Національний університет «Київський авіаційний інститут»

Факультет комп'ютерних наук та технологій
Кафедра інженерії програмного забезпечення

Тези доповідей

XXI Міжнародної науково-практичної конференції
здобувачів вищої освіти та молодих учених

Інженерія програмного забезпечення

Київ 2026

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
National University «Kyiv Aviation Institute»
Faculty of Computer Science and Technology

Abstracts of
XXI International
conference of higher education students
and young scientists

SOFTWARE ENGINEERING CONFERENCE

Kyiv 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЇ

Тези доповідей
XXI Міжнародної
науково-практичної конференції
здобувачів вищої освіти і молодих учених

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

Київ 2026

UDC 004.4(045)

SOFTWARE ENGINEERING CONFERENCE: Abstracts of reports of the XXI International Scientific and Practical Conference of Higher Education Graduates and Young Scientists, Kyiv, 2026, National University «Kyiv Aviation Institute», / S. Hnatyuk Editorial Board [etc.]. – K.: NU «KAI», 2026. – 158 p.

The materials of the scientific and practical conference contain a summary of the reports of scientific research works of students of higher education and young scientists in the field of "SOFTWARE ENGINEERING".

Program committee of the conference:

Petro STETSYUK, Doctor of Physical and Mathematical Sciences, Corresponding Member of the National Academy of Sciences of Ukraine, Head of the Department of the Institute of Cybernetics by V.M. Glushkov of the National Academy of Sciences of Ukraine, Kyiv, Ukraine

Serhiy HNATYUK, Doctor of Technical Sciences, Professor, Vice-Rector for Scientific Research and Technology Transfer, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Alina SAVCHENKO, Doctor of Technical Sciences, Professor, Head of the Department of Computer Information Technologies, Faculty of Computer Sciences and Technologies, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Oleksandr PROVATAR, Doctor of Physical and Mathematical Sciences, Professor, Head of the Department of Intelligent Software Systems, Faculty of Computer Sciences and Cybernetics, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

Oleksiy PYSARCHUK, Doctor of Technical Sciences, Professor, Professor of the Department of Computer Engineering, Faculty of Informatics and Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

Andriy BONDARCHUK, Doctor of Technical Sciences, Professor, Professor of the Department of Artificial Intelligence, Educational and Scientific Institute of Information Technologies, State University of Information and Communication Technologies, Kyiv, Ukraine

Olena CHEBANYUK, Doctor of Technical Sciences, Associate Professor, Professor of the Department of Software Engineering, Faculty of Computer Information Technologies, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Nataliia YEHORCHENKOVA, Doctor of Technical Sciences, Professor, researcher of Spatial Planning department, Institute of Management, Slovak University of Technology in Bratislava, Bratislava, Slovakia

Oksana MULESA, Doctor of Technical Sciences, Professor, Professor of the Department of Physics, Mathematics and Technology, University of Presov, Presov, Slovakia

Oleksandr KUCHANSKYI, Doctor of Technical Sciences, Professor, Professor of Computational and Data Science Department, Astana IT University, Astana, Kazakhstan

Henryk NOGA, Doctor of Habilitation, Professor, Director of the Institute of Technical Sciences, Head of the Department of Technical and Computer Education, University of the National Education Commission in Krakow, Krakow, Poland

Kamila KLUCZEWSKA-CHMIELARZ, PhD, Deputy Director of the Institute of Technical Sciences, University of the National Education Commission in Krakow, Krakow, Poland

Agnieszka KOWALSKA, PhD, Deputy Director of the Institute of Technical Sciences, University of the National Education Commission in Krakow, Krakow, Poland

Agnieszka GAJEWSKA, PhD, University of the National Education Commission in Krakow, Krakow, Poland

Tetiana KONRAD, PhD, University of the National Education Commission in Krakow, Krakow, Poland

Andrii FESENKO, Candidate of Technical Sciences (PhD), Associate Professor, Dean of the Faculty of Computer Sciences and Technologies, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Olena GRINENKO, Candidate of Technical Sciences (PhD), Associate Professor, Head of the Department of Software Engineering, Faculty of Computer Information Technologies, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Organizing committee of the conference:

Yana BIELOZOROVA, Candidate of Technical Sciences (PhD), Associate Professor, Associate Professor of the Department of Software Engineering, Faculty of Computer Science and Technology, National University “Kyiv Aviation Institute”, Kyiv, Ukraine

Larysa POSTAVNA, Assistant Professor, Department of Software Engineering, Faculty of Computer Science and Technology, National University “Kyiv Aviation Institute”, Kyiv, Ukraine

Editorial board

Chief editors:

Serhiy HNATYUK, Doctor of Technical Sciences, Professor, Vice-Rector for Scientific Research and Technology Transfer, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Yana BIELOZOROVA, Candidate of Technical Sciences (PhD), Associate Professor, Associate Professor of the Department of Software Engineering, Faculty of Computer Sciences and Technologies, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Deputy Chief Editors:

Olena GRINENKO, Candidate of Technical Sciences (PhD), Associate Professor, Head of the Department of Software Engineering, Faculty of Computer Sciences and Technologies, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Members of editorial board:

Alina SAVCHENKO, Doctor of Technical Sciences, Professor, Head of the Department of Computer Information Technologies, Faculty of Computer Sciences and Technologies, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Olena CHEBANYUK, Doctor of Technical Sciences, Associate Professor, Professor of the Department of Software Engineering, Faculty of Computer Sciences and Technologies, National University "Kyiv Aviation Institute", Kyiv, Ukraine

Tetiana KONRAD, PhD, University of the National Education Commission in Krakow, Krakow, Poland

Bild Redactor:

Yana BIELOZOROVA, Candidate of Technical Sciences (PhD), Associate Professor, Associate Professor of the Department of Software Engineering, Faculty of Computer Sciences and Technologies, National University "Kyiv Aviation Institute", Kyiv, Ukraine

© National University «Kyiv Aviation Institute», 2026

УДК 004.4(045)

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: Тези доповідей XXI Міжнародної науково-практичної конференції здобувачів вищої освіти і молодих учених, Київ, 2026, Національний університет «Київськиї авіаційний інститут» / Редакційна колегія: С. Гнатюк [та ін.]. – К.: НУ «КАІ», 2026. – 158 с.

Матеріали науково-практичної конференції містять узагальнення доповідей науково-дослідних робіт здобувачів вищої освіти та молодих учених у галузі «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ».

Програмний комітет конференції:

Стецюк Петро Іванович, доктор фізико-математичних наук, член-кореспондент НАН України, завідувач відділу Інституту кібернетики імені В.М. Глушкова НАН України, Київ, Україна

Гнатюк Сергій Олександрович, доктор технічних наук, професор, проректор з наукових досліджень та трансферу технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

Савченко Аліна Станіславівна, доктор технічних наук, професор, завідувач кафедри комп'ютерних інформаційних технологій, факультету комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

Провотар Олександр Іванович, доктор фізико-математичних наук, професор, завідувач кафедри інтелектуальних програмних систем, факультет комп'ютерних наук та кібернетики, Київський національний університет імені Тараса Шевченка, Київ, Україна

Писарчук Олексій Олександрович, доктор технічних наук, професор, професор кафедри обчислювальної техніки, Факультет інформатики та обчислювальної техніки, Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна

Бондарчук Андрій Петрович, доктор технічних наук, професор, професор кафедри штучного інтелекту, Навчально-науковий інститут Інформаційних технологій, Державний університет інформаційно-комунікаційних технологій, Київ, Україна

Чебанюк Олена Вікторівна, доктор технічних наук, доцент, професор кафедри інженерії програмного забезпечення, факультет комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

Єгорченкова Наталія Юріївна, доктор технічних наук, професор, дослідник департаменту Просторового планування, Інститут менеджменту, Словацький технічний університет в Братиславі, Братислава, Словаччина

Мулеса Оксана Юріївна, доктор технічних наук, професор, професор кафедри фізики, математики і техніки, Пряшівський університет, Пряшів, Словаччина

Кучанський Олександр Юрійович, доктор технічних наук, професор, професор департаменту обчислень та науки про дані, Astana IT University, Астана, Казахстан

Нога Генрик, доктор габілітований, професор, директор інституту технічних наук, завідувач кафедри технічної та комп'ютерної освіти, Університет комісії національної освіти в Кракові, Краків, Польща

Ключевська-Хмелярж Каміла, PhD, заступник директора Інституту технічних наук, Університет комісії національної освіти в Кракові, Краків, Польща

Ковальська Агнєшка, PhD, заступник директора Інституту технічних наук, Університет комісії національної освіти в Кракові, Краків, Польща

Гаєвська Агнєшка, PhD, Університет комісії національної освіти в Кракові, Краків, Польща

Конрад Тетяна Ігорівна, PhD, Університет комісії національної освіти в Кракові, Краків, Польща

Фесенко Андрій Олексійович, кандидат технічних наук, доцент, декан факультету комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

Гріненко Олена Олександрівна, кандидат технічних наук, доцент, завідувач кафедри інженерії програмного забезпечення, факультет комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

Організаційний комітет конференції:

Бєлозьорова Яна Андріївна, кандидат технічних наук, доцент, доцент кафедри інженерії програмного забезпечення, факультету комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

Поставна Лариса Петрівна, асистент кафедри інженерії програмного забезпечення, факультету комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

Редакційна колегія

Головні редактори:
Гнатюк Сергій Олександрович, доктор технічних наук, професор, проректор з наукових досліджень та трансферу технологій, Національний університет «Київський авіаційний інститут», Київ, Україна
Бєлозьорова Яна Андріївна, кандидат технічних наук, доцент, доцент кафедри інженерії програмного забезпечення, факультет комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

Заступник головних редакторів:
Гріненко Олена Олександрівна, кандидат технічних наук, доцент, завідувач кафедри інженерії програмного забезпечення, факультет комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

Члени редколегії:
Савченко Аліна Станіславівна, доктор технічних наук, професор, завідувач кафедри комп'ютерних інформаційних технологій, факультету комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна
Чебанюк Олена Вікторівна, доктор технічних наук, доцент, професор кафедри інженерії програмного забезпечення, факультет комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна
Конрад Тетяна Ігорівна, PhD, Університет комісії національної освіти в Кракові, Краків, Польща

Верстка:
Бєлозьорова Яна Андріївна, кандидат технічних наук, доцент, доцент кафедри інженерії програмного забезпечення, факультет комп'ютерних наук та технологій, Національний університет «Київський авіаційний інститут», Київ, Україна

© Національний університет «Київський авіаційний інститут», 2026

Scientific publication

Software Engineering Conference

Abstracts of
XXI International
conference of higher education students
and young scientists

Kyiv, 21-23 April 2026
Published in the author's edition

Наукова публікація

Інженерія програмного забезпечення

Тези доповідей
XXI Міжнародної
науково-практичної конференції здобувачів
вищої освіти і молодих учених

Київ, 21-23 квітня 2026
Публікується у авторській редакції

Національний університет «Київський авіаційний інститут» 03058. Київ – 58, проспект Любомира Гузара, 1
Свідоцтво про внесення до Державного реєстру України суб'єктів видавничої справи ДК № 977 від 05.07.2002

ЗМІСТ

УДК 004.75:004.891

КОНЦЕПТУАЛЬНА МОДЕЛЬ НЕЙРОІНСПІРОВАНОЇ СИСТЕМИ ДИНАМІЧНОГО УПРАВЛІННЯ РЕСУРСАМИ ДЛЯ ОБРОБКИ ВЕЛИКИХ ДАНИХ

Інна АВРАМЕНКО
Здобувачка вищої освіти першого рівня, кафедра КІТ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Артем ПОЛОЖЕНЦЕВ, к.т.н.

У статті запропоновано концептуальну модель нейроінспірованої системи динамічного виділення ресурсів для обробки великих даних із п'ятишаровою архітектурою за аналогією з організацією людського мозку. Модель інтегрує принципи нейропластичності, ієрархічної пам'яті та AI-керованого розподілу ресурсів. Порівняльний аналіз підтверджує перевагу запропонованого підходу над існуючими рішеннями за сукупністю критеріїв адаптивності, енергоефективності та масштабованості.

Ключові слова: *нейроінспірована система, динамічне виділення ресурсів, великі дані, нейропластичність, енергоефективність, хмарні обчислення.*

Вступ

Стрімке зростання обсягів даних вимагає принципово нових підходів до управління обчислювальними ресурсами. Існуючі рішення – хмарне автомасштабування (AWS, Google Cloud) [5] та нейроморфні чіпи (IBM TrueNorth, Intel Loihi) [1, 2] – вирішують лише окремі аспекти: перші забезпечують масштабованість, але позбавлені адаптивності на рівні даних; другі реалізують паралельну обробку, проте обмежені апаратною специфікою. Жодна з існуючих систем не поєднує нейроінспіровану адаптивність, багатопотокову інтеграцію та енергоефективне управління ресурсами в єдиній програмній архітектурі. Актуальність дослідження зумовлена необхідністю розробки концептуальної моделі, що усуває ці обмеження, запозичуючи принципи організації людського мозку без прив'язки до спеціалізованого апаратного забезпечення.

Ціль роботи

Метою роботи є формування та теоретичне обґрунтування концептуальної моделі нейроінспірованої системи динамічного виділення ресурсів для обробки великих даних, що об'єднує принципи нейропластичності, ієрархічної пам'яті, паралельної обробки та AI-керованого розподілу ресурсів [4]. Додатково формалізується адаптивна вагова функція планувальника та обґрунтовується механізм базового навантаження як засіб мінімізації обчислювальних витрат.

Матеріали та методи

У роботі використано методи системного аналізу та порівняльного дослідження існуючих архітектур обробки великих даних. Концептуальна модель побудована на основі аналогії з функціональною організацією людського мозку: принципи короткострокової та довгострокової пам'яті – з когнітивної психології, нейропластичність – з нейробіології, паралельна обробка сенсорних потоків – з нейрофізіології. Математична формалізація

планувальника ресурсів виконана методами теорії управління із застосуванням адаптивних вагових функцій [3].

Результати та обговорення

Запропонована модель має п'ятишарову архітектуру (Рис. 1), де кожен шар відповідає певній функції мозку.

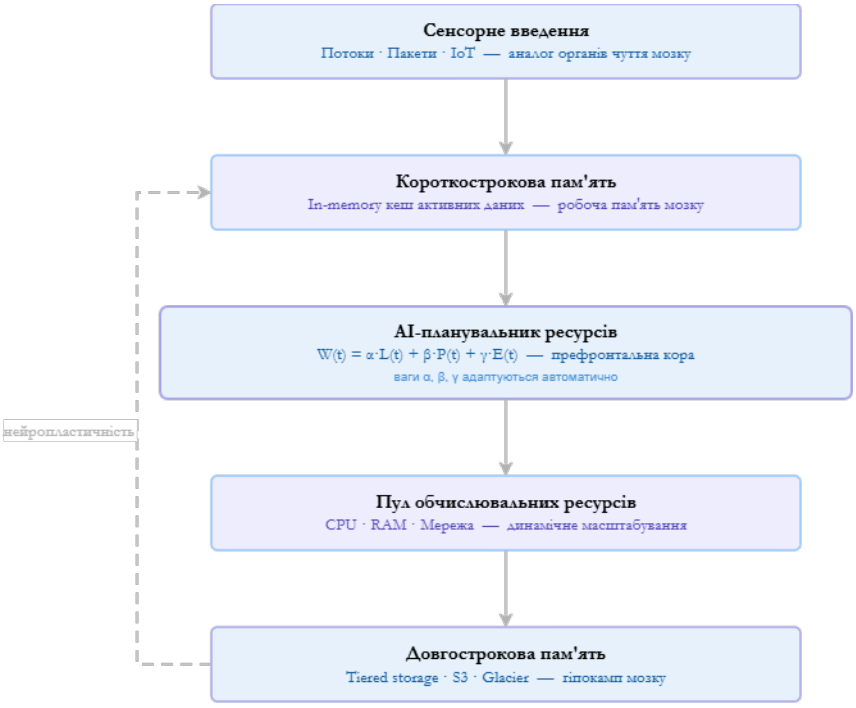


Рис. 1 – П’ятишарова архітектура нейроінспірованої системи

Ключовим елементом системи є AI-кора (шар 3) – адаптивний планувальник ресурсів, що реалізує принцип пластичності через динамічну зміну вагових коефіцієнтів. Алгоритм прийняття рішень зображено на Рис. 2.

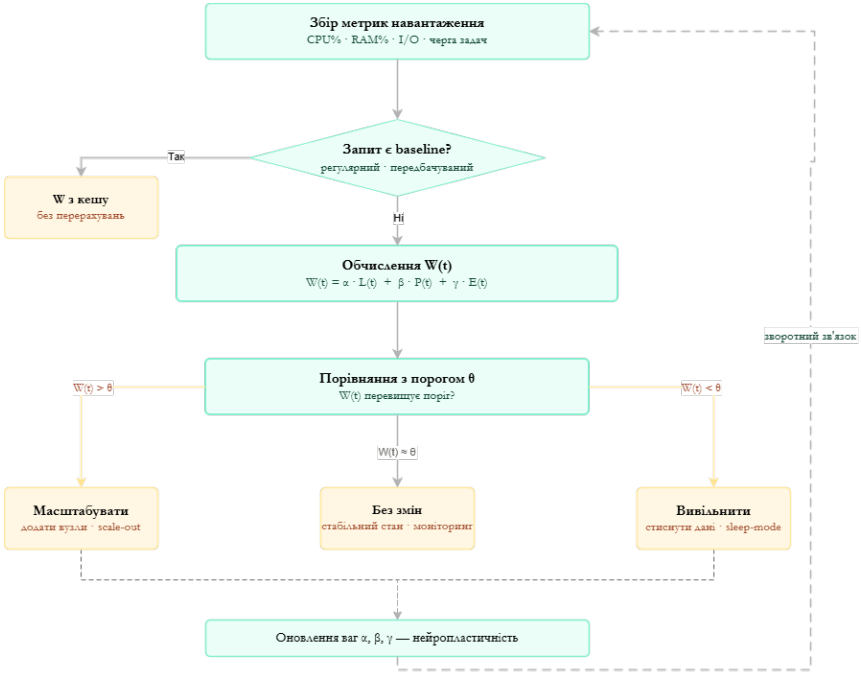


Рис. 2 – Алгоритм AI-планувальника ресурсів

Формальна модель планувальника визначається адаптивною ваговою функцією:

$$W(t) = \alpha \cdot L(t) + \beta \cdot P(t) + \gamma \cdot E(t), \quad (1)$$

де L(t) – поточне навантаження системи (CPU%, RAM%, I/O); P(t) – прогнозне навантаження на основі ML-моделі (LSTM-регресор); E(t) – коефіцієнт ефективності використання ресурсів; α, β, γ – вагові коефіцієнти, що адаптуються відповідно до принципу нейропластичності. Якщо W(t) перевищує поріг Θ – система масштабує пул ресурсів, при W(t) < Θ – вивільняє їх та переводить у режим енергозбереження.

Окремим механізмом є обробка базового навантаження (baseline load) – класу регулярних запитів (циклічна агрегація метрик, фоновий моніторинг). Для таких запитів W(t) береться з кешу без повного перерахунку – аналог автономних процесів стовбура мозку. Кожні N циклів baseline-запит проходить повну перевірку для захисту від поступових аномалій. Перерозподіл між рівнями пам'яті відбувається за критерієм частоти звернень: рідко використовувані дані стискаються та переміщуються з шару 2 (короткострокова) до шару 5 (довгострокова пам'ять) – аналогія консолідації пам'яті у мозку.

Порівняльний аналіз із існуючими системами наведено у Таблиці 1.

Таблиця 1

Порівняння систем обробки великих даних

Система	Нейро-пласт.	Пам'ять	Енергія	Масштаб
IBM TrueNorth	+	–	+	–
AWS Auto Scaling	–	–	±	+
Redis + S3/Glacier	–	+	±	±
Пропонована модель	**+**	**+**	**+**	**+**

Висновки

У роботі запропоновано концептуальну модель нейроінспірованої системи динамічного виділення ресурсів для Big Data з п'ятишаровою архітектурою за аналогією з організацією людського мозку. Ключовим теоретичним внеском є формалізація адаптивного планувальника через вагову функцію W(t), що реалізує принцип нейропластичності шляхом динамічного коригування коефіцієнтів α, β, γ. Механізм базового навантаження скорочує обчислювальні витрати для регулярних запитів за аналогією з автономними процесами стовбура мозку. Порівняльний аналіз підтверджує, що жодна з розглянутих систем не забезпечує одночасно нейроінспірованої адаптивності, ієрархічної пам'яті, енергоефективності та необмеженого масштабування. Запропонована модель може стати основою для розробки універсальної платформи обробки великих даних нового покоління.

Список використаних джерел

1. Abreu S., Shrestha S. B., Zhu Rui-Jie, Eshraghian J. *Neuromorphic principles for efficient large language models on Intel Loihi 2.* arXiv. 2025. URL: https://arxiv.org/abs/2503.18002 (дата звернення: 03.04.2026).

2. Hajizada E., Rager D., Shea T., Campos-Macias L., Wild A., Hüllermeier E., Sandamirskaya Y., Davies M. *Real-time continual learning on Intel Loihi 2.* arXiv. 2025. URL: https://arxiv.org/abs/2511.01553 (дата звернення: 03.04.2026).

3. Sefati S. S., Keymasi M., Craciunescu R., Maiduc S., Bayram M., Arasteh B. *Adaptive resource scheduling in multi-cloud computing using recurrent neural forecasting and memory-based metaheuristic optimization. Journal of Grid Computing*. 2025. Vol. 23, Article 26. URL: https://link.springer.com/article/10.1007/s10723-025-09812-7 (дата звернення: 03.04.2026).

4. Punniyamoorthy V., Agarwal A. K., Kumar B., Mazumder A., Kannan K., Saha S**.** *AI-driven cloud resource optimization for multi-cluster environments.* arXiv. 2025. URL: https://arxiv.org/abs/2512.24914 (дата звернення: 05.04.2026).

5. Jeong B., Jeong Y. S**.** *Autoscaling techniques in cloud-native computing: a comprehensive survey. Computer Science Review*. 2025. Vol. 58, p. 100791. URL: https://www.sciencedirect.com/science/article/abs/pii/S157401372500067X (дата звернення: 04.04.2026).

Відомості про автора

Авраменко Інна Віталіївна – здобувачка вищої освіти першого рівня, факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* нейроінспіровані системи та архітектури обробки даних, людино-машинна взаємодія (HCI) та нейроінтерфейси (BCI), кібербезпека та захист інформації в розподілених системах, застосування штучного інтелекту у медичних пристроях.

E-mail: 8464158@stud.kai.edu.ua

УДК 004.9

АНАЛІЗ СУЧАСНИХ СИСТЕМ/МЕТОДІВ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ КУР'ЄРСЬКОЇ СЛУЖБИ

Анастасія АНОХІНА
Здобувачка вищої освіти першого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Андрій ГІЗУН, професор, к.т.н.

У тезах розглядаються категорії показників ефективності доставки та методи їх аналізу. Проведено порівняння традиційного статистичного підходу з сучасними методами на основі Real-time аналітики та штучного інтелекту. Окреслено проблеми автоматизації «останньої милі» та обґрунтовано переваги впровадження спеціалізованих систем (TMS) для підвищення точності KPI.

Ключові слова: *кур'єрська служба, логістика, KPI, автоматизація, TMS.*

Вступ

Сучасні кур'єрські служби функціонують в умовах високої конкуренції, зростаючих вимог клієнтів та необхідності оптимізації витрат. Особливо критичним є етап «останньої милі», який визначає якість сервісу та рівень задоволеності клієнтів. У зв'язку з цим зростає потреба у впровадженні автоматизованих систем оцінювання ефективності доставки, що базуються на комплексному аналізі ключових показників діяльності (KPI).

Мета роботи

Метою роботи є проведення аналізу існуючих систем та методів оцінювання ефективності кур'єрських служб для виявлення їх недоліків і потенційних можливостей вдосконалення, а також обґрунтування доцільності впровадження автоматизованих систем збору та аналізу даних. Ефективність сучасної кур'єрської служби безпосередньо залежить від швидкості обробки даних та точності логістичних операцій. Традиційні методи контролю, що базуються на ретроспективному аналізі, вже не відповідають вимогам динамічного ринку. Це зумовлює необхідність переходу до проактивного управління логістикою. Для оцінювання ефективності доставки використовують різні категорії показників. До логістичних KPI належать час доставки (OTD), завантаженість транспорту (utilization rate), середній пробіг маршруту тощо. Наприклад, OTD відображає частку замовлень, виконаних у встановлений час, що є критичним для «останньої милі». До економічних KPI відносяться показники собівартості доставки: cost per delivery, витрати на транспорт і логістику, співвідношення витрат до доходу. Вони безпосередньо впливають на прибутковість компанії. Клієнтські KPI включають показники задоволеності та лояльності: CSAT і NPS, які відображають сприйняття якості сервісу кінцевим користувачем.

Традиційний підхід до оцінювання ефективності базується на ретроспективному аналізі даних (звіти, Excel), що характеризується простотою, але має суттєві недоліки - затримку в отриманні інформації та високий ризик помилок.

Сучасні методи передбачають використання real-time аналітики, GPS-трекінгу, IoT, Big Data та AI. Вони дозволяють оперативно виявляти відхилення та коригувати логістичні процеси в режимі реального часу.

Методи на основі машинного навчання забезпечують більш точне прогнозування показників ефективності, враховуючи складні залежності, зокрема вплив зовнішніх факторів (погода, трафік). Серед сучасних програмних рішень для автоматизації логістики виділяються Route4Me, Bringg, Relog та UIS.TMS. Вони забезпечують моніторинг KPI, візуалізацію даних та аналіз ефективності доставки.

Водночас існують проблеми автоматизації «останньої милі», зокрема:

- людський фактор (затримки синхронізації, вимкнення трекерів),
- фрагментація даних між різними системами,
- складність обробки нестандартних ситуацій (невдалі доставки, зміна адреси).

Висновки та пропозиції

Проведений аналіз показав, що традиційні методи оцінювання ефективності кур'єрських служб є недостатньо ефективними в умовах сучасного динамічного середовища, оскільки не забезпечують своєчасності та точності даних. Встановлено, що основними недоліками існуючих систем є фрагментація інформації, залежність від ручного введення даних та обмежені можливості оперативного аналізу.

З метою усунення зазначених проблем доцільним є впровадження автоматизованих систем, що інтегрують функції збору, обробки та аналізу даних у єдиному середовищі. Такі системи дозволяють:

- забезпечити узгодженість планових і фактичних показників,
- підвищити точність збору даних за рахунок автоматизації,
- скоротити вплив людського фактору,
- покращити оперативність прийняття управлінських рішень.

Отже, подальші дослідження доцільно спрямувати на розробку та впровадження інтегрованих аналітичних систем для підвищення ефективності кур'єрських служб.

Список використаних джерел

1. KPI в логістиці. URL: https://abmcloud.com/uk/kpi-v-lohistytsi.
2. Delivery performance KPI. URL: https://locus.sh/blogs/delivery-performance-kpi.
3. Last mile delivery service metrics. URL: https://fareye.com/resources/blogs/last-mile-delivery-service-metrics.
4. Real-time delivery tracking. URL: https://www.bringg.com/resources/blog/real-time-delivery-tracking.
5. Machine learning for delivery forecasting. *ScienceDirect*. 2024. URL: https://www.sciencedirect.com/science/article/pii/S2949863524000426.

Відомості про автора

Анохіна Анастасія Євгенівна – здобувачка вищої освіти першого рівня, кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інформаційні системи в логістиці, автоматизація бізнес-процесів, аналіз ключових показників ефективності.

E-mail: 7905974@stud.kai.edu.ua

УДК 004.8

ПОРІВНЯЛЬНИЙ АНАЛІЗ АРХІТЕКТУРИ RAG ТА КЛАСИЧНИХ МЕТОДІВ ПОШУКУ У СИСТЕМАХ ПИТАНЬ І ВІДПОВІДЕЙ

Нікіта БАБІК
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Любов БОРКОВСЬКА, к.т.н., доцент

У статті проведено порівняльний аналіз трьох підходів до побудови систем питань і відповідей на природній мові: повнотекстового пошуку BM25, донавчання мовних моделей та архітектури RAG (Retrieval-Augmented Generation). Розглянуто принципи роботи кожного підходу, їхні ключові обмеження та практичні критерії вибору залежно від характеристик бази знань. Встановлено, що RAG є найбільш гнучким рішенням для систем із динамічним контентом, оскільки не потребує перенавчання при оновленні документів і природно знижує ризик генерації хибних відповідей завдяки прив'язці до джерел.

Ключові слова: *RAG, Retrieval-Augmented Generation, BM25, fine-tuning, системи питань і відповідей, векторний пошук, великі мовні моделі, обробка природної мови, галюцинації моделі.*

Вступ

Організації накопичують великі обсяги документації, і звичайний текстовий пошук дедалі гірше справляється з природно-мовними запитами: він не розуміє синонімів і повертає нерелевантні результати, якщо у запиті вжито слово, якого немає в документі. На зміну йому прийшли системи питань і відповідей (QA), що повертають конкретну відповідь, а не список документів [1]. Серед підходів до побудови таких систем особливий інтерес становить архітектура RAG, яка зберігає знання зовні моделі й динамічно звертається до них при формуванні відповіді.

Ціль роботи

Мета роботи – порівняти архітектуру RAG із класичними підходами до побудови QA-систем: повнотекстовим пошуком BM25 та моделлю з донавчанням – за принципами роботи, ключовими обмеженнями та практичними критеріями вибору.

Матеріали та методи

Дослідження виконане у формі теоретичного аналізу наукових публікацій. BM25 оцінює важливість слова через частоту його входження в документ і рідкість у колекції, але не розуміє синонімів і не генерує відповідь – лише повертає найрелевантніший фрагмент тексту [1]. Fine-tuning передбачає навчання мовної моделі на розмічених парах «запитання–відповідь»: модель розуміє синонімічні запити, але при оновленні бази знань потребує повного перенавчання і поза межами навчальної вибірки схильна генерувати хибні відповіді [2]. RAG розділяє задачу на два етапи: спочатку система знаходить змістово близькі фрагменти через векторний пошук, потім мовна модель формулює зв'язну відповідь із посиланням на джерело; якщо потрібної інформації немає – система може про це повідомити [3].

Результати та обговорення

Аналіз виявив принципові відмінності між підходами. BM25 суттєво залежить від точності формулювання запиту і не підходить для систем із різноманітними природномовними зверненнями. Fine-tuning забезпечує хорошу точність у вузькій предметній області, однак кожне суттєве оновлення документів потребує нового циклу навчання, що може тривати години.

Головна практична перевага RAG – незалежність від перенавчання: додавання нових документів зводиться до їх індексації й займає хвилини. Водночас RAG природно знижує ризик «галюцинацій», оскільки відповідь будується лише на основі знайдених фрагментів. Втім, для невеликих статичних баз знань із чіткою термінологією класичний пошук може виявитись достатнім і значно дешевшим вибором, адже інфраструктура RAG суттєво складніша за звичайний BM25-індекс.

Висновки

BM25 залишається доцільним там, де термінологія фіксована і оновлення рідкі. Fine-tuning виправдовує себе у вузькоспеціалізованих задачах із наявним розміченим датасетом і стабільною базою знань. RAG є найбільш гнучким вибором для систем із різноманітними запитами і базами знань, що постійно оновлюються: він не потребує перенавчання і підтримує прозорість відповідей через посилання на джерела. Отримані висновки обґрунтовують вибір архітектури QA-систем, а перспективним напрямом розвитку є впровадження гібридних стратегій пошуку.

Список використаних джерел

1. RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture / A. Balaguer [та ін.]. 2024. URL: https://arxiv.org/abs/2401.08406 (дата звернення: 07.04.2026).

2. Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs / O. Ovadia [та ін.]. 2024. URL: https://arxiv.org/abs/2312.05934 (дата звернення: 07.04.2026).

3. Retrieval-Augmented Generation for Large Language Models: A Survey / Y. Gao [та ін.]. 2024. URL: https://arxiv.org/abs/2312.10997 (дата звернення: 07.04.2026).

Відомості про автора

Бабік Нікіта Петрович – здобувач вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* штучний інтелект, проєктування та розробка вебсистем.

E-mail: 7966709@stud.kai.edu.ua

УДК 004.4

КОМПОЗИТНА ФУНКЦІЯ ОЦІНЮВАННЯ КОРИСТУВАЧІВ У ГЕЙМІФІКОВАНИХ ЗАСТОСУНКАХ МОНІТОРИНГУ ВУГЛЕЦЕВОГО СЛІДУ

Валентин БАДАЙ
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Яна БЄЛОЗЬОРОВА
к.т.н., доц., доцент кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

Досліджено проблему конфлікту мотиваційних стимулів у гейміфікованих застосунках моніторингу особистого вуглецевого сліду. Показано, що три природні правила нарахування балів – за активність, абсолютний рівень CO_2e та відносне покращення – формують радикально різні рейтинги тих самих користувачів. Запропоновано композитну функцію оцінювання з min-max нормалізацією компонентів та зваженою сумою. На симуляції п'яти типових профілів за 30 днів продемонстровано, що композитна функція коректно вирізняє користувачів за реальним екологічним внеском.

Ключові слова: *вуглецевий слід, CO_2e, гейміфікація, лідерборд, нарахування балів, екологічний моніторинг.*

Вступ

Концепція вуглецевого сліду – кількісного показника у кг CO_2e [5] – стала основою для цілого класу мобільних і вебзастосунків самомоніторингу (Klima, JouleBug, Capture, Commons, Earth Hero, EcoTrack) [3, 4], а глобальний ринок таких рішень демонструє стійке зростання на рівні ~9% на рік [6]. Дослідження фіксують, що цифрові трекери здатні знижувати викиди користувачів у середньому на 23% за умови регулярного зворотного зв'язку [1]. Ключовим механізмом утримання залученості є гейміфікація – бали, рівні та публічні рейтинги [2]. Огляд провідних рішень показує, що жодне з них не реалізує одночасно точне вимірювання у кг CO_2e та повноцінний публічний лідерборд. При цьому сам спосіб нарахування балів є нетривіальною задачею: спрощений підхід веде до спотворень – найактивніший користувач не обов'язково найекологічніший, а користувач з уже низьким слідом не отримує винагороди за зусилля.

Ціль роботи

Формалізувати конфлікт стимулів при нарахуванні балів у гейміфікованих CO_2e-трекерах та запропонувати композитну функцію, що балансує активність, абсолютний рівень і відносне покращення.

Матеріали та методи

Виконано симуляцію п'яти умовних профілів користувачів за 30 днів у середовищі Python (numpy, pandas), seed = 42 для відтворюваності. Кожен профіль характеризується середнім CO_2e/день, дисперсією, ймовірністю логування та трендом зміни. Параметри обрано з типових діапазонів: важкий профіль 17–18 кг CO_2e/день (авто, опалення, м'ясний раціон), середній 8–10 кг, низький 4–5 кг (вегетаріанець без авто).

Порівнюються чотири стратегії нарахування:

1) Активність: $S_A = N$, де N – кількість днів з логом за період;

2) Абсолютний рівень: $S_P = 1 - \mu_{user} / \mu_{max}$, де μ_{user} – середній денний CO_2e даного користувача, μ_{max} – максимум μ_{user} по групі;

3) Покращення: $S_R = (\mu_{first} - \mu_{last}) / \mu_{first} \cdot 100\%$, де μ_{first}, μ_{last} – середні CO_2e за перший і останній тиждень відповідно;

4) Композитний: $S_C = w_A \cdot \hat{A} + w_P \cdot \hat{P} + w_I \cdot \hat{I}$, де $\hat{A}$ – min-max нормалізована активність, $\hat{I}$ – min-max нормалізоване покращення, $\hat{P}$ – інверсно нормалізований рівень CO_2e (низькі значення CO_2e → високі бали), усі три нормалізовані до [0, 100]; ваги $w_A = 0{,}25$, $w_P = 0{,}40$, $w_I = 0{,}35$ (сума ваг = 1, обрані евристично з пріоритетом реального рівня викидів). Константи масштабування у формулах 1–3 опущено, оскільки ранжування інваріантне до них.

Результати

Агреговані показники симуляції наведено у таблиці 1, рейтинги під кожною стратегією – на рис. 1.

Таблиця 1.

Агреговані показники симуляції за 30 днів

Користувач	Логів	Сер. CO_2e, кг/день	Покращення, %	Композит
Іван (забруднювач)	29	17,65	−5,8	23,1
Олена (еко, тиха)	17	4,30	+12,8	58,0
Петро (покращувач)	26	10,20	+30,4	74,6
Марія (середняк)	20	8,91	−1,8	35,8
Сергій (фарм-балів)	30	11,87	+8,3	56,0

Користувач, що логує щодня без зміни поведінки («фарм-балів»), посідає 1-ше місце за активністю, але лише 4-те – за абсолютним рівнем CO_2e.

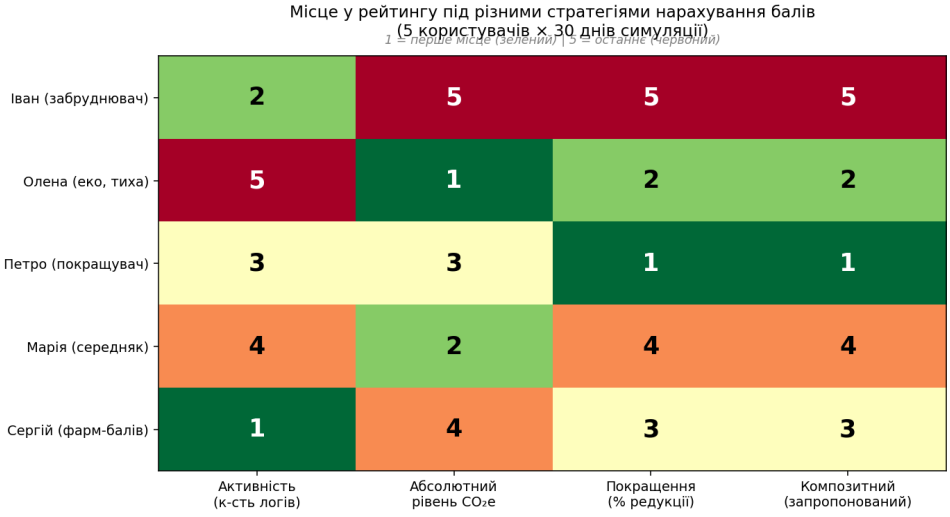


Рис. 1. Місця у рейтингу п’яти користувачів під чотирма стратегіями нарахування балів (симуляція 30 днів, seed = 42)

Користувач з найбільшим реальним покращенням (зниження CO_2e на 30,4%) виграє за критеріями покращення і композитом, але отримує лише 3-тє місце за абсолютним рівнем.

Найчистіший за фактом користувач – 1-й за абсолютним рівнем, але останній – за активністю. Жодна з трьох наївних стратегій не дає інтуїтивно прийнятного рейтингу. Композитна функція виносить на 1-ше місце користувача, що поєднує реальне зниження CO_2e з регулярним логуванням, а на останнє – забруднювача без змін поведінки.

Висновки

Числова симуляція показала, що три природні правила нарахування балів (за активністю, абсолютним рівнем CO_2e та відносним покращенням) формують різні рейтинги тих самих п'яти користувачів, причому переможець кожного правила опиняється на нижніх позиціях двох інших. Запропонована композитна функція з вагами 0,25/0,40/0,35 усуває цей конфлікт: на 1-ше місце виходить користувач з максимальним зниженням CO_2e (30,4%) у поєднанні з регулярним логуванням, на останнє – забруднювач без змін поведінки. Основне обмеження – евристичний вибір ваг.

Список використаних джерел

1. Hoffmann S., Lasarov W., Reimers H. Carbon footprint tracking apps. Does feedback help reduce carbon emissions? Journal of Cleaner Production. 2024. Vol. 434.
2. Adaji I., Idoko P., Adisa M. Insights from the Review of Apps that Influence Environmental Sustainability. Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization. 2024.
3. Top Apps for Tracking Your Carbon Footprint in 2025. Make An App Like. URL: https://makeanapplike.com/app-to-track-carbon-footprint/
4. Dash A. et al. EcoTrack: A Mobile Application for Real-Time Carbon Footprint Tracking and Sustainable Living. ResearchGate. 2024.
5. Ecological Footprint: methodology overview. Global Footprint Network. URL: https://www.footprintnetwork.org/our-work/ecological-footprint/
6. Carbon Footprint Management Market Industry Report 2030. Grand View Research.

Відомості про авторів:

Валентин Андрійович Бадай – здобувач вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси*: архітектура SaaS-застосунків, алгоритми ранжування, гейміфікація цифрових застосунків.

E-mail: 7924411@stud.kai.edu.ua

Бєлозьорова Яна Андріївна – доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення, обробка зображень, агрегація даних.

E-mail: yana.bielozorova@npp.kai.edu.ua

УДК 004.8

БЕЗПЕКА LLM-ЗАСТОСУНКІВ: ВИЯВЛЕННЯ ТА ЗАПОБІГАННЯ PROMPT INJECTION АТАКАМ

Олесь БОНДАР
Асистент кафедри інженерії програмного забезпечення
Національний університет «Київський авіаційний інститут»

У роботі досліджуються вразливості типу prompt injection у застосунках на основі великих мовних моделей (LLM). Проаналізовано основні вектори атак, методи їх виявлення та сучасні підходи до захисту. Запропоновано комбіновану стратегію захисту, що поєднує валідацію вхідних даних, ієрархію інструкцій і фільтрацію виводу з використанням додаткової LLM-моделі.

Ключові слова: *prompt injection, LLM, безпека застосунків, великі мовні моделі, захист від атак, RAG, штучний інтелект.*

Вступ

Стрімке поширення застосунків на основі великих мовних моделей (LLM) у корпоративному та споживчому секторах відкрило новий клас загроз інформаційної безпеки. Серед них особливе місце посідають prompt injection атаки – техніки маніпулювання вхідними даними, що дозволяють зловмисникам обходити системні інструкції та змушувати модель виконувати несанкціоновані дії [1].

На відміну від традиційних вразливостей програмного забезпечення, prompt injection атаки експлуатують фундаментальну архітектурну особливість LLM: модель не розрізняє системні інструкції від даних, наданих користувачем чи отриманих із зовнішніх джерел. Це робить задачу захисту принципово складнішою і такою, що не може бути вирішена стандартними засобами на кшталт екранування символів або перевірки типів [2].

Метою даної роботи є систематизація відомих векторів prompt injection атак, аналіз існуючих методів захисту та розроблення практичних рекомендацій для побудови захищених LLM-застосунків.

Ціль роботи

Метою дослідження є класифікація загроз типу prompt injection для LLM-застосунків, порівняльний аналіз методів їх виявлення та запобігання, а також формування рекомендацій щодо побудови захищеної архітектури LLM-систем. Завданнями роботи є:

- визначити основні класи prompt injection атак та їх механізми дії;
- проаналізувати сучасні підходи до виявлення та нейтралізації таких атак;
- порівняти ефективність різних стратегій захисту за ключовими критеріями;
- запропонувати комбіновану стратегію захисту для практичного застосування.

Матеріали та методи

Дослідження ґрунтується на аналізі наукової літератури, звітів з кібербезпеки провідних організацій (OWASP, NIST, Microsoft), а також відтворенні атак у контрольованому тестовому середовищі на базі моделей GPT-4o та Claude 3 Sonnet. Для класифікації атак використано методологію OWASP Top 10 for LLM Applications [3].

Виділяють два основні класи prompt injection атак: прямі та непрямі. Прямі атаки (direct prompt injection) відбуваються, коли зловмисник безпосередньо вводить у діалог з моделлю спеціально сформований запит, що змушує її ігнорувати системні інструкції. Типовий приклад – запит виду «Ignore all previous instructions and...». Непрямі атаки (indirect prompt injection) є складнішими: шкідливі інструкції вбудовуються в зовнішні дані, які модель опрацьовує в рамках RAG-систем чи при аналізі документів, веб-сторінок або електронних листів [1, 4].

Для оцінювання методів захисту застосовано такі критерії: ефективність (частка успішно відбитих атак у тестовій вибірці з 50 сценаріїв), складність реалізації та вплив на продуктивність системи.

Результати та обговорення

У ході тестування виявлено, що жоден одиночний метод захисту не забезпечує прийнятного рівня безпеки сам по собі. Input sanitization та валідація вхідних даних демонструють середню ефективність (~65%) і легко обходяться за допомогою кодування або перефразування шкідливих інструкцій. Підхід на основі ієрархії інструкцій (системний промпт з явними обмеженнями та маркуванням рівнів довіри) показав вищу ефективність (~80%), однак також вразливий до атак з обфускацією.

Найефективнішим виявився підхід LLM-as-a-judge, де окрема мовна модель аналізує як вхідний запит, так і вихід основної моделі перед поверненням відповіді користувачеві. Ефективність такого підходу склала ~91%, проте він потребує значних обчислювальних ресурсів і збільшує затримку відповіді на 40–60%.

Порівняльний аналіз методів захисту наведено у таблиці 1.

Таблиця 1

Порівняльна характеристика методів захисту від prompt injection

Метод захисту	Ефективність	Складність реалізації	Вплив на продуктивність
Input sanitization та валідація	Середня	Низька	Мінімальний
Instruction hierarchy (системний промпт)	Висока	Середня	Відсутній
LLM-as-a-judge (фільтрація виводу)	Висока	Висока	Помірний
Комбінований підхід	Дуже висока	Висока	Помірний

Результати підтверджують, що найбільш надійним є комбінований підхід, який поєднує всі три методи: попередню фільтрацію вхідних даних, ієрархію системних інструкцій з чітким розмежуванням рівнів довіри та перевірку виводу. Такий підхід забезпечує ефективність ~96% за рахунок взаємної компенсації слабких місць окремих методів.

Окремої уваги потребує проблема indirect prompt injection у RAG-системах. Зловмисний контент, вбудований у зовнішні документи, обходить традиційні фільтри, оскільки надходить з «довіреного» джерела – бази знань застосунку. Для захисту рекомендується впроваджувати сандбоксинг при обробці зовнішніх джерел та використовувати окрему модель для перевірки довіреності вилучених фрагментів.

Висновки

Дослідження підтвердило, що prompt injection атаки становлять системну загрозу для LLM-застосунків і не можуть бути усунені жодним одиночним методом захисту. На основі отриманих результатів сформульовано наступні практичні рекомендації:

1. Застосовувати принцип мінімальних привілеїв: LLM-агент має мати доступ лише до тих інструментів і даних, які необхідні для виконання конкретного завдання.
2. Впроваджувати явну ієрархію довіри в системному промпті з маркуванням джерел даних (системні інструкції, дані користувача, зовнішні джерела).
3. Використовувати окрему модель-«суддю» для перевірки виводу LLM перед поверненням відповіді в критично важливих застосунках.
4. Проводити регулярне red-team тестування з актуальними техніками атак, оскільки ландшафт загроз швидко еволюціонує.

Перспективними напрямами подальших досліджень є розробка спеціалізованих детекторів prompt injection на основі fine-tuning, а також дослідження захисту мультимодальних LLM-систем, де атаки можуть здійснюватися через зображення або аудіо-дані.

Список використаних джерел

1. Perez E., Ribeiro I. Ignore Previous Prompt: Attack Techniques For Language Models. NeurIPS 2022 Workshop. URL: https://arxiv.org/abs/2211.09527

2. Greshake K. et al. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. AISec 2023. URL: https://arxiv.org/abs/2302.12173

3. OWASP Top 10 for Large Language Model Applications. Version 1.1. URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/

4. Microsoft AI Red Team. Prompt Injection Attacks and Defenses in LLM-Integrated Applications. Microsoft Security Blog, 2024. URL: https://www.microsoft.com/en-us/security/blog/2024/prompt-injection

Відомості про автора

Бондар Олесь Анатолійович – Senior .NET Developer в IT-компанії Luxoft, асистент кафедри інженерії програмного забезпечення Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення, кібербезпека, великі мовні моделі.

