DOI: 10.34185/1991-7848.itmm.2024.01.075

РЕФАКТОРИНГ КРОС-ПЛАТФОРМНИХ ЗАСТОСУНКІВ З ВИКОРИСТАНЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Сирота О. А., Горячкін В.М.

Український державний університет науки і технологій, Україна

Анотація. *У сфері розробки програмного забезпечення продуктивність рефакторингу коду збільшується завдяки використанню штучного інтелект (ШІ). У процесі рефакторингу можна використовувати такі методи ШІ, як машинне навчання (ML), обробка природної мови (NLP) та генетичні алгоритми (GA). Кожен з методів має певний вплив на процес, як позитивний так і негативний. Зважаючи на це робота, що виконується ШІ, вимагає ретельного управління, щоб уникнути ряду проблем, наприклад "галюцинацій". Дослідження продовжують вивчати нові методи, оцінювати порівняльну ефективність та оптимізувати моделі ШІ для конкретних фреймворків і мов.*

Ключові слова: *рефакторинг, розробка, код, крос-платформні застосунки, штучний інтелект, машинне навчання, обробка природної мови, генетичний алгоритм.*

У сучасному світі взаємодія з різними додатками, розробленими для різних операційних систем і платформ, стала важливою частиною життя людей. Мова йде про додатки для Web, iOS, Android, MacOS, Windows і Linux, які використовуються користувачами в різних напрямках діяльності, таких як онлайн-покупки, замовлення доставки їжі, навчання, вирішення юридичних питань, керування "розумним" будинком або автомобілем і звичайно також розваги, такі як фільми і телепередачі. Розповсюдженим підходом до розробки таких додатків є використання крос-платформних технологій, що підтримують згадані платформи і переваги цього очевидні: швидка розробка, спільна база коду для різних платформ і порівняно нижчі витрати, ніж при нативній розробці. Даний підхід в розробці приваблює бізнес, бо задовольняє нагальні потреби.

Нажаль наведених переваг недостатньо для розробки насправді якісного програмного забезпечення. Існує багато факторів які в тій чи іншій мірі супроводжують процес розробки і негативно на нього впливають: недостатній рівень майстерності розробників, невдалий менеджмент, брак необхідних ресурсів, часта зміна вимог до майбутнього продукту, тощо. Слід мати на увазі,

що крос-платформні технології самі по собі також можуть мати певні недоліки як от помилки, проблеми в підтримці деякого функціоналу притаманного одній платформі та непритаманного іншій. Вище перераховані причини в результаті призводять до значного погіршення якості застосунку на виході. Можна виділити наступні проблеми:

- вихідний код поганої якості;
- недостатній рівень продуктивності програми, що виражається в повільній роботі інтерфейсу користувача, зависаннях, замалій кількості відрисованих кадрів за одиницю часу;
- надмірне використання пам’яті під час виконання, що призводить до екстреного завершення роботи;
- реалізація нового функціоналу займає непропорційно багато часу, через складнощі розуміння наявного коду;
- ріст команди розробників не достатньо впливає або майже не впливає на швидкість розробки вцілому;
- вартість реалізації програми невпинно зростає.

Розповсюдженим інструментом покращення внутрішньої структури коду без зміни бізнес-логіки є рефакторинг. Він поліпшує читабельність, спрощує обслуговування та розширення функцій, допомагає усунути технічні заборгованості, підвищує продуктивність і сприяє дотриманню стандартів кодування. В залежності від проекту застосування такого інструменту може бути досить складними процесом, особливо якщо мова йде про застарілий код та складний програмний продукт. Сприйняття великого обсягу коду для розробника є складною задачею, особливо коли це відбувається в процесі рефакторингу, бо крім розуміння всієї картини та утримання в розумі складного та заплутаного коду необхідно думати яким чином та в яких модулях мають бути внесені зміни. Слід зазначити, що нерідко це може призвести до появи нових помилок в функціональності, проблем сумісності або створення нових неефективних структур коду. В якийсь момент це може стати непосильним завданням через те, що це потребує значних ресурсів у вигляді часу та кількості розробників. Враховуючи те, що обробка коду, виявлення проблем, їх виправлення виконується вручну і вимагає повторного тестування, а успішність процесу безпосередньо залежить від рівня професійності

програміста – це є додатковими витратами для бізнесу. В гіршому випадку проект може закритись. Проте поява штучного інтелекту знаменує зміну всієї парадигми і обіцяє більш ефективний підхід.

Штучний інтелект з його можливостями надшвидкого предиктивного аналізу та обробки великих обсягів даних може використовуватись, щоб значною мірою автоматизувати процес рефакторингу коду. Підхід на основі штучного інтелекту полягає в тому, що інструменти ШІ проводять аналіз різного за обсягом коду, виявляють архітектурні помилки, порушення загальноприйнятих принципів програмування, такі як повторювані шаблони та автоматично проводять рефакторинг або пропонують можливі рішення розробнику [0]. В залежності від налаштувань інструментарію залученість розробників може бути як мінімум двох видів:

- ітеративний контроль роботи ШІ – ознайомлення з варіантами рішень, запропонованих ШІ інструментом та погодження чи відхилення його використання;
- конфігурація вхідних та перевірка вихідних даних – надання проекту для рефакторингу та додаткових конфігурацій, що впливають на цей процес та огляд результатів роботи ШІ;

Незалежно від обраного варіанта відбувається заощадження ресурсів в значній мірі.

