, в яких вони працюють. З розвитком ПЗ інструменти тестування на основі ШІ розвиваються в методи та моделі, забезпечуючи безперервне підвищення точності та охоплення тестування [2].

Важливим аспектом застосування ШІ в тестуванні ПЗ є його вміння створювати реалістичні та різноманітні тестові сценарії. Завдяки створенню синтетичних даних і симуляції різних взаємодій користувачів, ШІ сприяє комплексному тестуванню, яке точно відображає реальне використання. Це підвищує надійність результатів тестування, але й допомагає виявити потенційні вразливості, які можуть виникнути в різних сценаріях тестування. Також спостерігається розвиток фреймворків для автономного тестування (ФАТ), що спостерігається у роботах [3, 4]. ФАТ використовують методи, що керуються ШІ, для автономного проектування, виконання та оптимізації тестових кейсів. Автономність ФАТ полягає в тому, що зменшує залежність від людського втручання та фокусується на процессі тестування, також робить можливим обробку вимог до тестування, як у великих так і у складних програмних проектів (ПП) [3].

Інтелектуальна автоматизація тестування сьогодні є одним з помітних застосувань ШІ в тестуванні ПЗ [1]. Традиційні інструменти автоматизації тестування часто вимагають явного написання сценаріїв і заздалегідь визначених тестових кейсів. На відміну від них, інструменти автоматизації, що були інтегровані з ШІ, мають здатність до самонавчання та адаптації, що дає їм змогу орієнтуватися в динамічних інтерфейсах і реагувати на зміни в програмному забезпеченні без необхідності постійних ручних на-

лаштувань. Така адаптивність призводить до більш надійного та стійкого автоматизованого тестування, навіть в умовах частих оновлень і змін ПЗ.

Аналізуючи пріоритетність тестових кейсів на основі історичних патернів дефектів, змін у коді та критично важливих функцій, інтеграція ШІ у процес тестування зосереджується на ділянках з найбільшим потенціалом для виявлення критичних проблем. Такий цілеспрямований підхід підвищує ефективність процесів тестування і дозволяє більш раціонально розподіляти ресурси [4]. Здатність до самооптимізації ШІ не тільки прискорює процес тестування, але й мінімізує ризик пропуску критичних дефектів, що в кінцевому підсумку призводить до створення більш якісних програмних продуктів.

Оскільки програмний ландшафт продовжує розвиватися, роль ШІ в тестуванні стає все більш незамінною. Здатність алгоритмів ШІ обробляти складні тестові сценарії, передбачати потенційні проблеми і надавати глибоку аналітику дає командам розробників можливість приймати обґрунтовані рішення, забезпечуючи створення надійних і зручних для користувача програмних рішень.

Список використаних джерел:

1. Groz R., Simao A., Bremond N., Oriat, C. Revisiting AI and Testing Methods to Infer FSM Models of Black-Box Systems // ICSE'18: 40th International Conference on Software Engineering: AST '18: Proceedings of the 13th International Workshop on Automation of Software Test (May 28 - 29, 2018). Gothenburg, Sweden, 2018. P. 16–19. http://dx.doi.org/10.1145/3194733.3194736

2. Dangeti P. Statistics for Machine Learning: Techniques for Exploring Supervised, Unsupervised, and Reinforcement Learning Models with Python and R. Birmingham, UK : Packt Publishing, 2017. 426 p.

3. Alloghani M., Al-Jumeily D., Mustafina J., Hussain A., Aljaaf A.J. A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. Supervised and Unsupervised Learning for Data Science, Cham : Springer International Publishing, 2020. P. 3–21. DOI: https://doi.org/10.1007/978-3-030-22475-2_1.

4. Gullì A., Pal S. Deep Learning with Keras: implement neural networks with Keras on Theano and TensorFlow. Birmingham Mumbai : Packt Publishing, 2017. 303 p.

5. Yasnitsky L. N. Whether Be New "Winter" of Artificial Intelligence?. In: Antipova, T. (eds) Integrated Science in Digital Age. ICIS 2019. Lecture Notes in Networks and Systems, vol. 78. Springer, Cham. P. 13–17. https://doi.org/10.1007/978-3-030-22493-6_2