E-mail: oles.bondar@npp.kai.edu.ua

УДК 004.4

ВИКОРИСТАННЯ СУЧАСНИХ ТЕХНОЛОГІЙ ДЛЯ АНАЛІЗУ ЕФЕКТИВНОСТІ МОНІТОРИНГУ ХМАРНОЇ ІНФРАСТРУКТУРИ

Роман БОНДАРЕНКО
Здобувач вищої освіти другого рівня, кафедра ІТ
Сергій ГРІНЕНКО
к.т.н., доцент
Навчально-науковий інститут управління, технологій та правових наук
Національний транспортний університет

У роботі розглянуто проблему ефективного моніторингу хмарної інфраструктури в умовах її динамічності та розподіленого характеру, коли традиційні методи адміністрування стають недостатніми. Обґрунтовано доцільність застосування концепції спостережуваності (observability), що поєднує метрики, журнали та трасування запитів. Проаналізовано підходи white-box і black-box та показано перевагу їх поєднання. Визначено ключові критерії ефективності систем моніторингу: час виявлення й усунення проблем, масштабованість, якість візуалізації та сповіщень. Підтверджено практичну доцільність спільного використання Prometheus і Grafana для контролю стану хмарних застосунків.

Ключові слова: *моніторинг; хмарна інфраструктура; спостережуваність (observability); Prometheus; Grafana; метрики; розподілені системи.*

Вступ

Хмарні обчислювальні середовища широко використовуються для розгортання інформаційних систем і сервісів, що супроводжується зростанням складності їх інфраструктури та підвищенням вимог до контролю її стану. У таких умовах ефективний моніторинг стає ключовим інструментом забезпечення продуктивності, доступності та надійності систем.

Традиційні підходи до адміністрування, що базуються на ручному контролі або аналізі журналів подій, є недостатніми для сучасних розподілених систем. Виникає потреба у впровадженні комплексних засобів спостережуваності (observability), які забезпечують автоматичний збір, зберігання та аналіз телеметричних даних. До таких даних належать метрики, журнали та трасування запитів, що дозволяють оперативно виявляти збої та оцінювати продуктивність системи.

Одними з найбільш поширених інструментів моніторингу хмарної інфраструктури є Prometheus – система збору та зберігання метрик часових рядів, та Grafana – платформа для візуалізації та аналітики даних моніторингу. Їх спільне використання дозволяє реалізувати повний цикл спостереження за системою: від отримання показників роботи компонентів до побудови інформаційних панелей та налаштування сповіщень.

Особливістю хмарної інфраструктури є її динамічність, масштабованість і розподілений характер. Використання мікросервісної архітектури, контейнеризації та автоматичного масштабування призводить до постійної зміни складу системи, що ускладнює процес контролю її стану. Традиційні методи адміністрування, засновані на ручному аналізі журналів, є недостатніми, що обумовлює необхідність застосування концепції спостереження (observability), яка передбачає аналіз метрик, журналів та трасування запитів.

На відміну від традиційних дата-центрів, де сервери мають статичну конфігурацію і постійне розташування, у хмарному середовищі інфраструктура є динамічною. Віртуальні машини, контейнери та мережеві ресурси можуть створюватися і видалятися автоматично залежно від навантаження. У мікросервісній архітектурі система складається з великої кількості невеликих сервісів, що взаємодіють між собою через мережу, а її стан постійно змінюється.

Характерною особливістю є ефемерність ресурсів: екземпляри сервісів можуть існувати короткий час і автоматично замінюватися новими. Одночасно застосовується горизонтальне масштабування, при якому кількість екземплярів сервісу змінюється відповідно до навантаження. Компоненти системи часто розміщуються у різних дата-центрах або регіонах, а оркестратори контейнерів автоматично перезапускають та переміщують сервіси без участі адміністратора. Крім того, користувач не має фізичного доступу до обладнання і взаємодіє з інфраструктурою виключно через програмні інтерфейси.

Матеріали та методи

Розглянуто два основні підходи до моніторингу: white-box та black-box. White-box моніторинг забезпечує отримання детальної інформації про внутрішній стан системи, дозволяючи точно визначати причини збоїв, тоді як black-box орієнтований на перевірку доступності сервісів з точки зору користувача.

White-box (внутрішній) моніторинг передбачає отримання інформації безпосередньо з компонентів системи. Сервіси експортують технічні показники роботи, які збираються системою моніторингу через спеціальні інтерфейси.

Black-box (зовнішній) моніторинг розглядає систему як «чорну скриньку». Перевірка виконується шляхом імітації дій користувача: надсилання HTTP-запитів, тестових транзакцій або перевірки відкриття мережевих портів.

Їх поєднання дозволяє досягти більш повного уявлення про стан інфраструктури.

Результати

Важливим аспектом є визначення критеріїв ефективності систем моніторингу. До основних належать: надійність збору даних, швидкість виявлення проблем (Time to Detect), можливість їх усунення (Time to Resolution), масштабованість, гнучкість інтеграції, якість візуалізації та ефективність механізмів сповіщення. Оцінка цих параметрів дозволяє визначити придатність системи моніторингу для використання у конкретних умовах експлуатації.

Ефективний моніторинг у хмарній інфраструктурі повинен виконуватися автоматично, охоплювати всі компоненти системи та коректно працювати при зміні кількості вузлів. Важливим є швидке виявлення відмов і деградації продуктивності, можливість зберігання історичних даних для аналізу тенденцій та підтримка механізмів сповіщення про критичні відхилення параметрів. Окреме значення має зручне представлення інформації, яке дозволяє оператору швидко оцінити стан системи. Отже, специфіка хмарних середовищ зумовлює необхідність використання спеціалізованих систем моніторингу, орієнтованих на роботу з динамічною розподіленою інфраструктурою. Такі системи забезпечують своєчасне виявлення збоїв та контроль продуктивності сервісів, що є необхідною умовою стабільної роботи сучасних інформаційних систем.

Висновки

Аналіз ефективності систем моніторингу є необхідним для визначення їх придатності у конкретних умовах експлуатації. У хмарних та розподілених системах, де кількість компонентів і динаміка інфраструктури значні, критично важливо оцінювати не лише наявність даних, але й швидкість їх обробки, точність та можливість реагування на відмови, що свідчить про необхідність комплексного підходу, що поєднує різні моделі спостереження, використання сучасних інструментів та оцінювання за ключовими критеріями. Це забезпечує підвищення надійності, стабільності та продуктивності сучасних інформаційних систем та практичну доцільність використання сучасних застосунків Prometheus та Grafana для моніторингу хмарних додатків.

Список використаних джерел

1. Google SRE Book [Електронний ресурс]. – Режим доступу: https://sre.google/sre-book/monitoring-distributed-systems/

2. Документація Prometheus [Електронний ресурс]. – Режим доступу: https://prometheus.io/docs/introduction/overview/

3. The NIST Definition of Cloud Computing [Електронний ресурс]. – Режим доступу: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

4. OpenTelemetry: Observability primer [Електронний ресурс]. – Режим доступу: https://opentelemetry.io/docs/concepts/observability-primer/

5. Google SRE Book: Alerting on SLOs [Електронний ресурс]. – Режим доступу: https://sre.google/workbook/alerting-on-slos/

6. Prometheus Monitoring: Use Cases, Metrics, and Best Practices [Електронний ресурс]. - Режим доступу: https://www.tigera.io/learn/guides/prometheus-monitoring/

7. Grafana Cloud [Електронний ресурс]. – Режим доступу: https://grafana.com/docs/grafana-cloud/visualizations/

8. Guide on Grafana Common Features [Електронний ресурс]. – Режим доступу: https://staticintl.cloudcachetci.com/doc/pdf/product/pdf/1124_69439_en.pdf

9. Datadog: Infrastructure Monitoring Overview [Електронний ресурс]. – Режим доступу: https://www.datadoghq.com/knowledge-center/infrastructure-monitoring/

10. O'Reilly: Monitoring Distributed Systems [Електронна книга] Режим доступу: https://theswissbay.ch/pdf/Books/Computer%20science/O'Reilly/monitoring-distributed-systems.pdf

Відомості про авторів

Бондаренко Роман Сергійович – здобувач вищої освіти другого рівня, Навчально-науковий інститут управління, технологій та правових наук, Національний транспортний університет. *Наукові інтереси*: моніторинг; хмарна інфраструктура.

E-mail: rbondarenko211@gmail.com

Гріненко Сергій Анатолійович – доцент кафедри інформаційних технологій навчально-наукового інституту управління, технологій та правових наук, Національний транспортний університет. *Наукові інтереси*: інженерія програмного забезпечення.

E-mail: hrinenkosa@gsuite.duit.edu.ua

УКД 004.932

РОЗРОБКА НЕЙРОМЕРЕЖЕВОГО АЛГОРИТМУ РОЗПІЗНАВАННЯ ЕМОЦІЙНОГО СТАНУ ЛЮДИНИ ЗА ЗОБРАЖЕННЯМИ ЙОГО ОБЛИЧЧЯ

Любов БОРКОВСЬКА
к.т.н., доц., доцент кафедри інженерії програмного забезпечення
Національний університет «Київський авіаційний інститут»

У статті розглядається розробка нейромережевого алгоритму, який здійснює розпізнавання емоційного стану людини за зображеннями його обличчя.

***Ключові слова**: Keras, OpenCV, TensorFlow, Theano, CNTK.*

Вступ

Нині технології штучного інтелекту неминуче продовжують займати важливе місце в нашому житті: пошук найбільш оптимального маршруту, виконання побутових функцій, оптимізація бізнесу – лише малий список того, що вченим та програмістам вдалося автоматизувати та впровадити у повсякденне життя. Найбільший прорив у сфері штучного інтелекту здійснив напрямок розпізнавання об'єктів.

Технологія розпізнавання об'єктів дозволяє виконувати широкий спектр завдань, серед яких визначення номерів автомобілів, визначення об'єктів на дорогах, розпізнавання емоційного стану людини. Вирішення останнього завдання стало одним із найважливіших відкриттів у сфері розпізнавання об'єктів. Визначення емоцій людини за зображеннями особи було успішно впроваджено у сферу безпеки та у виробництво автомобілів з функцією автопілота.

Ціль роботи

Мета статті – розробити нейромережевий алгоритм, який здійснює розпізнавання емоційного стану людини за його зображеннями.

Навчання та тестування моделі

Keras – це високорівневий фреймворк для глибокого навчання, що дозволяє швидко створювати нейронні мережі. Фреймворк написаний мовою високорівневої Python і може працювати з різними бібліотеками для виконання обчислень на GPU, такими як, TensorFlow, Theano або CNTK. Keras надає простий інтерфейс для створення нейронної мережі та налаштування її шарів. Цей фреймворк підтримує безліч типів шарів нейронних мереж, наприклад, згорткові, рекурентні, шари підвибори та інші. Він також надає широкий вибір функцій активації, оптимізаторів та функцій втрат для гнучкого налаштування моделей.

OpenCV (OpenSource Computer Vision Library) – бібліотека комп'ютерного зору та обробки зображень з відкритим кодом. Написана високорівневою мовою програмування C++ і має інтерфейси для використання іншими мовами, включаючи Python. OpenCV надає безліч функцій та алгоритмів для роботи із зображеннями, серед яких виявлення об'єктів, розпізнавання облич, сегментація зображень, фільтрація. Бібліотека також підтримує роботу з відео та потоками зображень.

Технічне забезпечення

Навчання та тестування моделі здійснювалося за допомогою Google Colab – безкоштовного сервісу від Google, що дозволяє створювати та запускати Python-код у хмарному середовищі за допомогою браузера. Дане середовище надає доступ до графічних процесорів (GPU) і тензорних процесорів (TPU), що робить її корисною для навчання глибоких нейронних мереж та машинного навчання.

Результати та обговорення

Під час навчання моделей використовувалися додаткові функції, т.зв. зворотні виклики, які надає бібліотека Keras. Зворотні виклики дозволяють користувачеві налаштувати поведінку моделі під час навчання. У цій роботі під час навчання моделі були використані такі зворотні виклики:

- ModelCheckpoint – збереження моделі після кожної епохи;
- EarlyStopping – зупинка навчання моделі при досягненні зазначеної метрики певної точності;
- ReduceLROnPlateau – знижує швидкість навчання моделі, коли вказана метрика перестає покращуватись;

Висновки

Було розроблено нейромережевий алгоритм, який здійснює розпізнавання емоційного стану людини за його зображеннями. В архітектурі нейронної мережі є технологія пропускних з'єднань (skip connection), що дозволяє вирішити проблему загасаючого градієнта. Ефективність роботи алгоритму та його продуктивність можна оцінити за допомогою наступних метрик:

1. Точність. Вказує на правильність розпізнавання емоцій розробленою моделлю і є ймовірністю приналежності конкретного об'єкта до певного класу.

2. Крос-ентропійна функція помилки. Відображає різницю між фактичним результатом класифікації та очікуваним виходом. Реалізована модель була протестована на зображеннях, отриманих з послідовності відео через камеру пристрою.

Список використаних джерел

1. URL: https://viso.ai/deeplearning/vgg-very-deep-convolutional-networks/ (дата звернення 20.03.2026)

2. URL: https://www.geeksforgeeks.org/understandinggooglenet-model-cnn-architecture/ (дата звернення 10.03.2026)

Відомості про автора

Борковська Любов Олексіївна – доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення.

E-mail: liubov.borkovska@npp.kai.edu.ua

УДК 004.932

СИСТЕМНІ ОБМЕЖЕННЯ НЕЙРОННИХ АВАТАРІВ РЕАЛЬНОГО ЧАСУ: ЛАТЕНТНІСТЬ, ВАРТІСТЬ ТА VENDOR LOCK-IN

Андрій ВИНАРЧУК
Здобувач вищої освіти другого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Яна БЕЛОЗЬОРОВА, к.т.н., доц.

У роботі розглянуто основні інженерні проблеми створення високо-реалістичних аватарів на основі штучного інтелекту в сучасних програмних системах. Проаналізовано архітектурну складність таких рішень, особливості побудови мультимодальних систем, а також ключові обмеження, пов'язані із затримкою відповіді, ефектом «зловісної долини», високою вартістю експлуатації та залежністю від зовнішніх технологічних платформ. Розглянуто вплив регуляторних вимог до синтетичного контенту на архітектурні рішення.

Ключові слова: *нейронні аватари, штучний інтелект, мультимодальні системи, програмна інженерія, затримка відповіді, ефект зловісної долини, vendor lock-in, EU AI Act.*

Вступ

За прогнозами аналітиків ринку, обсяг сегменту цифрових аватарів перевищить 17 млрд доларів США до 2028 року [4]. Разом із тим технологічна зрілість таких систем суттєво відстає від комерційних очікувань, що робить аналіз їхніх інженерних обмежень особливо актуальним. У сучасних програмних системах технології штучного інтелекту все активніше застосовуються не лише для обробки тексту, а й для побудови інтерактивних цифрових персонажів, здатних відтворювати зовнішність, голос, міміку та поведінку людини. Такі системи прийнято називати ШІ-аватарами або нейронними аватарами – обидва терміни є рівнозначними у контексті цієї роботи й відображають загальноприйняту термінологію галузі. Вони знаходять застосування у сфері підтримки користувачів, маркетингу, освіти, цифрових сервісів та персоналізованої взаємодії з клієнтами. Особливу увагу привертають високо-реалістичні ШІ-аватари, оскільки вони забезпечують більш природний і переконливий формат спілкування між людиною та програмною системою. Водночас досягнення високого рівня реалістичності є складним інженерним завданням. Для цього необхідно поєднати в межах єдиної системи декілька складових: обробку тексту, синтез мовлення, візуальне відтворення аватара, узгодження міміки та мовлення, а також механізми передавання й відображення результату на пристрої користувача.

На відміну від багатьох традиційних програмних продуктів, у подібних системах користувач очікує швидкої, природної та узгодженої відповіді. Навіть незначна затримка, порушення узгодженості між голосом і візуальним рядом або нестабільність окремих компонентів істотно знижують якість сприйняття. Саме тому проблема створення високо-реалістичних ШІ-аватарів є актуальною не лише для галузі штучного інтелекту, а й для інженерії програмного забезпечення в цілому.

Ціль роботи

Мета роботи – аналіз основних технічних, архітектурних, економічних та регуляторних проблем створення високо-реалістичних ШІ-аватарів у сучасних програмних системах. Для

досягнення цієї мети поставлено такі завдання: визначити основні складові системи високо-реалістичного ШІ-аватара; проаналізувати труднощі інтеграції окремих компонентів у межах єдиної програмної системи; розглянути вплив затримки відповіді на сприйняття цифрового персонажа користувачем; дослідити ефект «зловісної долини» як фундаментальне обмеження рівня реалістичності; оцінити ризики залежності від зовнішніх постачальників технологічних рішень; охарактеризувати вплив вартості обчислень та інфраструктури на можливість практичного впровадження таких систем; проаналізувати регуляторні вимоги до синтетичного контенту та їх вплив на архітектуру системи.

Матеріали та методи

У роботі використано аналітичний підхід, що включає порівняльний аналіз архітектур сучасних систем нейронних аватарів (HeyGen, D-ID, Synthesia) та узагальнення підходів до побудови складних програмних систем з кількома каналами взаємодії. Методологічною основою дослідження є структурний аналіз архітектури ШІ-аватара як сукупності взаємопов'язаних програмних модулів. Узагальнену структуру типової системи наведено на рис. 1.

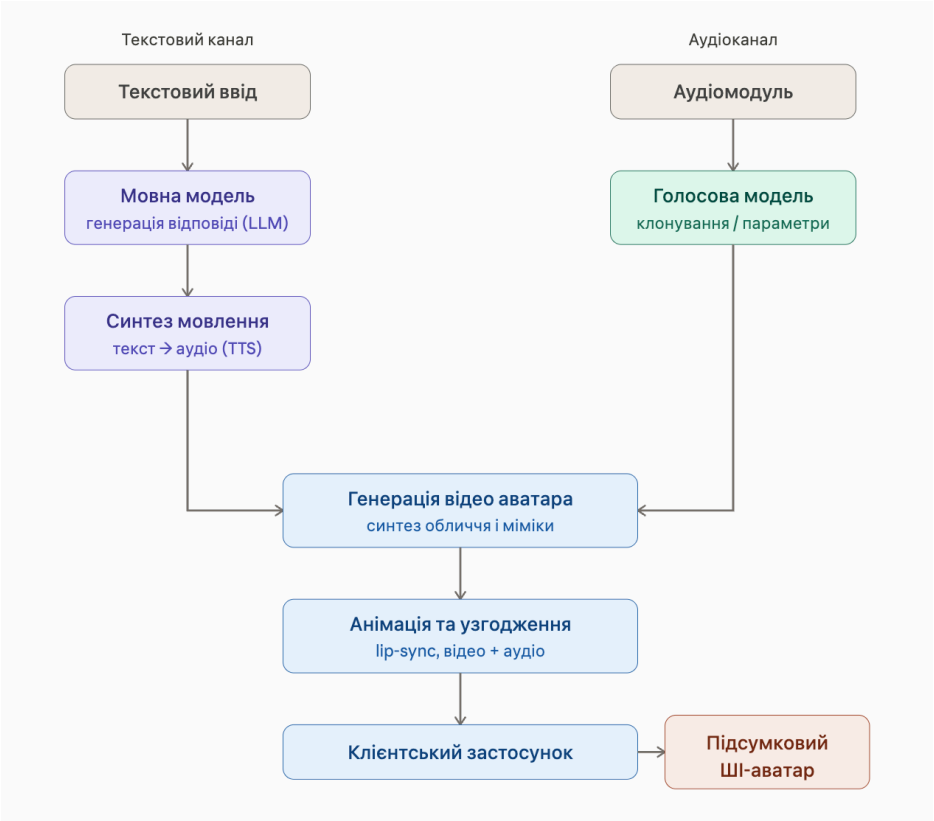


Рис. 1 – Структура типової системи високо-реалістичного ШІ-аватара

Типова система високо-реалістичного ШІ-аватара включає:

- компонент приймання й аналізу запиту користувача;
- мовну модель для формування відповіді;
- модуль синтезу мовлення;
- аудіомодуль для формування голосових характеристик персонажа;
- модуль візуальної генерації або анімації аватара;
- засоби узгодження звукового та візуального супроводу;
- механізми передавання даних;

– клієнтський застосунок для відображення результатів взаємодії.

Окремою складовою типової системи реалістичного ШІ-аватара є аудіомодуль, призначений для формування індивідуальних голосових характеристик цифрового персонажа. Його основна функція полягає у клонуванні голосу: на основі референсного аудіозапису система визначає параметри тембру, інтонації та ритму мовлення, які надалі передаються модулю синтезу мовлення. Водночас включення аудіомодуля є додатковим джерелом інженерних складнощів: обробка референсного аудіо потребує значних обчислювальних ресурсів, а залежність від зовнішніх сервісів клонування голосу формує додаткові ризики, пов’язані зі зміною умов доступу та вартості послуг.

Для дослідження також застосовано поділ основних проблем на технічні, архітектурні, економічні та організаційні. Це дало змогу розглядати ШІ-аватар не як окремий алгоритм, а як повноцінний програмний продукт із підвищеними вимогами до швидкодії, масштабованості, надійності та якості взаємодії з користувачем. Сучасні наукові праці показують, що створення реалістичних цифрових аватарів є комплексною задачею, яка поєднує генерацію візуального образу, синтез мовлення, відтворення міміки та узгодження кількох інформаційних модальностей у межах єдиної програмної системи. В оглядових дослідженнях, присвячених генерації відео з людиною та синтезу «мовних портретів», наголошується, що основними труднощами залишаються забезпечення візуальної правдоподібності, узгодженості рухів обличчя, часової стабільності кадрів і коректне оцінювання якості отриманого результату [1, 2]. Це дає підстави розглядати високо-реалістичний ШІ-аватар не як окрему модель, а як багатокомпонентну програмну систему, у якій кінцева якість залежить від узгодженої роботи всіх складових.

Важливу роль відіграє також синтез мовлення, оскільки саме він значною мірою визначає природність сприйняття цифрового персонажа користувачем. У науковому огляді з нейронного синтезу мовлення зазначено, що сучасні підходи суттєво підвищили природність і розбірливість штучного голосу, однак для практичного впровадження все ще важливими залишаються питання виразності, стійкості роботи, адаптивності та обчислювальної ефективності [3]. У поєднанні з візуальним модулем це означає, що під час розробки ШІ-аватара необхідно шукати баланс між якістю голосового супроводу, складністю інтеграції, вартістю обчислень і загальною стабільністю програмної системи.

Результати та обговорення

Однією з головних особливостей високо-реалістичних ШІ-аватарів є складність їхньої побудови. Якщо звичайна діалогова система може обмежуватися лише формуванням текстової відповіді, то ШІ-аватар потребує скоординованої роботи кількох підсистем. Генерація тексту, перетворення тексту на мовлення, анімація обличчя, візуалізація персонажа та передавання результату користувачу мають функціонувати узгоджено. Порушення роботи хоча б одного з цих компонентів негативно впливає на загальне сприйняття системи.

Це означає, що створення ШІ-аватара є задачею не лише добору ефективної моделі штучного інтелекту, а й правильної побудови всієї програмної архітектури. У таких системах важливо забезпечити стабільну взаємодію між модулями, передбачити обробку збоїв, механізми відновлення роботи, засоби контролю стану системи та накопичення технічної інформації про її функціонування. Без цього навіть технологічно складне рішення не забезпечить належної якості в реальних умовах використання.

Специфічною проблемою саме для аватарів є ефект «зловісної долини» (uncanny valley), описаний ще у 1970 році [5]. Що реалістичніший цифровий персонаж, то помітнішою

стає будь-яка дрібна помилка – неточний рух повіки, мікровідставання губ від звуку, неприродне кліпання. Користувач, який спокійно сприймає стилізованого аватара, різко відчуває дискомфорт від «майже людського». Для інженера це жорстке обмеження: система мусить або досягти майже бездоганної якості, або свідомо залишитись у зоні стилізації.

Найгостріше серед технічних обмежень проявляється затримка між дією користувача та відповіддю системи. Для цифрового персонажа ця характеристика має принципове значення, оскільки безпосередньо впливає на відчуття природності взаємодії. Дослідження у сфері взаємодії людини та комп'ютера показують, що затримка до 150 мс сприймається як природна реакція, тоді як понад 400 мс – критично знижує відчуття присутності та довіру до персонажа [7]. Затримка може виникати на різних етапах: під час обробки запиту, синтезу мовлення, побудови анімації, передавання даних та відображення результату на пристрої користувача.

Особливо складними є сценарії, у яких система повинна реагувати майже одразу після запиту. Для зменшення затримки використовують попередню підготовку окремих компонентів, буферизацію, оптимізацію передавання даних і часткове спрощення окремих елементів системи. Однак подібні рішення нерідко вимагають компромісу між якістю візуального чи звукового результату, вартістю обчислень і складністю реалізації.

Серйозним системним ризиком є також значна залежність від зовнішніх постачальників технологічних рішень. У багатьох сучасних системах окремі складові ШІ-аватара реалізуються не самостійно, а за допомогою сторонніх сервісів. Це можуть бути сервіси генерації тексту, синтезу мовлення, побудови аватарів, хмарної інфраструктури та обробки мультимедійних даних. Такий підхід прискорює створення першої працездатної версії продукту, проте водночас формує сильну залежність від зовнішніх платформ. У міжнародній практиці таке явище позначається терміном vendor lock-in – стан, за якого перехід на альтернативні рішення стає технічно та економічно невигідним.

Подібна залежність створює низку ризиків. До них належать зміна вартості послуг, оновлення інтерфейсів взаємодії, технічні обмеження, обмеження доступу до окремих функцій, а також загальна нестабільність роботи зовнішнього сервісу. У разі потреби переходу на інше рішення розробники можуть зіткнутися зі значними технічними та часовими витратами. Отже, при побудові високо-реалістичних ШІ-аватарів важливо враховувати не лише швидкість впровадження, а й довгострокову стійкість архітектури.

Суттєвим обмеженням також є висока вартість створення та подальшої підтримки подібних систем. Витрати виникають не лише на етапі початкової розробки, а й у процесі повсякденної експлуатації. Значні обчислювальні ресурси потрібні для роботи мовних моделей, синтезу мовлення, побудови анімації, обробки мультимедійних потоків і функціонування серверної інфраструктури. У результаті високо-реалістичний ШІ-аватар є суттєво дорожчим у підтримці, ніж звичайна текстова або голосова система.

Ця проблема особливо загострюється зі збільшенням кількості користувачів. Якщо на початковому етапі витрати можуть бути прийнятними, то при масштабуванні вони швидко зростають. Саме тому розробники змушені шукати баланс між ступенем реалістичності аватара, швидкістю відповіді системи та економічною доцільністю її використання.

Окремим чинником, який безпосередньо впливає на архітектурні рішення, є регуляторний тиск. Прийнятий у 2024 році Акт ЄС про штучний інтелект (EU AI Act) зобов'язує маркувати синтетичний відеоконтент і вести журнали його генерації [6]. Для розробників ШІ-аватарів це означає вбудовування механізмів водяного маркування

(watermarking) та аудиту безпосередньо в архітектуру системи, що додає як обчислювальне навантаження, так і складність інтеграції.

Окремої уваги заслуговує якість взаємодії з користувачем. Для користувача система сприймається як єдине ціле, а не як набір окремих технологічних рішень. Якщо голос звучить природно, але міміка виглядає неприродно, або якщо візуальний образ якісний, але відповідь з'являється із відчутною затримкою, загальне враження від системи погіршується. Отже, ключовим є не лише високий рівень окремих складових, а й узгодженість усієї взаємодії.

З інженерної точки зору це вимагає повного тестування сценаріїв використання, контролю якості на різних етапах роботи системи, аналізу технічних показників, фіксації помилок інтеграції та розробки механізмів безпечного спрощення функціональності в разі перевантаження або збоїв. У деяких випадках доцільно свідомо зменшити рівень складності окремих компонентів, якщо це дозволяє забезпечити вищу стабільність і передбачуваність роботи всього продукту.

Проведений аналіз показує, що основними проблемами створення високо-реалістичних ШІ-аватарів є складність архітектури, затримка відповіді, ефект зловісної долини, залежність від зовнішніх платформ, регуляторні вимоги, висока вартість підтримки та підвищені вимоги до якості взаємодії з користувачем. Усі ці чинники необхідно враховувати ще на етапі проєктування програмної системи.

Висновки

У результаті проведеного аналізу встановлено, що створення високо-реалістичних ШІ-аватарів є складним міждисциплінарним завданням, яке поєднує технології штучного інтелекту, мультимедійні засоби, мережеву взаємодію та методи інженерії програмного забезпечення.

Визначено, що основними обмеженнями таких систем є: складність інтеграції великої кількості взаємопов'язаних компонентів; затримка формування та відтворення відповіді (критичний поріг – 150–400 мс); ефект зловісної долини як фундаментальне обмеження рівня реалістичності; залежність від зовнішніх технологічних сервісів (vendor lock-in); регуляторні вимоги до синтетичного контенту (EU AI Act, 2024); висока вартість розробки та експлуатації.

Реалістичний ШІ-аватар – це не окрема модель, а інженерна система з жорсткими вимогами до узгодженості всіх компонентів. Подальші дослідження доцільно спрямувати на розробку уніфікованих архітектурних патернів для таких систем, методів зниження обчислювальних витрат без суттєвої втрати якості, а також на формування галузевих підходів до відповідності регуляторним вимогам щодо синтетичного контенту.

Список використаних джерел

1. Lei W., Wang J., Ma F., Huang G., Liu L. A Comprehensive Survey on Human Video Generation: Challenges, Methods, and Insights. arXiv preprint. 2024. arXiv:2407.08428.

2. Gowda S. N., Pandey D., Gowda S. N. From Pixels to Portraits: A Comprehensive Survey of Talking Head Generation Techniques and Applications. arXiv preprint. 2023. arXiv:2308.16041.

3. Tan X., Qin T., Soong F., Liu T.-Y. A Survey on Neural Speech Synthesis. arXiv preprint. 2021. arXiv:2106.15561.

4. MarketsandMarkets. Digital Human Avatar Market – Global Forecast to 2028. 2023. URL: https://www.marketsandmarkets.com

5. Mori M. The Uncanny Valley. IEEE Robotics & Automation Magazine. 2012. Vol. 19, No. 2. P. 98–100. (Original: Energy. 1970. Vol. 7(4). P. 33–35.)

6. European Parliament. Regulation (EU) 2024/1689 on Artificial Intelligence (EU AI Act). Official Journal of the European Union. 2024.

7. Nielsen J. Response Times: The 3 Important Limits. Nielsen Norman Group. 1993. URL: https://www.nngroup.com/articles/response-times-3-important-limits/

Відомості про автора:

Винарчук Андрій Володимирович – здобувач вищої освіти другого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси*: інженерія програмного забезпечення, штучний інтелект, проєктування програмних систем, мультимодальні цифрові технології.

E-mail: vinarchuk.a.v@gmail.com

УДК 37.091.26:004.021

АДАПТИВНИЙ АЛГОРИТМ КЛАСИФІКАЦІЇ ТИПОВИХ ПОМИЛОК КОРИСТУВАЧА ДЛЯ ФОРМУВАННЯ НАВЧАЛЬНИХ ЗАВДАНЬ

Олександр ВИНОГРАДОВ
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Наталія ШИБИЦЬКА
к.т.н., доц., доцент кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

Запропоновано підхід до побудови адаптивного алгоритму класифікації типових помилок користувача, у якому результат аналізу відповіді використовується не лише для фіксації факту помилки, а й для оновлення поточного профілю навчальних труднощів і параметрів наступного завдання. Алгоритм поєднує правила предметної області та інтерпретовані моделі машинного навчання, що дає змогу розрізняти концептуальні, процедурні та випадкові помилки, а також помилки, зумовлені когнітивним перевантаженням, і додатково виявляти стійкі повторювані патерни з урахуванням рівня знань користувача, історії відповідей, частоти повторення помилки та складності завдання.

Ключові слова: *адаптивний алгоритм, класифікація помилок користувача, формування навчальних завдань, профіль навчальних труднощів, інтерпретовані моделі, стан знань користувача.*

Вступ

У цифрових навчальних середовищах персоналізація часто зводиться до зміни загального рівня складності на основі підсумкового бала. Такий підхід недостатній, оскільки не враховує структуру індивідуальних помилок користувача, а отже не дозволяє відрізнити нестачу поняттєвого розуміння від процедурного збою або випадкової неуважності. Орієнтація освітнього процесу на потреби здобувача освіти узгоджується із загальними засадами сучасної освіти [1], а ідея адаптації навчального контенту до моделі знань користувача підтримується дослідженнями комп’ютерних систем навчання, у яких student model розглядається як основа організації індивідуалізованого навчального процесу [2]. Проте для практичної реалізації персоналізації недостатньо лише оцінити правильність відповіді; необхідно автоматизовано інтерпретувати тип помилки й пов’язати цю інтерпретацію з вибором наступного навчального кроку.

Ціль роботи

Метою роботи є розроблення адаптивного алгоритму класифікації типових помилок користувача для формування індивідуальних навчальних завдань. На відміну від статичних схем, запропонований підхід розглядає помилку користувача як динамічну подію, інтерпретація якої залежить від поточного рівня знань користувача, історії відповідей, повторюваності відхилень і когнітивного навантаження. Це дозволяє безпосередньо пов’язати результат класифікації з автоматизованим вибором теми, складності та форми подання наступного навчального кроку.

Матеріали та методи

Для інтерпретації помилок користувача доцільно використовувати дворівневу схему. На базовому рівні виділяються чотири класи помилок: концептуальні, процедурні, помилки неуважності та помилки, зумовлені когнітивним перевантаженням. Додатково враховується ознака повторюваності, яка фіксує стійкий патерн відхилень у межах певної підтеми. Концептуальна помилка вказує на неправильне розуміння правила, поняття або зв'язку між сутностями; процедурна - на порушення послідовності дій за збереження загального напряму розв'язання; помилка неуважності характеризується одиничним відхиленням за достатнього рівня володіння темою; помилка, зумовлена когнітивним перевантаженням, виявляє ситуацію, коли відповідь є неправильною не лише через незнання, а й через невідповідність складності завдання поточному стану користувача. Ознака повторюваності дає змогу відокремити одноразове відхилення від стійкого механізму помилки та використовується для адаптації наступних завдань.

Вхід алгоритму доцільно формалізувати як сукупність поточної спроби, стану знань, профілю труднощів та історії відповідей:

$$X_t = \langle A_t, K_t, E_t, H_t, C_t, M_t \rangle, \quad (1)$$

де A_t – параметри поточної відповіді; K_t – поточний стан знань; E_t – профіль типових помилок; H_t – історія попередніх взаємодій; C_t – контекст виконання поточного завдання (тема, рівень складності, час виконання, кількість спроб); M_t – параметри форми подання навчального матеріалу або завдання. Для побудови рішення використовуються ознаки тематичної належності завдання, рівня складності, часу виконання, кількості спроб, характеру неправильної відповіді, частоти повторення помилки користувача, динаміки успішності, стабільності результатів та поточного рівня знань користувача. Частина ознак обробляється правилами предметної області, які добре працюють для формальних патернів на кшталт синтаксичної помилки, невідповідності одиниць або хибної послідовності кроків. Для узагальнення складніших залежностей доцільно використати інтерпретовані моделі машинного навчання, зокрема дерево рішень, випадковий ліс або логістичну регресію [3-5].

Оцінювання рівня знань не слід зводити до одного числа. Доцільно підтримувати багатовимірний стан знань за темами, підтемами та рівнями складності. Його оновлення після кожної спроби може виконуватися рекурсивно:

$$K_{t+1} = clip\big((1 - \alpha_t)K_t + \alpha_t \cdot obs_t, 0{,}1\big), \quad (2)$$

де obs_t – результат поточної взаємодії з урахуванням правильності відповіді та тяжкості помилки, а α_t – ваговий коефіцієнт, що залежить від довіри до класифікації та новизни спостереження. Подібне уявлення стану знань узгоджується з підходами до структурування змісту навчання та побудови траєкторії руху в структурі знань залежно від вихідного рівня навченості [6].

Результати та обговорення

Логіка роботи алгоритму має циклічний характер: користувач виконує завдання; система фіксує результат і контекст відповіді; оновлює оцінку поточного рівня знань; виділяє ознаки помилки користувача; класифікує помилку; оновлює профіль труднощів; формує

наступне завдання; цикл повторюється. Принципово важливо, що адаптивність стосується не лише добору наступного завдання, а й самої інтерпретації помилки користувача. Одна й та сама неправильна відповідь не має фіксованого змісту.

Основну логіку адаптивного контуру, у якому класифікація помилки поєднується з оновленням стану знань, профілю навчальних труднощів і параметрів наступного завдання, наведено на рис. 1.

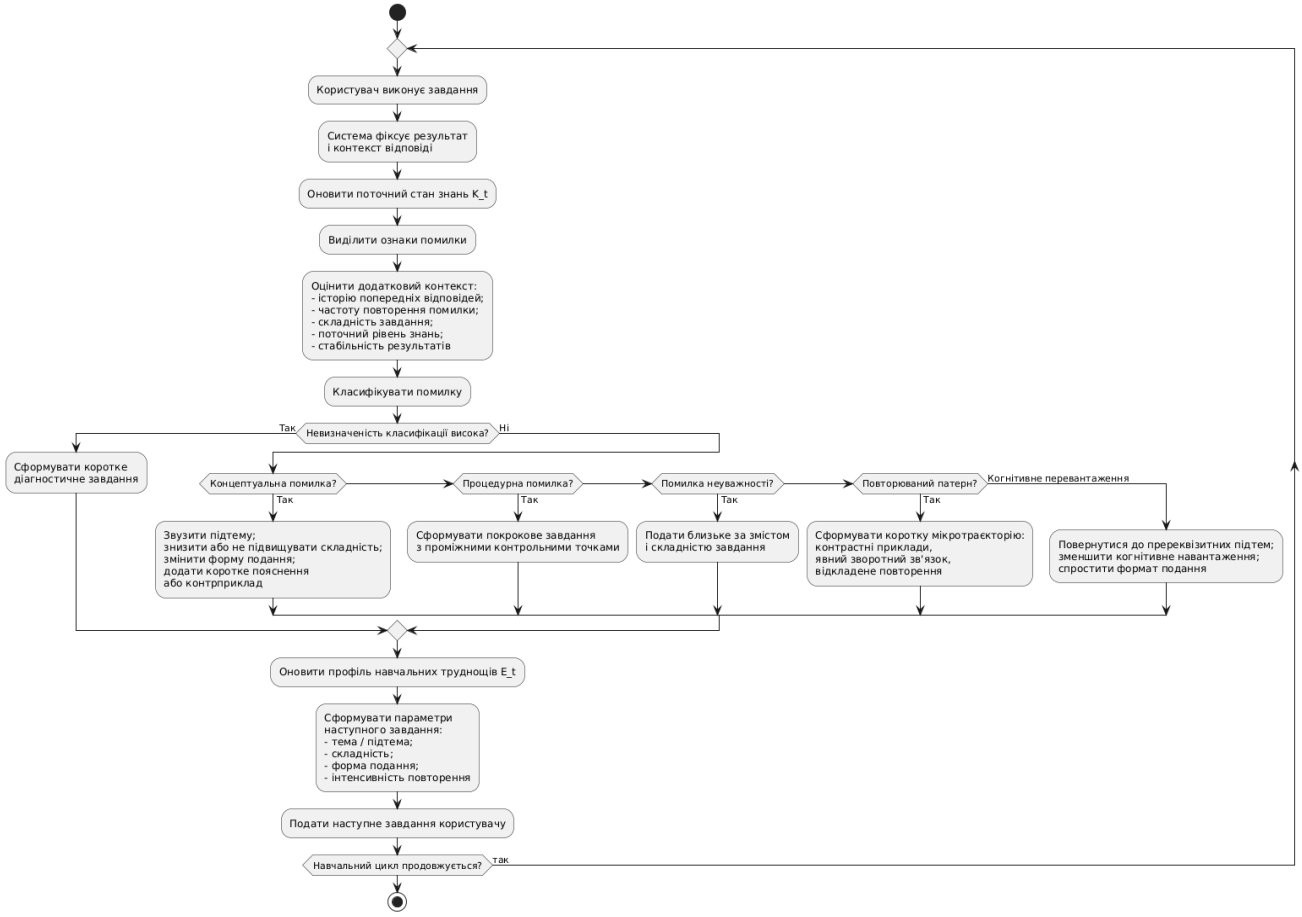


Рис. 1 – Основна частина адаптивного алгоритму класифікації типових помилок користувача та формування наступного навчального завдання

Наприклад, одиничну синтаксичну помилку в записі відповіді за високого рівня знань, стабільних попередніх результатів і відсутності повторення доцільно трактувати як помилку неуважності. Та сама помилка, якщо вона багаторазово повторюється в межах однієї підтеми, уже є підставою для інтерпретації процедурної помилки з ознакою повторюваності. Якщо ж неправильний принцип зберігається на різних формах подання завдань, помилку слід трактувати як концептуальну з відповідним повторюваним патерном. Аналогічно серія невдалих спроб лише на високому рівні складності за успішного виконання простіших підготовчих завдань свідчить не стільки про нерозуміння теми, скільки про когнітивне перевантаження. Отже, семантика помилки залежить від накопиченого профілю навчальних труднощів і не може визначатися ізольовано від історії взаємодії.

Результат класифікації безпосередньо впливає на параметри наступного завдання. Для концептуальної помилки доцільно зберігати тему, але звужувати підтему, знижувати або не підвищувати складність і змінювати форму подання матеріалу, додаючи коротке пояснення чи контрприклад. Для процедурної помилки доцільно формувати покрокові завдання з

проміжними контрольними точками. У разі помилки неуважності наступне завдання має бути близьким за змістом і складністю, оскільки різке повернення до простішого матеріалу буде необґрунтованим. Для стійкого повторюваного патерну ефективною є коротка мікротраєкторія з контрастних прикладів, явного зворотного зв'язку та відкладеного повторення. Для когнітивного перевантаження необхідним є повернення до пререквізитних підтем, зменшення когнітивного навантаження та корекція формату подання.

Стратегію формування наступного завдання можна подати узагальненою залежністю:

$$S_{t+1} = f(c_t, K_t, RP_t, U_t, \Delta d_t), \tag{3}$$

де c_t – базовий клас помилки; K_t – поточний стан знань; RP_t – індекс повторюваності; U_t – невизначеність класифікації; Δd_t – розрив між складністю завдання й поточними можливостями користувача. За високої невизначеності алгоритм повинен обирати не агресивну адаптацію, а коротке діагностичне завдання, що уточнює причину відхилення.

Таким чином, алгоритм працює як станозалежний контур прийняття рішень, у якому класифікація помилки, оновлення стану знань та генерація наступного завдання є взаємопов'язаними елементами. Це дозволяє перейти від реакції на окрему неправильну відповідь до формування індивідуальної мікротраєкторії навчання.

Висновки

Запропонований підхід дозволяє розглядати типову помилку користувача не як статичну мітку, а як результат адаптивної інтерпретації, що залежить від поточного стану знань, історії відповідей, частоти повторення помилки та складності завдання. Науково-прикладна цінність алгоритму полягає у поєднанні правил предметної області та інтерпретованих моделей машинного навчання для класифікації базового класу помилки, виявлення повторюваних патернів і керування параметрами наступного навчального завдання. Практичне значення підходу полягає у можливості формувати тему, складність, форму подання та інтенсивність повторення завдань відповідно до індивідуального профілю труднощів користувача.

Список використаних джерел

1. Про освіту : Закон України від 05.09.2017 № 2145-VIII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/2145-19 (дата звернення: 08.04.2026).

2. Shybytska N. Object-Oriented Model of Fuzzy Knowledge Representation in Computer Training Systems // *Advances in Computer Science for Engineering and Education VI* / eds. Z. Hu, I. Dychka, M. He. Lecture Notes on Data Engineering and Communications Technologies. Vol. 181. Cham : Springer, 2023. P. 116–125. DOI: 10.1007/978-3-031-36118-0_11.

3. Baker R. S., Inventado P. S. Educational Data Mining and Learning Analytics // *Learning Analytics: From Research to Practice* / eds. J. A. Larusson, B. White. New York : Springer, 2014. P. 61–75.

4. Breiman L. Random Forests // *Machine Learning*. 2001. Vol. 45, No. 1. P. 5–32. DOI: 10.1023/A:1010933404324.

5. Carvalho D. V., Pereira E. M., Cardoso J. S. Machine Learning Interpretability: A Survey on Methods and Metrics // *Electronics*. 2019. Vol. 8, No. 8. Article 832. DOI: 10.3390/electronics8080832.

6. Шибицька Н. М. Логіко-семантична модель структурування змісту навчання // *Modern international relations: current problems of theory and practice* : collective monograph / under general editorship of W. Welskop, Y. O. Voloshin. Łodz – Kyiv, 2021. P. 353–360.

Відомості про авторів

Виноградов Олександр Олегович – здобувач вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* Інтелектуальні системи підтримки прийняття рішень (DSS).

E-mail: 8008150@stud.kai.edu.ua

Шибицька Наталія Миколаївна – доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інтелектуальні системи навчання та експертні методи оцінювання, методи та моделі штучного інтелекту, нечітка математика та нейронні мережі.

E-mail: shibnatnik@ukr.net

УДК 004.77(043.2)

ПРОЄКТУВАННЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ УПРАВЛІННЯ ЗАЯВКАМИ ДЛЯ ВЕЛИКИХ СОЦІАЛЬНИХ ІНФРАСТРУКТУР: ДОСВІД ТА РЕЗУЛЬТАТИ

Станіслав ГОГУЛЯ
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Марія ВАСИЛЬЄВА, старший викладач

У роботі розроблено веборієнтовану систему збору та обробки заявок для студентського містечка на 5000 мешканців, що замінює паперовий документообіг. Система реалізована на стеку Node.js/PostgreSQL/React.js з інтеграцією Telegram Bot API та JWT-автентифікацією для розмежування ролей мешканця, виконавця та адміністратора. Впровадження дозволило скоротити середній час реакції на заявку з 7–10 до 1–2 діб і повністю усунути випадки втрати записів.

Ключові слова: *система управління заявками, студентське містечко, цифровізація, Node.js, React.js, PostgreSQL, Telegram Bot, JWT-автентифікація*

Вступ

Управління інфраструктурою студентського містечка, розрахованого на 5000 мешканців, є складним організаційним завданням, що вимагає ефективної координації між мешканцями та обслуговуючим персоналом. Традиційна модель фіксації заявок на ремонт та усунення несправностей через паперові журнали має суттєві недоліки: відсутність оперативного контролю стану виконання, втрата або пошкодження записів, неможливість аналізу навантаження та продуктивності. Відповідно до досліджень у сфері цифрової трансформації комунального господарства, автоматизація документообігу здатна скоротити час обробки заявок на 40–60% [1]. Зазначені проблеми зумовлюють актуальність розробки цифрової системи управління заявками, адаптованої до умов студентського містечка.

Ціль роботи

Метою роботи є проєктування та розробка веборієнтованої системи збору і обробки заявок для студентського містечка на 5000 мешканців, яка замінить паперовий документообіг, забезпечить систематизований облік і контроль виконання робіт, а також надасть адміністрації аналітичні інструменти для оцінки ефективності обслуговування.