В даному дослідженні розглядається декілька методів ШІ, які можна використовувати в процесі рефакторингу. До них можна віднести машинне навчання (ML), обробка природної мови (NLP) і генетичні алгоритми (GA). ML – це важлива підгалузь ШІ, яка навчена на наявних даних та попередніх результатах для прогнозування потенційних можливостей та наслідків в процесі рефакторингу. Базуючись на минулих результатх, моделі ML можуть виявляти закономірності та робити обґрунтовані прогнози щодо того, де рефакторинг може знадобитися в майбутньому [0]. NLP – це технологія ШІ, яка використовується для розуміння семантики коду та генераціх рекомендацій щодо його покращення в контексті читабельності. Це стає в нагоді коли розробникам доводиться працювати з застарілими або заплутаними системами, логіку коду яких дуже важко зрозуміти людині [0].

GA – це евристичний метод пошуку, який базується на принципах природної еволюції та вміє генерувати сукупність рішень для рефакторингу, аналізувати їх і обирати найбільш оптимальне на основі попередньо визначених критеріїв. Це надає можливість забезпечити вибір найбільш корректного рішення, враховуючи такі критерії як ефективність, розуміння, зручність внесення змін до коду [0].

Гарними прикладами використання описаних вище методів слугують такі розробки як GitHub Copilot або SapFix. SapFix – це гібридний інструмент Facebook що, може скоротити час налагодження для інженерів і прискорити розгортання нового програмного забезпечення. Він може автоматично пропонувати виправлення помилок для затвердження і був використаний для прискорення оновлення коду в додатку Facebook [0]. GitHub Copilot – це асистент кодування зі штучним інтелектом, розроблений Microsoft та OpenAi, який надає підказки, допомагаючи вам писати код більш ефективно. Він може доповнювати рядки або блоки коду, надавати поради щодо вирішення проблем, пояснювати як працює наявний код та проводити рефакторинг [0].

Підсумовуючи можна сказати, що рефакторинг із застосуванням ШІ дійсно впливає на продуктивність та якість цього процесу, шляхом застосування різних алгоритмів та зменшення залученості розробників. Проте важливим є той факт, що даний інструментарій все ще має певні проблеми та помилки і вимагає контролю з боку програмістів. Існує така проблема, що зветься “ШІ-галюцинація”, яка є прикладом проблеми яку не можна ігнорувати [0]. Крім цього постають питання який з методів ШІ треба використовувати, як цей вибір залежить від типу проекту чи вимог до нього та як це впливає на кінцевий результат.

Дослідження в цьому напрямку зосереджене на вивчені можливостей нових методів рефакторингу, що формуються через появу ШІ, проведенні порівняльних досліджень для оцінки ефективності підходів та оптимізації моделей штучного інтелекту для певних фреймворків та відповідних мов програмування.

ЛІТЕРАТУРА / REFERENCE

1. IBM. 9 ways developer productivity gets a boost from generative AI. URL: https://www.ibm.com/blog/developer-productivity/
2. IBM. What is machine learning (ML)? URL: https://www.ibm.com/topics/machine-learning
3. IBM. What is natural language processing (NLP)? URL: https://www.ibm.com/topics/natural-language-processing
4. Autoblocks. Genetic algorithm (GA). URL: https://www.autoblocks.ai/glossary/genetic-algorithm
5. Engineering at Meta. Finding and fixing software bugs automatically with SapFix and Sapienz. URL: https://engineering.fb.com/2018/09/13/developer-tools/finding-and-fixing-software-bugs-automatically-with-sapfix-and-sapienz/
6. GitHub Docs. About GitHub Copilot. URL: https://docs.github.com/en/copilot/about-github-copilot
7. IBM. What are AI hallucinations? URL: https://www.ibm.com/topics/ai-hallucinations

REFACTORING CROSS-PLATFORM APPLICATIONS USING ARTIFICIAL INTELLIGENCE

Oleksandr Syrota, Horiachkin Vadym,

Abstract. *In software development, the productivity of code refactoring is increased by the use of artificial intelligence (AI). AI methods such as machine learning (ML), natural language processing (NLP), and genetic algorithms (GA) can be used in the refactoring process. Each method has a certain impact on the process, both positive and negative. With this in mind, the work performed by AI requires careful management to avoid a number of problems, such as "hallucinations". Research continues to explore new methods, evaluate comparative effectiveness, and optimize AI models for specific frameworks and languages.*

Keywords*: refactoring, development, code, cross-platform applications, artificial intelligence, machine learning, natural language processing, genetic algorithm.*