Матеріали та методи

У процесі роботи застосовано методи аналізу предметної області, об'єктно-орієнтованого проєктування та прототипування інтерфейсів. Серверну частину реалізовано на основі фреймворку Node.js (Express) з реляційною базою даних PostgreSQL для зберігання заявок, категорій робіт та статусів виконання. Клієнтська частина побудована з використанням React.js, що забезпечує адаптивний інтерфейс для мешканців і адміністраторів. Для сповіщення персоналу про нові заявки інтегровано Telegram Bot API. Розмежування ролей (мешканець, виконавець, адміністратор) реалізовано засобами JWT-автентифікації. Дизайн бази даних передбачає зберігання повної історії змін статусів, що дозволяє проводити ретроспективний аналіз виконання [2].

Результати та обговорення

Розроблена система забезпечує повний цикл обробки заявки: від подачі мешканцем через вебформу або мобільний браузер до закриття заявки із підтвердженням. Мешканець може відстежувати поточний статус заявки в особистому кабінеті, а також залишати коментарі та оцінку якості виконання. Адміністратор отримує панель управління з розподілом заявок за категоріями (сантехніка, електрика, прибирання, загальне обладнання), пріоритетністю та будівлею. Виконавець бачить персональну чергу завдань із можливістю зміни статусу та завантаження фотофіксації результату. Система автоматично формує звіти про завантаженість персоналу, середній час закриття заявки та кількість повторних звернень. Тестування на реальних даних показало скорочення середнього часу реакції на заявку з 7–10 діб до 1–2 діб. Усунення паперових журналів повністю виключило випадки втрати заявок і дублювання записів.

Висновки

Розроблена система збору та обробки заявок є практичним інструментом цифровізації управління студентським містечком. Впровадження системи дозволяє повністю відмовитись від паперових журналів, систематизувати та пришвидшити виконання робіт, а також надає адміністрації прозорий контроль над процесами обслуговування. Перспективи подальшого розвитку передбачають інтеграцію з системою управління студентськими кімнатами, впровадження модуля планового технічного обслуговування та розробку мобільного застосунку для зручності мешканців.

Список використаних джерел

1. Laudon K. C., Laudon J. P. Management Information Systems: Managing the Digital Firm. 16th ed. Pearson, 2020. 672 p.

2. PostgreSQL Documentation. [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/

3. Telegram Bot API. Official Documentation. [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api

Відомості про автора

Гогуля Станіслав Сергійович – здобувач вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* проєктування та розробка вебсистем.

E-mail: 7844635@stud.kai.edu.ua

УДК 004.77 (043.2)

ЗАСТОСУВАННЯ ПАТЕРНА TRANSACTIONAL OUTBOX ДЛЯ ЗАБЕЗПЕЧЕННЯ КОНСИСТЕНТНОСТІ ДАНИХ У РОЗПОДІЛЕНИХ ВЕБСИСТЕМАХ

Вероніка ГОРБАЧ
Здобувачка вищої освіти першого рівня, кафедра ІПЗ
Вікторія ВОЛКОГОН
к.т.н., доц., доцент кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

У роботі досліджується синхронізація даних між реляційною базою даних та пошуковим індексом у розподілених вебсистемах. Проведено порівняльний аналіз підходів – синхронного подвійного запису, періодичного опитування та Change Data Capture. Обґрунтовано вибір патерна Transactional Outbox як оптимального рішення для гарантованої доставки повідомлень без ускладнення інфраструктури.

Ключові слова: *консистентність даних, Transactional Outbox, розподілені системи, Elasticsearch, PostgreSQL, CQRS, синхронізація даних, Change Data Capture, вебсистема.*

Вступ

Сучасні вебсистеми вимагають поєднання надійного зберігання структурованих даних та високошвидкісного багатофакторного пошуку. Використання виключно реляційних баз даних (наприклад, PostgreSQL) є неефективним для складного ранжування з динамічними ваговими коефіцієнтами. Це зумовлює необхідність побудови розподіленої гібридної архітектури, де паралельно з основною базою працює спеціалізований пошуковий рушій (наприклад, Elasticsearch). Основною інженерною проблемою таких систем стає забезпечення консистентності даних між незалежними вузлами. Класичний антипатерн синхронного «подвійного запису» у розподілених середовищах є ризикованим, оскільки мережеві або програмні збої неминуче призводять до незворотної розсинхронізації сховищ.

Ціль роботи

Метою дослідження є обґрунтування та вибір оптимального архітектурного методу забезпечення консистентності даних між реляційною базою та пошуковим індексом для майбутньої вебсистеми підтримки працевлаштування студентів, який унеможливить втрату даних при оновленні профілів без надмірного ускладнення інфраструктури.

Матеріали та методи

Оскільки використання класичних розподілених транзакцій (Two-Phase Commit) на рівні сучасних вебфреймворків є неефективним через жорстке блокування ресурсів, для проєктування архітектури обрано патерн Transactional Outbox [1]. Цей метод базуватиметься на використанні локальної транзакції бази даних PostgreSQL для одночасного виконання двох дій: збереження оновленої сутності у системі та фіксації повідомлення про це оновлення у спеціальній внутрішній таблиці-черзі (Outbox). Запропонований архітектурний потік повністю виключає зовнішні HTTP-запити до Elasticsearch під час відкритої транзакції. Натомість окремий фоновий процес періодично зчитуватиме невідправлені події з таблиці-черги, виконуватиме запити до пошукового рушія для оновлення індексу і маркуватиме події як

успішно опрацьовані. Такий підхід реалізує архітектурну модель CQRS (Command Query Responsibility Segregation) та забезпечує узгодженість у кінцевому підсумку [2].

Результати та обговорення

На етапі проєктування системи було проведено аналіз альтернативних підходів до синхронізації даних. Від впровадження синхронного подвійного запису було вирішено відмовитись через ризик критичної втрати даних у разі тимчасової недоступності пошукового кластера. Альтернативний метод періодичного опитування (Polling) наразі визнано неоптимальним, оскільки він створює надлишкове навантаження на реляційну базу та характеризується значною затримкою оновлення пошукової видачі. Високонадійним методом є використання інструментів Change Data Capture (CDC), які зчитують журнал транзакцій (Write-Ahead Log) на рівні самої БД. Однак розгортання додаткової інфраструктури потокової передачі даних робить цей підхід економічно та ресурсно недоцільним для проєктів, що не оперують надвеликими потоками даних. Отже, патерн Transactional Outbox обрано як найкращий інженерний компроміс. Він усуває проблему часткових відмов, гарантуючи доставку повідомлень до пошукового рушія (At-Least-Once delivery), та не потребує підключення зовнішніх брокерів повідомлень [3]. Логіка надійної синхронізації інкапсулюється на рівні існуючого бекенду.

Висновки

Проєктування підсистеми синхронізації на базі патерна Transactional Outbox дозволить ефективно розв'язати проблему консистентності даних між різнорідними сховищами у розподіленій вебсистемі. Запропонований підхід гарантуватиме, що жодне оновлення профілю не буде втрачено через тимчасові збої. Це забезпечить відмовостійку роботу аналізатора профілів та стабільний пошук без необхідності залучення ресурсомістких систем потокової обробки подій.

Список використаних джерел

4. Transactional outbox pattern. AWS Prescriptive Guidance. URL: https://docs.aws.amazon.com/prescriptive-guidance/latest/cloud-design-patterns/transactional-outbox.html (дата звернення: 04.04.2026).

5. Fowler M. CQRS. MartinFowler.com. URL: https://martinfowler.com/bliki/CQRS.html.

6. Implement the Transactional Outbox Pattern by Using Azure Cosmos DB. Microsoft Learn. URL: https://learn.microsoft.com/en-us/azure/architecture/databases/guide/transactional-outbox-cosmos (дата звернення: 04.04.2026).

Відомості про авторів

Горбач Вероніка Михайлівна – здобувачка вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* проєктування та розробка веб систем.

E-mail: 7936015@stud.kai.edu.ua

Волкогон Вікторія Олексіївна – доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення.

E-mail: viktoriia.volkohon@npp.kai.edu.ua

УДК 004.89

АЛГОРИТМ СЕМАНТИЧНОГО ПІДБОРУ МЕНТОРІВ ДО СТАРТАП-КОМАНД У CRM-СИСТЕМІ АКСЕЛЕРАТОРА

Каміла КОГУТ
Здобувачка вищої освіти другого рівня, кафедра ІПЗ
Валентина СКАЛОВА
Старший викладач кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

У статті розглядається проблема масштабованого підбору менторів для стартап-команд у межах CRM-системи стартап-акселератора. Запропоновано гібридний алгоритм семантичного матчингу на основі sentence embeddings та TF-IDF з ранжуванням за cosine similarity. Для оцінювання ефективності підходу було сформовано експериментальний датасет із 120 пар «команда–ментор». Результати експериментального дослідження показали, що запропонований алгоритм досягає значення NDCG@5 = 0,81, що перевищує результати базової тегової фільтрації. Модуль рекомендацій інтегровано у CRM-систему стартап-клубу у вигляді REST-сервісу.

Ключові слова: *рекомендаційна система, семантичний матчинг, embeddings, CRM, менторство, стартап-акселератор.*

Вступ

Стартап-клуби та акселератори активно використовують менторські програми для підтримки команд на різних етапах розвитку продукту. Однією з ключових проблем є ефективний підбір менторів відповідно до потреб конкретної стартап-команди. У більшості випадків процес виконується вручну адміністраторами або за допомогою базової фільтрації за тегами та категоріями. Такий підхід не масштабується при збільшенні кількості учасників екосистеми та не враховує семантичну близькість між суміжними доменами.

Наприклад, ментор із досвідом у fintech може бути корисним для стартапу в сфері insurtech або regtech, однак класична система фільтрації за тегами не визначить подібний зв’язок. У результаті частина потенційно релевантних менторів не потрапляє до списку рекомендацій.

Метою роботи є розробка алгоритму семантичного ранжування менторів для стартап-команд, здатного враховувати латентну суміжність доменів і технологій. Алгоритм реалізується як модуль рекомендацій у CRM-системі стартап-клубу з використанням сучасних методів обробки текстових даних та embeddings-моделей.

Матеріали та методи

Профіль кожного учасника формалізується вектором ознак F = {досвід, домени, технології, стадія спеціалізації}. Текстові поля (опис команди, bio ментора) перетворюються на щільні вектори за допомогою попередньо навченої моделі sentence-transformers/all-MiniLM-L6-v2 [1]. Для структурованих атрибутів застосовується зважена TF-IDF-схема. Результуючий вектор схожості отримується гібридним поєднанням обох підходів:

$$sim(T, M) = \alpha * cos(e\_T, e\_M) + (1 - \alpha) * tfidf(T, M) \quad (1)$$

де e_T та e_M – sentence embeddings команди та ментора відповідно, а $\alpha = 0{,}7$ визначено емпірично на валідаційній вибірці. Топ-N менторів із найвищим значенням sim потрапляють до рекомендованого списку. Модуль реалізовано на Python (FastAPI) і підключено до CRM через REST API.

Експериментальний датасет з 120 пар «команда-ментор» сформовано синтетично на основі профілів, згенерованих із відкритих джерел: bio менторів – з публічних профілів учасників акселераційних програм, опис команд – з агрегованих описів проєктів open-source та стартап-каталогів. Поля доменів і технологій нормалізовано за таксономією ESCO. Релевантність кожної пари визначалася формальним правилом: пара отримувала максимальний бал при збігу основного домену та підмножини технологій, нижчі бали – при семантичній суміжності доменів за класифікацією ESCO, та нульовий бал у решті випадків. Обмеженням є синтетичний характер ground-truth-розмітки: результати не можна напряму екстраполювати на реальні рекомендаційні задачі без додаткової валідації. Для уникнення витоку даних при підборі α датасет розділено на валідаційну та тестову частини у співвідношенні 1:2; метрики у Таблиці 1 обчислено на тестовій частині.

Для оцінювання алгоритму використовувалися метрики:

- NDCG@5 – оцінка якості ранжування рекомендацій;
- Precision@3 – точність рекомендацій у топ-3 результатах.

Результати та обговорення

Результати порівняння підходів наведено в Таблиці 1.

Таблиця 1

Порівняння підходів до підбору менторів

Метод	NDCG@5	Precision@3
Тегова фільтрація	0,66	0,58
TF-IDF	0,72	0,63
Sentence embeddings	0,78	0,71
Гібридний (proposed)	0,81	0,76

Запропонований гібридний алгоритм перевершує базові підходи за обома метриками. Особливо суттєва різниця проявляється у міждоменних парах: алгоритм коректно рекомендував fintech-менторів для insurtech- та regtech-команд у 87% випадків, тоді як тегова фільтрація – лише у 34%. Це підтверджує гіпотезу про те, що семантичний простір embeddings ефективно кодує суміжність доменів.

Час відповіді модуля у CRM становить менше 200 мс при базі до 500 менторів, що є прийнятним для інтерактивного використання.

Висновки

У роботі запропоновано гібридний алгоритм семантичного підбору менторів, що поєднує sentence embeddings та TF-IDF для ранжування рекомендацій у CRM-системі стартап-клубу.

Результати експериментального дослідження показали, що запропонований підхід забезпечує кращу якість рекомендацій порівняно з теговою фільтрацією та окремими методами текстового аналізу.

Практична реалізація алгоритму у вигляді REST-сервісу підтвердила можливість його інтеграції у CRM-системи стартап-акселераторів.

Перспективами подальших досліджень є врахування зворотного зв’язку менторів і команд, адаптивне налаштування вагових коефіцієнтів та використання великих мовних моделей для покращення рекомендацій.

Список використаних джерел

1. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of EMNLP 2019. URL: https://arxiv.org/abs/1908.10084 (дата звернення: 10.03.2026).

2. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge University Press, 2008. 544 с.

3. Järvelin K., Kekäläinen J. Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems. 2002. Vol. 20, No. 4. P. 422–446.

4. Aggarwal C. C. Recommender Systems: The Textbook. Springer, 2016. 498 с.

Відомості про авторів

Когут Каміла Вікторівна – здобувачка вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп’ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* рекомендаційні системи, NLP, CRM-системи.

E-mail: kamila.fly1703@gmail.com

Скалова Валентина Анатоліївна – старший викладач кафедри інженерії програмного забезпечення факультету комп’ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси*: інженерія програмного забезпечення, якість програмного забезпечення, супроводження, розгортання.

E-mail: valentyna.skalova@npp.kai.edu.ua

УДК 621.391

МЕТОД СТИСНЕННЯ ЦИФРОВИХ ЗОБРАЖЕНЬ У БАЗАХ ДАНИХ

Олена КОЛГАНОВА,
к.т.н., доц., доцент кафедри ІПЗ
Лідія ТЕРЕЩЕНКО,
к.т.н., доц., доцент кафедри кібербезпеки
Світлана КОРНІЄНКО
асистент, кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

Стаття присвячена розвитку і вдосконаленню методів стиснення зображень. Увагу приділено розвитку методу стиснення з втратами, де видаляються менш помітні дані, що дає високий коефіцієнт стиснення, але призводить до зниження якості. Основною задачею науковців стає знаходження такого алгоритмічного рішення, яке дозволить при більшому коефіцієнті стиснення залишити якомога вищу якість зображення.

Ключові слова: *алгоритм стиснення графічних даних, дискретне косинусне перетворення, багатомасштабний аналіз, сплайн-функції, зберігання даних.*

Вступ

Сьогодні алгоритми стиснення зображень є невід'ємною частиною сучасних інформаційних систем у різних галузях та сферах людської діяльності – від телекомунікацій та медицини до штучного інтелекту та оборонних технологій. Розвиток інформаційних технологій призвів до швидкого зростання обсягів мультимедійної інформації, саме тому стиснення зображень стає критично важливим компонентом цифрових систем у різних галузях [1, 2].

Основними сферами застосування методів стиснення зображень є:

1. Телекомунікації та мережеві сервіси. Застосування методів стиснення зображень може покращити пропускну здатність каналів зв'язку, зменшити затримки передачі даних та покращити якість обслуговування. Стиснення зображень дуже важливе для мобільних мереж з високим навантаженням та великою кількістю абонентів, мережевих сервісів та хмарних платформ, де ефективність стиснення впливає на швидкість доступу та стабільність послуг [1, 2, 3].

2. Супутникові та безпілотні авіаційні системи (БПЛА). Причиною використання методів стиснення зображень є обмежена пропускна здатність каналів зв'язку, енергетичні ресурси та високі вимоги до ефективності обробки даних. У супутникових системах така обробка зображень дозволяє ефективно передавати великі масиви даних дистанційного зондування Землі [1, 3]. У безпілотних літальних апаратах методи стиснення зображень є необхідною умовою для зменшення затримок у передачі потокового відео, забезпечення стабільного керування системою та зменшення навантаження на бортові системи. Важливими є такі адаптивні та інтелектуальні методи обробки зображень, які можуть змінювати параметри стиснення залежно від умов польоту, якості каналу зв'язку та обчислювальних ресурсів, тим самим сприяючи підвищенню надійності бортових радіосистем [1, 2].

3. Медицина. Ці методи відіграють ключову роль під час зберігання та передачі діагностичних даних (МРТ, КТ, рентгенівських, ультразвукових зображень) через медичні комп'ютерні мережі.

4. Системи комп'ютерного зору та штучного інтелекту. Методи стиснення зображень є основою для оптимізації зберігання навчальних зразків, зменшення обчислювальної потужності та прискорення передачі даних між вузлами обчислювальних систем. Алгоритми стиснення можуть бути безпосередньо включені до структури нейронних мереж, що дозволить об'єднати завдання кодування та розпізнавання образів в одне ціле.

5. Промисловий Інтернет речей (IIoT) та периферійні обчислення. Системи IIoT характеризуються використанням відеокамер високої роздільної здатності для окремих вузлів, що створює великі потоки даних. Стиснення зображень дозволяє зменшити навантаження на мережеву інфраструктуру за рахунок зменшення обсягу переданої інформації [16]. Адаптивні алгоритми стиснення на периферійних платформах сприяють реалізації концепції розумного виробництва Industry 4.0 [1].

6. Цифрові двійники. У концепції цифрових двійників візуальні дані використовуються для синхронізації фізичних процесів з їх віртуальними моделями. Стиснення даних створює передумови для можливості прогнозування збоїв, оптимізації технологічних процесів та підтримки управлінських рішень [1].

7. Радіолокаційне зображення. Алгоритми стиснення важливі на різних етапах обробки широкосмугових сигналів та зображень у численних радіолокаційних застосуваннях, включаючи радіолокаційну метеорологію.

Тому методи стиснення зображень є невід'ємною частиною Industry 4.0. Застосування цих методів може забезпечити масштабованість, адаптивність та ефективність інтелектуальної обробки візуальних даних.

Ціль роботи

Стиснення інформації є критично важливим у сучасному цифровому світі, оскільки обсяги даних постійно зростають, а ефективне управління цими даними стає все більш актуальним. Метою роботи є підвищення ефективності програмних засобів стиснення графічних даних в комп’ютерних системах передачі та зберігання інформації у базах даних. Основною задачею науковців стає знаходження такого математичного й алгоритмічного рішення, яке дозволить при більшому коефіцієнті стиснення залишити якомога вищу якість зображення.

Матеріали та методи

Сьогодні методи стиснення зображень переважно використовують три основні підходи: класичне стиснення на основі перетворення, відеоорієнтовані методи та глибоке навчання. Стандарт JPEG2000 є одним з найпоширеніших методів стиснення зображень і базується на дискретному вейвлет-перетворенні [2, 3]. На відміну від традиційного формату JPEG, заснованого на косинусному перетворенні, JPEG2000 дозволяє отримати кращу якість зображення при високих коефіцієнтах стиснення [1]. Основною перевагою цього стандарту є масштабованість та висока якість реконструкції, але метод вимагає високої обчислювальної складності.

Стандарт стиснення на основі HEVC (High Efficiency Video Coding) наразі активно використовується для стиснення зображень. Цей стандарт також має високу ефективність

завдяки складнішому прогнозуванню блоків та вдосконаленим методам стиснення ентропії. Однак недоліком цього стандарту є обчислювальна складність, зокрема під час декомпресії.

Стандарти, засновані на моделях стиснення глибокого навчання, побудовані на принципі використання автоенкодерів, згорткових нейронних мереж та трансформаторів. Ці підходи дозволяють досягти високої ефективності стиснення порівняно з класичними стандартами при низьких бітрейтах. Однак основним недоліком є тривалий процес навчання. Отже, огляд наукових джерел та прикладних напрямків демонструє, що методи стиснення зображень еволюціонували від класичних алгоритмів кодування до інтелектуальних, адаптивних та семантично орієнтованих підходів, інтегрованих з глибоким навчанням, кіберфізичними системами та технологіями Industry 4.0.

Результати та обговорення

Основні проблеми сучасних методів стиснення цифрових зображень полягають у компромісі між розміром файлу та якістю, складності обчислень, енергоспоживанні, а також у відсутності універсальних рішень для всіх типів зображень та умов використання. В майбутньому на вирішення цих проблем можуть вплинути нові технології, такі як штучний інтелект, квантові обчислення та нові стандарти стиснення, але залишаються значними викликами для галузі. Виходячи з проведеного огляду існуючих алгоритмів стиснення графічних даних та аналізу наведеного матеріалу, робиться постановка задачі дослідження: розробка власного кодеку з метою використання у системі стиснення графічної інформації для зберігання у базах даних.

Для боротьби із вказаними вище негативними явищами, що проявляються при стисненні графічних даних, перспективним є вдосконалення методів та алгоритмів компресії–декомпресії, у тому числі й несиметричних методів, у напрямку використання різних базисних функцій. Такі алгоритми повинні мати достатню швидкість обчислень для використання в режимі реального часу.

Цифрове стиснення є гнучкою технологією, оскільки рівні складності кодування, що використовуються, та ступінь стиснення можуть змінюватися залежно від програм. Основним принципом цифрового стиснення є використання так званої надмірності аудіо- або відеосигналу. Надмірність пояснюється тим, що аудіо та відео містять області зі схожими характеристиками. Таким чином, весь інформаційний потік можна умовно розділити на передбачувану частину (іншими словами, надмірність) та нову, непередбачувану частину (відому в теорії інформації як ентропія). Сума цих двох величин дає потік даних, зменшення якого залежатиме від того, наскільки добре ми зможемо зробити «прогноз». Теоретично можливо повністю виключити надмірність і залишити лише ентропію, але для цього знадобиться ідеальний алгоритм стиснення, який буде надзвичайно складним і займе невиправдано багато часу. Якщо ступінь стиснення настільки високий, що результуюча швидкість потоку даних стає меншою за ентропію, то інформація втрачається. На практиці коефіцієнт стиснення вибирається меншим за ідеальний, щоб забезпечити певний запас надійності. Це дає змогу використовувати простіші алгоритми та виконувати багаторазове відновлення/стиснення без помітної втрати якості. У побутовій техніці коефіцієнт стиснення може бути вищим, ніж у студійній, і якщо не потрібне багаторазове перезаписування, то деяка частина ентропії відкидається під час процесу стиснення [2].

Розглянемо стандартну послідовність операцій, яка використовується в алгоритмах стиснення зображень (на прикладі алгоритму JPEG). Основною операцією при стисненні з

втратами є ДКП в алгоритмі JPEG, або вейвлет-розклад у JPEG2000. В роботі пропонується на третьому кроці конвеєра застосувати сплайновий багатомасштабний розклад.

Розробка сплайнового багатомасштабного розкладу в роботі [3] дозволила під час розрахунку матриці планування за відомими формулами на кожному етапі отримати швидку реалізацію багатомасштабного аналізу з кратністю не 2 та зі змінною кратністю.

Висновки

У цій роботі розглянуто підхід до побудови алгоритму стиснення графічних даних для внутрішньої автоматизованої системи управління базою даних. Запропонована концепція враховує потребу в простоті, швидкому доступі до інформації, оскільки алгоритм має високу швидкодію. Представлена модель не є завершеним продуктом, однак може бути базою для подальшої реалізації. Реалізація такого проєкту сприятиме вдосконаленню алгоритмів компресії графічних даних та дозволить підвищити коефіцієнт стиснення.

Список використаних джерел

1. Bovik, A. Handbook of Image and Video Processing. Academic Press. USA, 2000. – 1429 p.

2. Burt, Peter J., Adelson, Edward H. The Laplacian Pyramid as a Compact Image Code, IEEE Transactions on Communication, Vol. Com-3l, No. 4, April 1983, pp. 532-540.

3. Y. Meyer, Wavelets and Operators, Cambridge University Press, Cambridge, 1993. – 225 p.

4. Shutko V., Tereshchenko L., Sitko A., Kravchenko V., Volkogon V., Zh. Vasylieva-Shalamova, Kolhanova O. Method for improving the efficiency of online communication systems based on adaptive multiscale transformation, 2020 10th International Conference ACIT’2020 (Germany, Deggendorf, September 16-18, 2020) – Conference Proceedings – PP. 824-829.

Відомості про авторів

Колганова Олена Олегівна – доцент кафедри інженерії програмного забезпечення факультету комп’ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* обробка зображень та сигналів інтелектуальні системи, математичне моделювання.

E-mail: olena.kolhanova@npp.kai.edu.ua

Терещенко Лідія Юріївна – доцент кафедри кібербезпеки факультету комп’ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* обробка зображень та сигналів інтелектуальні системи, математичне моделювання.

E-mail: lidiia.tereshchenko@npp.kai.edu.ua

Корнієнко Світлана Петрівна – асистент кафедри інженерії програмного забезпечення факультету комп’ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення, інтелектуальні системи.

E-mail: svitlana.korniienko@npp.kai.edu.ua

УДК 004.92

ДЕЯКІ АСПЕКТИ РОЗРОБКИ АЛГОРИТМІВ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ІГРОВИХ РІВНІВ

Максим КОНОНЕНКО
Здобувач вищої освіти другого рівня, кафедра ІТ
Сергій ГРІНЕНКО
к.т.н., доцент
Навчально-науковий інститут управління, технологій та правових наук
Національний транспортний університет

У роботі розглянуто аспекти розробки алгоритмів процедурної генерації контенту (PCG) для ігрових рівнів. Обґрунтовано доцільність PCG зниженням витрат розробки та підвищенням реіграбельності, особливо для жанру roguelike. Проаналізовано основні архітектурні підходи: функції когерентного шуму, генерацію на основі кімнат, клітинні автомати, дерева двійкового розбиття (BSP) та алгоритми пошуку з поверненням. Для оцінювання якості згенерованих рівнів запропоновано застосування методів теорії графів і пошукових алгоритмів – зокрема модифікованого пошуку в ширину (BFS) для перевірки прохідності, теплових карт для аналізу варіативності та евристик для розміщення об'єктів.

Ключові слова: *процедурна генерація контенту (PCG); ігрові рівні; алгоритми генерації; теорія графів; пошук у ширину (BFS); реіграбельність; roguelike.*

Вступ

Сучасна індустрія відеоігор активно використовує процедурну генерацію контенту (PCG) для автоматизованого створення ігрових рівнів, що дозволяє суттєво зменшити витрати ресурсів розробки та підвищити повторне використання ігор. Особливо це актуально для жанрів roguelike та інших динамічних ігрових систем, де кожен новий запуск повинен пропонувати унікальний досвід.

Процедурна генерація контенту (Procedural Content Generation, скорочено PCG) – це метод створення цифрових даних алгоритмічним шляхом, а не за допомогою прямого ручного маніпулювання чи редагування. У контексті розробки відеоігор PCG використовується для автоматизованого створення ігрових рівнів, текстур, 3D-моделей, квестів, правил гри та навіть музичного супроводу. Основна ідея полягає у використанні псевдовипадкових чисел у поєднанні з набором строгих математичних та логічних правил, що дозволяє комп'ютеру генерувати унікальний, але структурно коректний контент під час виконання програми.

Однією з головних причин широкого використання процедурної генерації в сучасній індустрії є суттєва економія ресурсів студії (Development Costs). Створення великомасштабних ігрових світів вручну вимагає тисяч годин монотонної роботи десятків левел-дизайнерів, 3D-художників та тестувальників. Автоматизація цього процесу дозволяє навіть невеликим незалежним (інді) студіям випускати масштабні проекти, успішно конкуруючи з великими AAA-компаніями. Замість того, щоб розставляти кожен об'єкт чи стіну на карті самостійно, розробник проектує «генератор», який робить це за нього за наперед заданими топологічними законами.

Іншим, ще більш значущим фактором є забезпечення надзвичайно високої реіграбельності (replayability). У традиційних іграх з фіксованим, лінійним дизайном рівнів

гравець, пройшовши гру один раз, досконало запам'ятовує розташування ворогів, пасток та нагород. Процедурна генерація фундаментально вирішує цю проблему: при кожному новому запуску (так званому «забігу») алгоритм створює абсолютно нову локацію. Гравець змушений покладатися не на завчену пам'ять, а на свої тактичні навички, реакцію та здатність швидко адаптуватися до непередбачуваних просторових умов.

Аналізуючи існуючі підходи в індустрії, варто виділити кілька ключових архітектурних методів генерації. Найвідомішим прикладом макро-генерації є використання функцій неперервного шуму (наприклад, Perlin Noise або Value Noise) для створення природних ландшафтів. Безперечним лідером цього підходу є гра Minecraft. Цей проект генерує практично нескінченний тривимірний воксельний світ, застосовуючи багатошарові математичні шуми для формування височин, гір, океанів, систем печер та логічного розподілу кліматичних біомів. Кожен згенерований світ визначається унікальним початковим числом – «сідом» (seed), що дозволяє гравцям ділитися генераціями та відтворювати абсолютно ідентичні світи на різних пристроях.

Інший, значно більш структурований підхід – це гібридна генерація на основі фрагментів або кімнат (Room-based / Chunk-based generation). У цьому випадку алгоритм не створює кожен блок з нуля. Замість цього він працює як конструктор, збираючи рівень із сотень невеликих «кімнат», які були заздалегідь спроєктовані та збалансовані левел-дизайнерами вручну. Відмінними прикладами такої системи є хітові інді-проекти Spelunky та Dead Cells. Програма випадковим чином обирає кімнати з бази даних, перевіряє їх на топологічну сумісність (щоб виходи з однієї кімнати логічно і фізично з'єднувалися з входами іншої) і вибудовує з них складний, багаторівневий лабіринт.

Окрім класичних платформерів та шутерів, алгоритми процедурної генерації активно інтегруються у жанри з глибокою тактичною та стратегічною складовою. До таких можна віднести сучасні автобатлери та RPG-проєкти. Наприклад, у концепціях ігор, що базуються на механіках синтезу персонажів та постійному подоланні поверхів (подібно до механік, описаних у манхві Pick Me Up!), процедурна генерація застосовується одразу на кількох рівнях абстракції. Алгоритм не лише створює симетричні або асиметричні бойові арени з різним тактичним рельєфом, але й процедурно підбирає хвилі супротивників, їхні синергії та нагороди. У таких випадках PCG виступає в ролі «віртуального ігрового режисера» (AI Director), який постійно аналізує математичний стан гри та адаптує згенерований контент під поточний прогрес користувача.

Методи та результати

Розглянуто основні підходи до процедурної генерації, серед яких: клітинні автомати, дерева двійкового розбиття (BSP), методи когерентного шуму та алгоритми пошуку з поверненням. Кожен із цих підходів формує різні типи ігрового простору – від органічних печер до структурованих підземель і лабіринтів.

Для оцінювання якості згенерованих рівнів запропоновано використання методів теорії графів та пошукових алгоритмів. Зокрема, застосування модифікованого алгоритму пошуку в ширину (BFS) дозволяє перевіряти прохідність рівня, визначати найкоротші маршрути та виявляти ізольовані області. Додатково використовується підхід теплових карт (heatmap), який дає змогу оцінити інтенсивність використання різних зон рівня та виявити альтернативні маршрути. Це дозволяє оцінювати не лише базову прохідність, але й тактичну варіативність ігрового простору. Важливим аспектом оцінювання є також логічність розміщення об’єктів.

Запропоновані евристичні методи аналізу геометрії рівня дозволяють автоматично визначати оптимальні позиції для ворогів, ресурсів і нагород, що підвищує якість геймплею.

Застосування математичних методів аналізу та алгоритмів пошуку дозволяє забезпечити якісне оцінювання процедурно згенерованих рівнів, що є ключовим фактором успішного використання PCG у сучасній ігровій індустрії.

Список використаних джерел

1. Togelius J., Yannakakis G. N., Stanley K. O., Browne C. Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games. 2011. Vol. 3, No. 3. P. 172–186.
2. Short T. X., Adams T. Procedural Generation in Game Design. Boca Raton : CRC Press, 2017. 334 p.
3. Hendrikx M., Meijer S., Van Der Velden J., Eliens A. Procedural content generation for games: A survey. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM). 2013. Vol. 9, No. 1. P. 1–22.
4. Smith G. Understanding procedural content generation: a design-centric analysis of the role of PCG in games. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2014. P. 917–926.
5. Yu D. Spelunky. Los Angeles : Boss Fight Books, 2016. 192 p.
6. Shaker N., Togelius J., Nelson M. J. Procedural Content Generation in Games. Cham : Springer, 2016. 208 p.

Відомості про авторів

Кононенко Максим Андрійович – здобувач вищої освіти другого рівня, Навчально-науковий інститут управління, технологій та правових наук, Національний транспортний університет. *Наукові інтереси*: моніторинг; хмарна інфраструктура.

E-mail: kononenkomaksim.a@gmail.com

Гріненко Сергій Анатолійович – доцент кафедри інформаційних технологій навчально-наукового інституту управління, технологій та правових наук, Національний транспортний університет. *Наукові інтереси*: інженерія програмного забезпечення.

E-mail: hrinenkosa@gsuite.duit.edu.ua

УДК 005.334 : 004.415.2

СУЧАСНІ МЕТОДИ ОЦІНЮВАННЯ ЙМОВІРНОСТІ ТА НАСЛІДКІВ РИЗИКІВ У ПРОГРАМНИХ ПРОЄКТАХ

Інна КОРНІЙЧУК
Здобувачка вищої освіти другого рівня, кафедра ІПЗ
Інна ЛАЩ
Здобувачка вищої освіти другого рівня, кафедра ІПЗ
Наталія ШИБИЦЬКА
к.т.н., доц., доцент кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

У роботі розглянуто сучасні підходи до оцінювання ризиків у програмних проєктах із використанням метрик розробки, методів машинного навчання та інтеграції у CI/CD-процеси. Запропоновано комплексний підхід, що поєднує кількісні моделі, аналіз дефектів та AI-алгоритми для підвищення точності прогнозування ризиків.

Ключові слова: *управління ризиками, програмні проєкти, CI/CD, машинне навчання, метрики якості ПЗ, дефекти інформаційних систем, AI.*

Вступ

У сучасних умовах розробка програмного забезпечення (ПЗ) є складним, багаторівневим та динамічним процесом, що супроводжується значною кількістю невизначеностей. Кожен програмний проєкт функціонує в умовах змінних вимог, обмежених ресурсів та високих вимог до якості, що зумовлює виникнення різноманітних ризиків. Такі ризики можуть впливати на строки виконання, вартість реалізації, якість програмного продукту та загальний успіх проєкту. До основних типів ризиків належать технічні, організаційні, ресурсні, а також ризики, пов’язані з людським фактором.

Ціль роботи

Метою роботи є підвищення точності оцінювання ризиків у програмних проєктах шляхом застосування інтелектуальних методів аналізу даних, метрик розробки та інтеграції з CI/CD-процесами.

Завдання дослідження: аналіз існуючих підходів до оцінювання ризиків; формалізація кількісної моделі ризику; дослідження можливостей використання метрик; застосування алгоритмів машинного навчання; інтеграція оцінювання ризиків у CI/CD.

Матеріали та методи

В роботы досліджується процес управління ризиками у програмних проєктах та методи оцінювання ризиків на основі метрик та AI. Оцінювання ризиків здійснюється на основі базової моделі. Також застосовуються: математичне очікування ризику; умовна ймовірність; кореляційний аналіз. Для проведення аналізу застосовуються такі метрики: кількість комітів, складність коду, щільність дефектів, рівень покриття тестами та кількість невдалих збірок. Методи дослідження: порівняльний аналіз, статистичний аналіз, а також методи машинного навчання (Random Forest, XGBoost, нейронні мережі).

Результати та обговорення

Результати проведеного аналізу свідчать, що традиційна модель ризик-менеджменту, побудована на детермінації ймовірності та ступеня впливу, характеризується високою доступністю, проте не повною мірою відображає динамічні зміни в життєвому циклі проєкту.

Хоча стандартна матриця «ймовірність – вплив» залишається найбільш поширеним інструментом, вона не враховує стохастичну природу змін у проєкті. Впровадження системи метрик забезпечує можливість об'єктивної верифікації поточного стану проєкту, ідентифікації критичних компонентів системи, а також предиктивного моделювання потенційних дефектів.

Додатково встановлено, що спрощеність традиційного підходу, не дивлячись на його практичність, обмежує можливості глибокого аналізу ризиків у складних та динамічних проєктних середовищах. Зокрема, відсутність урахування часових змін, взаємозалежностей між ризиками та впливу зовнішніх факторів може призводити до заниження або некоректної оцінки ризикових подій.

Натомість застосування системи метрик дозволяє формувати більш комплексне уявлення про стан проєкту, забезпечує моніторинг ключових показників у динаміці, підвищує точність виявлення критичних зон і створює передумови для більш обґрунтованого прогнозування можливих відхилень і дефектів.

Порівняння методів:

Інтеграція в CI/CD дозволяє: отримувати метрики в реальному часі; автоматично оцінювати ризики; впроваджувати risk-gates. Комбінування підходів забезпечує: підвищення точності; адаптивність; зниження ризиків. Розглянемо більш детально особливості методів оцінювання ризиків.

Управління ризиками є критично важливим елементом процесу розробки ПЗ. Одним із ключових етапів цього процесу є оцінювання ризиків, яке передбачає визначення ймовірності їх виникнення та оцінку можливих наслідків. Традиційні підходи до оцінювання ризиків часто базуються на експертних оцінках і якісному аналізі, що може призводити до суб'єктивності та недостатньої точності результатів.

Для підвищення об'єктивності та точності оцінювання доцільно застосовувати кількісні моделі. Узагальнено рівень ризику може бути визначений як:

$$R = P \times I, \tag{1}$$

де R – рівень ризику, P – ймовірність виникнення ризику, I – величина його впливу на проєкт. Така формалізація дозволяє здійснювати порівняльний аналіз ризиків, визначати пріоритети їх обробки та обґрунтовувати управлінські рішення.

Використання метрик у оцінюванні ризиків. Сучасні програмні проєкти генерують значні обсяги даних у процесі розробки, що відкриває можливості для автоматизації оцінювання ризиків. Використання метрик програмного проєкту, таких як кількість змін у коді, частота комітів, складність модулів, щільність дефектів, покриття тестами, швидкість виконання задач та кількість невдалих збірок, дозволяє сформувати об'єктивну основу для аналізу.

Ймовірнісні та статистичні методи. Для більш глибокого аналізу доцільно застосовувати умовну ймовірність, яка визначається як

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{2}$$

Формула походить із класичної теорії ймовірностей і дозволяє оцінити ймовірність виникнення ризику за наявності певного фактора. У програмних проєктах це дає змогу, наприклад, визначити ймовірність появи дефектів за умови високої складності коду або частих змін, що підвищує точність оцінювання ризиків. Крім того, узагальнену оцінку ризиків можна здійснювати за допомогою математичного сподівання:

$$E(R) = \sum_{i-1}^{n} P_i \times I_i \quad (3)$$

Ця формула є базовою в математичній статистиці та дозволяє врахувати сукупність різних ризиків, їх імовірності та впливи, формуючи інтегральну оцінку ризиковості проєкту.

Кореляційний аналіз. Для виявлення залежностей між метриками застосовується коефіцієнт кореляції:

$$r = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sigma_x \sigma_y} \quad (4)$$

Даний статистичний показник дозволяє визначити силу зв'язку між різними характеристиками проєкту, наприклад між складністю коду та кількістю дефектів або між частотою комітів і стабільністю системи. Використання кореляційного аналізу допомагає ідентифікувати ключові фактори ризику та підвищити обґрунтованість управлінських рішень.

Автоматизований збір метрик здійснюється за допомогою сучасних інструментів розробки та управління проєктами. Системи контролю версій, платформи безперервної інтеграції, інструменти аналізу якості коду та трекінгу задач забезпечують накопичення великої кількості даних, які можуть використовуватися для оцінювання ризиків у реальному часі [1]. Це дозволяє перейти від статичного до динамічного аналізу стану проєкту.

Аналіз дефектів. Особливе значення має оцінювання дефектів як індикаторів технічних ризиків. Дефекти ПЗ безпосередньо впливають на якість продукту та можуть призводити до серйозних наслідків, включаючи збої системи або втрату даних [2]. Аналіз історичних даних про дефекти дозволяє виявляти закономірності їх виникнення, визначати найбільш проблемні компоненти системи та прогнозувати ймовірність появи нових помилок.

Використання ітераційних підходів до розробки, зокрема Agile-методологій, сприяє зниженню ризиків за рахунок коротких циклів розробки та постійного зворотного зв'язку [3]. Кожна ітерація дозволяє перевіряти проміжні результати, виявляти дефекти на ранніх етапах та коригувати подальші дії. Це значно зменшує ймовірність накопичення критичних проблем.

Роль CI/CD. Важливим елементом сучасного процесу розробки є безперервна інтеграція та доставка (CI/CD). Використання CI/CD забезпечує автоматичну збірку, тестування та розгортання ПЗ при кожній зміні коду [4]. Це дозволяє оперативно виявляти помилки, контролювати якість коду та отримувати актуальні метрики у реальному часі. Таким чином, CI/CD виступає джерелом даних для оцінювання ризиків і водночас інструментом їх зниження.

Застосування штучного інтелекту. Подальший розвиток підходів до оцінювання ризиків пов'язаний із застосуванням методів штучного інтелекту. Машинне навчання дозволяє аналізувати великі обсяги даних і виявляти складні залежності, які складно визначити традиційними методами [5]. На основі метрик програмного проєкту можуть будуватися моделі для оцінювання ймовірності виникнення ризиків:

$$P = f(x_1, x_2, \dots, x_n), \quad (5)$$

де $x_1 \dots x_n$ — набір метрик, що характеризують стан проєкту.

Для реалізації таких моделей можуть використовуватися різні алгоритми машинного навчання, зокрема логістична регресія, дерева рішень, Random Forest, градієнтний бустинг та нейронні мережі. Кожен із цих методів має свої переваги та може застосовуватися залежно від характеру даних і задачі. Наприклад, дерева рішень забезпечують інтерпретованість результатів, тоді як нейронні мережі дозволяють моделювати складні нелінійні залежності.

Алгоритми градієнтного бустингу, зокрема XGBoost та LightGBM, демонструють високу ефективність на структурованих даних проєктних метрик, що робить їх особливо придатними для задач класифікації ризиків у програмній інженерії , що робить їх особливо придатними для задач класифікації ризиків у програмній інженерії [5].

Кожен із зазначених методів має певні обмеження. Дерева рішень забезпечують інтерпретованість, але можуть бути недостатньо точними на складних даних. Нейронні мережі здатні моделювати нелінійні залежності, проте функціонують як «чорна скринька», що ускладнює пояснення результатів. Традиційна ж кількісна модель $R = P \times I$, не враховує динаміки проєкту та семантичного контексту. З метою наочного порівняння зазначених підходів їх основні характеристики узагальнено в таблиці:

Таблиця 1

Порівняння методів

Метод	Характеристики/Особливості	Переваги	Недоліки
Класична модель «ймовірність – вплив»	Базується на оцінці ймовірності виникнення ризику та величини його впливу	Простота; зрозумілість; швидкість застосування	Не враховує динаміку проєкту; обмежена точність; статичність
Ймовірнісні та статистичні методи	Використання умовної ймовірності, математичне очікування (expected value), статистичних підходів	Більш точне оцінювання; врахування множини факторів	Потребують даних; складність реалізації
Кореляційний аналіз	Визначення залежностей між метриками проєкту	Виявлення ключових факторів ризику; обґрунтованість рішень	Не показує причинно-наслідкові зв’язки
Метрики програмного проєкту	Аналіз кількісних показників (зміни коду, дефекти, покриття тестами тощо)	Об’єктивність; можливість автоматизації; реальний стан системи	Залежність від якості даних; не враховує контекст
Методи машинного навчання	Використання моделей (Random Forest, XGBoost, нейронні мережі) для прогнозування ризиків	Висока точність; виявлення складних залежностей; адаптивність	Складність; потреба у великих даних; інтерпретованість (особливо НМ)
Аналіз дефектів	Дослідження історії помилок і їх закономірностей	Прогнозування проблемних зон; покращення якості ПЗ	Обмеження лише технічними ризиками
Інтеграція в CI/CD	Автоматичний збір і аналіз даних у реальному часі	Актуальність; автоматизація; безперервний моніторинг	Залежність від інфраструктури; складність впровадження
Комбінований підхід	Поєднання метрик, статистики, ML та CI/CD	Висока точність; адаптивність; комплексність	Складність реалізації; потреба в інтеграції систем

Комбінування методів. Саме тому комбінування підходів дозволяє нівелювати ці недоліки. Ансамблеві методи підвищують точність прогнозування. Поєднання класифікації та регресії формує повноцінний ризиковий профіль: перша визначає категорію ризику, друга – кількісно оцінює вплив на терміни та бюджет. Інтеграція AI-моделей у CI/CD забезпечує безперервний моніторинг ризиків у реальному часі, а NLP-методи додають до аналізу семантичну інформацію з текстових даних. Такий комплексний підхід створює синергію між інтерпретованістю, точністю та динамічністю аналізу, що є критичним для управління ризиками в сучасних програмних проєктах.

Оцінювання наслідків ризиків. Окрім оцінювання ймовірності ризиків, важливим є аналіз їх наслідків. Для цього можуть застосовуватися методи регресійного аналізу, що дозволяють оцінити вплив ризиків на ключові показники проєкту, такі як тривалість розробки, кількість дефектів або витрати ресурсів [6]. Поєднання моделей класифікації та регресії дозволяє комплексно оцінювати ризики. Зокрема, класифікаційні моделі визначають категорію ризику (критичний, помірний, низький), тоді як регресійні моделі кількісно оцінюють його потенційний вплив на терміни та бюджет проєкту, що разом формує повноцінну картину ризикового профілю.

Особливу роль у комплексному оцінюванні відіграє автоматизований аналіз дефектів з використанням AI-методик [7]. Інтелектуальні системи здатні не лише фіксувати факт виникнення дефекту, але й класифікувати його за критичністю, автоматично встановлювати зв’язки між схожими помилками та прогнозувати ймовірність рецидиву у суміжних модулях. Використання NLP-методів (обробки природної мови) для аналізу коментарів до коду, повідомлень про помилки та звітів про тестування дозволяє видобувати додаткову семантичну інформацію, яка недоступна при традиційному кількісному аналізі.

Комбінування різних підходів, зокрема ансамблевих методів машинного навчання, дозволяє підвищити точність прогнозування. Інтеграція таких моделей у процес CI/CD дає можливість здійснювати безперервний аналіз ризиків у реальному часі. При кожній зміні коду система може автоматично оновлювати оцінки ризиків, що забезпечує адаптивність та актуальність результатів. Методи інтеграції AI-компонентів у пайплайн CI/CD включають впровадження автоматичних “gates” – порогових перевірок на основі передбачених ризиків, що блокують розгортання у разі перевищення критичного рівня ризику та тим самим підвищують захист кінцевого продукту.

Підвищення захисту програмних проєктів є невід’ємним результатом впровадження інтегрованої системи оцінювання ризиків. Поєднання автоматизованого збору метрик, AI-аналізу дефектів та ансамблевих моделей прогнозування дозволяє своєчасно ідентифікувати не лише технічні, але й безпекові ризики – вразливості коду, небезпечні залежності від сторонніх бібліотек, аномальні патерни у поведінці системи. Таким чином, AI-методики виступають не лише інструментом управління якістю, але й засобом підвищення захищеності (security) ПЗ на всіх етапах його розробки [7].

Запропонований підхід до оцінювання ризиків передбачає поєднання автоматизованого збору метрик, аналізу дефектів, використання ітераційних практик розробки та застосування методів штучного інтелекту. Така інтеграція дозволяє реалізувати проактивне управління ризиками, що базується на даних і забезпечує своєчасне виявлення потенційних проблем. Архітектура запропонованої системи охоплює три рівні: рівень збору даних (агенти моніторингу метрик, інтеграція з Git, JIRA, SonarQube), рівень аналізу (ML-моделі оцінювання ймовірності та наслідків ризиків) та рівень прийняття рішень (автоматичні сповіщення, дашборди ризиків, рекомендаційні механізми для команди проєкту).

Висновки

У роботі запропоновано комплексний підхід до оцінювання ризиків у програмних проєктах, що поєднує підходи програмної інженерії та методи машинного навчання. Проведений аналіз сучасних методів оцінювання ризиків дозволив розробити модель, яка забезпечує не лише визначення поточного стану проєкту, а й прогнозування розвитку ризиків у динаміці. Запропонований підхід базується на автоматизованому зборі метрик процесу розробки програмного забезпечення, аналізі дефектів, використанні ітераційних практик (Agile, DevOps) та застосуванні методів штучного інтелекту. Це дає змогу підвищити точність оцінювання ймовірності виникнення ризикових подій і визначення їхнього впливу на якість програмного продукту.

У межах дослідження вдосконалено модель кількісного оцінювання ризиків за рахунок інтеграції метрик, що відображають динаміку змін програмного коду, стабільність збірок і результати тестування, з алгоритмами машинного навчання, здатними виявляти приховані залежності між характеристиками процесу розробки та появою дефектів.

Отримані результати свідчать, що поєднання традиційних методів із підходами штучного інтелекту дозволяє підвищити точність прогнозування ризиків, зменшити кількість дефектів, оптимізувати використання ресурсів і покращити якість програмного продукту.

Список використаних джерел

1. Ramaswamy S. DevOps Metrics that Matter: A Data-Driven Approach to Performance Measurement and Team Productivity. URL: https://www.researchgate.net/publication/394035674_DevOps_Metrics_that_Matter_A_Data-Driven_Approach_to_Performance_Measurement_and_Team_Productivity (дата звернення: 06.04.2026)

2. Malhotra R., Khanna M. A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools // Elsevier (ScienceDirect), 2022.

3. Schwaber K., Beedle M. Agile Software Development with Scrum. – Upper Saddle River: Prentice Hall, 2002. – С. 11–18.

2. Humble J. Continuous Delivery. – Boston: Addison-Wesley, 2016. – С. 63–70. 5. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // arXiv. – 2016.

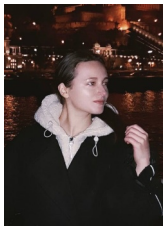
3. Stellman A., Greene J. Applied Software Project Management. – Sebastopol: O'Reilly Media, 2016. – С. 120–135.

4. Yeboah J., Popoola S. Efficacy of static analysis tools for software defect detection on open-source projects. URL: https://www.researchgate.net/publication/380757271_Efficacy_of_static_analysis_tools_for_software_defect_detection_on_open-source_projects (дата звернення: 07.04.2026)

Відомості про авторів

Корнійчук Інна Сергіївна – здобувачка вищої освіти другого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія та тестування програмного забезпечення.

E-mail: 7459063@stud.kai.edu.ua



Лащ Інна Олегівна – здобувачка вищої освіти другого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія та тестування програмного забезпечення.

E-mail: 5264270@stud.kai.edu.ua

Шибицька Наталія Миколаївна – доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інтелектуальні системи навчання та експертні методи оцінювання, методи та моделі штучного інтелекту, нечітка математика та нейронні мережі.

E-mail: shibnatnik@ukr.net

UDC 004.891

EXPLANATION METHODS FOR RECOMMENDATIONS IN SEMANTIC RESUME–JOB MATCHING SYSTEMS

Yuliia KOSTINA
Bachelor's Degree student, Software Engineering Department
National University «Kyiv Aviation Institute»
Scientific supervisor – Yana BIELOZOROVA, PhD

This paper examines the core principles of explanation methods applied to AI-driven resume–job matching systems, with a focus on the black-box problem inherent in modern HR recommendation pipelines. An analysis is conducted of the main approaches to generating interpretable recommendations – including feature-based, contrastive, and example-based explanations – within the context of semantic matching based on sentence embeddings. The applicability of each approach is assessed in terms of fidelity to the model's decision logic and comprehensibility for end users such as recruiters and candidates. A market review of six major HR-tech platforms reveals that while explainability is increasingly recognised as a priority, its implementation remains uneven across the industry.

Keywords: *explainable AI, recommender system, semantic matching, sentence embeddings, HR-tech, SHAP, attention weights.*

Introduction

Automated resume screening powered by machine learning has become a common fixture in modern recruitment pipelines. However, the opacity of such systems – the so-called black-box problem – draws justified criticism from practitioners and researchers alike. Recruiters and candidates are typically unable to understand why a particular resume–job pair received a specific ranking score, which undermines trust and limits the practical utility of these systems.

Legislative pressure reinforces the urgency of this problem. Under the EU Artificial Intelligence Act (Regulation 2024/1689, Annex III, point 4), automated candidate selection systems are classified as high-risk AI systems, for which explainability is a mandatory requirement. Accordingly, the task of generating interpretable explanations shifts from a desirable feature to a legal obligation for developers of HR-tech solutions.

The goal of this work

The goal of this work is to survey existing approaches to explaining recommendations in semantic matching systems, review how leading commercial platforms address this challenge in practice, and assess the current state of the market with respect to explainability standards.

Models and Methods

Three main classes of explanations for recommender systems are examined in this work:

- Feature-based explanations answer the question: "Which skills and qualifications matched, and with what weight?" For example: "Python – high match (0.91), project management – moderate match (0.64)" (values illustrative). This approach is the most intuitively accessible for recruiters and aligns naturally with their existing evaluation frameworks.

• Contrastive explanations address the question: "Why this job, and not another?" They are formulated as comparisons, highlighting what differentiates the recommended match from alternatives. This type of explanation is particularly valuable for ranking justification.

• Example-based explanations rely on precedent: "Candidates with a similar profile to yours have successfully been placed in position Y." They increase user trust through analogy with real historical cases stored in the system.

From a technical standpoint, the most prominent implementation approaches include SHAP-based feature attribution, decomposition of embedding similarity across skill-specific subspaces, and attention weights from transformer models. The latter provide token-level granularity but require additional aggregation to produce human-readable concept-level explanations. A key evaluation criterion across all approaches is fidelity – the degree to which the explanation faithfully reflects the model's actual decision logic rather than offering a post-hoc rationalisation.

Results: Market Analysis

To assess the current state of explainability in practice, six major HR-tech platforms were analysed based on publicly available product documentation, official statements, technical blog posts, and independent reviews published in 2024–2025. Platforms were selected based on their market share in enterprise recruiting and the availability of public documentation on their AI features. The evaluation focused on four dimensions: whether a matching score is shown to the user, what type of explanation (if any) is provided, whether the explanation is accessible to candidates, and to whom the explanation is directed.

The analysis reveals several notable patterns. First, feature-based explanations dominate the market – all platforms that provide any explanation at all do so by surfacing which skills or competencies contributed to the match score. None of the six platforms reviewed offers contrastive or example-based explanations as a standard product feature, despite these being well-established in the academic XAI literature.

Second, there is a significant gap between recruiters and candidates in terms of explanation access. Only one platform out of six (HireVue) explicitly addresses candidate-facing transparency in its published AI Explainability Statement; most platforms direct explanations exclusively to recruiters or HR professionals. Candidates – who are the most directly affected party – remain largely excluded from the explanation loop.

Third, the depth of explainability varies considerably. Eightfold AI represents the most technically advanced approach, publishing detailed engineering documentation of its three-step matching pipeline (semantic embeddings, structured feature extraction, and explainable inference). At the other end of the spectrum, Workday's Candidate Skills Match tool exposes only a coarse categorical label – "Strong," "Good," "Fair," or "Low" – without any breakdown of which factors drove the classification. This limited transparency has become part of a broader legal challenge to automated screening: in Mobley v. Workday (2023), opacity of the underlying scoring is among the factors cited in the discrimination claim.

Fourth, regulatory compliance is increasingly driving disclosure. HireVue's AI Explainability Statement was published in 2022 and updated in 2024, directly in response to growing regulatory scrutiny – though independent commentary (CDT, 2022) notes that such statements often fall short of meaningfully explaining the system's behaviour. SAP SuccessFactors similarly established an AI Ethics Steering Committee and formally classifies hiring-related AI as a high-risk use case subject to mandatory review and bias auditing.

Overall, the market review confirms that while explainability is becoming a recognised priority, its implementation remains shallow and recruiter-centric. The gap identified – absence of contrastive and example-based explanations, limited candidate-facing transparency, and variable fidelity of existing feature-based approaches – represents a clear opportunity for further research and product development.

Table 1.

Explainability features of major HR-tech platforms (2024–2025)

Platform	Score shown to user	Explanation type	Audience	Candidate-facing	Notes
LinkedIn Hiring Assistant	Yes (skill match signals)	Feature-based	Recruiter	Partial	Matching logic visible to both recruiter and candidate per 2025 product docs
HireVue	Competency scores	Feature-based (competencies)	Recruiter, Candidate	Yes	Published AI Explainability Statement (2022, updated 2024); substance of such statements has drawn academic scrutiny [7]
Workday Candidate Skills Match	Categorical: Strong / Good / Fair / Low	Feature-based (skills only)	Recruiter	No	Score granularity is limited; defendant in Mobley v. Workday (2023), a class action alleging discriminatory outcomes from automated screening widely cited in discussions of AI screening opacity
Eightfold AI	Match Score (0–100)	Feature-based, semantic embedding decomposition	Recruiter	No	Technically most advanced: 3-step pipeline (embeddings → structured features → explainable inference)
SAP SuccessFactors	Recommendation with justification	Feature-based (XAI techniques)	HR professional	No	AI Ethics Steering Committee reviews high-risk use cases; applies bias-mitigation tooling
Greenhouse	No score; structured scorecard	Rule-based (structured criteria)	Recruiter, Hiring manager	N/A	Focuses on DEI compliance and structured hiring; no AI-generated match score exposed to user

Conclusions

This paper examined three classes of explanations for semantic resume–job matching systems and reviewed how leading commercial platforms currently address the explainability challenge in

practice. The market analysis of six major HR-tech tools reveals that feature-based explanations are the dominant – and in most cases, the only – form of explanation offered. Contrastive and example-based approaches, well-established in academic research, remain absent from production systems. Candidate-facing transparency is largely neglected, despite candidates being the primary stakeholders in AI-driven hiring decisions.

These findings suggest that the field is at an early stage of maturity with respect to XAI in HR-tech. The gap between what the literature proposes and what platforms implement is substantial. Our further work will focus on designing candidate-facing explanation interfaces, evaluating the comprehensibility of different explanation types across user groups, and developing standardised fidelity benchmarks that can be applied across platforms – particularly in the context of the EU AI Act's mandatory explainability requirements for high-risk AI systems.

References

1. Ribeiro M. T., Singh S., Guestrin C. "Why Should I Trust You?": Explaining the predictions of any classifier // Proceedings of KDD. 2016.

2. Lundberg S., Lee S.-I. A unified approach to interpreting model predictions // Advances in Neural Information Processing Systems. 2017. Vol. 30.

3. Reimers N., Gurevych I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks // Proceedings of EMNLP. 2019.

4. European Parliament. EU Artificial Intelligence Act. Regulation (EU) 2024/1689. 2024.

5. HireVue. AI Explainability Statement (2024 edition). [Online: https://www.hirevue.com/blog/hiring/hirevue-ai-explainability-and-ethics].

6. Eightfold AI. AI-powered talent matching: The tech behind smarter and fairer hiring. Engineering Blog, 2024. [Online: https://eightfold.ai/engineering-blog/ai-powered-talent-matching-the-tech-behind-smarter-and-fairer-hiring/].

7. Center for Democracy and Technology. HireVue "AI Explainability Statement" Mostly Fails to Explain What it Does. 2022. [Online: https://cdt.org/insights/hirevue-ai-explainability-statement-mostly-fails-to-explain-what-it-does/].

8. Tang et al. Explainable person-job recommendation systems: A systematic review of 85 studies. Preprint, 2025.

Information about the author

Yuliia Kostina – Bachelor's Degree student of the Software Engineering Department, Faculty of Computer Science and Technologies, National University “Kyiv Aviation Institute”. *Research interests:* software engineering, artificial intelligence, recommender system, semantic matching.

E-mail: 8009004@stud.kai.edu.ua

УДК 004.9

ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ОПТИМІЗАЦІЇ ГУМАНІТАРНОЇ ЛОГІСТИКИ В УМОВАХ ВОЄННОГО СТАНУ

Лілія КОСТЮЧЕНКО
Здобувачка вищої освіти першого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Володимир ТАЛАЛАЄВ, доцент, к.т.н.

У роботі розглядаються можливості використання інформаційних технологій для оптимізації гуманітарної логістики в умовах воєнного стану. Актуальність дослідження зумовлена необхідністю забезпечення своєчасного та безпечного доставлення гуманітарної допомоги в умовах руйнування транспортної інфраструктури, наявності небезпечних територій та постійної зміни логістичної ситуації. Запропоновано підхід до автоматизації маршрутизації з урахуванням просторових, часових і безпекових обмежень.

Ключові слова: *гуманітарна логістика, інформаційні технології, оптимізація маршрутів, волонтерські організації, воєнний стан, автоматизована система, маршрутизація.*

Вступ

Повномасштабна війна в Україні суттєво ускладнила доставлення гуманітарної допомоги населенню. Руйнування транспортної інфраструктури, обмеження руху окремими територіями, зміна безпекової ситуації та необхідність оперативного реагування на потреби людей створюють додаткові труднощі для волонтерських і благодійних організацій [1]. Ефективність гуманітарної логістики за таких обставин залежить не лише від наявності ресурсів, а й від здатності швидко аналізувати ситуацію та приймати обґрунтовані рішення. Традиційне ручне планування маршрутів часто не дозволяє оперативно враховувати всі чинники перевезення гуманітарних вантажів: небезпечні або тимчасово недоступні території, часові обмеження роботи волонтерів, стан дорожньої інфраструктури та потребу пріоритетного обслуговування окремих населених пунктів. Тому особливого значення набуває використання сучасних інформаційних технологій, здатних автоматизувати збір, аналіз та обробку логістичної інформації.

Метою роботи є дослідження можливостей застосування інформаційних технологій для оптимізації гуманітарної логістики в умовах воєнного стану та визначення підходів до автоматизованого планування маршрутів доставки гуманітарної допомоги з урахуванням часових, ресурсних і безпекових обмежень.

Матеріали та методи

Методологічною основою дослідження є системний аналіз логістичних процесів, методи оптимізації маршрутів, технології обробки просторових даних та принципи проєктування інформаційних систем. Дослідження спрямоване на визначення можливостей використання сучасних інформаційних технологій для підвищення ефективності гуманітарних перевезень в умовах воєнного стану.

Гуманітарна логістика в умовах збройного конфлікту характеризується високим рівнем невизначеності, швидкою зміною умов перевезень та необхідністю оперативного реагування

на нові потреби населення. Для забезпечення ефективного управління такими процесами доцільним є використання інформаційних систем, які інтегрують дані з різних джерел та автоматизують процес прийняття рішень.

Інформаційне забезпечення гуманітарної логістики передбачає використання даних про населені пункти та пункти видачі допомоги, стан дорожньої інфраструктури, наявність небезпечних або тимчасово окупованих територій, характеристики транспортних засобів та графіки роботи волонтерів. Для аналізу таких даних можуть застосовуватися бази даних, геоінформаційні технології, цифрові карти та алгоритми оптимізації. Геоінформаційні системи дають змогу візуалізувати маршрути, виявляти небезпечні ділянки та оцінювати доступність точок доставки. Алгоритми маршрутизації, зокрема задачі класу Vehicle Routing Problem [2], дозволяють формувати маршрути з урахуванням кількох критеріїв одночасно.

У практичній реалізації подібних задач можуть використовуватися спеціалізовані програмні засоби, зокрема бібліотеки для розв’язання задач оптимізації маршрутів, які підтримують часові обмеження, обмеження ресурсів та багатокритеріальний вибір рішень. Це створює можливість автоматично пропонувати маршрут, який не лише є зручним за відстанню, а й є придатним з погляду безпеки та максимально наближеним до реальних умов виконання гуманітарних перевезень.

Процес оптимізації гуманітарних маршрутів включає збір та актуалізацію інформації про транспортну мережу, визначення доступних і небезпечних територій, аналіз наявних ресурсів, формування множини можливих маршрутів та вибір оптимального рішення за сукупністю визначених критеріїв. У разі зміни безпекової ситуації або появи нових потреб система повинна забезпечувати можливість оперативного перепланування маршрутів.

Результати та обговорення

Аналіз показав, що застосування інформаційних технологій у гуманітарній логістиці дозволяє перейти від ручного планування до автоматизованого прийняття рішень та значно підвищує її ефективність в умовах воєнного стану. Насамперед це проявляється у скороченні часу на підготовку маршрутів і зменшенні кількості помилок під час ручного опрацювання логістичної інформації – що особливо важливо в кризових умовах, коли затримка навіть на кілька годин може зменшити ефективність допомоги або створити додаткові ризики для виконавців [3].

Одним із ключових напрямів цифровізації є автоматизація планування маршрутів. Алгоритми оптимізації одночасно враховують декілька параметрів: відстань між пунктами доставки, прогнозований час перевезення, завантаженість транспортних засобів, пріоритетність отримувачів допомоги та рівень безпеки маршрутів. У результаті формується найбільш доцільний маршрут, що забезпечує своєчасне доставлення гуманітарних вантажів із мінімальними витратами ресурсів.

Важливим результатом використання інформаційних систем є підвищення оперативності реагування на зміни. Отримання актуальної інформації про дорожню ситуацію, пошкодження інфраструктури або нові небезпечні зони дозволяє швидко коригувати маршрути та уникати ризиків для волонтерів і транспортних засобів, що підвищує безпеку логістичних операцій та знижує ймовірність зриву доставки.

Інтеграція геоінформаційних технологій із системами управління логістикою дозволяє координаторам візуалізувати маршрути на електронних картах, швидко оцінювати ситуацію, контролювати виконання перевезень і приймати обґрунтовані рішення, а також точніше розподіляти транспортні ресурси між різними напрямками доставки.

Практична цінність такого підходу полягає у зменшенні навантаження на координаторів, підвищенні швидкості ухвалення рішень, прозорості логістичного процесу та кращому використанні людських і транспортних ресурсів, що критично важливо для волонтерських організацій. В умовах воєнного стану оптимальний маршрут не завжди означає найкоротший – у пріоритеті можуть бути безпечність, об'їзд небезпечних ділянок, дотримання часових обмежень та гарантоване доведення вантажу до кінцевої точки. Тому інформаційна система повинна підтримувати багатокритеріальну оптимізацію, а не лише пошук найкоротшого шляху.

Таким чином, використання інформаційних технологій у гуманітарній логістиці забезпечує скорочення часу планування, підвищення безпеки доставки, зменшення навантаження на координаторів та покращення якості управлінських рішень. Комплексне застосування інформаційних систем, геоінформаційних технологій та алгоритмів оптимізації створює передумови для більш ефективного функціонування волонтерських і гуманітарних організацій в умовах воєнного стану.

Висновки

Дослідження показало, що оптимізація гуманітарної логістики в сучасних реаліях України не може спиратися виключно на критерії економічної ефективності (мінімізацію пального чи часу) – головним критерієм стає безпека. Використання сучасних засобів обробки даних, геоінформаційних систем та алгоритмів оптимізації маршрутів дозволяє автоматизувати процес планування перевезень, враховувати безпекові обмеження та оперативно реагувати на зміни логістичної ситуації.

Комплексне застосування інформаційних технологій підвищує ефективність гуманітарних перевезень, скорочує час доставлення допомоги та раціоналізує використання ресурсів волонтерських організацій, створюючи передумови для якісного логістичного забезпечення населення в умовах воєнного стану та інших кризових ситуацій.

Список використаних джерел

1. Трансформація гуманітарної логістики в Україні під впливом воєнних дій // Економіка та суспільство. URL: https://economyandsociety.in.ua/index.php/journal/article/view/2948/2866

2. Toth P., Vigo D. Vehicle Routing: Problems, Methods, and Applications. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics, 2014. 463 p.

3. Challenges of Logistical Support in the Frontline Territories of Ukraine. ResearchGate. URL: https://www.researchgate.net/publication/393491593_Challenges_of_Logistical_Support_in_the_Frontline_Territories_of_Ukraine

Відомості про автора

Костюченко Лілія Віталіївна – здобувачка вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення.

E-mail: 5713315@stud.kai.edu.ua

UDC 005.8:005.591.6:005.94:004.8

EXTENDING AGILE OPERATING MODELS FOR AI-AUGMENTED PROJECT ENVIRONMENTS: A COMPARATIVE EXTENSION OF THE DATA-EMPIRICAL AGILITY MODEL

Oleh LUKUTIN
Senior Lecturer
University of Economics and Law «KROK»

The article examines the extension of Agile Operating Models in the context of artificial intelligence integration within project environments. The Data-Empirical Agility Model (DEAM) is proposed as a conceptual framework that enables the incorporation of AI-driven mechanisms into Agile feedback cycles. A comparative analysis of existing Agile Operating Models is conducted to identify their limitations in supporting AI-augmented processes. The study substantiates the necessity of transitioning from traditional empirical approaches to data-empirical mechanisms for managing organizational and project adaptation in digitally transforming environments.

Keywords: *Agile technologies, Project management, Agile Operating Models, Artificial Intelligence, AI-augmented organizations, organizational adaptation, Data-Empirical Agility Model (DEAM), digital transformation.*

Introduction

Agile Operating Models have become a foundational mechanism for enabling adaptability, iterative delivery and value-oriented governance in contemporary organizations. Frameworks such as Agile Product Operating Model (APOM) and Evidence-Based Management (EBM) provide structured approaches for aligning organizational strategy, delivery processes and performance measurement cycles [2], [3]. These models emphasize transparency, inspection and adaptation as core principles for sustainable value delivery [11].

The increasing adoption of Artificial Intelligence (AI) technologies introduces new structural and operational dynamics that challenge traditional feedback and governance mechanisms. AI-enabled systems generate continuous streams of predictive signals, anomaly detections, and workflow insights that influence organizational behavior beyond standard iteration cycles [9]. Traditional Agile feedback loops, primarily dependent on retrospective analysis and periodic evaluation, may become insufficient under conditions of continuous sensing and automated decision support [5].

In AI-augmented environments, organizational learning becomes more dynamic and data-intensive, requiring integrated architectures that support both human judgment and machine-generated insights. This transformation aligns with contemporary theories of team learning and organizational adaptation [12]. Despite the maturity of Agile Operating Models, there remains a conceptual gap in supporting continuous, data-driven organizational learning within AI-enabled contexts. Therefore, extending Agile Operating Models to incorporate adaptive learning mechanisms represents an important research direction for improving organizational responsiveness and resilience [1].

Purpose of the research

The purpose of this research is to comparatively analyze established Agile Operating Models and identify structural gaps related to AI-driven feedback integration, followed by positioning the

Data-Empirical Agility Model (DEAM) as an extension framework enabling continuous adaptive learning in AI-augmented organizations.

The research objectives include:

- analyzing conceptual and operational characteristics of APOM and EBM [2], [3];
- identifying structural limitations of existing models under AI-driven conditions [5];
- defining architectural mechanisms required for continuous learning [1];
- positioning DEAM as an extension model supporting predictive and feedback-driven governance.

Materials and methods

The study applies principles of Design Science Research (DSR), focusing on conceptual modeling and comparative framework analysis [1]. The research integrates theoretical evaluation of Agile Operating Models with structural synthesis of feedback-driven learning architectures.

The comparative evaluation is based on three reference models:

- Agile Product Operating Model (APOM) [3];
- Evidence-Based Management (EBM) [2];
- Data-Empirical Agility Model (DEAM) [10].

Evaluation criteria include: governance structure; feedback architecture; decision latency characteristics; learning mechanisms; AI integration capability [9]; resistance handling mechanisms; measurement signal granularity.

These criteria enable systematic identification of structural differences and conceptual extensions required for AI-augmented environments [7].

Results and discussion

Comparative analysis of established Agile Operating Models demonstrates strong capabilities in organizing product delivery and measuring value outcomes. However, these models primarily rely on iterative inspection cycles and retrospective learning mechanisms [2], [3]. Under conditions of continuous data generation, these mechanisms may lead to delayed response patterns and fragmented feedback interpretation [5].

Table 1

Comparative Positioning of DEAM, APOM and EBM

Dimension	APOM (Agile Product Operating Model)	EBM (Evidence-Based Management)	DEAM (Data-Empirical Agility Model)
Conceptual Foundation	Product-centric operating structure	Outcome-based performance management	Data-empirical adaptive learning architecture
Primary Objective	Optimize product value delivery	Improve organizational outcomes	Enable continuous adaptive learning in AI-augmented environments
Governance Logic	Product and portfolio alignment	Value-driven decision governance	Feedback-centric adaptive governance

Table 1 (cont.)

Feedback Model	Iterative, sprint-based feedback	Metric-driven periodic feedback	Continuous multi-layer feedback integration
Decision Latency	Iteration-dependent (time-boxed)	Review-cycle dependent	Real-time or near-real-time adaptive response
Learning Mechanism	Retrospective learning cycles	Metric interpretation and adjustment	Predictive and reinforcement-based learning cycles
Measurement Scope	Product and delivery performance	Value outcomes and capabilities	Multi-dimensional signals (flow, value, risk, resistance)
AI Integration Capability	Implicit or tool-level	Partial (data-driven but human-centric)	Explicitly embedded AI sensing and prediction layers
Handling Structural Resistance	Implicit, experience-driven	Partially addressed via metrics	Explicit detection and reintegration into learning cycles
Adaptation Strategy	Iterative process refinement	Outcome-based adjustment	Continuous system-level adaptation
Operational Dynamics Representation	Limited dynamic modeling	Static measurement cycles	Explicit operational dynamics modeling (Fig. 1)
Feedback Architecture Complexity	Moderate	Moderate	High, multi-loop adaptive architecture (Fig. 2)
Human – AI Collaboration Model	Human-dominant	Human-guided decision support	Symbiotic human–AI co-learning architecture
Scalability Mechanism	Portfolio alignment structures	Capability measurement scaling	Signal-based scaling across organizational layers
Predictive Capability	Minimal	Limited forecasting	Built-in predictive learning loops
Suitability for AI-Augmented Organizations	Partial	Moderate	High
Model Type	Operating Model	Management Framework	Adaptive Learning Extension Framework
Role in This Study	Baseline reference model	Comparative performance model	Proposed extension model

The results indicate that existing models provide structured governance and measurement practices but demonstrate limited support for continuous feedback assimilation and predictive adaptation. In particular, structural resistance phenomena – such as coordination delays, workflow bottlenecks, and decision conflicts – are typically treated as operational anomalies rather than learning inputs [7].

To address these limitations, the Data-Empirical Agility Model introduces a continuous co-learning architecture integrating human decision-making processes with AI-generated predictive signals [10].

The foundational learning architecture of the Data-Empirical Agility Model (DEAM) has been previously introduced and described in detail in earlier research [10].

Therefore, the present study focuses on the operational and feedback dynamics that extend the established DEAM framework in AI-augmented organizational environments.

Building upon the previously established DEAM learning architecture [10], the operational dynamics within AI-augmented environments demonstrate the importance of timely feedback integration and adaptive response mechanisms. Delayed interpretation of feedback signals may lead to increased structural resistance and reduced responsiveness [7].

Figure 1 illustrates the sequential flow of signals from operational environments through sensing, interpretation, adaptation, and resistance detection stages. Structural resistance is represented as an integral feedback component rather than an isolated failure condition. This representation supports the concept of resistance reintegration as a mechanism for continuous learning [10].

Another key architectural component of DEAM involves unified governance feedback loops that integrate both human and AI-driven decision inputs. Such integration supports predictive governance and proactive adaptation strategies [9].

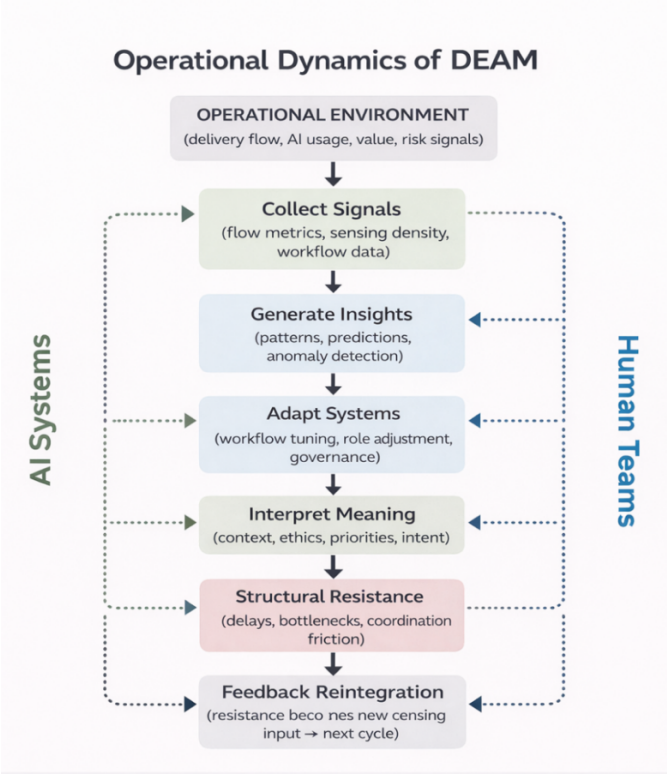


Figure 1 – Operational Dynamics of the Data-Empirical Agility Model (DEAM)

Figure 2 demonstrates the interaction between performance metrics, predictive insights, governance decisions, and outcome signals. The architecture emphasizes multi-directional feedback flows that enable real-time adaptation across organizational layers.

Overall, the findings suggest that extending Agile Operating Models with continuous sensing mechanisms significantly improves adaptive capacity and supports more resilient organizational behavior in AI-enabled environments [1].

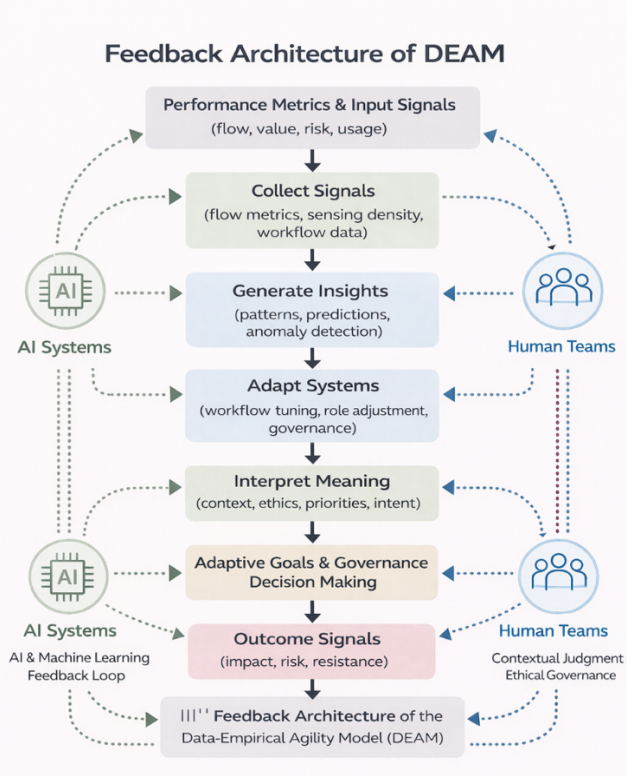


Figure 2 – Feedback Architecture of the Data-Empirical Agility Model (DEAM)

Conclusions

The conducted analysis demonstrates that existing Agile Operating Models provide structured mechanisms for product alignment and performance measurement but require architectural extensions to operate effectively within AI-augmented environments [2], [3].

Comparative evaluation identified several structural gaps, including limited predictive capability, delayed feedback integration, and insufficient mechanisms for managing structural resistance [5]. The proposed Data-Empirical Agility Model addresses these limitations by introducing a continuous learning architecture that integrates human judgment and AI-driven sensing [10].

Positioning DEAM as an extension framework enables compatibility with existing Agile practices while supporting the transition toward data-empirical governance models [1].

Future research directions include empirical validation of DEAM through experimental implementation in AI-enabled Agile environments and quantitative evaluation of learning cycle efficiency.

References

1. Hevner A. R., March S. T., Park J., Ram S. Design science in information systems research // MIS Quarterly. 2004. Vol. 28, No. 1. P. 75–105. URL: https://www.jstor.org/stable/25148625

2. Scrum.org. Evidence-Based Management Guide. Scrum.org, 2020. URL: https://www.scrum.org/resources/evidence-based-management-guide

3. Scrum.org. Agile Product Operating Model (APOM). Scrum.org, 2023. URL: https://www.scrum.org/resources/agile-product-operating-model

4. Sutherland J., Sutherland J. J. Scrum: The art of doing twice the work in half the time. New York : Crown Business, 2014.

5. Rigby D. K., Sutherland J., Noble A. Agile at scale // Harvard Business Review. 2018. Vol. 96, No. 3. P. 88–96. URL: https://hbr.org/2018/05/agile-at-scale

6. Kersten M. Project to product: How to survive and thrive in the age of digital disruption. Portland : IT Revolution Press, 2018.

7. Forsgren N., Humble J., Kim G. Accelerate: The science of lean software and DevOps. Portland: IT Revolution Press, 2018.

8. Denning S. The age of Agile. New York : AMACOM, 2018.

9. Russell S., Norvig P. Artificial intelligence: A modern approach. 4th ed. Harlow : Pearson, 2021.

10. Lukutin O., Michkivskyy S. Designing AI-augmented learning systems for Agile organizations: The Data-Empirical Agility Model (DEAM). 2025. URL: https://snku.krok.edu.ua/index.php/vcheni-zapiski-universitetu-krok/article/view/1202

11. Schwaber K., Sutherland J. The Scrum Guide. Scrum.org, 2020. URL: https://scrumguides.org/scrum-guide.html

12. Edmondson A. C. Teaming: How organizations learn, innovate, and compete in the knowledge economy. San Francisco : Jossey-Bass, 2012.

Information about the author

Oleh Lukutin – Senior Lecturer at the Department of Computer Science, University "KROK", Kyiv, Ukraine; Agile Coach and Delivery Manager at GlobalLogic. *Research interests:* Agile transformation, AI-augmented organizational learning, Project Management, Evidence-Based Management (EBM) and AI integration in software engineering.

E-mail: oleglv@krok.edu.ua

УДК 004.4

ВИБІР АЛГОРИТМУ ІНТЕРВАЛЬНОГО ПОВТОРЕННЯ ДЛЯ ВЕБПЛАТФОРМИ ПІДГОТОВКИ ДО ІСПИТІВ

Тарас МАРЧЕНКО
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Яна БЄЛОЗЬОРОВА, к.т.н., доц.

У роботі проведено порівняльний аналіз трьох алгоритмів інтервального повторення – системи Лейтнера, SuperMemo-2 (SM-2) та Free Spaced Repetition Scheduler (FSRS) – у контексті їх інтеграції у вебплатформу для підготовки до іспитів. Розглянуто принципи роботи кожного алгоритму, ключові параметри та особливості реалізації. Порівняння виконано за метриками точності прогнозування пригадування (Log-loss, RMSE), середньою кількістю повторень та технічними характеристиками. Результати показують, що FSRS перевершує SM-2 за точністю прогнозу та потребує приблизно на 25 % менше повторень, тоді як система Лейтнера, попри простоту реалізації, поступається обом алгоритмам за ефективністю. Зроблено висновок про доцільність використання FSRS у сучасних навчальних вебзастосунках попри вищу обчислювальну та архітектурну складність.

Ключові слова: *інтервальне повторення, FSRS, SuperMemo, система Лейтнера, флеш-картки, крива забування, вебплатформа, прогнозування пригадування.*

Вступ

Поширеним методом підготовки до іспитів залишається механічне заучування великого обсягу матеріалу у стислі строки (так зване «зубріння»). Його обмежена ефективність відома щонайменше з XIX століття: у класичних експериментах Германа Еббінгауза із безглуздими складами через 20 хвилин після заучування у пам'яті лишалося близько 60 % матеріалу, через годину – близько 45 %, а через добу – лише третина; у довгостроковій пам'яті без повторень закріплювалося приблизно 15 % [1]. Подальші дослідження кривої забування з часом сформували принцип інтервального повторення – техніки, у якій матеріал повторюється через інтервали, що поступово зростають. Цей принцип ліг в основу алгоритмів планування повторень: системи Лейтнера, родини SuperMemo (зокрема SM-2) та FSRS, які сьогодні є найбільш поширеними у навчальних застосунках.

Метою роботи є порівняльний аналіз провідних алгоритмів інтервального повторення для визначення їх переваг, недоліків і доцільності інтеграції у навчальну вебплатформу підготовки до іспитів.

Матеріали та методи

Першим і найпростішим методом інтервального повторення є система Лейтнера, запропонована німецьким журналістом і науковим популяризатором Себастіаном Лейтнером у 1972 році. Її суть полягає у розміщенні флеш-карток у послідовність «коробок» відповідно до рівня засвоєння кожної з них. У разі успішного пригадування картка переходить до коробки наступного рівня, у разі помилки – повертається на нижчий рівень. Кожен наступний рівень повторюється з більшим інтервалом [2].

Іншим поширеним і вже формалізованим алгоритмом є SuperMemo, розроблений польським дослідником Пьотром Возняком у 1985 році. Алгоритм пройшов кілька ітерацій (остання версія – SM-20, 2026 рік), однак найбільш широко використовуваною досі залишається версія SM-2 [3]. У SM-2 кожна картка характеризується трьома параметрами:

1. n – кількість поспіль успішних повторень картки (з оцінкою ≥ 3);
2. EF – фактор легкості, що визначає швидкість зростання інтервалу між повтореннями (початкове значення 2,5);
3. I – інтервал у днях до наступного повторення.

Під час сеансу користувачеві пропонуються картки, з моменту останнього повторення яких минуло щонайменше I днів. Користувач оцінює пригадування за шкалою від 0 до 5 (0–2 – градації неправильної відповіді, 3–5 – правильної), на основі чого перераховуються EF та I [3].

Найбільш сучасним з розглянутих є алгоритм FSRS (Free Spaced Repetition Scheduler), розроблений з 2022 року Jarrett Ye [5] та спільнотою Open Spaced Repetition; перша публічна версія (FSRS v1) вийшла 18 вересня 2022 року, а з листопада 2023 року алгоритм є типовим планувальником у Anki (починаючи з версії 23.10). FSRS базується на трикомпонентній моделі пам'яті (DSR) і прогнозує момент, коли ймовірність пригадування картки опуститься до цільового значення (за замовчуванням – 90 %). Кожна картка описується трьома змінними [4]:

1. *D (Difficulty)* – складність картки для конкретного користувача;
2. *S (Stability)* – стабільність пам'яті: кількість днів до моменту, коли ймовірність пригадування знизиться до 90 %;
3. *R (Retrievability)* – поточна прогнозована ймовірність пригадування картки.

На відміну від SM-2, FSRS має набір ваг (у поточній версії FSRS-6 – близько двох десятків параметрів), значення яких можуть автоматично оптимізуватися під історію повторень конкретного користувача методом градієнтного спуску після накопичення ~1000 повторень [4].

Для обґрунтованого вибору алгоритму під вебплатформу порівняння проведено за такими критеріями: Log-loss та RMSE (метрики точності прогнозу пригадування – менше значення відповідає кращій моделі), середня кількість повторень для досягнення цільового рівня запам'ятовування, а також технічні характеристики (кількість параметрів, потреба у зберіганні історії повторень, обчислювальна складність).

Результати

Як видно з таблиці 1, система Лейтнера не наведена у стовпцях точності, оскільки вона не генерує ймовірності пригадування, а оперує лише фіксованими рівнями коробок. У цьому її головна перевага (простота реалізації, відсутність потреби у зберіганні історії повторень) і водночас головний недолік: за ефективністю вона істотно поступається математично обґрунтованим алгоритмам. SM-2 у свою чергу помітно поступається FSRS і за точністю прогнозу (Log-loss, RMSE), і за кількістю необхідних повторень: за даними бенчмарку, FSRS-6 з типовими вагами дозволяє досягти цільового рівня запам'ятовування приблизно на 25 % меншою кількістю повторень, ніж SM-2 [6, 7].

Платою за точність FSRS є вища архітектурна та обчислювальна складність: окрім трьох змінних стану, алгоритм використовує близько двох десятків ваг, а для персоналізації параметрів потрібен оптимізатор, який обробляє повну історію повторень користувача (порядку 1000 і більше). У контексті вебплатформи це означає необхідність зберігати журнал повторень у базі даних і періодично (за рекомендацією – 3–4 рази на рік або при подвоєнні

обсягу повторень) перенавчати параметри. Проте для типових колекцій карток цей оверхед є прийнятним і не створює суттєвих обмежень для бекенду; крім того, доступні готові опенсорсні реалізації алгоритму на TypeScript, Python, Rust, Go тощо (під ліцензією MIT), які істотно спрощують інтеграцію.

Таблиця 1

Порівняння алгоритмів інтервального повторення

Алгоритм	Log-loss	RMSE	Повторення (відн. SM-2)	Параметри	Журнал повторень у БД
Система Лейтнера	–	–	умовно базис	1 (рівень коробки)	не потрібен
SM-2	0,4900	14,84 %	базис (100 %)	3 (n, EF, I)	потрібен
FSRS-6 (default)	0,3468	7,86 %	~75 % (на 25 % менше)	3 змінні + ~21 вага	потрібен
FSRS-6 (recency w)	0,3198	4,37 %	≤ 75 %	3 змінні + ~21 вага	потрібен; оптимізатор

Висновки

Проведений порівняльний аналіз показує, що для інтеграції у навчальну вебплатформу підготовки до іспитів найбільш доцільним є алгоритм FSRS. Він демонструє кращу точність прогнозування пригадування, ніж SM-2 (нижчі значення Log-loss і RMSE), та потребує приблизно на 25 % менше повторень для досягнення того ж цільового рівня запам'ятовування. Система Лейтнера, незважаючи на простоту реалізації, не витримує конкуренції за ефективністю і може розглядатися лише як резервний варіант для мінімалістичних рішень.

Підвищені вимоги FSRS до пам'яті та обчислень компенсуються наявністю готових опенсорсних реалізацій під MIT-ліцензією, що знижує вартість інтеграції до прийнятного рівня. Напрямами подальшої роботи є практична інтеграція FSRS у прототип вебплатформи, оцінка масштабованості оптимізатора параметрів на реальному навантаженні та порівняння UX-сценаріїв з планувальником на базі SM-2.

Список використаних джерел

1. Forgetting curve. Wikipedia. URL: https://en.wikipedia.org/wiki/Forgetting_curve (дата звернення: 18.04.2026).

2. Leitner system. Wikipedia. URL: https://en.wikipedia.org/wiki/Leitner_system (дата звернення: 18.04.2026).

3. SuperMemo. Wikipedia. URL: https://en.wikipedia.org/wiki/SuperMemo (дата звернення: 18.04.2026).

4. ABC of FSRS. GitHub: open-spaced-repetition/fsrs4anki Wiki. URL: https://github.com/open-spaced-repetition/fsrs4anki/wiki/ABC-of-FSRS (дата звернення: 18.04.2026).

5. Ye J. A Stochastic Shortest Path Algorithm for Optimizing Spaced Repetition Scheduling. Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22). 2022. P. 4381–4390.

6. SRS Benchmark. URL: https://expertium.github.io/Benchmark.html (дата звернення: 18.04.2026).

7. FSRS vs SM-2. Diane. URL: https://www.diane.app/en/guides/fsrs-vs-sm2 (дата звернення: 18.04.2026).

Відомості про автора:

Тарас Олександрович Марченко – здобувач вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси*: інженерія програмного забезпечення, освітні технології, алгоритми інтервального повторення.

E-mail: 8150611@stud.kai.edu.ua

УДК 004.451

ЗАСТОСУВАННЯ СТАТИЧНОГО АНАЛІЗУ DEEP LEARNING ДЛЯ ПОШУКУ ПРИХОВАНИХ ПОМИЛОК ВИКОРИСТАННЯ ПАМ'ЯТІ КОДОМ C++

Василь МЕЛЬНИК
к. ф-м. н., доцент, викладач-методист
Волинський фаховий коледж НУХТ
Наталія БАГНЮК
к.т.н., доцент КІ та Б, ЛНТУ
Оксана КОРЧУК
викладач-методист
Волинський фаховий коледж НУХТ

У статті викладено особливості та переваги застосування статичного аналізу Deep learning на базі нейронних мереж для виявлення прихованих помилок використання пам'яті програмним кодом C++. Обговорено залежність його продуктивності, можливості відображення та зменшення хибно-позитивних проявів.

Ключові слова: *статичний аналіз коду, використання пам'яті, пошук помилок, Deep learning.*

Вступ

Помилкове використання пам'яті кодом C++ не виявлене традиційними аналізаторами може приводити до збоїв у роботі застосунків, їх непередбачувану поведінку, відкриваючи вразливості безпеки [1,2]. Deep learning разом з нейронними мережами здатний виявляти приховані помилки, використовуючи збережений досвід його застосування на мільйонах зразків коду [3].

В ході аналізу програмного коду помилки керування пам'яттю є найскладнішими. Аналізатори стикаються зі складною арифметикою вказівників, умовним розподілом, багатопотоковими взаємодіями, проблемами її звільнення і доступом до неї іншими процесами. Існує проблема переповнення буфера пам'яті та запису даних за його межі чи помилкові спроби звільнити пам'ять двічі, відкриваючи її витік розіменуванням вказівників [4].

Ціль роботи

Метою роботи є застосування статичного аналізу Deep learning для проєктів C++ і виявлення прихованих помилок використання пам'яті кодом перед промисловим розгортанням, покращуючи безпеку та заощаджуючи час розробника.

Особливості налаштування та виконання аналізу Deep learning

Deep learning разом з нейро-мережею (SonarQube, CodeClimate, CodeRabbit) та штучним інтелектом аналізує код як шаблони, перетворюючи його у числове представлення, аналізуючи зв'язки і покроковий розподіл пам'яті, відзначаючи підозрілості. Засобом Deep learning для аналізу безпеки використання пам'яті кодом є DeepCode, поєднаний із засобами GitHub, GitLab та IDE-плагінами з Docker [5], засобом CodeGuru – для посилання на кодову базу AWS, інтегруючись з CI/CD та direct API.

Інтегруючи статичний аналіз Deep Learning з проєктом C++ необхідно:

1) обрати інструмент з урахуванням вимог проєкту, його розмірності, потреби в інтеграції з ресурсами і пам'яттю;

2) встановити ресурс Infer Neural разом з моделями нейронних мереж [5];

3) налаштувати нейронні моделі використання пам'яті, створивши файл конфігурації [6];

4) виконати аналіз коду разом з побудовою проєкту.

Обговорення результатів Deep learning

Статичний аналіз Deep learning вимагає більше обчислювальних ресурсів: 8-16 GB оперативної пам'яті, мінімум 4 ядра, 2-5 GB – для реалізації моделей, довший час, ніж для традиційного аналізу. Реалізація інтегрує поступове виконання аналізу коду, запуск збірок на сервері, використання спільного репозиторію моделей та аналіз коду лише на критичних шляхах. Для великих проєктів аналіз реалізується лише для змінених файлів, додаючи функції інкрементального аналізу, а повний аналіз – у вигляді щодобових тестових збірок.

Для візуалізації результатів Deep learning взято помилковий код з динамічним об'єктом і відсутністю оператора `delete[]` (рис. 1). Засоби виводу подають графіки викликів із виділенням шляхів появи проблем, точки розподілу та звільнення пам'яті, рівні достовірності нейронної мережі для кожного помилкового виявлення.

Моделі Deep learning генерують і хибно-позитивні результати, які можна зменшувати встановленням порогів достовірності, налаштовуючи моделі перевірки на *конкретний проєкт*, використовуючи зворотній зв'язок.

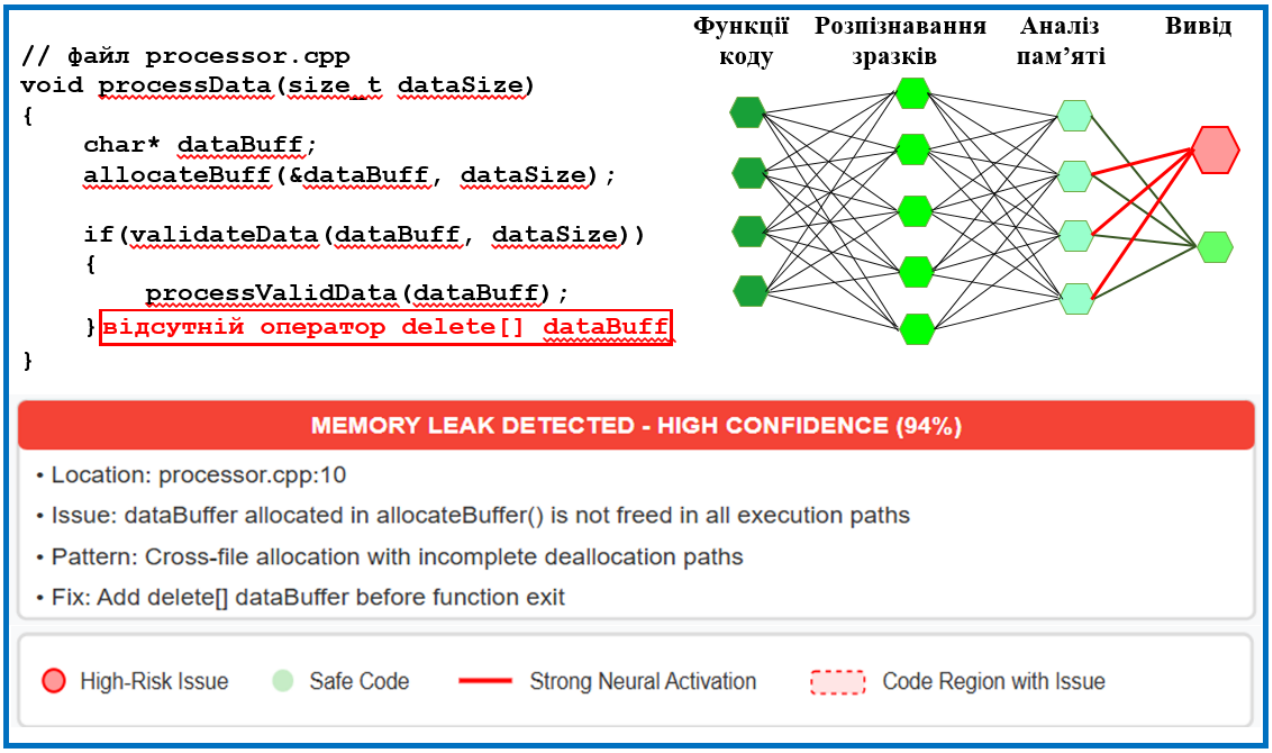


Рис.1 – Виявлення помилки пам'яті в коді з динамічним об'єктом

Отримано результати поліпшення використання пам'яті для кількох великих C++-проєктів: зменшення кількості збоїв їх роботи – понад 71%, на 89% зростання точності виявлення помилок, у 3 рази пришвидшення виявлення помилок, зменшення хибних спрацьовувань більше ніж на 40% порівнянно з традиційним аналізом.

Висновки

Аналіз Deep learning значно покращує знаходження прихованих помилок використання пам’яті C++-кодом через залучення шаблонів, їх аналітичного співставлення, які традиційні інструменти не виявляють. Поєднання його з нейро-мережами пришвидшує їх виявлення, скорочуючи час налагодження, якість, надійність та безпеку коду.

Список використаних джерел

1. Коваленко А.А., Куценко Т.Г., Шаповал А.С., Ситник О.В., Ні Я.С. Огляд методів аналізу безпечного програмного забезпечення. Control, Navigation and Communication Systems. Х.: ХНУР, 2025, №1. – с.156-161.

2. URL: https://github.com/google/oss-fuzz (date of access: 11.07.2025).

3. URL: https://cwe.mitre.org/data/definitions/762.html (date of access: 12.07.2025).

4. URL: https://arxiv.org/abs/2301.05869 (date of access: 12.07.2025).

5. URL: https://github.com/facebook/infer/tree/main (date of access: 17.08.2025).

6. URL: https://en.cppreference.com/w/cpp/language/ memory_model.html (date of access: 12.09.2025).

Відомості про авторів:

Мельник Василь Михайлович – кандидат фізико-математичних наук, доцент, викладач-методист відділення інформатики та комп’ютерної техніки Волинського фахового коледжу НУХТ. *Наукові інтереси:* дослідження сокетної взаємодії для реалізації продуктивності мереж, крос-платформенної взаємодії та систем розподілених обчислень.

E-mail: melnykvasyl@yahoo.com

Багнюк Наталія Володимирівна – кандидат технічних наук, доцент кафедри комп’ютерної інженерії та безпеки Луцького національного технічного університету. *Наукові інтереси:* комп'ютерні мережі та системи, захист інформації, системи моніторингу, обробка інцидентів та реагування, хмарні технології.

E-mail: bagnyuknata@gmail.com

Корчук Оксана Ігорівна – викладач-методист відділення інформатики та комп’ютерної техніки Волинського фахового коледжу НУХТ. *Наукові інтереси:* комп'ютерні мережі та системи.

E-mail: korchukoi@gmail.com

УДК 004.9:658.8

ОСОБЛИВОСТІ ПРОЄКТУВАННЯ CRM-СИСТЕМ ДЛЯ МАЛОГО ТА СЕРЕДНЬОГО БІЗНЕСУ

Валерія МЕЛЬНИЧЕНКО
Здобувачка вищої освіти першого рівня, кафедра ІПЗ
Наталія ШИБИЦЬКА
к.т.н., доц., доцент кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

У тезах розглянуто проєктування CRM-систем для малого та середнього бізнесу з урахуванням балансу між простотою, швидкістю впровадження та гнучкістю. Показано, що для малого бізнесу важливі низька вартість, простота використання та інтеграції з ключовими сервісами, тоді як для середнього – зростає роль автоматизації, аналітики та масштабованості. Проведено порівняльний аналіз CRM-рішень (KeepinCRM, KeyCRM, Pipedrive, Creatio). Встановлено, що «коробкові» CRM більше підходять малому бізнесу, а no-code платформи – середньому з складнішими процесами.

Ключові слова: *CRM, малий та середній бізнес, проєктування інформаційних систем, воронка продажів, автоматизація, інтеграції, no-code, багатоклієнтна архітектура (multi-tenant).*

Вступ

У сучасних умовах розвитку цифрових технологій CRM-системи є важливим інструментом автоматизації бізнес-процесів, управління клієнтською базою та підвищення ефективності продажів. Особливої актуальності набуває проєктування CRM-рішень для малого та середнього бізнесу, оскільки такі підприємства мають обмежені ресурси та потребують простих, доступних, але водночас функціональних систем. Невдало спроєктована CRM може не лише не забезпечити очікуваного ефекту, а й ускладнити роботу підприємства, що підкреслює важливість дослідження підходів до її розробки та впровадження.

У цифровій економіці CRM-системи виступають не лише як «база контактів», а як ядро управління взаємодією з клієнтами, контролю продажів, сервісного обслуговування та маркетингових комунікацій. Крім того, вони є інструментом стандартизації щоденних операцій – від обробки лідів до ведення угод і замовлень, виконання зобов'язань та організації повторних продажів. Для малого та середнього бізнесу це має особливе значення, оскільки такі підприємства зазвичай функціонують в умовах високої конкуренції та обмежених ресурсів і потребують швидкого результату від впровадження цифрових інструментів без тривалих та витратних проєктів.

Мета роботи

Метою роботи є аналіз особливостей проєктування CRM-систем для малого та середнього бізнесу та визначення ключових критеріїв ефективності таких систем на основі порівняння сучасних CRM-рішень, які репрезентують різні підходи: «швидкий старт та простота» (KeepinCRM, KeyCRM), «sales-first CRM з розвиненою роботою з воронкою та автоматизаціями» (Pipedrive) і «платформний no-code підхід з орієнтацією на моделювання процесів» (Creatio).

Матеріали та методи

У роботі застосовано метод порівняльного аналізу CRM-систем, зокрема KeepinCRM, KeyCRM, Pipedrive та Creatio, з урахуванням їх функціональних і архітектурних особливостей. Методологічною основою дослідження є поєднання ключових підходів до проєктування систем для малого та середнього бізнесу:

- Процесно-орієнтований підхід: узгодження логіки системи з реальними етапами угод.
- Користувацько-орієнтований підхід: забезпечення зручності та швидкого освоєння UI.
- Модульний підхід: можливість поетапного впровадження функцій та інтеграцій.
- Масштабований підхід: підтримка зростання кількості користувачів та обсягів даних.
- Інтеграційний підхід: взаємодія із зовнішніми сервісами (API, маркетплейси).
- Гнучкий підхід до кастомізації: адаптація полів та воронки під специфіку галузі.

Результати та обговорення

На основі проведеного аналізу (табл.1) пропонується концепція універсальної гібридної CRM-системи, що нівелює проблему «технологічного розриву» при зростанні бізнесу.

Узагальнюючи, ключова відмінність між «легкими» CRM (орієнтованими на швидке впровадження) та «платформними» CRM полягає у тому, який саме тип складності переноситься на користувача. У легких системах складність часто «захована» у готових сценаріях (ліди/замовлення/доставка), що скорочує час запуску. У платформних системах користувач отримує інструменти моделювання, але зростає важливість передпроєктного аналізу процесів, ролей, прав доступу і управління змінами – інакше гнучкість перетворюється на джерело ризиків. Це узгоджується з тим, що для малого та середнього бізнесу типовими проблемами є слабко визначені цілі, ресурсні обмеження та недостатньо продумані інтеграції, а для середнього бізнесу – складніші процеси і розподілені структури.

Основна ідея полягає у створенні єдиної екосистеми з динамічним рівнем складності інтерфейсу та функціоналу. Ключові характеристики рішення:

- Конструктор рівнів (Tiered Architecture): Для мікробізнесу система функціонує в «легкому» режимі (аналогічно KeepinCRM) – з простими тригерами та базовими воронками за доступною ціною. При масштабуванні бізнес не змінює платформу, а просто активує блок «Розширені бізнес-процеси».
- Інтеграція BPMN-дизайнера: У розширеному режимі користувач отримує інструментарій рівня Creatio для гнучкого моделювання складних міжфункціональних процесів, зберігаючи при цьому всі накопичені дані.
- Безшовна міграція функцій: Головною перевагою є відсутність потреби в перенесенні клієнтських баз, історії лідів чи налаштованих інтеграцій. Система масштабується «всередині» себе, що радикально знижує витрати на послуги інтеграторів та ризики втрати даних.
- Економічна адаптивність: Вартість володіння системою (TCO) корелює з поточними потребами бізнесу: оплата за базовий функціонал для початківців та преміальна тарифікація за доступ до інструментів управління процесами для великих компаній.

Таблиця 1

Порівняльна характеристика CRM-систем та концепції універсальної гібридної CRM-системи

№ п/п	CRM-системи	Модель / підходи проєктування	Етап розвитку бізнесу	Особливості
1	KeepinCRM	Користувацько-орієнтована легка CRM швидкого старту з базовою кастомізацією	Мікро-, малий та середній бізнес; початковий рівень цифровізації	«Все в одній системі», низький бар'єр входу, безкоштовний доступ на 1 користувача, типові воронки продажів, налаштування полів без складного програмування
2	Pipedrive	Процесно-орієнтована, інтеграційна sales pipeline CRM	Малий та середній бізнес; зріліші продажі з потребою в дисципліні роботи з угодами	Фокус на воронці продажів, активностях і наступних кроках, workflow automation, широка екосистема інтеграцій
3	KeyCRM	Користувацько-орієнтована, модульна інтегрована CRM для e-commerce	Малий, середній та зростаючий онлайн-бізнес; операційно зрілі e-commerce процеси	«Єдине вікно» комунікацій, замовлення, склад, відправки, інтеграції з маркетплейсами та службами доставки, налаштування без програмістів
4	Creatio	Процесно-орієнтована, інтеграційна, масштабована no-code CRM-платформа для моделювання процесів	Середній і великий бізнес; високий рівень процесної зрілості та міжфункціональної взаємодії	BPMN/process designer, AI, no-code, побудова наскрізних бізнес-процесів, аналітика та оптимізація вузьких місць
5	Універсальна гібридна CRM-система	Користувацько-орієнтований, модульний, масштабований, інтеграційний, процесно-орієнтований. Адаптивна багаторівнева CRM-платформа	Усі рівні зрілості бізнесу – від мікробізнесу до великого підприємства	«Легкий» режим для старту, розширений режим із BPMN-дизайнером, безшовне масштабування без міграції даних, економічна адаптивність

Такий підхід забезпечує життєздатність системи на всіх етапах життєвого циклу підприємства, роблячи її інвестиційно привабливим рішенням як для стартапів, так і для сталих корпорацій.

Висновки

Ефективне проєктування CRM-систем для малого та середнього бізнесу має виходити з реальної зрілості процесів і ресурсних обмежень компанії, а не лише з переліку «функцій у прайсі». Для малого бізнесу найбільш критичними є простота, швидкість запуску, мінімальна вартість входу та наявність готових інтеграцій, що підтримують основні операційні сценарії; це відповідає підходам CRM-рішень, орієнтованих на швидкий старт і типові процеси продажу/торгівлі. Для середнього бізнесу зростає потреба у системній автоматизації та

формалізації процесів, ролей і правил, що робить доцільними платформні no-code рішення з можливістю моделювання складних workflow/BPM-процесів та подальшої оптимізації. Водночас зростає складність передпроєктного аналізу та управління змінами, а отже потрібна дисципліна постановки вимог і дизайну даних. Ключовою проблемою проєктування CRM для малого та середнього бізнесу є досягнення балансу між простотою використання та функціональною гнучкістю: надмірна складність демотивує користувачів і підвищує ризики провалу, тоді як недостатня гнучкість створює бар'єри для масштабування. Практично оптимальним є підхід поетапного розвитку: старт із «легкої» CRM та/або MVP-функціоналу з чітким планом розширення (інтеграції, автоматизації, ролі, аналітика) або міграції на платформну модель у міру ускладнення процесів. Запропонована концепція універсальної гібридної CRM-системи полягає в об'єднанні переваг «легких» CRM швидкого старту, орієнтованих на простоту використання та готові інтеграції, із можливостями платформних no-code рішень щодо гнучкого моделювання, автоматизації та масштабування бізнес-процесів у межах єдиної системи.

Список використаних джерел

5. KeepinCRM. Офіційний сайт: позиціювання для малого/середнього бізнесу, складові системи та безкоштовний користувач [Електронний ресурс]. – Режим доступу: https://keepincrm.com/deals.

6. KeyCRM. Офіційний сайт: позиціювання, єдине вікно комунікацій, модулі онлайн-торгівлі та інтеграції [Електронний ресурс]. – Режим доступу: https://shelfy.com.ua/programs/keycrm/.

7. Pipedrive. Офіційні сторінки: позиціювання як CRM для малого бізнесу та можливості workflow automation [Електронний ресурс]. – Режим доступу: https://www.pipedrive.com/ru/products/sales.

8. Creatio. Офіційний сайт та опис технологій no-code: автоматизація бізнес-процесів, process designer з BPMN [Електронний ресурс]. – Режим доступу: https://www.creatio.com/ua/crm.

Відомості про авторів

Мельниченко Валерія Миколаївна – здобувачка вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення.

E-mail: lerokmelnichenko@gmail.com

Шибицька Наталія Миколаївна – доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інтелектуальні системи навчання та експертні методи оцінювання, методи та моделі штучного інтелекту, нечітка математика та нейронні мережі.

E-mail: shibnatnik@ukr.net

УДК 004.41:005.52

СТУДІЯ АНАЛІТИКИ І ДИЗАЙНУ: ДОМЕННА АРХІТЕКТУРА І ЗАДАЧІ АНАЛІТИЧНОГО ЗАБЕЗПЕЧЕННЯ ТЕХНОЛОГІЧНОГО СТЕКУ ЦИФРОВИХ B2B-ПРОЄКТІВ

Іван МІНЯЙЛО
Здобувач вищої освіти другого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Володимир ТАЛАЛАЄВ, доцент, к.т.н.

У роботі запропоновано модель «Студії аналітики і дизайну» для формалізації процесу доменної декомпозиції складних B2B-систем на основі DDD. Визначено критерії розподілу функціональності між кастомними мікросервісами та Low-Code інструментами, що дозволяє уникнути архітектурного боргу в гібридних системах.

Ключові слова: *Domain-Driven Design, Event Storming, мікросервіси, Low-Code, Power Platform, B2B-системи, Anti-Corruption Layer.*

Вступ

Сучасні цифрові B2B-платформи характеризуються зростаючою складністю: десятки взаємопов'язаних мікросервісів, тисячі бізнес-правил, гетерогенні команди розробки. У цьому контексті управління складністю стає самостійною задачею, яка вимагає системного аналітичного підходу. Предметно-орієнтоване проєктування (Domain-Driven Design, DDD) пропонує стратегічний інструментарій для декомпозиції складних систем, проте в практиці B2B-проєктів виникає характерний розрив: архітектори використовують DDD-термінологію для опису результату, але аналітичний процес, що веде до цього результату, залишається неформалізованим.

Додатковою проблемою є гібридний технологічний стек, де частина системи реалізується кастомним кодом, а частина - Low-Code інструментами (зокрема Microsoft Power Platform). Існуючі підходи або ігнорують Low-Code як архітектурний елемент, або надмірно розширюють його застосування на ключові домени, що призводить до деградації продуктивності та vendor lock-in.

Ціль роботи

Мета роботи - сформулювати доменну модель «Студії аналітики і дизайну», яка на основі принципів DDD визначає: (а) які аналітичні задачі необхідні для доменної декомпозиції; (б) якими методами та артефактами ці задачі вирішуються; (в) як результати аналітичної роботи перетворюються на обґрунтовані рішення щодо технологічного стеку та архітектури інтеграцій у гібридних B2B-системах.

Матеріали та методи

Дослідження базується на стратегічних патернах DDD: Bounded Context (обмежений контекст із власною моделлю та мовою), Context Map (карта відносин між контекстами), класифікація піддоменів (Core, Supporting, Generic). Основним методом аналітичного дослідження обрано Event Storming - структурований процес спільного моделювання домену

з бізнес-експертами, що має два рівні: Big Picture (виявлення меж контекстів) та Design Level (глибоке моделювання окремого контексту).

Запропонована модель Студії передбачає чотириетапний аналітичний pipeline: доменне інтерв'ю, Event Storming сесія, класифікація піддоменів та формування критеріїв вибору стеку. Ключовою відмінністю від класичного Center of Excellence є фокус на дослідженні домену, а не на контролі та стандартизації.

Результати та обговорення

Студія аналітики і дизайну – це постійно діючий формат спільної роботи, де доменний аналітик виступає фасилітатором дослідження. На етапі доменного інтерв'ю аналітик ставить бізнес-експертам структуровані питання: які процеси створюють цінність для клієнта, де виникають конфлікти між підрозділами, які терміни вживаються по-різному. На етапі Event Storming учасники розміщують бізнес-події на часовій шкалі, після чого аналітик виявляє кластери подій та визначає межі контекстів за ознаками зміни мови та відповідальності.

На основі виявлених контекстів аналітик проводить класифікацію піддоменів за чотирма критеріями: унікальність процесу, складність логіки, частота змін та критичність для бізнесу. Результат фіксується у матриці рішень (Табл. 1).

Таблиця 1

Матриця класифікації піддоменів та вибору стеку

Піддомен	Тип	Цінність	Обґрунтування	Стек
Ціноутворення B2B	Core	Висока	Складні інваріанти, 3 моделі ціни	.NET + Rich Domain Model
Маршрутизація замовлень	Core	Висока	Конкурентна перевага	Java/Node.js + EDA
Погодження документів	Supporting	Середня	Лінійний CRUD-workflow	Power Automate + Power Apps
Управління доступами	Generic	Низька	Стандартний протокол	Azure AD B2C / SaaS

Критерії застосовності Power Platform, сформульовані аналітиком, включають: складність логіки (лінійні CRUD-сценарії vs. розгалужені обчислення з інваріантами), навантаження (до 500 запитів/добу vs. понад 1000/годину), вимоги до тестування (мінімальні vs. критичні unit/integration тести) та безпеку (стандартні RBAC vs. кастомні ACL) .

Інтеграційний рівень між кастомними мікросервісами та Power Platform потребує Anti-Corruption Layer (ACL) – трансляційного шару, що запобігає забрудненню доменної моделі деталями Low-Code системи. Типова топологія: мікросервіси генерують Domain Events у Azure Service Bus, Azure Functions виступають як ACL, Power Automate підписується на події та виконує операції у Dataverse. Задача аналітика – визначити, які саме дані перетинають кордон контексту та зафіксувати це в Integration Contract Registry.

Ключовим ризиком гібридної архітектури є «архітектурний податок»: підтримка ACL-трансляторів та черг повідомлень вимагає висококваліфікованих інженерів. Аналітик повинен обґрунтувати, що вартість інтеграційного прошарку не перевищує економію від використання Low-Code для підтримуючих доменів. Іншим ризиком є повзуча складність у Power Platform, де бізнес-правила неконтрольовано розподіляються між Dataverse, Power Apps та Power

Automate. Студія встановлює пороги складності (не більше 3 рівнів вкладеності умов) та ініціює рефакторинг у кастомний сервіс при їх перевищенні.

Висновки

Запропонована модель Студії аналітики і дизайну переміщує фокус із архітектурного результату на аналітичний процес. Доменна декомпозиція є аналітичною задачею: Bounded Contexts виявляються через дослідження мови бізнесу, а не технічний аналіз коду. Event Storming забезпечує структуровану процедуру перетворення бізнес-знань на архітектурні артефакти. Класифікація піддоменів є рішенням першого порядку: Power Platform обґрунтована для Supporting та Generic доменів, але її застосування у Core Domains призводить до незворотного архітектурного боргу. Інтеграційний рівень (ACL, Message Brokers) має власну економічну вартість, яку аналітик повинен кількісно обґрунтувати.

Список використаних джерел

1. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2003. 560 p.
2. Vernon V. Implementing Domain-Driven Design. Addison-Wesley, 2013. 656 p.
3. Richardson C. Microservices Patterns. Manning Publications, 2018. 520 p.
4. Brandolini A. Introducing EventStorming. Leanpub, 2021.
5. Microsoft Corporation. Power Platform licensing and limits. 2024. URL: https://learn.microsoft.com/en-us/power-platform/ (дата звернення: 15.01.2025).
6. Newman S. Building Microservices. 2nd ed. O'Reilly Media, 2021. 616 p.

Відомості про автора:

Міняйло Іван Святославович – здобувач вищої освіти другого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси*: IT-projects management, інтелектуалізація.

E-mail: van.minyaylo@gmail.com

УДК 004.415.2

АЛГОРИТМ АВТОМАТИЗОВАНОЇ АГРЕГАЦІЇ РІЗНОРІДНИХ ЗАПИСІВ ІНГРЕДІЄНТІВ У СИСТЕМАХ ПЛАНУВАННЯ ХАРЧУВАННЯ

Олександра МІРОШНИЧЕНКО
Здобувачка вищої освіти першого рівня, кафедра ІПЗ
Яна БЄЛОЗЬОРОВА
к.т.н., доц., доцент кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

У статті досліджується проблема семантичного дублювання записів при агрегації різнорідних множин інгредієнтів у системах планування харчування. Запропоновано трьохетапний алгоритм, що включає нормалізацію рядкових ідентифікаторів на основі словника синонімів, конвертацію кількісних значень до базових одиниць СІ через конверсійну таблицю та агрегацію з подальшою серіалізацією у форматі JSON. Проаналізовано часову складність алгоритму та визначено домінуючу складову.

Ключові слова: *агрегація даних, нормалізація рядків, конвертація одиниць виміру, усунення дублікатів, REST API, Spring Boot, системи планування харчування.*

Вступ

Системи планування харчування оперують структурованими колекціями рецептів, де кожен запис містить перелік інгредієнтів із кількісними значеннями та одиницями виміру. При формуванні зведеного списку покупок з декількох рецептів виникає задача злиття гетерогенних множин: один семантичний об'єкт може бути представлений лексично відмінними рядковими ідентифікаторами та несумісними одиницями виміру у різних джерелах [3, 4]. Наївне об'єднання без нормалізації породжує семантичні дублікати, що знижує якість вихідних даних та утруднює подальшу обробку. Задача дедуплікації та агрегації подібних записів є частковим випадком загальнішої проблеми інтеграції даних [5], що активно досліджується у контексті вебплатформ і мікросервісної архітектури.

Ціль роботи

Метою дослідження є проєктування, формальний опис та верифікація алгоритму автоматизованої агрегації різнорідних записів інгредієнтів, що забезпечує:

1) лексичну нормалізацію рядкових ідентифікаторів;
2) конвертацію кількісних значень до єдиної системи одиниць;
3) агрегацію та серіалізацію результату для REST API вебплатформи.

Матеріали та методи

Алгоритм реалізовано мовою Java [1] з використанням фреймворку Spring Boot [2]. Персистентний рівень побудовано на Spring Data JPA (Hibernate) поверх СУБД PostgreSQL. Для нормалізації рядків застосовано підхід на основі словника синонімів у поєднанні з токенізацією та фільтрацією службових слів – методологія, що відповідає практикам entity resolution [3]. Основні технологічні рішення наведено у таблиці 1.

Таблиця 1

Технологічний стек системи

Компонент	Технологічне рішення	Роль у системі
Backend Framework	Spring Boot 3 (Java 17) [2]	Серверна логіка, IoC-контейнер, REST API
Data Layer	Spring Data JPA/Hibernate	ORM, керування транзакціями, PostgreSQL
Security	Spring Security + JWT	Аутентифікація, RBAC (ролі: admin/author/consumer)
Frontend	JS ES6+, HTML5, CSS3	SPA-інтерфейс, взаємодія з API через fetch()
Алгоритмічний модуль	AggregationService (Java)	Нормалізація, конвертація одиниць, агрегація, серіалізація

Результати та обговорення

Розроблений алгоритм AggregationService виконується у три послідовні етапи.

1. *Нормалізація рядкових ідентифікаторів.* Кожен вхідний рядок приводиться до нижнього регістру, токенізується за пробільними символами та очищується від службових слів. Отриманий токен перевіряється у структурі HashMap<String, String> (словник синонімів, 2000 записів) і замінюється канонічною формою. Якщо відповідного запису всловнику не знайдено, токен використовується без змін.

2. *Конвертація кількісних значень та агрегація.* Числові значення перераховуються до базових одиниць СІ (маса → кг, об'єм → л) через конверсійну таблицю з підтримуваних одиниць виміру. Записи з однаковим канонічним ідентифікатором попередньо сортуються за ключем, після чого об'єднуються за один прохід. Це дозволяє уникнути повторного перегляду колекції та знизити обсяг проміжних обчислень.

3. *Зворотна конвертація та серіалізація.* Агреговані значення переводяться до зручних для відображення одиниць виміру та впорядковуються за категоріями предметної області. Сформований результат серіалізується у JSON-відповідь відповідного REST-ендпоінту вебплатформи.

Загальна часова складність алгоритму визначається кількістю рецептів у плані, середньою кількістю інгредієнтів у кожному рецепті та логарифмічним множником, що вноситься етапом сортування перед агрегацією. Обсяг використовуваної пам'яті є лінійним відносно загальної кількості записів і визначається розміром проміжних колекцій під час обробки.

Верифікацію коректності нормалізації виконано методом ручної розмітки: сформовано еталонний набір з 1840 записів інгредієнтів (120 рецептів), кожному з яких експертно призначено канонічну форму. Точність вимірювалась як частка записів, канонічна форма яких збіглась з еталоном, серед усіх оброблених записів. Для початкового словника з 800 термінів отримано точність 94,3%; після розширення до 2000 термінів – 97,8%. Основним джерелом помилок залишаються лексичні варіації регіонального походження.

Ефективність усунення дублікатів оцінено як відношення розміру агрегованої колекції до розміру колекції при наївному злитті без нормалізації. На тестовому наборі встановлено, що запропонований алгоритм усуває 73 % семантичних дублікатів порівняно з базовим підходом прямого об'єднання.

Навантажувальне тестування проведено засобом Apache JMeter із симуляцією 50 одночасних користувачів та 1000 запитів. Тестовий план охоплював 21 рецепт із середнім

числом інгредієнтів 10 на рецепт. Середній час відповіді склав 24 мс, а у 95% випадків не перевищував 38 мс, що відповідає вимогам інтерактивних вебзастосунків.

Висновки

Запропонований трьохетапний алгоритм нормалізації, агрегації та серіалізації різнорідних записів інгредієнтів забезпечує точність нормалізації до 97,8% на наборі 1840 записів та усуває 73% семантичних дублікатів порівняно з прямим об'єднанням без нормалізації. Інтеграція модуля у REST API на базі Spring Boot забезпечує час відповіді не більше 38 мс у 95% випадків, що відповідає вимогам інтерактивних вебзастосунків. Напрямом подальших досліджень є заміна словникового підходу на векторні моделі семантичної схожості (Word2Vec, FastText) для підвищення стійкості нормалізації до нових лексичних варіацій.

Список використаних джерел

1. Bloch J. Effective Java. – 3rd ed. – Addison-Wesley Professional, 2018. – 412 p.

2. Walls C. Spring in Action. – 6th ed. – Manning Publications, 2022. – 520 p.

3. Christen P. Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. – Springer, 2012. – 270 p.

4. Dong X. L., Srivastava D. Big Data Integration // Proceedings of the VLDB Endowment. – 2013. – Vol. 6, № 11. – P. 1188–1189. DOI: 10.14778/2536222.2536248.

5. Rahm E., Bernstein P. A. A survey of approaches to automatic schema matching // The VLDB Journal. – 2001. – Vol. 10. – P. 334–350. DOI: 10.1007/s007780100057.

Відомості про авторів

Мірошниченко Олександра Олександрівна – здобувачка вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* backend-розробка на Java, алгоритми обробки та інтеграції даних.

E-mail: am.miroxa@gmail.com

Бєлозьорова Яна Андріївна – доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення, обробка зображень, агрегація даних.

E-mail: yana.bielozorova@npp.kai.edu.ua

УДК 004.2(043.2)

КРИТЕРІЇ ВИБОРУ МІКРОСЕРВІСНИХ СИСТЕМ ДЛЯ ТЕСТУВАННЯ МЕТОДІВ АРХІТЕКТУРНОЇ РЕКОНСТРУКЦІЇ

Артем МОРГУН
Здобувач вищої освіти третього рівня, кафедра ІПЗ
Науковий керівник – Андрій ГІЗУН, професор, к.т.н.

Запропоновано критерії вибору тестових мікросервісних систем для оцінювання методів реконструкції архітектури програмного забезпечення: архітектурна складність, наявність телеметрії, відтворюваність, реалістичність та оцінюваність. Порівняльний аналіз Sock Shop, TeaStore та DeathStarBench показав відсутність універсального рішення та обґрунтував доцільність їх комбінованого використання.

Ключові слова: *мікросервісна архітектура, реконструкція архітектури, тестові системи, порівняльний аналіз, телеметрія.*

Вступ

Мікросервісна архітектура є широко застосовуваним підходом до розробки сучасних розподілених систем завдяки своїй гнучкості, масштабованості та незалежності компонентів. Водночас така декомпозиція ускладнює аналіз, супровід і еволюцію систем із великою кількістю сервісів. Це зумовлює потребу в методах реконструкції архітектури програмного забезпечення, які дозволяють відновити структуру системи, залежності між сервісами та їхню поведінку для подальшого аналізу й планування її розвитку з метою зменшення накопичення технічного боргу.

У наукових дослідженнях для оцінювання таких методів широко використовуються [1] тестові мікросервісні системи (вихідний код і розгорнуте середовище), зокрема Sock Shop [2], TeaStore [3] та DeathStarBench [4]. Однак їхній вибір часто здійснюється без чітко визначених критеріїв, що може негативно впливати на валідність отриманих результатів.

Ціль роботи

Формування критеріїв вибору мікросервісних систем для тестування методів реконструкції архітектури програмного забезпечення.

Матеріали та методи

У роботі застосовано аналітичний підхід та порівняльний аналіз для узагальнення практик використання мікросервісних систем і оцінювання їхньої придатності для задач реконструкції архітектури програмного забезпечення.

Результати та обговорення

З метою оцінювання придатності тестових мікросервісних систем для тестування нових методів реконструкції архітектури програмного забезпечення було проведено порівняльний аналіз найбільш поширених систем: Sock Shop, TeaStore та DeathStarBench.

Результати оцінювання структуровано у вигляді п’яти агрегованих критеріїв: архітектурна складність, наявність телеметрії, відтворюваність, реалістичність та оцінюваність.

Архітектурна складність характеризує рівень декомпозиції системи на окремі сервіси та рівень взаємодій між ними. Наявність телеметрії відображає доступність і повноту даних під час роботи, зокрема логів, метрик і трасувань, необхідних для виконання динамічної реконструкції. Відтворюваність визначає можливість розгортання системи та повторення експериментів у контрольованих умовах. Реалістичність оцінює ступінь відповідності системи практикам побудови реальних мікросервісних застосунків, зокрема щодо характеру навантаження та типових сценаріїв взаємодії. Оцінюваність характеризує здатність системи забезпечувати об'єктивну перевірку точності результатів реконструкції, зокрема завдяки наявності еталонної архітектури або можливості її відновлення.

Таблиця 1

Відповідність наявних програмних систем критеріям

Критерій	Sock Shop	TeaStore	DeathStarBench
Архітектурна складність	Середня 13 контейнерів	Середньо-низька 7 контейнерів	Висока ~27 контейнерів
Наявність телеметрії	Дуже висока Трасування, логи, метрики	Дуже висока Трасування, логи, метрики	Висока Трасування, але 1%
Відтворюваність	Середньо-низька Застарілі версії	Висока декілька шляхів	Середня потрібна збірка wrk2 та 8 – 16 ГБ RAM
Реалістичність	Висока Продукт	Середня Інфраструктура	Середня Інфраструктура
Оцінюваність	Висока Open Api, test plan	Дуже висока MASCOTS'18, 20+ робіт	Висока ASPLOS'19 + Thrift IDL

TeaStore визначено як найбільш придатну тестову систему для задач архітектурної реконструкції. Це зумовлено наявністю повної еталонної моделі, представленої у рецензованій публікації на MASCOTS'18, реєстру сервісів як еталонного джерела даних, а також вбудованої телеметрії. Додатковою перевагою є низька вартість розгортання та інструментація на рівні методів, яка забезпечує більш детальну інформацію порівняно з RPC-орієнтованим трасуванням.

DeathStarBench доцільно використовувати як додаткову систему для стрес-тестування підходу. Його перевагами є значно більша топологія (приблизно 27 контейнерів у бенчмарку socialNetwork), використання кількох мов програмування, а також застосування широко поширених інструментів телеметрії (OpenTracing, Jaeger). Крім того, IDL-файли Thrift можуть слугувати надійним джерелом опису контрактів між сервісами.

Sock Shop, попри наявність розвинених засобів телеметрії (Prometheus, Grafana, Jaeger, EFK) та реалістичну предметну область електронної комерції, доцільно використовувати лише як допоміжну тестову систему. Основною причиною є припинення підтримки репозиторію в грудні 2023 року, а також застарілі версії базових образів.

Таким чином, TeaStore є найбільш придатною тестовою системою для перевірки результатів реконструкції завдяки наявності еталонних представлень. DeathStarBench доцільно використовувати для оцінювання роботи методу на великомасштабних і різнорідних системах. Sock Shop варто розглядати як допоміжну тестову систему, зокрема для сценаріїв, що передбачають роботу із застарілою кодовою та інфраструктурною базою.

Висновки

У роботі визначено критерії оцінювання тестових мікросервісних систем: архітектурна складність, телеметрія, відтворюваність, наближеність до реальних умов і можливість перевірки результатів. Порівняльний аналіз Sock Shop, TeaStore та DeathStarBench показав відсутність універсального рішення й необхідність компромісу між простотою використання та реалістичністю. Обґрунтовано комбінований підхід: TeaStore як основна система для перевірки результатів реконструкції, DeathStarBench – для оцінювання методу на великомасштабних і різнорідних системах, Sock Shop як допоміжна система для сценаріїв із застарілою кодовою та інфраструктурною базою.

Список використаних джерел

1. Zhou X., Peng X., Xie T., Sun J., Xu C., Ji C., Zhao W. Benchmarking microservice systems for software engineering research // In: Proceedings of the 40th ACM/IEEE International Conference on Software Engineering (ICSE 2018). – New York : Association for Computing Machinery, 2018. – P. 323–324. – https://doi.org/10.1145/3183440.3194991

2. Smith S., Robinson E., Frederiksen T., Stevens T., Cerny T., Bures M., Taibi D. Benchmarks for End-to-End Microservices Testing // In: Proceedings of the 17th IEEE International Conference on Service-Oriented System Engineering (SOSE 2023). – Piscataway : IEEE, 2023. – P. 60–66. – https://doi.org/10.1109/SOSE58276.2023.00013

3. Bliudze S., De Palma G., Giallorenzo S., Lanese I., Zavattaro G., Zemtsop Ndadji B. A. Adaptable TeaStore // In: Proceedings of the 1st Workshop on Adaptable Cloud Architectures (WACA 2025). – Open Publishing Association, 2025. – (Electronic Proceedings in Theoretical Computer Science, EPTCS; vol. 438). – P. 1–14. – https://doi.org/10.4204/EPTCS.438.1

4. Sedghpour M. R. S., Obeso Duque A., Cai X., Skubic B., Elmroth E., Klein C., Tordsson J. HydraGen: A microservice benchmark generator // In: Proceedings of the 2023 IEEE 16th International Conference on Cloud Computing (CLOUD 2023). – Piscataway: IEEE, 2023. – P. 189–200. – https://doi.org/10.1109/CLOUD60044.2023.00030

Відомості про автора

Моргун Артем Вікторович – здобувач вищої освіти третього рівня, кафедра інженерії програмного забезпечення Факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* мікросервісна архітектура, тестові системи, порівняльний аналіз.

E-mail: 2175566@stud.kai.edu.ua

УДК 004.056.55:004.42

АРХІТЕКТУРНА МОДЕЛЬ ІНТЕГРАЦІЇ ПОСТКВАНТОВИХ АЛГОРИТМІВ ЦИФРОВОГО ПІДПИСУ У МІКРОСЕРВІСНУ АРХІТЕКТУРУ НА ОСНОВІ JWT-АВТЕНТИФІКАЦІЇ У СЕРЕДОВИЩІ .NET

Микола ОНАЙ
Доцент кафедри ПЗКС, к.т.н.
Федір ПЕТРЕНКО
Аспірант 1 курсу кафедри ПЗКС
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

У роботі розглядається проблема інтеграції постквантових алгоритмів цифрового підпису стандартів NIST у мікросервісну архітектуру на основі JWT-автентифікації у середовищі .NET. Досліджено обмеження алгоритмів RS256 і ES256 в умовах квантових загроз, проаналізовано підходи до PQC-міграції та виявлено їх практичну несумісність з існуючими підходами у корпоративних системах. Запропоновано архітектурну модель PQC-JWT, що реалізує оброблення постквантового підпису на рівні API Gateway: верифікацію ML-DSA-65 з кешуванням результатів, генерацію ідентифікатора сесії на основі SHA-256 хешу, відмову від передачі токена до сервісів та передачу верифікованих атрибутів користувача через заголовки X-Claims. Це забезпечує ізоляцію PQC-логіки на рівні єдиного компонента. Модель формує основу для подальшого експериментального дослідження методу оптимізації постквантової JWT-автентифікації у середовищі .NET 10.

Ключові слова: *постквантова криптографія, цифровий підпис, еліптична криптографія, програмне забезпечення, JWT-автентифікація, мікросервісна архітектура, ML-DSA, NIST PQC, .NET 10, API Gateway.*

Вступ

Сьогодні мікросервісна архітектура є одним з основних підходів до побудови сучасних розподілених програмних систем. При цьому важливим питанням забезпечення безпеки таких систем є захист автентифікації, яка традиційно реалізується на основі JSON Web Token (JWT) з алгоритмами цифрового підпису RS256 або ES256 [1]. Більшість сучасних хмарних платформ та корпоративних систем використовують саме цей механізм як основу для верифікації ідентичності при комунікації між сервісами. Проте проблема криптографічної стійкості JWT-автентифікації набуває нової актуальності у зв’язку з розвитком квантових обчислень.

Алгоритм Шора, реалізований на достатньо потужному квантовому комп’ютері, дозволяє ефективно розв’язати задачі факторизації та дискретного логарифмування, на яких ґрунтується стійкість алгоритмів RSA та ECDSA [2]. Це означає, що існуючі механізми підпису JWT-токенів стають вразливими. Загроза набуває практичного характеру вже сьогодні через атаку типу «harvest now, decrypt later». Перехоплені токени можуть бути скомпрометовані після появи достатніх квантових обчислювальних ресурсів [3].

У 2024 році NIST затвердив два стандарти цифрового підпису: ML-DSA (FIPS 204) [4] та SLH-DSA (FIPS 205) [5]. FALCON (майбутній FN-DSA, FIPS 206) відібраний для стандартизації, однак наразі перебуває на стадії draft. Заміна класичного підпису на постквантовий у JWT-автентифікації породжує нові проблеми: збільшення розміру токена на

фіксовані +3.3 КБ для ML-DSA-65 незалежно від обсягу даних, порушення зворотної сумісності з існуючою інфраструктурою та додаткові витрати обчислювальних ресурсів у кожному сервісі для верифікації.

Ціль роботи

Метою роботи є розроблення архітектурної моделі інтеграції постквантових алгоритмів цифрового підпису у мікросервісну JWT-автентифікацію у середовищі .NET 10, яка забезпечує квантову стійкість до атак алгоритму Шора та загроз типу HNDL, реалізує обробку постквантового підпису на рівні API Gateway з централізованим управлінням сесіями, та забезпечує повну ізоляцію криптографічної логіки від downstream-сервісів без змін у їх коді.

Матеріали та методи

У процесі виконання роботи використовувались матеріали стандартів NIST щодо постквантової криптографії, зокрема специфікації алгоритмів ML-DSA [4] та SLH-DSA [5], а також чинні специфікації RFC 7515 [6], RFC 7518 [7], RFC 7519 [1], що визначають структуру та механізми підпису JSON Web Token.

Для аналізу характеристик алгоритмів цифрового підпису застосовувався метод порівняльного аналізу за такими критеріями: розмір публічного ключа, розмір підпису, накладні витрати операцій підпису та верифікації при розподіленому виконанні на кожному сервісі, рівень безпеки за класифікацією NIST, а також доступність вбудованої підтримки у платформі .NET 10. Для порівняння використовувалися класичні алгоритми ES256 та RS256, які є стандартом для JWT-автентифікації у мікросервісних системах.

Для створення архітектурної моделі використовувався метод структурного проєктування розподілених систем із урахуванням принципів криптогнучкості та керованої міграції. Архітектурні рішення базувалися на аналізі обмежень мікросервісних систем, зокрема розміру HTTP-заголовків, швидкодії верифікації токенів та ізоляції криптографічної логіки від downstream-сервісів і керування сесіями на рівні Gateway.

Практична реалізація запропонованої моделі орієнтована на платформу .NET 10 з використанням вбудованого класу MLDsa для операцій підпису та верифікації ML-DSA, а також стандартних механізмів кешування для реалізації сесійного шару на рівні API Gateway.

Результати та обговорення

Аналіз існуючих підходів до PQC-міграції у JWT-автентифікації виявив їх суттєві практичні обмеження. Повна заміна Identity Provider на PQC-сумісний потребує одночасного оновлення всіх downstream-сервісів та клієнтів, оскільки змінюється формат токена та механізм верифікації на кожному вузлі системи [11]. Перехід від JWT до mTLS із PQC-сертифікатами вирішує проблему на транспортному рівні, однак кардинально змінює протокол автентифікації та є несумісним з OAuth 2.0/OIDC [12]. Sidecar-proxy підхід ізолює PQC-логіку на рівні мережевої інфраструктури, проте не впливає на квантову стійкість підпису JWT-токена, який продовжує передаватися між сервісами у незахищеному форматі [11]. Жоден із розглянутих підходів не забезпечує інтеграцію PQC без суттєвих змін у всіх вузлах системи, що визначає наукову новизну запропонованого рішення – ізоляції PQC-логіки на рівні єдиного компонента (API Gateway) зі збереженням існуючого коду downstream-сервісів.

З точки зору моделі загроз JWT-автентифікація у мікросервісній архітектурі вразлива до двох окремих векторів квантових атак: атаки Шора [2] на приватний ключ IdP через

публічний JWKS-endpoint та HNDL-атаки [3] на перехоплені Bearer-токени у транзитному трафіку між сервісами. Слід підкреслити, що захист на рівні транспорту (TLS) не нейтралізує жодного із цих векторів; квантовостійкий підпис необхідний саме на рівні JWT.

Серед фіналізованих стандартів NIST для JWT-контексту найбільш збалансованим виявився ML-DSA-65 [8] (таблиця 1). Алгоритм забезпечує рівень безпеки NIST Level 3 (еквівалент AES-192), є детермінованим і має вбудовану підтримку у .NET 10 через клас MLDsa. Розмір підпису 3293 байта вписується у стандартний ліміт HTTP-заголовків (8 КБ), що є важливим для Bearer-токенів. SLH-DSA відхилено через підпис 7856 байтів та experimental статус API SlhDsa (SYSLIB5006) у .NET 10. FALCON (FN-DSA) не розглядався як нефіналізований стандарт (draft FIPS 206).

Таблиця 1

Порівняння алгоритмів цифрового підпису для JWT [4,5]

Алгоритм	**Тип крипто-графії**	**Математична основа**	**Розмір відкритого ключа (Байти)**	**Розмір підпису (Байти)**	**Швидкість (Підпис / Перевірка)**	**Оцінка для використання у JWT**
RS256 (RSA-2048)	Класична	Факторизація цілих чисел	256	256	Повільно / Дуже швидко	Класичний базовий алгоритм
ES256 (ECDSA P-256)	Класична	Еліптичні криві	64	64	Швидко / Швидко	Сучасний галузевий стандарт
EdDSA (Ed25519)	Класична	Еліптичні криві	32	64	Дуже швидко / Дуже швидко	Високопродуктивна альтернатива ES256
ML-DSA-44 (Рівень 2)	Постквантова	Решітки (Lattices)	1312	2420	Швидко / Дуже швидко	Перспективний PQC-стандарт.
ML-DSA-65 (Рівень 3)	Постквантова	Решітки (Lattices)	1952	3293	Швидко / Дуже швидко	Основний рекомендований стандарт NIST.

На основі проведеного аналізу, наведеного у таблиці 1, запропоновано архітектурну модель PQC-JWT, що реалізує два ключові механізми. Перший – гібридний підпис JWT-токена: IdP генерує токен з двома незалежними підписами, постквантовим ML-DSA-65 та класичним RS256 [9], що забезпечує можливість поступової міграції у перехідний період відповідно до підходу гібридних схем. Другий – обробка постквантового підпису на рівні API Gateway: Gateway верифікує ML-DSA-65 підпис та кешує результат верифікації за SHA-256 хешем токена з TTL, що відповідає залишковому часу дії токена; витягує атрибути користувача та передає downstream-сервісам виключно заголовки X-Claims і X-Session-Id. JWT-токен не передається за межі Gateway. Downstream-сервіси читають X-Claims для реалізації бізнес-логіки та прокидають X-Session-Id у вихідних запитах, що дозволяє Gateway централізовано валідувати сесію при міжсервісних викликах. Архітектуру моделі наведено на рис. 1.

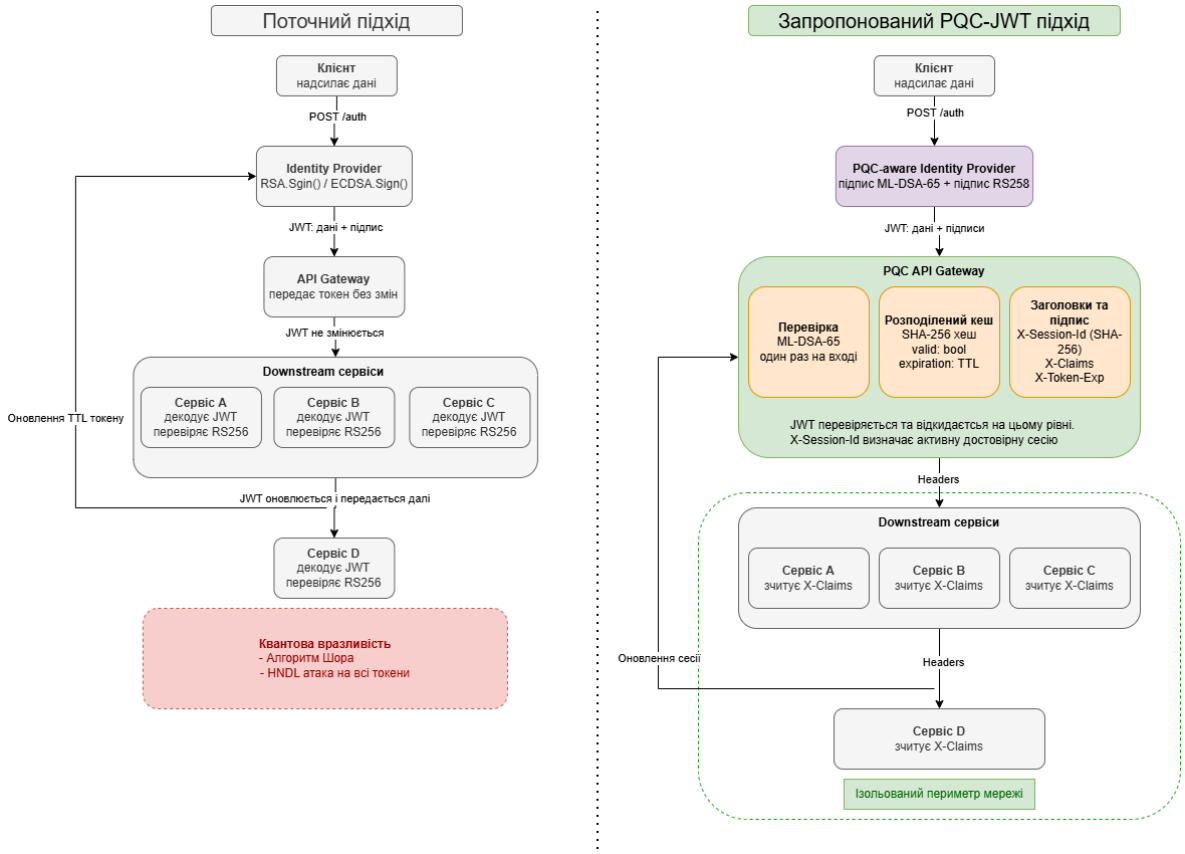


Рис. 1 – Архітектурна схема моделі PQC-JWT: порівняння поточного та зпропонованого підходу

Важливою характеристикою моделі є підтримка декларативної конфігурації через appsettings.json (Options Pattern), що дозволяє змінювати параметри PQC-автентифікації: вибір алгоритму підпису, параметри кешування сесій та TTL без перекомпіляції сервісів. Гібридний режим підпису є єдиним підтримуваним на перехідний період, оскільки дозволяє IdP поступово мігрувати до постквантових алгоритмів без одночасного оновлення всієї інфраструктури. Запропонована модель формує теоретичну та архітектурну основу для подальшого розроблення та експериментальної оцінки методу оптимізації постквантової JWT-автентифікації у мікросервісній архітектурі.

Висновки

Проведений аналіз показав, що жоден з існуючих підходів до PQC-міграції не вирішує проблему без значних змін в існуючій інфраструктурі. Виявлено два незалежні вектори квантових загроз для JWT-автентифікації у мікросервісах: атака Шора на ключ IdP та HNDL-атака на Bearer-токени в транзитному трафіку, що вимагає захисту саме на рівні підпису токена.

За результатами порівняльного аналізу фіналізованих алгоритмів NIST PQC визначено ML-DSA-65 як оптимальний для JWT-контексту у .NET 10: стабільний вбудований API (MLDsa), детермінований підпис та прийнятний розмір токена відрізняють його від SLH-DSA; FALCON виключено як не фіналізований стандарт.

Запропоновано архітектурну модель PQC-JWT, що реалізує два механізми: гібридний підпис (ML-DSA-65 + RS256) для забезпечення поступової міграції IdP та обробку постквантового підпису на рівні API Gateway з передачею downstream-сервісам заголовків X-Claims та X-Session-Id замість JWT-токена. Це забезпечує повну ізоляцію криптографічної

логіки від downstream-сервісів та централізоване управління сесіями без змін у коді мікросервісів.

Подальші дослідження передбачають практичну реалізацію запропонованої моделі у середовищі .NET 10 та експериментальну оцінку методу оптимізації постквантової JWT-автентифікації: вимірювання швидкодії підпису та верифікації ML-DSA-65, пропускної здатності API Gateway під навантаженням та ефективності кешування сесій порівняно з локальною верифікацією JWT у кожному сервісі.

Список використаних джерел

1. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. – IETF, 2015.
2. Shor P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. // SIAM Journal on Computing. – 1997. – Vol. 26, No. 5. – P. 1484-1509.
3. Kagai F., Branch P., But J., Allen R. Harvest-Now, Decrypt-Later: A Temporal Cybersecurity Risk in the Quantum Transition. // Telecom. – 2025. – Vol. 6, No. 4. – Art. 100. – DOI: 10.3390/telecom6040100.
4. NIST FIPS 204. Module-Lattice-Based Digital Signature Standard (ML-DSA). – NIST, 2024.
5. NIST FIPS 205. Stateless Hash-Based Digital Signature Standard (SLH-DSA). – NIST, 2024.
6. Jones M. JSON Web Signature (JWS). RFC 7515. – IETF, 2015.
7. Jones M. JSON Web Algorithms (JWA). RFC 7518. – IETF, 2015.
8. Ducas L., Kiltz E., Lepoint T., Lyubashevsky V., Schwabe P., Seiler G., Stehlé D. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. // IACR Transactions on Cryptographic Hardware and Embedded Systems. – 2018. – Vol. 2018, No. 1. – P. 238–268.
9. Bindel N., Herath U., McKague M., Stebila D. Transitioning to a Quantum-Resistant Public Key Infrastructure. // Post-Quantum Cryptography. PQCrypto 2017. Lecture Notes in Computer Science, vol. 10346. – Springer, Cham, 2017. – P. 384-405.
10. Newman S. Building Microservices: Designing Fine-Grained Systems. – 2nd ed. – O'Reilly Media, 2021.
11. Weller D., Schellekens R. Incorporating Post-Quantum Cryptography in a Microservice Environment. Semantic Scholar – University of Amsterdam, OS3, 2020. – URL: https://rp.os3.nl/2019-2020/p13/report.pdf.
12. Alnahawi N., Müller J., Oupický J., Wiesmaier A. A Comprehensive Survey on Post-Quantum TLS. // IACR Communications in Cryptology. – 2024. – Vol. 1, No. 2. – DOI: 10.62056/ahee0iuc.

Відомості про авторів

Онай Микола Володимирович – доцент, к.т.н., доцент кафедри програмного забезпечення комп'ютерних систем. *Наукові інтереси:* скінченні поля, криптографія, криптографія еліптичних кривих, задача дискретного логарифмування, кібербезпека, математично-алгоритмічна оптимізація криптографічних методів.

E-mail: onay@pzks.fpm.kpi.ua

Петренко Федір Володимирович – аспірант 1 курсу кафедри програмного забезпечення комп'ютерних систем. *Наукові інтереси:* постквантова криптографія, кібербезпека розподілених систем, хмарна архітектура на базі платформи .NET/Azure, оптимізація криптографічних методів у середовищах з обмеженими ресурсами.

E-mail: petrenkofedia@gmail.com

UDC 004.4

USAGE OF EVENT-DRIVEN ARCHITECTURE FOR IMPLEMENTING COMPLEX SYSTEMS IN GAME DEVELOPMENT

Volodymyr OSYCHNIUK
Bachelor's Degree student, Software Engineering Department
National University «Kyiv Aviation Institute»
Scientific supervisor – Yana BIELOZOROVA, PhD

This paper examines event-driven architecture (EDA) in modern game development, focusing on how Unity, Unreal Engine, and Godot 4 implement event-based communication between decoupled gameplay systems. It addresses the limitations of direct inter-object dependencies in large-scale games and shows how EDA enables indirect communication so systems like UI, audio, physics, AI, and analytics can respond independently to shared events. The paper surveys key patterns associated with EDA, including the observer pattern, publish–subscribe messaging, event queues and command objects, and evaluates their trade-offs in real-time, frame-based environments. The analysis highlights differences in abstraction, tooling, and performance across engines, and identifies a common tendency for developers to introduce additional messaging layers such as event buses or mediators to manage complexity and further reduce coupling in large projects.

Keywords: *event-driven architecture, EDA, game development, Unity, Unreal Engine, Godot, observer pattern, event bus, signals, publisher-subscriber.*

Introduction

Modern video games sit among the most structurally complex pieces of software ever shipped. A production-grade title may coordinate hundreds of independent systems simultaneously – physics, animation, audio, networking, AI, UI, achievements, analytics – each of which must respond fluidly to changes in game state without dragging the rest of the codebase into a morass of hard-wired dependencies.

The most straightforward approach to inter-system communication – direct method calls between objects – scales poorly. When a player character dies, the audio system needs to trigger a sound, the UI needs to display a respawn countdown, and the analytics pipeline needs to log the event. Connecting each of these directly to the player object creates a web of hard dependencies that violates separation of concerns and makes every system involved harder to test or replace in isolation.

Event-driven architecture inverts this relationship. Rather than a producer reaching out to consumers directly, it emits an event into a shared medium and trusts that any interested subscriber will react independently. This decoupling is the central value proposition of EDA – and the reason it has become a foundational design pattern in both enterprise software and game development.

The goal of this work

This paper surveys the theory underpinning EDA, describes its principal implementation patterns with attention to their practical trade-offs, and compares how Unity, Unreal Engine, and Godot 4 support it at the engine level. Where gaps or inconsistencies are identified, the paper also examines how developer communities have responded.

Models and Methods

Several implementation patterns commonly associated with event-driven architecture are examined and compared throughout this work. Although these patterns share the common goal of reducing direct coupling between systems, they differ substantially in how events are routed, consumed, and synchronised.

The *observer pattern* establishes a shared event channel through which producers emit typed event objects without holding any reference to consumers. Subscribers register interest in a specific event type and are notified automatically upon publication. This is the most broadly applicable of all patterns and serves as the foundation for all engine-native implementations examined in this paper.

The *event queue* interposes a FIFO buffer between publication and consumption, deferring dispatch to a controlled moment in the game loop – typically the start of the following frame. This eliminates the mid-frame state mutations and cascading race conditions that arise when synchronous observer chains fire during physics or animation updates. The trade-off is added latency in event delivery, which matters in latency-sensitive subsystems.

The *command pattern* represents actions as serializable objects that may be queued, replayed, or reversed at a later time. Although not inherently event-driven in isolation, command-based systems are frequently integrated with EDA in network replication, input recording, and deterministic replay systems where auditability and temporal ordering are essential requirements.

From an implementation standpoint, the three leading engines approach these patterns at substantially different levels of abstraction. Unity exposes raw C# event delegates for maximum throughput, a serialisable UnityEvent wrapper for Inspector-driven wiring, and the ScriptableObject event channel pattern for communication that crosses scene boundaries. Unreal Engine offers Blueprint Event Dispatchers for designer-facing publish-subscribe workflows and C++ TMulticastDelegate structures for lower-level event dispatch, supplemented in UE5 by the Gameplay Message Router – a centralised, tag-based messaging subsystem. Godot 4 integrates signals as a first-class node communication primitive; typed signal declarations, introduced in Godot 4.0, improve static validation and editor tooling compared with earlier dynamically typed signal definitions.

Results

One recurring pattern across all three engines is the frequent use of intermediary messaging layers to reduce direct dependencies between gameplay systems. Figure 1 illustrates one common implementation style in which publishers emit events into a central dispatcher that forwards them to interested subscribers without requiring direct object references.

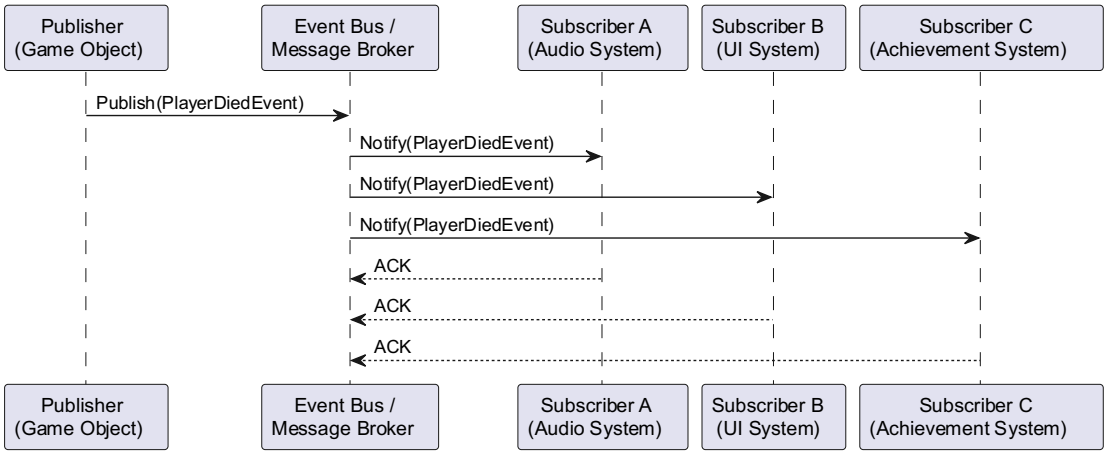


Figure 1. PlayerDiedEvent propagated from publisher to three independent subscriber systems via a centralised event bus

While publish-subscribe architectures do not inherently require a centralised global bus, many large game projects adopt some form of shared mediation layer to simplify cross-system communication.

In practice, the three engines approach this need differently. Unity developers commonly implement ScriptableObject event channels; Unreal Engine 5 includes the Gameplay Message Router as a built-in messaging subsystem; and Godot projects frequently rely on Autoload singletons or signal aggregation patterns. Although these approaches differ substantially in implementation details and engine integration, they reflect a shared preference for reducing direct coupling between independently evolving gameplay systems.

A second finding concerns abstraction level and its relationship to coupling. The static structure underlying all three implementations is captured in Figure 2: the IEventListener interface, implemented independently by systems such as AudioSystem and UISystem, is the architectural seam that makes substitution and independent testing possible. What the diagram also makes apparent is the deliberate absence of any direct link between subscriber systems – precisely the property that erodes when teams bypass the bus and introduce direct inter-system calls for convenience.

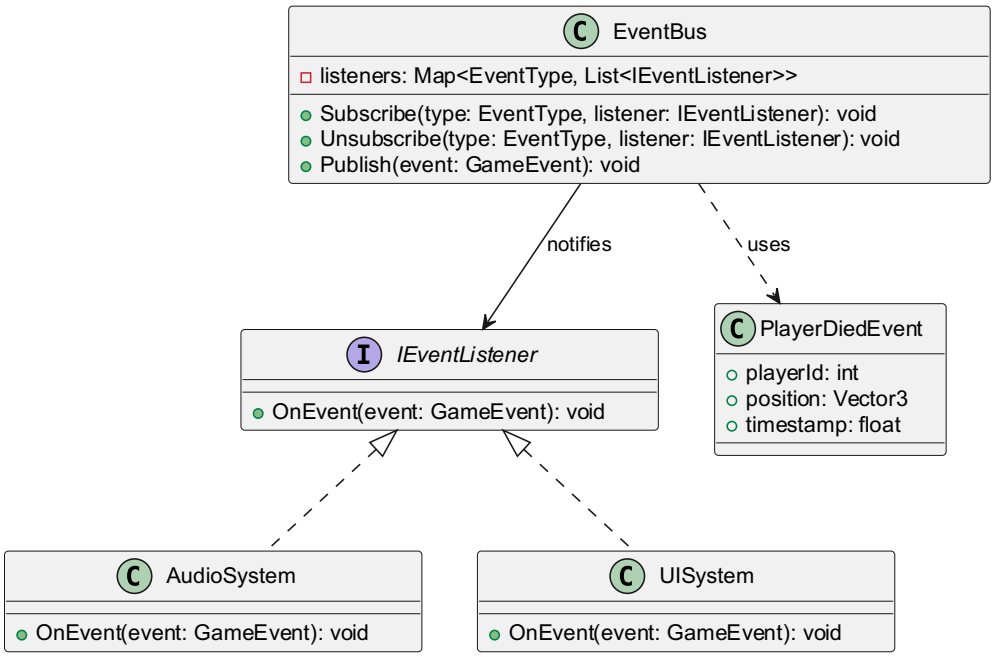


Figure 2. IEventListener interface implemented by AudioSystem and UISystem; EventBus manages subscription and dispatch

Performance characteristics further differentiate the engines beyond what the API surface alone reveals. Unity's UnityEvent abstraction introduces additional runtime overhead compared with raw C# delegates due to its serialisation and invocation mechanisms, making delegates generally preferable for high-frequency notifications. Unreal Engine's native C++ delegate systems are designed for low-overhead dispatch in performance-critical code paths, while Blueprint dispatchers trade some runtime efficiency for editor accessibility. Godot's signal system occupies a middle ground: native signals are lightweight for typical gameplay usage, while C# integrations in Godot 4 improve interoperability with statically typed workflows.

Conclusions

This paper surveyed event-driven architecture as a foundational design approach for managing complexity in game systems and compared its implementation across the three leading engines. The

analysis confirms that EDA encompasses multiple related communication patterns – including observer relationships, publish-subscribe messaging, event queues and command-oriented workflows – each suited to different gameplay and tooling requirements and carrying distinct performance and maintainability trade-offs.

Unity offers the greatest flexibility but demands the highest architectural discipline from its developers; the ScriptableObject event channel pattern has emerged as the community's de facto standard for large-scale projects. Unreal Engine provides the most integrated and visually transparent EDA tooling through Blueprint Dispatchers and, in UE5, the Gameplay Message Router. Godot 4's typed signals represent a mature, engine-native approach that prioritises simplicity and discoverability, making it the most accessible entry point for developers encountering these patterns for the first time.

References

1. Bilas S. A Data-Driven Game Object System. Game Developers Conference, 2002.
2. Hipple R. Game Architecture with Scriptable Objects. Game Developers Conference, 2017.
3. Epic Games. Unreal Engine 5 Documentation: Delegates and Events. [Online: https://docs.unrealengine.com/5.0/en-US/delegates-and-lamba-functions-in-unreal-engine], 2024.
4. Unity Technologies. Unity Scripting API: UnityEvent. [Online: https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html], 2024.
5. Godot Engine Documentation. Signals. [Online: https://docs.godotengine.org/en/stable/getting_started/step_by_step/signals.html], 2024.
6. Nystrom R. Game Programming Patterns. Genever Benning, 2014. [Online: https://gameprogrammingpatterns.com].

Information about the author

Volodymyr Osychniuk – Bachelor's Degree student of the Software Engineering Department, Faculty of Computer Science and Technologies, National University “Kyiv Aviation Institute”. *Research interests:* software engineering, game development, compliance recording software, schedule management.

E-mail: 7947174@stud.kai.edu.ua

УДК 004.8:004.42

ПРАКТИЧНЕ ВИКОРИСТАННЯ ШІ-АСИСТЕНТІВ ПРОГРАМУВАННЯ: GITHUB COPILOT VS CLAUDE CODE

Петро ПАНЧЕНКО
Асистент кафедри інженерії програмного забезпечення
Національний університет «Київський авіаційний інститут»

Проведено порівняльний аналіз двох провідних на 2026 рік ШІ-асистентів програмування – GitHub Copilot та Claude Code. Розглянуто сфери застосування обох інструментів у середовищах розробки, ключові можливості, доступні моделі штучного інтелекту та сформульовано рекомендації щодо вибору раціональної конфігурації для розробника програмного забезпечення.

Ключові слова: *ШІ-асистенти програмування, GitHub Copilot, Claude Code, агентне кодування, інтегроване середовище розробки, генеративний штучний інтелект.*

Вступ

Розвиток генеративного штучного інтелекту перетворив ШІ-асистентів для розробки програмного забезпечення зі звичайних автодоповнювачів коду на повноцінні агентні системи, здатні виконувати багатоетапні задачі: читати кодові бази, одночасно редагувати кілька файлів, виконувати консольні команди та проводити рефакторинг. Станом на 2026 рік ринок агентного кодування (agentic coding) фактично сформувався навколо двох парадигм: глибока інтеграція в інтегроване середовище розробки з широким мультимодельним вибором (підхід GitHub Copilot) та термінал-нативний агент із власним розширеним набором інструментів (підхід Anthropic Claude Code). Обидва продукти зустрілися на спільному полі – Claude Code випустив повноцінні плагіни для середовищ розробки, а GitHub Copilot надав розробникам доступ до моделей Anthropic, – тому коректна постановка питання сьогодні полягає не стільки у виборі «одне або інше», скільки у визначенні раціональної комбінації цих інструментів для конкретного типу задач.

Цілі та методи дослідження

Мета роботи – провести порівняльний аналіз GitHub Copilot та Claude Code за такими аспектами: сфери застосування обох інструментів у середовищах розробки, ключові можливості, доступні моделі штучного інтелекту, а також формулювання рекомендацій щодо вибору раціональної конфігурації. Джерелами слугували офіційна документація обох продуктів, корпоративні блоги Microsoft/GitHub та Anthropic, незалежні бенчмарки коду (SWE-bench Verified, SWE-bench Pro, Terminal-Bench 2.0) та аналітичні матеріали станом на квітень 2026 року.

Результати дослідження

Сфери застосування у середовищах розробки. Основна відмінність між інструментами полягає не в наявності окремих функцій, а в тому, де саме розробник може ними скористатися. GitHub Copilot, будучи історично надбудовою над редактором, охоплює найширший спектр середовищ: VS Code, Visual Studio, родина JetBrains (IntelliJ IDEA, PyCharm, WebStorm,

GoLand, PhpStorm), Eclipse, Xcode, Vim/Neovim, Azure Data Studio. Автодоповнення коду доступне у всіх перелічених середовищах, тоді як інтерактивний чат – у VS Code, JetBrains та Visual Studio. Claude Code розпочинався як суто термінальний інструмент і донині зберігає термінальну парадигму як основну, проте на початку 2026 року Anthropic офіційно випустив розширення для інтеграції в середовища розробки [1]. Розширення для VS Code, що вже налічує понад два мільйони користувачів, надає нативний графічний інтерфейс. Розширення для родини JetBrains залишається у статусі бета-версії [2] та реалізований не як повноцінне нативне розширення, а як обгортка над командним рядком: команда claude запускається у вбудованому терміналі редактора, після чого плагін додає інтеграцію з вікном порівняння змін і поділ діагностики. Для Visual Studio офіційного розширення Anthropic не існує.

Дослідження виявило три категорії середовищ. До першої належать редактори, де доступні обидва інструменти з нативною інтеграцією – VS Code та його надбудови. Це найбільш гнучкий випадок, оскільки розробник може комбінувати інструменти у щоденній роботі. Друга категорія – середовища, де обидва інструменти присутні, проте Claude Code має нижчий рівень інтеграції; це родина JetBrains. Тут типовою є ситуація, коли GitHub Copilot забезпечує автодоповнення та чат у звичній формі, а Claude Code викликається через термінальну сесію для складніших задач. До третьої категорії належать середовища, де доступний лише GitHub Copilot – Eclipse, Xcode, Azure Data Studio. Для цих середовищ Claude Code не має окремої інтеграції, але CLI продовжує працювати у будь-якому вбудованому терміналі, тому формально «недоступним» його назвати не можна.

Можливості та цінова політика. Обидва продукти підтримують ключові функції сучасного AI-асистента: автодоповнення коду, інтерактивний чат, агентний режим для багатоетапних задач, режим планування, автоматичне рев'ю запитів на злиття та інтеграцію зі сторонніми інструментами через протокол Model Context Protocol. Принципова відмінність полягає у глибині агентного стеку: Claude Code пропонує специфічні механізми, які у GitHub Copilot відсутні або реалізовані у спрощеному вигляді – субагенти з власним системним промптом і обмеженим набором інструментів, хуки (детерміновані обробники подій життєвого циклу, що тригеряться, наприклад, на запис файлу або виконання команди), навички у форматі описових файлів, ізоляцію субагентів через тимчасові гілки git та контекстне вікно у мільйон токенів для моделі Sonnet 4.6. GitHub Copilot, зі свого боку, має значно тісніший зв'язок із екосистемою GitHub.

Цінова політика обох продуктів передбачає кілька рівнів. GitHub Copilot пропонує безкоштовний тариф з обмеженнями, базовий індивідуальний за 10 доларів на місяць, розширений за 39 доларів, та командні плани від 19 доларів на користувача [3]. Claude Code не має безкоштовного тарифу, базова підписка коштує 20 доларів на місяць, далі йдуть тарифи Max за 100 та 200 доларів і командні плани від 100 доларів на користувача [4].

Доступні моделі та їх продуктивність. Перелік моделей у двох продуктах принципово відрізняється. Claude Code оперує виключно з моделями Anthropic – Opus 4.7, Opus 4.6, Sonnet 4.6, Haiku 4.5, – що забезпечує глибоку взаємну оптимізацію між агентом та моделями, які з ним працюють. GitHub Copilot підтримує мультипровайдерський пул, що включає лінійки OpenAI (GPT-4.1, GPT-5 mini, GPT-5.3-Codex, GPT-5.4, GPT-5.5), Anthropic (Sonnet 4.5/4.6, Opus 4.5/4.6/4.7, Haiku 4.5), Google (Gemini 2.5 Pro, Gemini 3.1 Pro) та xAI (Grok 4)[5]. Узагальнюючи результати незалежних бенчмарків коду станом на квітень 2026 року, можна виокремити такі тенденції. На стандартизованих задачах розв'язання реальних запитів GitHub (SWE-bench Verified) топові моделі обох провайдерів – GPT-5.5 від OpenAI та Opus 4.7 від Anthropic – фактично йдуть на одному рівні, з відривом близько одного відсоткового пункту,

що знаходиться в межах статистичної похибки бенчмарку [6]. На складнішому мультимовному варіанті (SWE-bench Pro), де перевіряються складніші реалістичні задачі з кількох мов програмування, Opus 4.7 утримує помітне лідерство. На термінальних задачах та задачах автоматизації командного рядка (Terminal-Bench 2.0) лідирує конфігурація OpenAI Codex CLI з GPT-5.5 [6]. Для повсякденного програмування модель Sonnet 4.6 від Anthropic виявилася найкращим компромісом між якістю та вартістю серед усіх протестованих варіантів – її показники відстають від флагманських Opus та GPT-5.5 на одиниці відсоткових пунктів, тоді як вартість токенів істотно нижча [7]. Для простих задач (документування, прості рефакторинги, перетворення форматів) економічно виправданим є використання найшвидших моделей – Haiku 4.5 та GPT-5 mini, які достатні для рутинних операцій і обходяться у багато разів дешевше.

Висновки

GitHub Copilot та Claude Code віддзеркалюють дві відмінні філософії агентного кодування, які перестали бути взаємовиключними. Перший оптимізований для щоденних коротких ітерацій із сильним автодоповненням і широкою підтримкою середовищ розробки; другий – для довгих автономних задач на великих кодових базах, де перевагу дають велике контекстне вікно, ізольовані субагенти та визначена сфера контролю. Раціональний вибір залежить від профілю використання. Для індивідуального розробника з помірним бюджетом і потребою у швидкому автодоповненні та періодичному доступі до моделей Anthropic найвигіднішим є базовий тариф GitHub Copilot, що покриває автодоповнення безлімітно та надає квоту преміум-запитів до моделей провідних провайдерів. Для інженера, який систематично виконує великі агентні задачі – рефакторинг монорепозиторіїв, міграції фреймворків, генерацію комплексних тестових наборів, дослідження незнайомих кодових баз, – оптимальною є комбінація обох продуктів: Copilot для інтерактивної роботи в середовищі розробки плюс Claude Code для делегування складних задач термінальному агенту з повним доступом до субагентів та інтеграцій Model Context Protocol. Для команд від п’яти розробників фінансово і функціонально виправдано базою брати корпоративний тариф GitHub Copilot для всіх членів команди та доповнювати його кількома ліцензіями Claude Code Max для технічних лідів і архітекторів, що працюють з найскладнішими рефакторингами.

Список використаних джерел

1. Use Claude Code in VS Code. Claude Code Docs. – URL: https://code.claude.com/docs/en/vs-code (дата звернення: 04.04.2026).

2. Claude Code [Beta] Plugin for JetBrains IDEs. JetBrains Marketplace. – URL: https://plugins.jetbrains.com/plugin/27310-claude-code-beta- (дата звернення: 04.04.2026).

3. GitHub Copilot. Plans & pricing. – URL: https://github.com/features/copilot/plans (дата звернення: 04.04.2026).

4. Plans & Pricing. Claude by Anthropic. – URL: https://claude.com/pricing (дата звернення: 04.04.2026).

5. Supported AI models in GitHub Copilot. GitHub Docs. – URL: https://docs.github.com/en/copilot/reference/ai-models/supported-models (дата звернення: 04.04.2026).

6. SWE-bench Leaderboards. – URL: https://www.swebench.com/ (дата звернення: 04.04.2026).

7. Best AI for Coding (2026): Every Model Ranked by Real Benchmarks. MorphLLM. – URL: https://www.morphllm.com/best-ai-model-for-coding (дата звернення: 04.04.2026).

Відомості про автора:

Панченко Петро Дмитрович – асистент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення.

E-mail: petro.panchenko@npp.kai.edu.ua

UDC 004.41:004.8

TEST-DRIVEN DEVELOPMENT METHOD AND TOOLS ENHANCED BY ARTIFICIAL INTELLIGENCE

Vladyslav PECHENENKO
First-year PhD student, Software Engineering Department
National University «Kyiv Aviation Institute»
Scientific supervisor – Olena GRINENKO, PhD

This paper examines the integration of Artificial Intelligence technologies into the Test-Driven Development methodology to improve test design, defect detection, and test suite maintenance. The proposed method combines the classical Red-Green-Refactor cycle with AI-based requirement analysis, automated test generation, risk-oriented test prioritization, and continuous validation while preserving the developer's central role in quality assurance.

Keywords: *Test-Driven Development (TDD), Artificial Intelligence, AI-assisted testing, automated test generation, defect prediction, software quality.*

Introduction

The growing complexity of modern software systems makes quality assurance an essential part of the development process. In this context, Test-Driven Development (TDD) remains an important engineering practice because it requires developers to write tests before production code and then improve the implementation through iterative refactoring [1]. This Red-Green-Refactor cycle supports early defect detection, clearer requirements understanding, and better software design [2].

At the same time, traditional TDD has several practical limitations. Manual test creation requires considerable time and effort, especially in systems with numerous edge cases, security restrictions, and continuously changing requirements. Developers often focus on standard scenarios, whereas boundary conditions, exceptional inputs, and high-risk interactions remain insufficiently tested. As a result, maintaining broad and effective test coverage becomes difficult in real-world projects [2].

Recent progress in Artificial Intelligence (AI), particularly in machine learning, natural language processing, and large language models, opens new possibilities for improving testing activities. AI can assist in interpreting requirements, generating candidate test cases, highlighting defect-prone components, and refining existing test suites [3, 4].

Therefore, integrating AI into TDD can make the process faster, more scalable, and more effective while preserving the developer's control over software quality.

Purpose of the research

The purpose of this study is to propose and evaluate an AI-enhanced TDD method that extends the classical TDD process with intelligent support mechanisms. The proposed method is intended to improve test generation, strengthen defect detection, and reduce the maintenance effort required for test suites in modern software projects.

Materials and methods

The proposed approach extends traditional TDD with four interconnected AI-supported stages: requirement understanding, AI-assisted test generation, defect prediction and optimization, and continuous validation. This structure combines classical testing discipline with intelligent support mechanisms that improve the efficiency and completeness of the development process. The general workflow of the proposed method is presented in Figure 1.

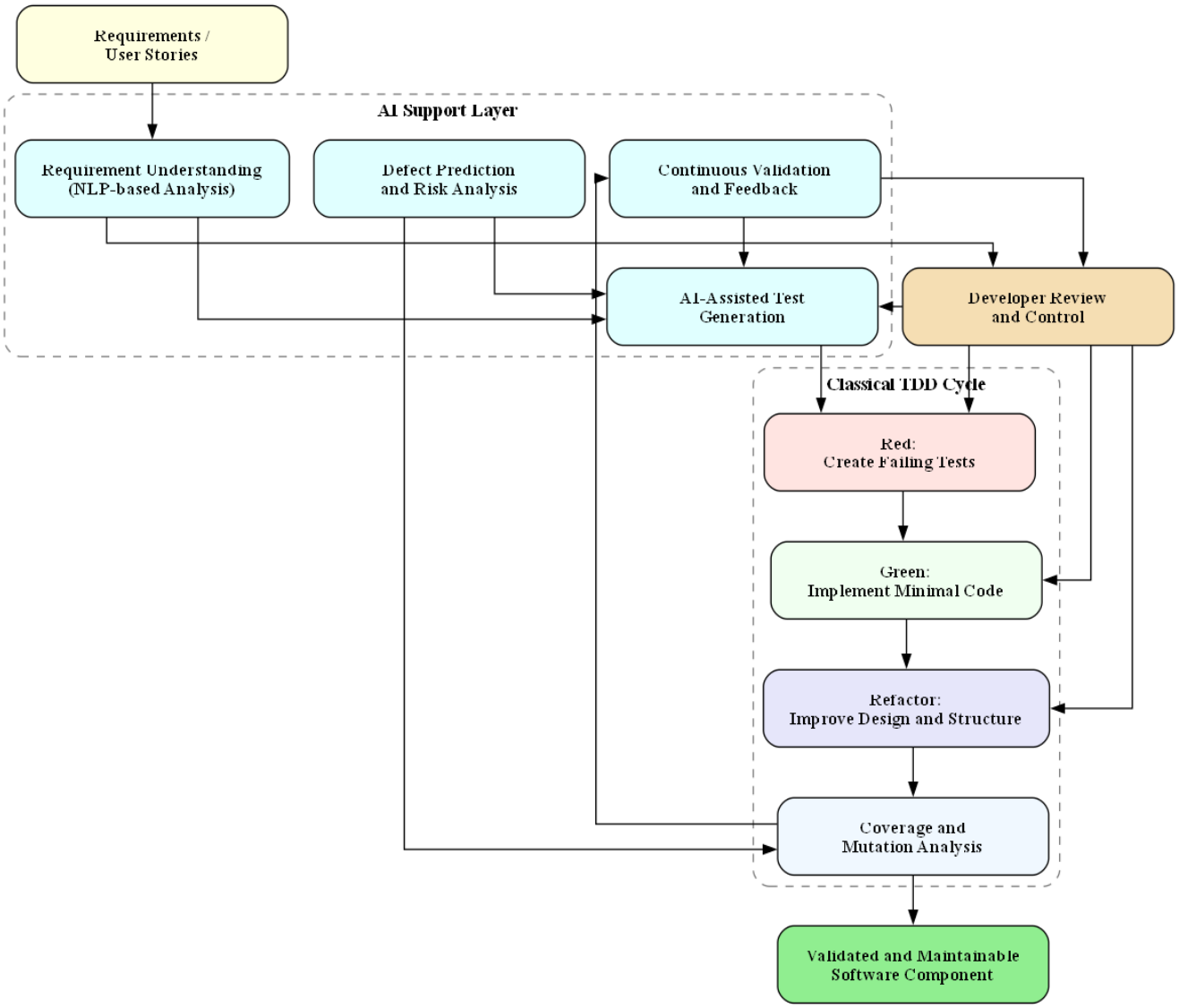


Fig. 1 – AI-enhanced Test-Driven Development workflow

At the first stage, natural language processing is used to analyze user stories and functional requirements and convert them into structured, testable scenarios. At the second stage, AI-based tools generate test skeletons and suggest unit and integration test cases, including edge and negative scenarios that are often missed during manual design [3]. At the third stage, machine learning techniques help identify high-risk components and support prioritization of testing activities. In addition, code coverage and mutation testing are used to assess the effectiveness of the generated test suite [4]. At the fourth stage, continuous feedback from test execution results is used to refine and update test cases while keeping the developer in control of validation and decision-making.

The proposed method was evaluated on an ASP.NET Core 8 password-reset module using xUnit, FluentAssertions, Moq, Coverlet, and Stryker.NET. The selected module consists of practical scenarios such as token generation, token expiration, invalid input handling, repeated reset attempts,

and rate limiting. This makes it suitable for evaluating the effectiveness of AI-assisted TDD in a realistic case.

Results and discussion

The evaluation showed that the AI-enhanced TDD method provides several practical benefits. First, AI-generated test templates reduced the implementation time for initial test drafting by approximately 40%, enabling developers to concentrate on business logic instead of repetitive boilerplate code. Second, test coverage increased by around 20%, especially in scenarios that involve invalid input, expired tokens, and request throttling. Third, the mutation score increased by about 15%, which indicates stronger fault-detection capability and better assertion quality within the test suite.

The results also demonstrated that AI support can reduce the maintenance burden of tests. When implementation details changed during refactoring, AI-assisted suggestions helped adapt outdated tests more quickly. However, the study also confirmed that AI-generated tests still require careful human review, since automatically generated assertions may be incomplete or may not fully reflect the business intent of the software component.

Overall, the findings suggest that AI is most useful in TDD as a collaborative assistant rather than as an autonomous replacement for the developer. It can accelerate routine work, identify overlooked scenarios, and improve test effectiveness, but final validation and responsibility for software quality must remain with the engineer.

Conclusions

The study confirms that integrating Artificial Intelligence into Test-Driven Development can significantly enhance the efficiency and robustness of software testing. By combining requirement analysis, automated test generation, defect prediction, and continuous feedback with the classical TDD cycle, the proposed method improves productivity, broadens test coverage, and strengthens test quality.

The experimental results demonstrate that AI-enhanced TDD is a promising direction for modern software engineering, especially in projects where rapid feedback and reliable testing are critical. Future work may focus on applying the proposed method to larger systems, extending it to other platforms, and integrating AI-assisted testing more deeply into CI/CD environments.

References

1. Beck, K. Test-Driven Development: By Example. Boston: Addison-Wesley Professional, 2003. 220 p.

2. Janzen D. S., Saiedian H. Test-driven development: Concepts, taxonomy, and future direction. Computer. 2005. Vol. 38, No. 9. P. 43–50. DOI: 10.1109/MC.2005.314.

3. Anand S., Burke E. K., Chen T. Y., Clark J., Cohen M. B., Grieskamp W., Harman M., Harrold M. J., McMinn P. An orchestrated survey of methodologies for automated software test case generation // *Journal of Systems and Software*. 2013. Vol. 86, No. 8. P. 1978–2001. DOI: 10.1016/j.jss.2013.02.061.

4. Amalfitano D., Faralli S., Hauck J. C. R., Matalonga S., Distante D. Artificial Intelligence Applied to Software Testing: A Tertiary Study. ACM Computing Surveys. 2023. Vol. 56, No. 3. Article 58. 38 p. DOI: 10.1145/3616372

Information about the author

Vladyslav Pechenenko – first-year PhD student of the Software Engineering Department, Faculty of Computer Science and Technologies, National University “Kyiv Aviation Institute”. *Research interests:* software engineering, artificial intelligence.

E-mail: 2240515@stud.kai.edu.ua

УДК 004.89:37.091.3

НАУКОВО-ДОСЛІДНИЦЬКИЙ ПРОЄКТ ЯК ОСНОВА ФОРМУВАННЯ БАЗ ЗНАНЬ

Альона ПОЛЯКОВА
Здобувачка вищої освіти другого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Володимир ТАЛАЛАЄВ, доцент, к.т.н.

У роботі обґрунтовується роль науково-дослідницького проєкту (НДП) як системоутворювального елементу формування баз знань аналітики і дизайну цифрових проєктів у магістерському циклі підготовки за ОПП «Інформаційна аналітика та управління цифровими проєктами». Запропоновано структурну модель бази знань НДП як організованого репозиторію доменних онтологій, аналітичних патернів і проєктних рішень. Визначено механізми накопичення, актуалізації та повторного використання знань у циклі магістерської підготовки.

Ключові слова: *науково-дослідницький проєкт, база знань, доменна онтологія, аналітика цифрових проєктів, магістерська підготовка, проєктне навчання, knowledge management, інформаційна аналітика.*

Вступ

Сучасні вимоги до підготовки магістрів у сфері інформаційної аналітики та управління цифровими проєктами передбачають не лише формування предметних компетентностей, а й здатність систематизувати, зберігати та повторно використовувати набуті знання. Це ставить на порядок денний питання про інструменти, які б органічно поєднували дослідницьку діяльність студентів із формуванням структурованих баз знань.

Науково-дослідницький проєкт (НДП) у магістерському циклі виступає не лише формою практичної підготовки, а й генератором знань – онтологій предметної галузі, аналітичних моделей, архітектурних рішень, що поповнюють базу знань кафедри і програми. Таким чином, НДП стає центральним механізмом формування, верифікації та трансляції знань в освітньому процесі.

Ціль роботи

Метою роботи є формалізація структури бази знань науково-дослідницького проєкту як основи системи управління знаннями магістерського циклу підготовки за ОПП «Інформаційна аналітика та управління цифровими проєктами», а також визначення компонентів, механізмів поповнення та використання такої бази знань.

Матеріали та методи

Методологічну основу дослідження складають:

- концепція Knowledge Management (KM) та онтологічної інженерії;
- підходи доменної інженерії та доменного моделювання;
- концепція Project-Based Learning (PBL) та дослідницького навчання;
- системний підхід до проєктування освітніх середовищ з підтримкою знань.

Структура бази знань НДП. База знань НДП є структурованим репозиторієм, що охоплює шість взаємопов'язаних компонентів: онтологію предметної галузі (домену цифрових проєктів), множину аналітичних патернів і шаблонів рішень, репозиторій проєктних артефактів (моделей, специфікацій, прототипів), правила та обмеження предметної галузі, метрики оцінювання якості знань і компетентностей, а також версійний реєстр змін та актуалізації знань.

Онтологічний компонент охоплює концепти предметної галузі (бізнес-аналітика, цифровий продукт, стейкхолдер, вимоги, архітектура тощо), відношення між ними та формальні обмеження й правила виведення. Саме онтологія забезпечує семантичну узгодженість знань, накопичених у різних спринтах і різними командами магістрантів.

Механізм накопичення та актуалізації знань. Поповнення бази знань відбувається циклічно в межах кожного спринту НДП: артефакти, що генеруються командою (моделі, аналітичні звіти, прототипи), проходять верифікацію та інтегруються до відповідних компонентів бази знань. Таким чином, кожен завершений спринт збагачує репозиторій новими патернами і прецедентами.

Версійний реєстр забезпечує передачу накопичених знань від однієї когорти магістрантів до наступної, формуючи безперервний освітній репозиторій, що відображає еволюцію предметної галузі. Це дозволяє уникнути «амнезії знань» між поколіннями студентів і підтримувати актуальність онтологій.

Результати та обговорення

1. *Роль НДП у формуванні баз знань.* НДП генерує три типи знань: декларативні (онтологія домену, специфікації вимог), процедурні (методики аналізу, алгоритми проєктування) та прецедентні (кейси успішних рішень, патерни помилок). Поєднання цих типів у єдиній базі знань забезпечує комплексний характер підготовки.
2. *Аналітичний компонент бази знань.* База знань НДП включає аналітичні патерни задач бізнес-аналізу, моделі даних цифрових B2B-проєктів, шаблони дашбордів і звітів, а також репозиторій методів data-driven прийняття рішень. Це дозволяє магістрантам спиратися на попередній досвід команд при вирішенні аналогічних задач.
3. *Дизайн-компонент бази знань.* Проєктна діяльність у НДП акумулює архітектурні рішення, UX/UI-патерни, специфікації інтерфейсів і прототипи, що утворюють дизайн-репозиторій, доступний для повторного використання в наступних проєктах.
4. *Інтеграція з навчальними дисциплінами.* База знань НДП є спільним ресурсом для дисциплін циклу ОПП: «Аналітика даних», «Проєктування ІС», «Управління цифровими проєктами» тощо. Це забезпечує горизонтальну інтеграцію навчального процесу навколо єдиної предметної бази.
5. *Освітній ефект.* Реалізація НДП з орієнтацією на формування бази знань підвищує практичну спрямованість навчання, забезпечує відповідність вимогам ринку та інтегрує освітню й дослідницьку діяльність студентів у єдиний процес.

Висновки

Науково-дослідницький проєкт є не лише формою практичного навчання, а й системоутворювальним механізмом формування баз знань аналітики і дизайну в магістерському циклі ОПП «Інформаційна аналітика та управління цифровими проєктами». Запропонована структура бази знань дозволяє організувати накопичені знання, забезпечити їх версійний контроль та повторне використання в наступних проєктних циклах.

Інтеграція НДП з базою знань кафедри створює підґрунтя для побудови інтелектуального освітнього середовища та CASE-платформи підтримки підготовки магістрів, що відповідає актуальним вимогам ринку цифрових продуктів і послуг.

Список використаних джерел

1. Варенко В. М. Інформаційно-аналітична діяльність: навч. посіб. / В. М. Варенко. – К.: Університет «Україна», 2014.

2. Довгий С. О., Стрижак О. Є., Палагін О. В. та ін. Комп'ютерні онтології та їх використання у навчальному процесі: монографія. – К.: Інститут обдарованої дитини НАПН України, 2013.

3. Тесля Ю. М. Управління проєктами: підручник / Ю. М. Тесля, Є. І. Харченко. – К.: НТУУ «КПІ ім. Ігоря Сікорського», 2020.

4. Козубцов І. М. Методологія наукових досліджень: конспект лекцій для магістрів спеціальності 122 «Комп'ютерні науки» / І. М. Козубцов. – Луцьк: ЛНТУ, 2022.

Відомості про автора:

Полякова Альона Віталіївна – здобувачка вищої освіти другого рівня, кафедра інженерії програмного забезпечення факультету комп’ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси*: управління проєктами, аналітика цифрових проєктів, проєктне навчання.

E-mail: 7428187@stud.kai.edu.ua

УДК 004.85:005.8

ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПРОГНОЗУВАННЯ ПЕРЕВИЩЕННЯ БЮДЖЕТУ В IT-ПРОЄКТАХ: ПОРІВНЯННЯ З МЕТОДОМ EARNED VALUE MANAGEMENT

Денис РУДНИК
Здобувач вищої освіти другого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Яна БЄЛОЗЬОРОВА, к.т.н., доц.

У роботі досліджено підходи до прогнозування перевищення бюджету в IT-проєктах із використанням класичного методу Earned Value Management (EVM) та сучасних методів штучного інтелекту. Особливу увагу приділено здатності підходів виявляти відхилення на ранніх етапах життєвого циклу проєкту. Проведено порівняльний аналіз точності прогнозування та часу виявлення ризиків. Результати демонструють доцільність використання ШІ як інструменту раннього попередження перевищення бюджету.

Ключові слова: *управління проєктами, IT-проєкти, штучний інтелект, EVM, cost overrun, прогнозування бюджету.*

Вступ

Проблема перевищення бюджету залишається однією з ключових у сфері управління IT-проєктами [1]. Навіть за наявності формалізованих підходів до планування та контролю витрат значна частина проєктів виходить за межі початково визначених бюджетних обмежень. Це зумовлено високою невизначеністю, частими змінами вимог та складною взаємодією технічних і організаційних факторів.

Класичним інструментом контролю вартості є метод Earned Value Management, який дозволяє оцінювати стан проєкту на основі співвідношення запланованих та фактичних показників [2]. Незважаючи на свою поширеність, цей підхід орієнтований переважно на фіксацію вже наявних відхилень, що обмежує його застосування для раннього прогнозування.

Розвиток методів штучного інтелекту відкриває нові можливості для аналізу динаміки проєктів [3]. Використання моделей машинного навчання дозволяє враховувати велику кількість факторів та виявляти приховані закономірності, що робить можливим більш раннє прогнозування перевищення бюджету. Незважаючи на поширеність EVM, цей метод базується на припущенні про лінійність розвитку проєкту, що в умовах сучасних Agile-процесів знижує точність прогнозів. Методи ШІ, натомість, здатні враховувати технічний борг, складність коду та когнітивні упередження менеджерів, що робить актуальним їх порівняльне дослідження.

Ціль роботи

Метою роботи є кількісне порівняння точності та своєчасності виявлення перевищення бюджету в IT-проєктах методом Earned Value Management та ансамблевими моделями машинного навчання (Random Forest, XGBoost) і рекурентними нейронними мережами (LSTM) на основі відкритих даних реальних IT-проєктів.

Матеріали та методи

У дослідженні розглянуто проблему прогнозування перевищення бюджету в IT-проєктах у контексті поєднання традиційних методів управління та сучасних підходів штучного інтелекту. Зокрема, проаналізовано застосування методу Earned Value Management як базового інструменту контролю витрат та можливості його доповнення інтелектуальними моделями прогнозування.

Експериментальна частина базується на відкритому датасеті ISBSG R1 (International Software Benchmarking Standards Group, 2016), що містить дані 225 реальних IT-проєктів, у тому числі фактичні витрати, плановий бюджет та ознаки перевищення [4]. Вибірку розділено на навчальну (80%) та тестову (20%) частини до будь-якої аугментації. Задача прогнозування сформульована як бінарна класифікація: визначення факту перевищення бюджету більш ніж на 10% від планового значення. Набір ознак включає: показники EVM на проміжних контрольних точках (CPI, SPI), розмір команди, кількість змін вимог, тип методології розробки та показники якості.

Оцінювання стану проєкту здійснювалося із використанням методу EVM, який дозволяє визначити співвідношення між запланованими та фактичними витратами на різних етапах реалізації. Аналіз проводився у часовому розрізі, що дало змогу встановити момент виникнення відхилень та оцінити їх вплив на загальну вартість проєкту.

Паралельно було застосовано підхід на основі штучного інтелекту, який передбачає використання моделей машинного навчання для прогнозування фінальних витрат. Для цього сформовано набір ознак, що відображають ключові характеристики проєкту, зокрема динаміку виконання задач, інтенсивність змін вимог, рівень завантаження команди та показники якості розробки. Зокрема, для побудови моделі було використано ансамблеві методи машинного навчання (Random Forest та XGBoost), що забезпечують високу точність на структурованих даних, а також рекурентні нейронні мережі типу LSTM для аналізу динаміки часових рядів проєктних показників. Для оцінки невизначеності та імовірнісного прогнозування фінальних витрат застосовано інтеграцію методу Монте-Карло з навченими ML-моделями.

Порівняння підходів здійснювалося шляхом паралельного аналізу результатів, отриманих на різних етапах виконання проєкту. Особлива увага приділялася моменту виявлення відхилення від бюджету та точності прогнозу, що дозволило оцінити ефективність кожного підходу в умовах динамічного середовища IT-проєкту.

Застосований підхід забезпечив комплексне дослідження проблеми, поєднавши класичні методи контролю з можливостями інтелектуального аналізу даних. Для узагальнення результатів наведено порівняльні таблиці 1 – 3.

Таблиця 1

Комплексне порівняння традиційного (EVM) та інтелектуального (AI) підходів

Критерій порівняння	Традиційний підхід (EVM/Людина)	Інтелектуальний підхід (ML/ШІ)	Рівень переваги
Точка виявлення ризику	50–60% готовності проєкту	30–40% готовності проєкту	ШІ (раніше на 20%)
Точність прогнозу	Середня (базується на минулих витратах)	Вища на 15–25% (враховує 50+ параметрів)	ШІ

Продовження Табл.1

Джерела даних	Бюджетні звіти, тайм-трекінг	Git-репозиторії, Jira, досвід розробників, складність задач	ШІ (ширше охоплення)
Чутливість до Scope Creep	Низька (реагує постфактум)	Висока (аналізує темпи появи нових вимог)	ШІ
Обробка невизначеності	Суб'єктивна оцінка менеджера	Імовірнісне моделювання (Monte Carlo + ML)	ШІ
Вартість впровадження	Низька (достатньо Excel)	Висока (потребує інфраструктури та навчання)	EVM/Людина
Інтерпретованість	Прозора (прості формули)	Складна (ефект «чорної скриньки»)	EVM/Людина

Таблиця 2

Спектр факторів, що враховуються при прогнозуванні перевищення бюджету

Категорія фактора	Що бачить Людина (EVM)	Що бачить ШІ (Deep Learning)
Персонал	Кількість годин у звіті	Кореляція між досвідом розробника та швидкістю закриття багів
Процес	Відхилення від графіка (SPI)	Аномалії в циклах Code Review та частоті комітів
Вимоги	Кількість змінених задач	Семантичний аналіз описів задач на предмет нечіткості формулювань
Якість	Кількість знайдених дефектів	Накопичення технічного боргу та його вплив на майбутню вартість

Таблиця 3

Результати порівняльного оцінювання моделей бінарної класифікації на тестовій вибірці

Модель	Accuracy	AUC-ROC	F1-score	Точка виявлення ризику	Перевага над EVM
EVM (baseline)	0,64	0,66	0,61	53% готовності	–
Random Forest	0,76	0,79	0,74	38% готовності	Acc +19%
XGBoost	0,81	0,84	0,79	35% готовності	Acc +27%
LSTM	0,79	0,82	0,77	33% готовності	Acc +23%
XGBoost + Monte Carlo	0,83	0,87	0,81	32% готовності	Acc +30%; найкращий

Результати та обговорення

Отримані результати (табл. 3) демонструють стабільну перевагу методів ШІ над EVM за всіма показниками точності на тестовій вибірці. Найкращий результат продемонстрував

ансамбль XGBoost з імовірнісним моделюванням Монте-Карло: Accuracy = 0,83 проти 0,64 у EVM (+30%), AUC-ROC = 0,87, F1-score = 0,81 проти 0,61. Ключовою перевагою є момент виявлення ризику: модель XGBoost+Monte Carlo ідентифікує потенційне перевищення бюджету вже на 32% готовності проєкту порівняно з 53% для EVM – випередження майже на 20 відсоткових пунктів.

Разом із тим використання ШІ пов'язане з необхідністю якісних навчальних даних та складністю інтерпретації – моделі типу XGBoost дають відповідь «чи буде перевищення», але не пояснюють причини без додаткового SHAP-аналізу. Це обумовлює доцільність комбінованого підходу, у якому EVM забезпечує прозорий контроль поточного стану, а ШІ виступає системою раннього попередження.

Аналіз важливості ознак (feature importance) показав, що найбільший вплив на прогноз мають: частота змін вимог (25%), накопичений технічний борг (19%) та відхилення SPI у перших трьох спринтах (17%). Саме ці фактори залишаються поза полем зору класичного EVM, що пояснює його нижчу чутливість до ранніх сигналів перевищення бюджету.

Висновки

У результаті проведеного дослідження встановлено, що метод Earned Value Management є ефективним інструментом контролю виконання бюджету, але обмеженим у здатності до раннього прогнозування (Accuracy = 0,64, AUC-ROC = 0,66, точка виявлення ризику – 53% готовності). Запропонований підхід на основі ансамблю XGBoost з моделюванням Монте-Карло досягає Accuracy = 0,83 та AUC-ROC = 0,87, виявляючи ризики вже на 32% готовності проєкту – на 21 відсотковий пункт раніше за EVM.

Практичне значення роботи полягає у можливості підвищення ефективності управління IT-проєктами шляхом використання інтелектуальних моделей прогнозування. Найбільш доцільним є комбіноване застосування EVM та ШІ, що забезпечує як контроль поточного стану, так і своєчасне виявлення потенційних відхилень.

Перспективи подальших досліджень пов'язані з використанням більш складних моделей машинного навчання та їх інтеграцією у системи управління проєктами, а також із валідацією запропонованого підходу на реальних промислових датасетах.

Список використаних джерел:

1. Kerzner H. *Project management: a systems approach to planning, scheduling, and controlling*. – Hoboken: John Wiley & Sons, 2017. – 848 p.

2. Wysocki R. K. *Effective project management: traditional, agile, extreme*. – Indianapolis: Wiley Publishing, 2019. – 720 p.

3. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. – 2016. – P. 785–794.

Відомості про автора:

Денис Миколайович Рудник – здобувач вищої освіти другого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси*: управління проєктами, IT-проєкти, прогнозування бюджету, штучний інтелект.

E-mail: 5739122@stud.kai.edu.ua

УДК 004.089

СУЧАСНІ МЕТОДИ ВИЯВЛЕННЯ РОЗБІЖНОСТЕЙ МІЖ ДОКУМЕНТАМИ КАНОНІЧНОЇ ФОРМИ

Дмитро СИРЕЙЩІКОВ,
аспірант, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Олена КОЛГАНОВА, к.т.н., доц.

Завдання виявлення розбіжностей у документах канонічної форми сьогодні набуло особливого значення через перехід від простої цифровізації до автономного управління знаннями. Якщо раніше ми порівнювали тексти, щоб знайти друкарські помилки, то сьогодні це фундамент для довіри до автоматизованих систем. Враховуючи недоліки існуючих синтаксичних чи мережевих методів, а також ризики сліпого використання великих мовних моделей, запропоновано використати гібридний підхід для диференціації документів канонічної форми.

Ключові слова: *канонічна форма, виявлення розбіжностей, машинне навчання, гібридний підхід, семантичний аналіз, обробка природної мови (NLP).*

Вступ

Сьогодні ми все частіше стикаємося з необхідністю переводити «брудні» або неструктуровані дані у канонічний вигляд (стандартизовані XML-схеми чи реляційні таблиці). Проте сама по собі канонізація – це лише половина справи. Наступним, не менш складним етапом, є автоматизоване порівняння цих документів для пошуку розбіжностей [2]. Це критично для юридичної практики, верифікації наукових праць та коректної інтеграції великих баз даних [1,4]. Канонічна форма зазвичай передбачає стандартизований вигляд документа, що дозволяє зосередитися на змістовних відмінностях, ігноруючи особливості форматування.

Ціль роботи

Метою роботи є розробка гібридного підходу для виявлення розбіжностей між документами канонічної форми, прагнучи усунути недоліки існуючих відокремлених методів, а саме, текстуальних методів (рівень символів та слів), структурних методів (рівень розмітки) та семантичних методів (рівень змісту).

Матеріали та методи

На сьогодні існує кілька основних груп методів, кожна з яких має свої «вузькі місця»:

- Синтаксичні та лексичні методи (наприклад, міри на базі TF-IDF або аналіз n-грам) здатні виявляти подібність на рівні слів, але вони часто ігнорують глибокий семантичний контекст і специфічну структуру документа [1, 3].
- Структурні та мережеві методи (наприклад, аналіз графів цитувань або зв'язків) є ефективними для виявлення зовнішніх кореляцій між документами, але повністю ігнорують їхній безпосередній текстовий зміст [1].
- Онтологічні підходи (наприклад, системи концептуальної візуалізації) дозволяють порівнювати документи на базі доменних онтологій, проте їхня ефективність жорстко

обмежена повнотою та актуальністю створеної бази знань (наприклад, для вузьких спеціалізованих сфер) [3].

Результати та обговорення

Новий поштовх цій темі дало машинне навчання. Використання графових ембедингів (Node2Vec), нейромережевих моделей для тексту (Doc2Vec, SciBERT) та великих мовних моделей (LLM) дозволяє здійснювати глибоку тематичну сегментацію та виявляти семантичні розбіжності між окремими фактами чи аргументами у документах [1]. Однак покладатися лише на алгоритми штучного інтелекту чи LLM ризиковано: такі моделі можуть генерувати зовні коректні, але фактично хибні дані («галюцинації»), продукуючи приховані помилки, які вкрай важко відловити під час автоматичного порівняння [2].

З огляду на це, найбільш перспективним напрямом є розробка гібридного підходу. Ідея полягає у поєднанні структурного аналізу з семантичними можливостями ML-моделей. Дослідження підтверджують, що саме така комбінація (текстова схожість + мережеві зв'язки) дає результати, які найбільше збігаються з оцінками реальних експертів [1]. Таке поєднання дозволяє ефективно використовувати багатовимірні властивості документів, які неможливо проаналізувати якимось одним методом [1].

Запропоновано концепцію вирівнювання по осях для категоризації типів вхідних даних та характеристики розширеної області застосування нашої системи порівняно з існуючими інструментами. Декларативна граматика складається з умов збігу, які визначають повторювані шаблони вхідних комірок, та операцій вилучення, які визначають, як збіги значень відображаються у вихідній таблиці.

Висновки

Враховуючи недоліки відокремлених синтаксичних чи мережевих методів, а також ризики сліпого використання великих мовних моделей, розробка гібридного підходу є оптимальним рішенням для виявлення розбіжностей між документами канонічної форми. Поєднання методів машинного навчання для семантичного аналізу з детермінованими алгоритмами перевірки структури дозволить підвищити точність порівняння документів, спростити обробку великих масивів даних та забезпечити прозорість (explainability) автоматизованих рішень.

Список використаних джерел:

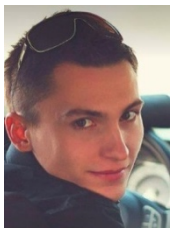
1. Bhattacharya P., Ghosh K., Pal A., Ghosh S. Methods for Computing Legal Document Similarity: A Comparative Study. (2020). URL: https://arxiv.org/abs/2004.12307 (Last accessed: 04.04.2026)

2. Xiong K., Huang C. A., Wybrow M., Wu Y. TableCanoniser: Interactive Grammar-Powered Transformation of Messy, Non-Relational Tables to Canonical Tables. CHI '25 (2025). URL: https://dl.acm.org/doi/10.1145/3706598.3714321 (Last accessed: 04.04.2026)

3. Zhang X., Chandrasegaran S., Ma K.-L. ConceptScope: Organizing and Visualizing Knowledge in Documents based on Domain Ontology. CHI '21 (2021). URL: https://arxiv.org/abs/2003.05108 (Last accessed: 04.04.2026)

4. Jain S. et al. SCIREX: A Challenge Dataset for Document-Level Information Extraction. (2020). URL: https://arxiv.org/abs/2005.00512 (Last accessed: 04.04.2026)

Відомості про автора



Сирейщіков Дмитро Олексійович – аспірант кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення.

E-mail: 8390671@stud.kai.edu.ua

UDC 004. 413.4

NETWORK-OBSERVABLE DATA EXFILTRATION FROM KUBERNETES CLUSTERS: MAPPING MITRE ATT&CK TECHNIQUES TO CLUSTER EGRESS PATHS

Ihor SKOSTARIEV
PhD student, Software Engineering Department
Olena GRINENKO
PhD, Associate Professor
National University «Kyiv Aviation Institute»

This paper maps MITRE ATT&CK exfiltration techniques (TA0010) to the concrete network egress paths through which data leaves a Kubernetes cluster, validating the mapping against documented campaigns (Scarleteel, TeamTNT, Kinsing, Dero). The coverage analysis shows that most techniques traverse standard pod egress and are network-observable, but reveals three structural blind spots invisible to network-only monitoring: exfiltration via private cloud fabric, link-local IMDS credential harvesting, and attacker-initiated retrieval through exposed surfaces. Complete detection coverage therefore requires combining network telemetry with cloud audit logs and node-level visibility.

Keywords: *Kubernetes, data exfiltration, MITRE ATT&CK, network egress, container security, threat detection.*

Introduction

Kubernetes has become the dominant orchestration platform for cloud-native workloads, and threat intelligence confirms that attackers have followed [1], [2]. The initial-access and lateral-movement phases of Kubernetes intrusions are well-catalogued through MITRE ATT&CK for Containers and vendor threat matrices. The exfiltration phase, however, has received less systematic attention from a Kubernetes-architecture standpoint.

MITRE ATT&CK tactic TA0010 [3] catalogues exfiltration techniques organized by method. Vendor matrices and covert-channel taxonomies [4] extend this with protocol-level detail. Yet none maps techniques to the network egress path through which data physically leaves a Kubernetes cluster. A network-based detector must be positioned relative to a concrete egress path, not an abstract technique label. Without an egress-path model tied to Kubernetes networking, it is impossible to specify what a monitoring tool can and cannot observe.

This paper maps TA0010 techniques to Kubernetes network paths, grounds the mapping in documented campaigns, and produces a detection coverage analysis identifying structural blind spots in network-only monitoring.

Purpose of the research

The objectives are: (1) to map MITRE ATT&CK TA0010 exfiltration techniques to network paths in Kubernetes cluster architecture; (2) to specify detection visibility per path; and (3) to validate the mapping against documented campaigns.

Materials and methods

MITRE ATT&CK TA0010 techniques [3] serve as the starting taxonomy. Each technique was assessed for Kubernetes applicability and mapped to a network-path model defined below.

Techniques inapplicable to containerized environments (T1011, T1052) are excluded. Behavioral modifiers (T1020, T1029, T1030) describe transfer patterns rather than network channels and are not mapped. The network-path model is constructed from Kubernetes networking specifications, cloud provider documentation, service-mesh architecture references, and government hardening guidance [9], [10]. The mapping is validated against publicly documented attack campaigns targeting Kubernetes clusters [2], [5], [6], [7], [8]. Academic covert-channel taxonomies [4] provide supplementary framing.

Results and discussion

The model comprises three outbound egress routes and three adjacent architectural patterns.

Path 1 – Standard pod egress. Pod traffic traverses the CNI plugin (Calico, Cilium, Flannel), is source-NAT'd at the node, and exits via the cloud provider's internet gateway. Observable at node-level and VPC flow logs.

Path 2 – Service-mesh egress gateway. A sidecar proxy intercepts outbound traffic and routes it through a dedicated egress gateway - a single choke point amenable to L7 monitoring.

Path 3 – VPC private endpoint. AWS PrivateLink, GCP Private Service Connect, and Azure Private Endpoints route pod traffic to cloud services over private fabric, bypassing the public internet. Invisible to public-egress monitors; observable only via cloud audit logging.

Pattern A – IMDS credential acquisition. The Instance Metadata Service (169.254.169.254) is a link-local credential source, not an exfiltration route. A pod reaching IMDS obtains temporary cloud IAM credentials, enabling subsequent exfiltration over Path 1 or Path 3.

Pattern B – Exposed management surface (inbound). Internet-facing API servers, kubelets, or dashboards allow attackers to connect inward and receive data in response payloads. Detection requires ingress monitoring and API audit logs.

Pattern C – In-cluster staging. Data aggregated inside the cluster before a single outbound transfer. The staging is not network-observable; only the eventual egress event is.

Mapping TA0010 techniques to K8s network paths. Table 1 summarizes the mapping. Key observations follow.

T1567 – Exfiltration Over Web Service. T1567.002 (Cloud Storage): a pod uploads data to S3/GCS/Blob. Via Path 1, visible in VPC flow logs; via Path 3, invisible to public-egress monitors - only cloud audit logs capture it. This path divergence is K8s-specific. T1567.003 (SaaS): a pod posts to Pastebin, Telegram, Discord; traverses Path 1. *Evidence*: Scarleteel exfiltrated data from S3 [5]; Kinsing uses HTTP-based C2 [2].

T1041 – Exfiltration Over C2 Channel. Malware reuses its C2 channel for exfiltration over IRC or HTTPS (Path 1). IRC is anomalous in clusters and highly detectable; HTTPS-based C2 requires domain-based policy. *Evidence*: TeamTNT operated IRC-based C2 from Kubernetes nodes [6, 7].

T1048 – Exfiltration Over Alternative Protocol. DNS exfiltration: data encoded in query labels directed to an attacker-controlled nameserver. CoreDNS forwards unresolved queries upstream without restriction, making K8s structurally permissive of this technique. Traverses Path 1 (UDP 53). *Evidence*: DNS tunneling is well-established [4]; no K8s-specific campaign has been publicly attributed.

T1537 – Transfer Data to Cloud Account. Two-phase chain: a pod harvests credentials from IMDS (Pattern A), then calls cloud APIs (Path 1 or 3). The IMDS call is invisible to VPC monitoring. *Evidence*: TeamTNT conducted mass IMDS credential harvesting [6]; Scarleteel used IMDS credentials for S3 exfiltration [5]. IMDSv2 with hop limit 1 blocks pod-level access.

T1572 – Protocol Tunneling . A pod runs Ngrok, Cloudflared, or reverse SSH, establishing an encrypted tunnel bypassing egress rules via HTTPS (Path 1). Detectable via DNS to relay domains and long-lived connections. *Evidence*: documented in cloud workload compromises [1].

Attacker-initiated pull via exposed management surface. An attacker queries data through an exposed API server or dashboard (Pattern B) - inbound, not egress. Does not map to any TA0010 technique. *Evidence*: the Dero campaign exploited exposed K8s API servers at scale [8]. This represents a gap in the TA0010 taxonomy for container environments.

Table 1

ITRE ATT&CK TA0010 Techniques Mapped to Kubernetes Network Paths

MITRE technique	K8s manifestation	Network path	Campaign evidence	Detection visibility
T1567.002 Exfiltration to Cloud Storage	Pod uploads to S3/GCS/Blob via SDK	Path 1 or Path 3	Scarleteel (2023)	Path 1: VPC flow logs; Path 3: cloud audit logs only
T1567.003 Exfiltration to SaaS	Pod POSTs to Pastebin, Telegram, Discord	Path 1	Kinsing (2023)	DNS lookups; outbound volume anomaly
T1041 Exfiltration Over C2 Channel	IRC or HTTPS C2 doubles as exfil	Path 1	TeamTNT (2021)	IRC: highly detectable; HTTPS: domain policy
T1048 Exfiltration Over Alternative Protocol	DNS-encoded data via CoreDNS	Path 1 (UDP 53)	No public K8s example	CoreDNS logs; query entropy anomaly
T1537 Transfer to Cloud Account	IMDS creds then cloud API calls	Pattern A then Path 1/3	TeamTNT (2021); Scarleteel (2023)	Node-level logs; CloudTrail
T1572 Protocol Tunneling	Ngrok / Cloudflared / reverse SSH	Path 1	No public K8s example	DNS to relay domains; long-lived sessions
No TA0010 technique	Management surface pull (inbound)	Pattern B (inbound)	Dero (2023)	Ingress monitoring; API audit logs

Detection coverage analysis. Techniques traversing Path 1 (T1567, T1041, T1048, T1572) are observable at VPC flow logs or node-level capture.

Three structural blind spots exist:

- T1567.002 via Path 3: data leaves via private cloud fabric. Requires cloud audit log integration (CloudTrail, GCP Audit Logs).
- T1537 IMDS phase (Pattern A): a link-local call invisible to VPC monitoring. Requires node-level telemetry (eBPF) or IMDSv2 enforcement.
- Management surface pull (Pattern B): an ingress problem. Requires API server audit logs. No TA0010 technique captures this pattern.

A network-only detector covers Path 1 techniques but misses exfiltration via VPC private endpoints (Path 3), IMDS credential acquisition (Pattern A), and attacker-initiated retrieval through exposed surfaces (Pattern B). Complete coverage requires cloud audit log integration, API server audit logs, and node-level visibility.

Conclusions

Mapping TA0010 techniques to Kubernetes network paths reveals that most applicable techniques traverse standard pod egress (Path 1) and are network-observable. Campaigns - Scarleteel, TeamTNT, Kinsing, Dero - confirm that attackers favor high-bandwidth, low-complexity techniques. Three structural blind spots exist: VPC private endpoints (Path 3), IMDS credential acquisition

(Pattern A), and attacker-initiated retrieval through exposed management surfaces (Pattern B). The last pattern lacks a dedicated TA0010 technique, representing a taxonomy gap for container environments. The mapping allows practitioners to specify detection coverage in concrete terms. Future work should explore automated path classification using eBPF-based pod flow tracing.

References

1. Clark M. 2023 Global Cloud Threat Report: Cloud Attacks are Lightning Fast. Sysdig, Inc. 2023. URL: https://sysdig.com/blog/2023-global-cloud-threat-report/ (accessed: 09.03.2026).

2. Microsoft. Threat Matrix for Kubernetes. URL: https://microsoft.github.io/Threat-Matrix-for-Kubernetes/ (accessed: 09.03.2026).

3. MITRE. Exfiltration, Tactic TA0010 - Enterprise. MITRE ATT&CK. URL: https://attack.mitre.org/tactics/TA0010/ (accessed: 09.03.2026).

4. Mazurczyk W., Caviglione L. Information Hiding as a Challenge for Malware Detection. IEEE Security & Privacy. 2015. Vol. 13, No. 2. P. 89-93.

5. Sysdig Threat Research Team. SCARLETEEL: Operation Leveraging Terraform, Kubernetes, and AWS for Data Theft. Sysdig Blog. 2023. URL: https://sysdig.com/blog/cloud-breach-terraform-data-theft/ (accessed: 09.03.2026).

6. Trend Micro Research. Tracking the Activities of TeamTNT: A Closer Look at a Cloud-Focused Malicious Actor Group. Trend Micro. 2021. URL: https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/teamtnt-activities-probed (accessed: 09.03.2026).

7. Chen J., Sasson A., Zelivansky A. Hildegard: New TeamTNT Cryptojacking Malware Targeting Kubernetes. Unit 42, Palo Alto Networks. 2021. URL: https://unit42.paloaltonetworks.com/hildegard-malware-teamtnt/ (accessed: 09.03.2026).

8. Mechtinger A., Berkovich S., Tikochinski G. Pause Off My Cluster: DERO Cryptojacking Takes a New Shape. Wiz Research. 2024. URL: https://www.wiz.io/blog/dero-cryptojacking-campaign-adapts-to-evade-detection (accessed: 09.03.2026).

9. NSA, CISA. Kubernetes Hardening Guidance. Ver. 1.2. National Security Agency. 2022. URL: https://media.defense.gov/2022/Aug/29/2003066362/-1/-1/0/CTR_KUBERNETES_HARDENING_GUIDANCE_1.2_20220829.PDF (accessed: 09.03.2026).

10. CNCF TAG Security. Cloud Native Security Whitepaper. Ver. 2. Cloud Native Computing Foundation. 2022. URL: https://github.com/cncf/tag-security/tree/main/community/resources/security-whitepaper/v2 (accessed: 09.03.2026).

Information about the authors:

Ihor Skostariev – postgraduate student at the Software Engineering Department of National university «Kyiv aviation institute». *Research interests:* kubernetes, eBPF, network traffic analysis.

E-mail: 9016404@stud.kai.edu.ua

Olena Grinenko – Candidate of Technical Sciences (PhD), Associate Professor, Head of the Department of Software Engineering, Faculty of Computer Information Technologies, National University «Kyiv Aviation Institute». *Research interests:* Software engineering, data analytics.

E-mail: olena.hrinenko@npp.kai.edu.ua

УДК 004.434

ЗАСТОСУВАННЯ ПРЕДМЕТНО-ОРІЄНТОВАНИХ МОВ ДЛЯ ЗАДАЧ ПЕРЕТВОРЕННЯ ТЕКСТУ

Михайло СТЕЦЮК
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Марія ВАСИЛЬЄВА, старший викладач

У роботі розглянуто проблему транслітерації та переписування текстів за допомогою наявних програмних засобів. Показано недоліки регулярних виразів і ICU Transliterator, зокрема складність підтримки правил і проблеми послідовного застосування замін. Запропоновано підхід до створення предметно-орієнтованої мови для опису правил пошуку й заміни з урахуванням контексту, множин символів і пріоритезації правил. Наведено приклад синтаксису такої мови та окреслено можливі сфери її використання.

Ключові слова: *предметно-орієнтована мова, DSL, транслітерація, перетворення тексту, регулярні вирази, формальні мови.*

Вступ

Незважаючи на велике різноманіття програмного забезпечення, що з'явилося впродовж десятиліть існування комп'ютерів, як універсальних програмних продуктів, так і спеціалізованих рішень, що базуються на власних алгоритмах, галузь продовжує розвиватися та робити інновації, адже існують певні класи задач, для яких готові інструменти залишаються недостатніми. Типовими прикладами таких задач є конфігурація інфраструктури, опис апаратного забезпечення, стилізація документів, електронні обчислення та управління даними, і вирішують їх зазвичай застосуванням предметно-орієнтованих мов (DSL) – комп'ютерних мов, що спеціалізуються на завданнях певної предметної області. Є кілька відомих прикладів таких мов, які набули популярності: CSS, SQL, Verilog, MATLAB, Maple, GraphViz, Gherkin тощо [1]. Існують DSL для перетворення XML-документів, моделей чи кодових файлів, такі як Tom, DMS, QVT [2], проте системи переписування рядків досі здебільшого є суто теоретичними концепціями та реалізуються автоматами Поста або алгоритмом Маркова [3].

Ціль роботи

Метою статті є проаналізувати поточні рішення проблеми транслітерації текстів та розкрити тему доменно-специфічних (предметно-орієнтованих) мов для розв'язання широкого кола завдань, зокрема на прикладі деякої формальної мови для переписування рядків на основі визначень правил.

Матеріали та методи

Транслітерація (передача написання) та транскрипція (передача вимови іноземних слів або власних назв) на сьогоднішній день потребують словників для заміни символів чи бібліотек, які часто працюють за наперед визначеними стандартами і не сприймають контекст, або важкими технологіями машинного навчання. За потреби мати власні правила доводиться

писати алгоритми мовами програмування загального призначення, що робить їх важкими для подальшої підтримки, внесення змін та сприйняття людиною.

Базовим рішенням є регулярні вирази, що дозволяють шукати та замінювати текст на основі лаконічних шаблонів, однак вони потребують послідовного дублювання інструкцій для кожного правила. Ще одним популярним рішенням є набір бібліотек ICU для підтримки Unicode у мовах C/C++ та Java, а точніше їхній компонент Transliterator, який підтримує багато видів відносин між символами (сприймає контекст) та дозволяє зручно описувати сукупність правил [4]. Обидва рішення, натомість, мають суттєві проблеми, спричинені послідовним застосуванням правил, що називаються feeding та bleeding order – коли правило А створює вхідні умови для виконання правила Б або усуває їх відповідно. Наприклад, за наявності правил «a → b, b → c» рядок «ab» буде помилково замінено на «cc» замість очікуваного «bc».

Таким чином, на основі описаних недоліків наявних програмних засобів та функціональних потреб у сфері транслітерації (робота з природними мовами), деяка реалізація пропонованої теоретичної предметно-орієнтованої мови за дизайном має:

- надавати простий синтаксис для опису сукупності правил пошуку і заміни;
- підтримувати задання контексту («перед», «після», «в кінці слова» тощо), класи символів («будь-яка літера», «цифра»), множини («будь-який символ із даних»), оператори АБО і НЕ, групування, квантифікацію («один або більше разів»);
- алгоритмічно вирішувати пріоритезацію і паралельність застосувань правил;
- потенційно мати інтеграцію з деякою мовою програмування або інший спосіб розширення функцій.

Враховуючи високу ефективність рушіїв регулярних виразів, доцільно базувати DSL на них та запозичити частину синтаксису, зокрема екранування, оператор “|” та квантифікатори. Проте, маючи велику кількість легкодоступних символів на клавіатурі, пропонується розширити мову різними видами дужок, операціями над множинами, коментарями, директивами обробки і шаблонами, щоб задавати імена повторюваним частинам.

Результати та обговорення

На рис. 1 показано фрагмент прикладу синтаксису спроєктованої мови, де [] позначають перевірку контексту (lookahead), <> задають множину символів, | є оператором АБО, # – коментарем, а → розділено пошук і заміну.

```
# Cyrillic Polish

a   →   a
ą[<kgtdc>|d<zż>|cz]     →   он
ą[<pb>]                 →   ом
ą[<śźć>|dź|<szc>i|dzi]  →   онь
ą[<łł>]                 →   о
```

Рис. 1 – Приклад мови перетворення тексту

Як зазначено у коментарі, цей код виконає запис польської мови кирилицею, що може бути корисним, наприклад, в освіті для навчання правилам читання польських літер у словах. Мова може знайти застосування в оформленні документів чи картографічних сервісах.

Висновки

У статті було проаналізовано проблеми у наявних інструментах переписування тексту та запропоновано підхід до розв’язання даного класу задач шляхом розробки предметно-орієнтованої мови.

Список використаних джерел

1. Domain-specific language. Wikipedia. 2004. URL: https://en.wikipedia.org/wiki/Domain-specific_language (дата звернення: 20.03.2026).

2. Transformation language. Wikipedia. 2004. URL: https://en.wikipedia.org/wiki/Transformation_language (дата звернення: 20.03.2026).

3. Semi-Thue system. Wikipedia. 2005. URL: https://en.wikipedia.org/wiki/Semi-Thue_system (дата звернення: 20.03.2026).

4. Transforms. ICU Documentation. URL: https://unicode-org.github.io/icu/userguide/transforms/general/rules.html (дата звернення: 20.03.2026).

Відомості про автора:

Стецюк Михайло Олександрович – здобувач вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп’ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси*: інженерія програмного забезпечення, DSL, формальні мови, транслітерація текстів.

E-mail: 8027350@stud.kai.edu.ua

UDC 004.4

NEURAL NETWORK LIFECYCLE AS ANALOGY TO SOFTWAREDEVELOPMENT LIFECYCLE: FINE-TUNING AS MAINTENANCE

Oleksandr SUKHOVYI
First-year PhD student, Software Engineering Department
National University «Kyiv Aviation Institute»
Scientific supervisor – Olena GRINENKO, PhD

This paper establishes a formal analogy between the neural network lifecycle and the software development lifecycle. It argues that fine-tuning, continual learning, and domain adaptation can be treated as forms of software maintenance. This perspective supports applying software engineering practices such as version control, regression testing, and change management to neural network lifecycle management.

Keywords: *neural networks, software development lifecycle, SDLC, MLOps, fine-tuning, continual learning, software maintenance.*

Introduction

The rapid proliferation of neural network (NN) based systems has brought increasing attention to their management beyond the initial training phase. Despite the technical differences between traditional software and machine learning models, both share a fundamental property: they are developed, deployed, maintained, and eventually retired. The software engineering community has long established formal frameworks for managing this process – most notably the Software Development Lifecycle (SDLC) covering requirements analysis, design, implementation, testing, deployment, maintenance, and end-of-life [1].

In contrast, the lifecycle management of neural networks, while partially addressed by emerging MLOps practices [2], has not been systematically examined through the lens of classical software engineering methodology. This gap is significant: treating neural networks as static artifacts overlooks their dynamic, evolving operational existence and prevents the transfer of decades of accumulated software engineering knowledge to the machine learning domain.

Purpose of the research

The goal of this research is to establish a formal analogy between the neural network lifecycle and the software development lifecycle, and to demonstrate that fine-tuning – along with related techniques such as continual learning and domain adaptation – constitutes a form of software maintenance applied to learned models.

Materials and methods

To construct the analogy, we review established SDLC models – including the classical waterfall model, iterative development, and agile frameworks – and compare their phases with the stages of neural network development and operation as described in recent MLOps and deep learning literature. This comparison is conducted at the level of phase objectives and artifacts rather than implementation details, to identify structural correspondences that hold across different instantiations of both lifecycles. We further examine the ISO/IEC 14764 standard classification of software maintenance types – corrective, adaptive, perfective, and preventive – and map each category to

corresponding fine-tuning strategies used in the neural network domain. The mapping is evaluated based on the intent and trigger of each activity: whether it is reactive or proactive, whether it targets a known defect or a shifted operating environment, and whether its goal is to restore, preserve, or improve the quality of the artifact.

Results and discussion

The proposed lifecycle analogy is summarized in Table 1. The requirements phase of SDLC corresponds to problem formulation and dataset specification in neural network development, where the task, performance targets, and data sources are defined. The design phase maps to architecture selection, including the choice of model family, depth, and connectivity. Implementation corresponds to initial training, where model parameters are optimized on the training dataset. Testing and validation directly parallel model evaluation, covering metrics-based assessment on held-out data including performance, fairness, and robustness. Deployment maps to model serving in production infrastructure.

Table 1

SDLC phase to neural network lifecycle analogy

SDLC phase	Neural Network Lifecycle phase
Requirements analysis	Problem formulation & dataset specification
Architecture design	Model architecture selection
Implementation	Initial training
Testing & validation	Model evaluation
Deployment	Model serving
Maintenance	Fine-tuning & continual learning
End-of-life	Model retirement

The maintenance analogy is particularly rich. ISO/IEC 14764 defines four maintenance types [3] that map directly onto fine-tuning strategies.

Corrective maintenance (fixing defects) corresponds to fine-tuning on misclassified or edge-case samples, where known model failures are addressed through targeted retraining on problematic inputs.

Adaptive maintenance (adapting to environment changes) maps to domain adaptation, where a model is fine-tuned on data from a shifted distribution to preserve performance under concept drift.

Perfective maintenance (improving performance beyond initial requirements) corresponds to performance fine-tuning, such as quantization-aware retraining or accuracy improvement on specific subpopulations.

Preventive maintenance (preemptive quality improvement) maps to continual learning with regularization, where the model is proactively updated to prevent catastrophic forgetting and performance degradation [4].

This mapping suggests that the tools and workflows developed for fine-tuning neural networks can be understood and organized using the established vocabulary and processes of software maintenance engineering. For instance, the practice of maintaining a model registry with versioned checkpoints directly mirrors source control in software development; evaluation on a held-out regression test suite before deploying a fine-tuned model mirrors pre-release testing in a continuous integration pipeline; and staged rollout strategies such as canary deployments apply equally to model updates and software releases. Conversely, mature software engineering practices that have been refined over decades – version control, change management, regression testing, traceability, and

impact analysis – can be systematically adapted and applied to neural network lifecycle management. Adopting this perspective enables organizations to treat model updates with the same rigor and discipline as software patches, reducing the risk of silent performance degradation and improving the reproducibility of the development process.

Just as software systems are never truly «finished» after initial release – requiring patches, updates, and eventual migration or retirement – neural networks deployed in production environments require continuous attention throughout their operational life. Data distributions shift as the real world evolves, causing a phenomenon known as concept drift that erodes predictive performance over time. User requirements change, demanding capabilities that were not part of the original training objective. New edge cases and adversarial inputs emerge that expose previously undetected weaknesses in the model. Hardware and infrastructure constraints evolve, requiring re-optimization of the model for new deployment targets. In all of these scenarios, the engineering response – incremental updates, targeted fixes, systematic monitoring, and structured rollback procedures – is structurally identical whether the artifact under management is a codebase or a trained model. Recognizing this equivalence is not merely an academic observation: it provides a principled basis for applying the full arsenal of software maintenance methodology to the challenge of keeping neural network systems reliable, accurate, and aligned with their operational context over time.

Conclusions

We have proposed a systematic analogy between the neural network lifecycle and the software development lifecycle, demonstrating that fine-tuning and continual learning occupy a role equivalent to software maintenance. This perspective offers several practical benefits: it allows practitioners to leverage mature software engineering methodologies for model management, provides a common language for cross-disciplinary teams, and motivates the application of formal maintenance classification to the neural network domain. Future work should investigate the development of standards and tooling for neural network lifecycle management that explicitly incorporate software engineering principles.

References:

1. Sommerville I. Software Engineering. 10th ed. Pearson, 2016. 816 p.
2. Kreuzberger D., Kühl N., Hirschl S. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. IEEE Access. 2023. Vol. 11. P. 31866–31879. URL: https://doi.org/10.1109/ACCESS.2023.3262138.
3. ISO/IEC 14764:2006. Software Engineering – Software Life Cycle Processes – Maintenance. Geneva: ISO, 2006.
4. Van De Ven G. M., Tuytelaars T., Tolias A. S. Three types of incremental learning. Nature Machine Intelligence. 2022. Vol. 4, No. 12. P. 1185–1197. URL: https://doi.org/10.1038/s42256-022-00568-3.

Information about the author

Oleksandr Sukhovyi – first-year PhD student of the Software Engineering Department, Faculty of Computer Science and Technologies, National University "Kyiv Aviation Institute". *Research interests:* Knowledge distillation, model compression, neural networks.

E-mail: 1462622@stud.kai.edu.ua

УДК 004:378:005.8:004.9 (045)

НАВЧАЛЬНО-ДОСЛІДНІ ПРОЄКТИ ЯК ФОРМА ПРОЄКТНОГО НАВЧАННЯ СТУДЕНТІВ МАГІСТЕРСЬКОГО РІВНЯ ЗА ОПП «ІНФОРМАЦІЙНА АНАЛІТИКА ТА УПРАВЛІННЯ ЦИФРОВИМИ ПРОЄКТАМИ»

Володимир ТАЛАЛАЄВ
к.т.н., доц., доцент кафедри інженерії програмного забезпечення
Національний університет «Київський авіаційний інститут»
Лариса ПОСТАВНА
асистент кафедри інженерії програмного забезпечення
Національний університет «Київський авіаційний інститут»

У роботі розглядається можливість і доцільність застосування технології проєктного навчання як ефективного засобу практико-орієнтованого навчання в системі магістерської підготовки фахівців в сфері інформаційної аналітики і дизайну цифрових проєктів. В основу такого навчання покладено розгортання і виконання студентами навчально дослідницького проєкту (НДП) створення студії аналітики і дизайну цифрових B2B проєктів. В НДП в якості базової обрана Agile-методології з використанням поширеного фреймворка Scrum.

Ключові слова: *проєктне навчання, навчально-дослідні проєкти, Agile-методологія, доменна інженерія, цифрові проєкти, інформаційно-аналітичне забезпечення цифрових проєктів, інтелектуалізація процесів аналітики.*

Вступ

Цифровізація економіки та широке впровадження технологій штучного інтелекту трансформують як зміст професійної діяльності фахівців з інженерії програмного забезпечення, так і вимоги до їх підготовки. Виникає необхідність переходу від дисциплінарно-орієнтованих моделей навчання до інтегрованих, діяльнісних моделей, здатних забезпечити формування компетентностей у контексті реальних цифрових проєктів.

У цьому контексті особливого значення набуває проєктна технологія навчання, яка передбачає організацію освітнього процесу навколо виконання комплексних практичних завдань. Її розвитком є навчально-дослідні проєкти (НДП), які поєднують освітню, дослідницьку та інженерну діяльність і виступають ядром підготовки магістрів за ОПП «Інформаційна аналітика та управління цифровими проєктами».

Ціль роботи

Метою роботи є розробка та формалізація моделі навчально-дослідного проєкту як базової форми реалізації проєктної технології навчання, а також визначення його структурних, процесних та аналітичних компонентів у системі магістерської підготовки.

Матеріали та методи

Методологічну основу дослідження складають:

- концепція Project-Based Learning (PBL);
- підходи доменної інженерії;
- концепція data-driven та AI-driven цифрових проєктів;

– системний та процесний підхід до моделювання складних систем.

Формалізоване визначення НДП

Навчально-дослідний проєкт визначимо як систему:

$$NDP = \langle D, S, B, P, A, K \rangle, \quad (1)$$

де: D – доменна модель (онтологія предметної області); S – множина спринтів; B – множина беклогів задач; P – множина процесів (життєвий цикл проєкту); A – аналітичне забезпечення; K – база знань проєкту.

Модель спринтової організації

Спринт описується як:

$$S_i = \langle G_i, T_i, R_i, M_i \rangle, \quad (2)$$

де: G_i – цілі спринту; T_i – множина задач; R_i – результати (артефакти); M_i – метрики оцінювання.

Структура беклогів

Беклог представимо як:

$$B = \bigcup_{j=1}^{n} B_j \, , B_j = \{t_{j1}, t_{j2}, \ldots, t_{jm}\}, \quad (3)$$

де кожен беклог B_j відповідає певному класу задач:

– доменний аналіз;
– бізнес-аналітика;
– інтелектуалізація (AI/ML);
– проєктування ІС.

Модель потоків забезпечення

Введемо множину потоків:

$$P = \{FStr, BAStr, IAStr, BKStr\}, \quad (4)$$

де: $FStr$ – потік інженерних рішень; $BAStr$ – аналітичний потік; $IAStr$ – потік інтелектуалізації; $BKStr$ – потік накопичення знань.

Взаємодія потоків задається відображенням:

$$\Phi: (BAStr, IAStr) \rightarrow FStr, \quad (5)$$

Модель аналітичного забезпечення

Аналітичне забезпечення визначимо як:

$$A = \langle Data, Models, Metrics \rangle. \quad (6)$$

де: $Data$ – дані; $Models$ – аналітичні та ML-моделі; $Metrics$ – система метрик.

Результати та обговорення

1. Інтеграційна роль НДП. Запропонована модель дозволяє інтегрувати: освітні дисципліни, практичну діяльність, дослідницьку роботу студентів.

2. Формування компетентностей. НДП забезпечує розвиток: системного мислення, аналітичних навичок, компетентностей у сфері AI та data-driven рішень, навичок командної роботи.

3. Орієнтація на цифрові проєкти. Модель НДП відповідає характеристикам цифрових проєктів: використання даних як основного ресурсу, інтеграція AI-рішень, автоматизація процесів.

4. Формування знань та їх акумуляція. База знань забезпечує: повторне використання рішень, формування навчальних курсів, накопичення доменних онтологій.

5. Освітній ефект. Реалізація НДП дозволяє: підвищити практичну спрямованість навчання, забезпечити відповідність вимогам ринку, інтегрувати освіту і науку.

Висновки

Навчально-дослідні проєкти є ефективною формою реалізації проєктної технології навчання в магістратурі, а формалізована модель НДП дозволяє системно описати його структуру, процеси та результати. Використання доменно-орієнтованого та Agile підходів забезпечує узгодженість навчального процесу з практикою цифрових проєктів.

Запропонований підхід створює основу для побудови інтелектуальних освітніх середовищ і CASE-платформ підтримки навчання.

Список використаних джерел

1. Thomas J. W. A review of research on project-based learning. 2020.
2. ISO/IEC 15939:2017. Systems and software engineering – Measurement process. Geneva: ISO, 2017.
3. Van Vliet H. Software engineering: principles and practice. 2021.

Відомості про авторів:

Талалаєв Володимир Опанасович – доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* аналітика і дизайн цифрових проєктів.

E-mail: VATalalaev@gmail.com

Поставна Лариса Петрівна – асистент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* бізнес-аналітика, ризикостійкість програмних проєктів.

E-mail: larysa.postavna@npp.kai.edu.ua

УДК 004.8:004.41

ПОРІВНЯЛЬНИЙ АНАЛІЗ ПЛАНУВАННЯ ІТ-ПРОЄКТУ ЛЮДИНОЮ ТА СИСТЕМАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Валерія ХАРІНА
Здобувачка вищої освіти другого рівня, кафедра ІПЗ
Валентина СКАЛОВА
Старший викладач кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

У дослідженні розглянуто застосування технологій штучного інтелекту в управлінні ІТ-проєктами, зокрема для планування проєкту, оцінювання термінів виконання завдань та ідентифікації ризиків. Проведено порівняльний аналіз результатів, отриманих людиною та системами штучного інтелекту (ChatGPT-4.0 та Gemini 3.0), включно з кількісною оцінкою відхилень у термінах, повноті декомпозиції та оцінці потенційних загроз проєкту.

Ключові слова: *управління проєктами, ІТ-проєкти, штучний інтелект, планування проєкту, оцінювання термінів, ідентифікація ризиків, WBS, ChatGPT, Gemini.*

Вступ

У сучасних умовах стрімкого розвитку інформаційних технологій та зростання складності ІТ-проєктів особливої актуальності набуває підвищення ефективності процесів управління проєктами. Планування робіт, оцінювання термінів виконання завдань та ідентифікація ризиків значною мірою залежать від досвіду менеджера проєкту.

Розвиток технологій штучного інтелекту (ШІ) відкриває нові можливості для організації процесів розробки програмного забезпечення. Системи ШІ здатні аналізувати великі обсяги даних, формувати структуру проєкту, прогнозувати терміни виконання завдань та виявляти потенційні загрози.

У зв'язку з цим виникає потреба у дослідженні ефективності використання штучного інтелекту в процесах планування ІТ-проєктів та порівнянні його результатів із традиційним підходом, що базується на управлінських рішеннях людини.

Ціль роботи

Метою цієї роботи є проведення експериментального аналізу процесу планування проєкту «Автоматизована система управління студентським містечком ДУ КАІ» за участю людини та систем штучного інтелекту (ChatGPT-4.0, Gemini 3.0) з метою кількісної оцінки точності прогнозування термінів, повноти декомпозиції робіт, ідентифікації ризиків та швидкості формування плану.

Матеріали та методи

Для дослідження було обрано кваліфікаційний проєкт «Автоматизована система управління студентським містечком ДУ КАІ». Система передбачає взаємодію кількох ролей користувачів: адміністратор університету, адміністратор гуртожитку, студенти, експлуатаційний відділ. Також у проєкті реалізовано модуль авторизації, який забезпечує розмежування прав доступу до інформації.

Функціональні можливості системи включають такі модулі:

- Авторизація: реєстрація та вхід користувачів;
- Адміністратор університету: перегляд оголошень, їх розміщення та видалення, перегляд заяв на поселення, перегляд мешканців гуртожитку, облік порушень, управління користувачами системи, вихід з системи.
- Адміністратор гуртожитку: перегляд оголошень, управління електронною чергою, реєстрація поселення студента, облік мешканців, виселення студента, створення порушень правил проживання, облік порушень, закріплення інвентарю, облік інвентарю, перегляд поданих заяв на технічне обслуговування, перегляд звітів про технічне обслуговування, вихід з системи.
- Студент: перегляд оголошень, подача заяви на поселення, запис в електронну чергу до адміністратора гуртожитку, електронна перепустка, перегляд особистої інформації, подача заяви на технічне обслуговування, вихід з системи;
- Експлуатаційний відділ: перегляд оголошень, заяви на технічне обслуговування, створення звітів про технічне обслуговування, вихід з системи.

У рамках експерименту було сформовано 3 варіанти планування проєкту:

- план розроблений людиною;
- план, сформований із використанням системи ChatGPT-4.0;
- план, сформований із використанням системи Gemini 3.0.

Для забезпечення відтворюваності дослідження всі три підходи виходили з однакового технічного завдання. Системам ШІ було надано ідентичний промпт: «Створи детальний план розробки системи управління студентським містечком з такими модулями: авторизація, адміністратор університету, адміністратор гуртожитку, студент, експлуатаційний відділ. Команда: 2–4 особи. Методологія: Waterfall. Результат: WBS із оцінкою тривалості кожного етапу у робочих днях». Якість планів оцінювалась за трьома критеріями: повнота (кількість виділених підзадач), реалістичність (відповідність оцінок фактичному досвіду) та час формування плану.

Результати планування проєкту, виконаного людиною та системами штучного інтелекту, представлені у вигляді таблиці 1. У ній наведено структуру робіт та оцінку тривалості їх виконання у робочих днях для кожного з підходів, а також відсоткове відхилення оцінок ШІ від плану людини. Позначення «−» означає, що відповідний етап або підзадача не були виділені відповідним підходом.

Таблиця 1

Порівняння плану виконання проєкту, створеного різними підходами

Завдання	Термін виконання (робочі дні)				
	Людина	ChatGPT-4.0	Відхил.	Gemini 3.0	Відхил.
1. Аналіз вимог	11	6	-45%	8	-27%
1.1 Збір вимог	5	2	-60%	3	-40%
1.2 Визначення ролей користувачів	−	1	−	−	−
1.3 Документування функціональних вимог	2	2	+0%	−	−
1.4 Документування нефункціональних вимог	2	1	-50%	−	−

Завдання	Термін виконання (робочі дні)				
	Людина	ChatGPT-4.0	Відхил.	Gemini 3.0	Відхил.
1.5 Документування бізнес-вимог	1	–	–	–	–
1.6 Документування системних вимог	1	–	–	–	–
1.7 Створення технічного завдання	–	–	–	5	–
2. Проєктування системи	16	7	-56%	14	-12%
2.1 Проєктування архітектури системи	6	5	-17%	3	-50%
2.2 Проєктування бази даних	5	2	-60%	4	-20%
2.3 Дизайн системи	5	–	–	7	+40%
3. Розробка	41	16	-61%	32	-22%
3.1 Розробка бази даних	5	1	-80%	–	–
3.2 Реалізація модуля «Авторизація»	3	3	+0%	4	+33%
3.3 Реалізація модуля «Студент»	8	3	-62%	12	+50%
3.4 Реалізація модуля «Адміністрація гуртожитку»	10	4	-60%	6	-40%
3.5 Реалізація модуля «Адміністрація університету»	10	3	-70%	6	-40%
3.6 Реалізація модуля «Експлуатаційний відділ»	5	2	-60%	4	-20%
4. Тестування	16	7	-56%	13	-19%
4.1 Інтеграційне тестування	3	2	-33%	5	+67%
4.2 Модульне тестування	13	4	-69%	5	-62%
4.3 Перевірка коректності ролей доступу	–	1	–	–	–
4.4 Тестування безпеки	–	–	–	3	–
5. Документація	15	5	-67%	8	-47%
5.1 Створення керівництва користувача	5	3	-40%	4	-20%
5.2 Створення технічної документації	10	2	-80%	4	-60%
ЗАГАЛЬНА КІЛЬКІСТЬ РОБОЧИХ ДНІВ:	**99**	**41**	**-59%**	**75**	**-24%**

Аналіз отриманих результатів свідчить про суттєві відмінності у підходах до планування проєкту. ChatGPT-4.0 занизив загальний термін виконання на 59% порівняно з планом людини, Gemini 3.0 – на 24%. Це свідчить про стабільну тенденцію систем ШІ до оптимістичної оцінки тривалості проєктних робіт.

Щодо повноти декомпозиції: людина виділила 18 підзадач, ChatGPT-4.0 – 17 (але серед них є унікальні, такі як «Визначення ролей користувачів»), Gemini 3.0 – 15. ChatGPT-4.0 пропустив документування системних та бізнес-вимог, дизайн системи, а також тестування

безпеки. Gemini 3.0 – документування функціональних і нефункціональних вимог, розробку бази даних. Обидві системи схильні недооцінювати аналітичні та документаційні етапи на користь технічної реалізації.

Додатково в рамках дослідження було проведено аналіз ризиків ІТ-проєкту, визначених людиною та системами штучного інтелекту, з метою оцінки повноти та релевантності їх ідентифікації. Системам ШІ було надано ідентичний промпт (у межах попередньо сформованого діалогу, присвяченого плануванню проєкту): «Ідентифікуй потенційні внутрішні та зовнішні ризики для розробки ІТ-проєкту системи управління студентським містечком».

Результати ідентифікації ризиків, виконаного людиною та системами штучного інтелекту, представлені у вигляді таблиці 2. Позначення «+» вказує на наявність ідентифікації відповідного ризику, тоді як «–» означає його відсутність.

Таблиця 2

Порівняння ідентифікації ризиків, визначених різними підходами

Ризик	Суб'єкт ідентифікації ризику		
	Людина	**ChatGPT-4.0**	**Gemini 3.0**
Внутрішні			
Некоректна оцінка термінів виконання завдань	+	+	-
Неефективний розподіл ресурсів проєкту	+	+	-
Перевищення бюджету розробки	+	-	-
Недостатність ресурсного забезпечення	+	+	+
Висока складність реалізації системи	+	+	+
Вразливості безпеки	+	+	+
Помилки в проєктуванні архітектури або БД	+	+	+
Недостатнє тестування	-	+	-
Зовнішні			
Зміна вимог замовника	+	+	+
Зміни в нормативних документах ЗВО	+	+	-
Нестабільне електропостачання	+	-	-
Технічні збої обладнання	+	+	-
Обмежений доступ до реальних даних	-	+	+
Форс-мажорні обставини (стихійні лиха)	+	+	-
Низька ІТ-грамотність користувачів	-	-	+
Інфраструктурні обмеження університету	-	-	+
ЗАГАЛЬНА КІЛЬКІСТЬ ВИЗНАЧЕНИХ РИЗИКІВ:	**12**	**12**	**8**

Аналіз результатів ідентифікації ризиків свідчить про наявність як спільних, так і відмінних підходів між людиною та системами штучного інтелекту. Загальна кількість визначених ризиків у людини та ChatGPT-4.0 є однаковою – 12, тоді як Gemini 3.0 ідентифікував меншу кількість ризиків – 8, що може свідчити про менш повну декомпозицію потенційних загроз проєкту.

У межах внутрішніх ризиків усі підходи однаково визначили ключові технічні аспекти. Водночас ChatGPT-4.0 виокремив ризик недостатнього тестування, тоді як Gemini 3.0 не

врахував ризики, пов’язані з плануванням (оцінка термінів, розподіл ресурсів). Людина, у свою чергу, охопила ширший спектр управлінських ризиків, включаючи бюджетні обмеження.

Щодо зовнішніх ризиків, усі підходи ідентифікували зміну вимог замовника, що є ключовим аспектом. При цьому ChatGPT-4.0 виявив додаткові ризики, пов’язані з доступом до даних, а Gemini 3.0 – низьку грамотність користувачів та інфраструктурні обмеженнями, що свідчить про його орієнтацію на експлуатаційний контекст системи. Водночас людина більш повно врахувала фактори зовнішнього середовища, зокрема технічні та енергетичні обмеження.

Отримані результати підтверджують, що системи штучного інтелекту здатні ефективно ідентифікувати типові технічні ризики, однак можуть пропускати окремі управлінські та контекстно-залежні загрози.

Аналіз витраченого часу свідчить, що системи штучного інтелекту значно перевершують людину за швидкістю виконання задачі планування. Зокрема, людина витратила на формування плану та ідентифікацію ризиків орієнтовно 180 хвилин, ChatGPT-4.0 – близько 3 хвилин, Gemini 3.0 – близько 2 хвилини. Таким чином, системи ШІ виконали задачу у 60–80 разів швидше порівняно з людиною. Водночас скорочення часу на планування за допомогою ШІ супроводжується зниженням точності термінів виконання, що потребує подальшого коригування з боку менеджера проєкту.

Результати та обговорення

У результаті проведеного дослідження було отримано три варіанти планування ІТ-проєкту, сформовані людиною та системами штучного інтелекту – ChatGPT-4.0 і Gemini 3.0. Порівняльний аналіз показав відмінності у структурі проєкту, рівні деталізації задач та оцінці термінів виконання.

План, розроблений людиною, характеризується більш реалістичною оцінкою тривалості виконання робіт. Системи штучного інтелекту формують структурований та деталізований план, виділяючи додаткові етапи, які не завжди враховуються при ручному плануванні. Проте обидві системи демонструють стабільну тенденцію до заниження загальних термінів: ChatGPT-4.0 – на 59%, Gemini 3.0 – на 24%.

Особливо показовою є різниця у підходах до документаційних етапів: ChatGPT-4.0 відвів на документацію лише 5 днів (−67% від плану людини), Gemini 3.0 – 8 днів (−47%). Це підтверджує, що системи ШІ систематично недооцінюють обсяг аналітичної та документаційної роботи.

Аналіз ідентифікації ризиків показав, що людина та ChatGPT-4.0 визначили однакову кількість ризиків – 12, тоді як Gemini 3.0 ідентифікував менше – 8. Усі підходи охопили ключові технічні ризики, проте ШІ пропускали окремі управлінські та контекстні аспекти. Результати свідчать, що системи ефективні для виявлення типових технічних загроз, але потребують доповнення людини для повного визначення ризиків.

Разом з тим, системи ШІ виконали задачу планування та ідентифікації ризиків у 60–80 разів швидше порівняно з людиною, що підтверджує їх ефективність як інструменту швидкого прототипування проєктної документації.

Висновки

У результаті проведеного дослідження здійснено кількісний порівняльний аналіз процесу планування ІТ-проєкту, виконаного людиною та системами штучного інтелекту.

Встановлено, що ChatGPT-4.0 занижує загальні терміни проєкту на 59%, Gemini 3.0 – на 24% порівняно з планом досвідченого фахівця.

Системи ШІ ефективно виконують декомпозицію проєкту на етапи та підзадачі, проте стабільно недооцінюють аналітичні та документаційні роботи. ChatGPT-4.0 має вищий рівень деталізації, Gemini 3.0 демонструє більш наближені до реальності часові оцінки.

Дослідження показало, що ChatGPT-4.0 і людина визначили однакову кількість ключових ризиків – 12, тоді як Gemini 3.0 виявив менше – 8. При цьому всі підходи успішно охопили основні технічні ризики, але системи ШІ могли не враховувати окремі управлінські та планувальні аспекти. Щодо зовнішніх ризиків, було підтверджено важливість зміни вимог замовника, тоді як інші фактори, такі як доступ до даних та інфраструктурні обмеження, системи ШІ оцінювали по-різному.

Практична рекомендація за результатами дослідження: системи ШІ доцільно використовувати на початковому етапі планування для швидкої генерації WBS-структури (економія часу у 60–80 разів), після чого менеджер проєкту має скоригувати терміни з урахуванням реального контексту. Особливу увагу слід приділяти аналітичним і документаційним етапам, які системи ШІ стабільно недооцінюють, а також перевіряти та доповнювати ідентифікацію ризиків для повної оцінки проєкту. Перспективою подальших досліджень є тестування на більшій вибірці проєктів та з використанням agile-методологій.

Список використаних джерел:

1. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide). – PMI, 2021.
2. Perkusich M. et al. Intelligent software project management: A systematic literature review // Information and Software Technology. – 2019. – Vol. 116. – P. 1–26.
3. Dwivedi Y.K. et al. Artificial Intelligence (AI): Multidisciplinary perspectives on emerging challenges // International Journal of Information Management. – 2021. – Vol. 57.
4. ChatGPT [Електронний ресурс]. – URL: https://chatgpt.com
5. Gemini [Електронний ресурс]. – URL: https://gemini.google.com

Відомості про авторів:

Валерія Валеріївна Харіна – здобувачка вищої освіти другого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси*: управління проєктами, ІТ-проєкти, штучний інтелект.

E-mail: 7373650@stud.kai.edu.ua

Скалова Валентина Анатоліївна – старший викладач кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси*: інженерія програмного забезпечення, якість програмного забезпечення, супроводження, розгортання.

E-mail: valentyna.skalova@npp.kai.edu.ua

УДК 004.42

ЧАТ-БОТ ДЛЯ АВТОМАТИЗАЦІЇ ВІДПОВІДЕЙ НА ТИПОВІ ЗАПИТАННЯ СТУДЕНТІВ КАФЕДРИ

Михайло ЦИБА
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Данііл ХОДАКОВ, доцент, к.т.н.

У статті розглядається проєктування чат-бота для автоматизації відповідей на типові запитання студентів кафедри. Проаналізовано існуючі підходи до побудови діалогових систем та обґрунтовано вибір технологічного стеку. Описано спроєктовану архітектуру рішення та очікуваний ефект від його впровадження.

Ключові слова: *чат-бот, обробка природної мови, NLP, автоматизація, Telegram Bot API, кафедра.*

Вступ

Сучасний навчальний процес у закладах вищої освіти супроводжується значним обсягом адміністративних та організаційних запитань із боку студентів. Більшість таких запитань є типовими та повторюваними: розклад занять, терміни здачі курсових і лабораторних робіт, вимоги до оформлення документів, контакти викладачів, порядок проходження практики тощо. Обробка цих запитань потребує значних часових ресурсів від співробітників кафедри, що особливо відчутно в пікові навчальні періоди – під час сесії або перед захистом кваліфікаційних робіт.

У той самий час, за даними досліджень у сфері освітніх технологій, до 75–80% звернень студентів до адміністрації кафедри стосуються стандартних питань, відповіді на які не змінюються протягом семестру [1]. Це свідчить про високий потенціал автоматизації даного процесу. Ефективним рішенням є впровадження чат-бота на основі технологій штучного інтелекту та обробки природної мови (NLP), здатного обробляти запити студентів у режимі реального часу без участі людини.

Використання месенджерів, зокрема Telegram, як платформи для взаємодії є виправданим з огляду на їх широку поширеність серед студентської аудиторії. За статистикою, понад 90% студентів регулярно користуються Telegram, що робить цей канал комунікації найбільш доступним і зручним для цільової аудиторії [2].

Ціль роботи

Метою роботи є проєктування та розробка чат-бота для автоматизації відповідей на типові запитання студентів кафедри ІПЗ, що дозволить: знизити навантаження на персонал кафедри; скоротити час очікування відповіді студентом; забезпечити цілодобову доступність інформаційного сервісу; підвищити загальну якість інформаційного обслуговування.

Для досягнення поставленої мети вирішувались наступні завдання: аналіз предметної галузі та виявлення типових категорій запитань; дослідження та порівняння існуючих технологій побудови чат-ботів; розробка архітектури системи; формування та наповнення бази знань; реалізація та тестування прототипу; оцінка ефективності розробленого рішення.

Матеріали та методи

У роботі проаналізовано три основні підходи до побудови чат-ботів: сценарійний (rule-based), на основі машинного навчання та гібридний. Детальне порівняння цих підходів наведено в розділі “Результати та обговорення”. Для вирішення поставленого завдання пропонується гібридний підхід, який поєднує переваги сценарійної логіки для структурованих запитів і NLP-моделі для обробки довільного тексту.

Порівняльний аналіз існуючих фреймворків для розробки чат-ботів наведено у Таблиці 1. За критеріями підтримки української мови, простоти інтеграції з Telegram та наявності відкритого вихідного коду пропонується використати фреймворк Rasa Open Source у поєднанні з бібліотекою python-telegram-bot.

Таблиця 1

Порівняльний аналіз фреймворків для розробки чат-ботів

Фреймворк	Підтримка мов	Інтеграція з Telegram	Відкритий код	Складність налаш-тування
Rasa Open Source	Багатомовний (укр.+)	Так	Так (Apache 2.0)	Середня
Dialogflow (Google)	Багатомовний	Так (через webhook)	Ні (хмарний)	Низька
Microsoft Bot Framework	Багатомовний	Так	Так (MIT)	Висока
ChatterBot (Python)	Обмежена	Через бібліотеку	Так (BSD)	Низька

Для реалізації системи планується використати мову програмування Python 3.11. База знань формуватиметься на основі реальних запитань студентів кафедри ІПЗ, зібраних методом анкетування та аналізу звернень через соціальні мережі кафедри. Попереднє опитування (85 респондентів) дозволило виявити 5 основних тематичних категорій запитів, на основі яких планується сформувати навчальний датасет із понад 200 прикладів.

Класифікацію намірів користувача (intent classification) планується реалізувати на основі попередньо натренованої моделі multilingual BERT з дообчанням на зібраному датасеті. Спроєктована архітектура системи наведена на Рис. 1.

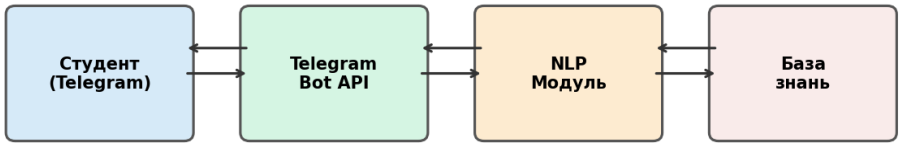


Рис. 1 – Архітектура чат-бота

Система складається з чотирьох основних модулів: інтерфейс користувача (Telegram Bot API), модуль обробки природної мови (NLP-модуль на базі BERT), база знань та менеджер діалогу. Взаємодія між компонентами відбувається через REST API. У разі, якщо рівень впевненості моделі у відповіді нижчий за встановлений поріг (0.65), запит автоматично переадресовується відповідальному співробітнику кафедри.

Результати та обговорення

На етапі проєктування системи було проведено аналіз альтернативних підходів до реалізації модуля обробки природної мови. Перший розглянутий підхід – повністю сценарійний (rule-based) чат-бот із заздалегідь визначеними деревами діалогу. Його перевагою є висока передбачуваність відповідей та простота підтримки. Проте такий підхід виявився нежиттєздатним для поставленого завдання: він не здатний обробляти довільно сформульовані запити та вимагає ручного опису кожного сценарію, що за наявності понад 200 унікальних варіантів запитань є недоцільним.

Другий підхід – використання генеративних великих мовних моделей (LLM) через зовнішній API. Він забезпечує найвищу гнучкість відповідей, однак має суттєві обмеження для даного контексту: залежність від стороннього хмарного сервісу, відсутність контролю над точністю відповідей у вузькоспеціалізованій предметній галузі та потенційні ризики розголошення конфіденційних даних кафедри. Крім того, відповіді LLM можуть містити “галюцинації” – хибну, але переконливо сформульовану інформацію, що є неприпустимим для офіційного інформаційного сервісу [3].

Обраний гібридний підхід на базі Rasa Open Source поєднує переваги обох методів: NLP-модуль класифікує наміри користувача та витягує сутності з тексту, тоді як менеджер діалогу керує сценарієм відповіді на основі заздалегідь визначених правил. Така архітектура забезпечує детерміновані та перевіряємі відповіді в межах бази знань кафедри, водночас дозволяючи студентам формулювати запити довільно, без необхідності дотримуватись строгого синтаксису. Порівняння підходів з точки зору ключових критеріїв наведено на Рис. 2.

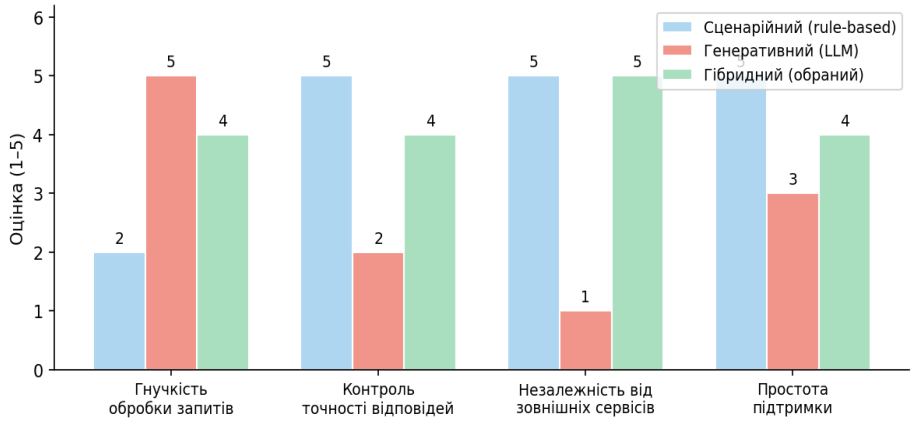


Рис. 2 – Порівняння підходів до реалізації чат-бота за ключовими критеріями

Окремо розглянуто питання обробки запитів, що виходять за межі бази знань. Від підходу повної відмови у відповіді (“Я не знаю”) було вирішено відмовитись як від такого, що знижує корисність системи. Натомість спроєктовано механізм ескалації: якщо рівень впевненості класифікатора у відповіді нижчий за встановлений поріг, запит автоматично переадресовується відповідальному співробітнику кафедри з відповідним повідомленням студенту. Це гарантує, що жоден запит не залишиться без відповіді.

Висновки

Проєктування чат-бота на базі гібридного підходу із застосуванням Rasa Open Source та Telegram Bot API дозволить ефективно вирішити проблему інформаційного перевантаження персоналу кафедри. Спроєктована система забезпечуватиме цілодобову

автоматичну обробку типових студентських запитань без участі людини, що суттєво скоротить час очікування відповіді – з кількох годин до миттєвої реакції для більшості звернень.

Обраний механізм ескалації нерозпізнаних запитів гарантуватиме, що жодне звернення студента не залишиться без відповіді. При цьому співробітники кафедри концентруватимуть свою увагу лише на нетипових, складних або нових запитаннях, які дійсно потребують участі людини. Такий підхід оптимально балансує між рівнем автоматизації та якістю обслуговування.

Запропонована архітектура є масштабованою: після введення системи в експлуатацію база знань може поповнюватись новими категоріями запитань без зміни програмного коду. Перспективами подальшого розвитку є інтеграція з університетськими інформаційними системами (електронний деканат, система розкладу) для надання актуальних даних у режимі реального часу, а також впровадження механізму активного навчання – автоматичного вдосконалення моделі на основі накопичених реальних звернень.

Список використаних джерел

1. Adamopoulou E., Moussiades L. An Overview of Chatbot Technology. IFIP Advances in Information and Communication Technology. 2020. Vol. 584. P. 373–383.

2. Telegram Messenger. Telegram Statistics 2024. [Електронний ресурс]. – Режим доступу: https://telegram.org/blog.

3. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805. 2019.

4. Rasa Technologies. Rasa Open Source Documentation. [Електронний ресурс]. – Режим доступу: https://rasa.com/docs/.

5. python-telegram-bot. Official Documentation. [Електронний ресурс]. – Режим доступу: https://docs.python-telegram-bot.org.

Відомості про авторів:

Циба Михайло Євгенович – здобувач вищої освіти другого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси*: розробка програмного забезпечення, штучний інтелект, чат-боти.

E-mail: 8080942@stud.kai.edu.ua

УДК 004.932:621.9.06

ЗАСТОСУВАННЯ АЛГОРИТМІВ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ ВИЯВЛЕННЯ ТА АВТОМАТИЧНОГО КОРИГУВАННЯ ДЕФЕКТІВ ФРЕЗЕРУВАННЯ НА CNC ВЕРСТАТІ З КОНВЕЄРОМ

Костянтин ЦЮЦЮРА
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Дмитро ГОЛОЛОБОВ, к.ф.-м.н., доцент

У роботі розглядається теоретична архітектура програмно-апаратного комплексу для безперервного моніторингу якості фрезерування на CNC верстатах, інтегрованих у конвеєрні лінії. Запропоновано метод автоматичного виявлення та коригування дефектів виробництва.

Ключові слова: *автоматизація виробництва, CNC-обробка, виявлення дефектів, комп'ютерний зір.*

Вступ

Сучасні виробничі лінії часто використовують системи для фрезерування деталей з CNC (ЧПК – числово-програмним керуванням), які інтегровані з конвеєрними системами, що забезпечує безперервний цикл виготовлення. Проте, процес фрезерування піддається впливу неоднорідності матеріалу, зносу інструменту, вібрацій та інших змінних, що призводить до виникнення дефектів обробки. Ранні конвеєрні лінії покладались на візуальний контроль з боку оператора, але, з підвищенням швидкостей конвеєра, такий підхід не є ефективним. Впровадження систем комп’ютерного зору дозволяє автоматизувати процес виявлення дефектів, проте більшість рішень лише сигналізують про помилку. На сьогодення актуальним є створення замкнутих систем, які здатні не лише виявляти дефекти, а й автоматично коригувати команди управління для усунення дефекту на наступних деталях.

Ціль роботи

Мета тези – проаналізувати та запропонувати теоретичну модель та архітектуру замкнутої системи, здатної виявляти дефекти фрезерування та генерувати скориговані інструкції для наступних деталей.

Матеріали та методи

Для розуміння переваг, які надаватиме запропонована система, необхідно детальніше розглянути підходи, які використовуються на існуючих CNC конвеєрних верстатах. Умовно їх можна поділити на дві групи: без автоматизованого контролю якості та з автоматизованим контролем якості деталі. Перша група включає в себе верстати, які не виконують контролю якості готової деталі та повністю покладаються в цьому на оператора. При цьому, якщо розвинулась така несправність верстату, яка не спричиняє спрацювання базових систем безпеки (тобто знос фрези, неточність фіксуючого механізму, люфт в направляючих чи передачах, тощо), верстат продовжуватиме виготовляти дефектні деталі, поки оператор не помітить проблему та не зупинить конвеєр для ремонту. Друга група верстатів має складніший підхід: вони мають вбудовані системи контролю якості (часто реалізовані за допомогою

комп'ютерного зору), які здатні виявляти дефектні деталі. При виявленні дефекту реакція відрізняється від верстата до верстата, від простої автоматичної зупинки конвеєра до скидання дефектних деталей у окремі ємності від придатних та продовження роботи (Рис. 1).

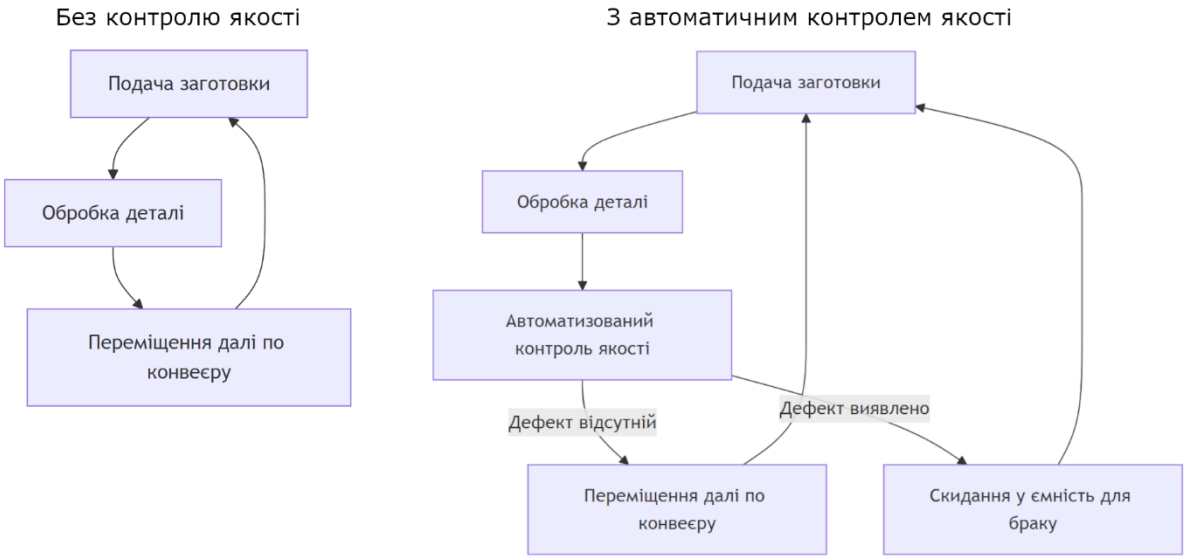


Рис. 1 – Діаграми роботи існуючих верстатів

З даного опису можна виділити спільну рису обох груп верстатів – якщо брак не є одиничним, яким би дрібним відхилення від еталону не було, його виправлення потребуватиме зупинки конвеєра та ручного налаштування чи ремонту.

Для вирішення даної проблеми пропонується подальше розвинення верстатів з автоматизованим контролем якості деталі та додавання до них системи зворотного зв'язку, яка зможе, у певному діапазоні, впливати на програму керування та автоматично виправляти деякі дефекти (наприклад, компенсувати люфт, зміщення по осі) без втручання оператора чи інженерів з ремонту.

Пропонована система складається з модуля контролю та обчислювального модуля:

- Модуль контролю – це оптична станція спостереження, яка стоїть одразу за станцією обробки та фіксує результат. Це може бути камера високої роздільної здатності з низькою витримкою, навколо якої встановлено достатнє освітлення. В залежності від потреб, камера може бути кольоровою або чорно-білою, може мати поляризаційний фільтр для мінімізації відблисків і т. д.
- Обчислювальний модуль – це програмне забезпечення, яке відповідає за інтерпретацію знятого зображення, визначення відхилення деталі від еталонної та, за потреби, генерацію оновлених параметрів для програми керування CNC верстата.

Важливо відзначити, що для надійної компенсації дефектів обробки вони повинні бути консистентними протягом обробки кількох деталей. Система, при виявленні дефекту на одній-двох деталях підряд, може вносити дуже слабкі коригування у програму і збільшувати їх лише після того, як буде підтверджено, що вплив коригувань на обробку є очікуваним. Такий підхід необхідний для забезпечення асимптотичної стійкості системи, не дозволяючи системі перейти у нестійкий стан через накопичення мікродефектів.

Розглянемо роботу такої системи на базовому прикладі: верстат з CNC використовується для фрезерування кола з діаметром 10 мм по центру квадратної заготовки з розмірами 20 на 20 мм. Основними проблемами при такій процедурі є можливе зміщення

фрезерованого кола від центру заготовки та нерівномірність самого кола (Рис. 2). Причинами даних дефектів можуть бути неврахований люфт по осі, невчасне спрацювання датчика нульового положення осі, тощо.

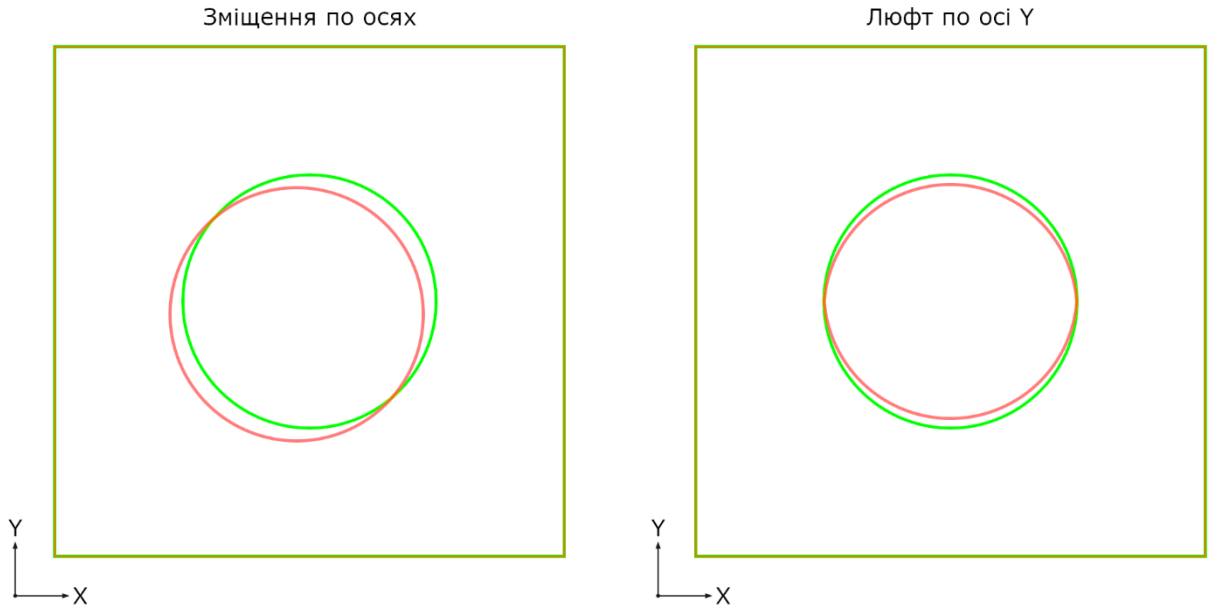


Рис. 2 – Можливі дефекти у порівнянні з еталоном

Алгоритм обробки дефекту складається з таких кроків:

1. Отримання зображення обробленої деталі.
2. Застосування алгоритму виділення контурів для знаходження фізичних меж обробки та самої деталі.
3. Апроксимація знайденого контуру та його накладання на еталонну 2D-модель обробленої деталі.
4. Визначення значення відхилення та порівняння його з порогом толерантності. Математично критерій наявності дефекту описується за формулою 1.

$$D = \max|P_{detected} - P_{ideal}| > \varepsilon \quad (1)$$

де: $P_{detected}$ – координати точок знайденого контуру, P_{ideal} – координати еталонної моделі, ε – максимально допустиме відхилення.

5. Якщо відхилення перевищує допустиме, то фіксується дефект. Дефектна деталь скидається у окрему ємність для браку.
6. При фіксації дефекту обчислювальний модуль визначає нові параметри для програми керування та дає команду на обробку наступної деталі.
7. Якщо наступна деталь оброблена з допустимим відхиленням, дефект вважається усунутим. Якщо наступна деталь оброблена з меншим відхиленням, але все ще недопустимим, то знову виконується генерація нових параметрів. Якщо наступна деталь оброблена з іншим, неочікуваним відхиленням, то конвеєр зупиняється, бо це ймовірніше за все вказує на критичну несправність у механіці верстата.

Результати та обговорення

Запропонована архітектура дозволяє перетворити CNC-верстат із розімкненим контуром керування на інтелектуальну систему зі зворотним зв’язком (Рис. 3).

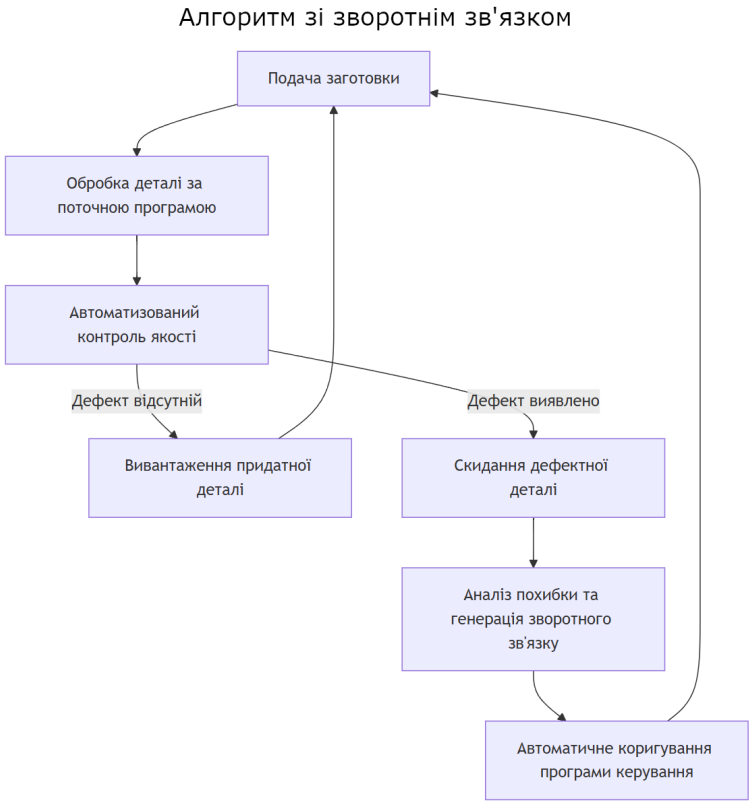


Рис. 3 – Діаграма роботи зі зворотним зв’язком

Головною перевагою такої системи є здатність до саморегулювання: вона автоматично компенсує поступовий знос механіки верстата, завдяки чому значно продовжується автономна робота конвеєра без втручання оператора чи інженерів з ремонту.

Проте, дана система має певні обмеження:

- імплементація є складною і сильно залежить від конкретного верстата,
- обробка зображень вимагає потужних локальних обчислювальних вузлів, здатних виконувати всі операції у реальному часі,
- висока чутливість до калібрування модуля контролю.

Можливості подальшого розвитку системи після впровадження:

- перехід від звичайних методів контурного аналізу до використання нейронних мереж, що дозволить системі фіксувати не лише геометричні відхилення, а й більш складні візуальні дефекти (тріщини, відколи, перегрів заготовки, структурні неоднорідності, тощо).
- прогнозування оптимального часу для обслуговування верстата ще до того, як він почне генерувати брак, завдяки збору і аналізу даних про необхідні коригування з часом.

Висновки

Використання алгоритмів комп’ютерного зору для створення замкнених систем керування CNC-верстатами є перспективним напрямком автоматизації виробництва. Розроблена концептуальна модель демонструє, як аналіз цифрових зображень дозволяє не

лише виявляти факт браку, а й автоматично адаптувати параметри обробки для його уникнення у майбутньому.

Список використаних джерел

1. Аніщенко М. В. Системи числового програмного керування : підручник / М. В. Аніщенко ; Нац. техн. ун-т "Харків. політехн. ін-т". – Харків : Факт, 2024. – 416 с.

2. Open Source Computer Vision Library (OpenCV). URL: https://opencv.org (дата звернення: 10.04.2026).

Відомості про автора:

Цюцюра Костянтин Сергійович – здобувач вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення, автоматизація виробництва.

E-mail: 6187727@stud.kai.edu.ua

УДК 004.8:004.41

AI-АСИСТОВАНА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Ілона ШЕВЧЕНКО
к.т.н., доцент, доцент кафедри ІПЗ
Національний аерокосмічний університет «ХАІ»

Проаналізовано сучасний стан AI-асистованої розробки програмного забезпечення та розглянуто ключові інструменти на основі великих мовних моделей. Оцінено їх вплив на продуктивність розробників, якість коду, а також визначено основні переваги, ризики та перспективи подальшого розвитку цієї галузі.

Ключові слова: *AI-асистована розробка, великі мовні моделі, інструменти розробки.*

Вступ

Стрімкий розвиток штучного інтелекту (Artificial Intelligence, AI) протягом останніх років докорінно змінив підходи до розробки програмного забезпечення. Інструменти на основі великих мовних моделей (GitHub Copilot, Cursor, Amazon CodeWhisperer тощо) стали невід'ємною частиною робочого процесу сучасного розробника. Вони здатні генерувати код, виправляти помилки, пояснювати складні конструкції та автоматизувати рутинні завдання.

Особливого значення набуває AI-асистована розробка в умовах зростаючого попиту на програмні продукти та дефіциту кваліфікованих фахівців. Інтеграція AI у процес розробки відкриває нові можливості для підвищення ефективності команд і зменшення часу виведення IT-продукту на ринок. Зазначимо, що це породжує нові виклики, пов'язані з якістю коду, безпекою та питаннями відповідальності.

Метою роботи є аналіз сучасного стану AI-асистованої розробки програмного забезпечення, дослідження ключових інструментів і технологій, а також оцінка їх впливу на продуктивність розробників та якість кінцевого продукту. Доповідь спрямована на формування цілісного уявлення про переваги та обмеження використання AI при розробці, а також на визначення перспективних напрямів розвитку цієї галузі.

Основний матеріал

Сучасні AI-інструменти для розробки можна поділити на кілька категорій:

1) автодоповнення коду (Copilot, Tabnine), які пропонують рядки або блоки коду в режимі реального часу;

2) чат-асистенти (ChatGPT, Claude), призначені для пояснення коду, генерації алгоритмів і вирішення задач у форматі діалогу;

3) інтегровані середовища розробки з вбудованим AI (Cursor, Replit), що забезпечують наскрізну підтримку від написання до тестування коду.

Дослідження компанії GitHub показали, що розробники, які використовують Copilot, виконують завдання на 55% швидше порівняно з тими, хто не користується AI-підтримкою [1]. Особливо суттєвим є прискорення при написанні шаблонного коду, написанні юніт-тестів та роботі з незнайомими бібліотеками. Проте варто зазначити, що переваги є найбільш відчутними для розробників середнього рівня, тоді як досвідчені фахівці отримують менший відносний приріст продуктивності.

Важливим аспектом є також зниження когнітивного навантаження: розробники

витрачають менше зусиль на пошук документації та приклади, зосереджуючись на логіці та архітектурі рішення, що сприяє зменшенню втоми та підвищенню задоволеності від роботи.

Результати та обговорення

Крім перелічених переваг, AI-асистована розробка несе в собі певні ризики:

1) згенерований код може містити вразливості безпеки або використовувати застарілі патерни; дослідження Стенфордського університету показало, що розробники, які покладаються на AI-підказки, частіше допускають помилки безпеки [2];

2) надмірна залежність від AI може гальмувати розвиток фундаментальних навичок у початківців [3];

3) існують питання щодо ліцензування коду, згенерованого на основі навчальних даних з відкритих репозиторіїв.

Окремою проблемою є галюцинації моделей – генерація коду з неіснуючими функціями або бібліотеками, що вимагає від розробника критичного ставлення до кожної пропозиції AI та обов'язкової перевірки результатів.

Висновки

AI-асистована розробка програмного забезпечення є однією з найбільш трансформаційних тенденцій сучасної IT-індустрії. Вона суттєво підвищує продуктивність розробників, прискорює цикл розробки та знижує поріг входу для початківців. Разом із тим ефективне використання AI-інструментів потребує критичного мислення, розуміння їх обмежень та збереження відповідальності розробника за якість і безпеку коду.

Перспективи галузі пов'язані з подальшою інтеграцією AI у всі етапи життєвого циклу програмного забезпечення – від збору і аналізу вимог до супроводу та рефакторингу. Впровадження AI-інструментів у процеси розробки стає важливим чинником формування конкурентоспроможності IT-компаній. Однак індустрії необхідно виробити стандарти та найкращі практики щодо безпечного та відповідального використання штучного інтелекту у розробці програмного забезпечення.

Список використаних джерел

1. Peng S., Kalliamvakou E., Cihon P., Demirer M. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv. 2023. URL: https://arxiv.org/abs/2302.06590.

2. Perry N., Srivastava M., Kumar D., Boneh D. Do Users Write More Insecure Code with AI Assistants? Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23). New York: ACM, 2023. P. 2785–2799. URL: https://arxiv.org/abs/ 2211.03622.

3. Shen J. H., Tamkin A. How AI Assistance Impacts the Formation of Coding Skills. arXiv. 2026. URL: https://arxiv.org/ abs/2601.20245.

Відомості про автора:

Шевченко Ілона Володимирівна – доцент кафедри інженерії програмного забезпечення Національного аерокосмічного університету «ХАІ». *Наукові інтереси*: інженерія програмного забезпечення, штучний інтелект, великі мовні моделі.

E-mail: ilona.shevchenko@gmail.com

УДК 004.65

ВАЛІДАЦІЯ ДАНИХ У MONGODB: ВІД ГНУЧКОСТІ ДО ОБМЕЖЕНЬ

Тимур ШЕВЧЕНКО
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Національний аерокосмічний університет «ХАІ»
Науковий керівник – Ілона ШЕВЧЕНКО, к.т.н., доцент

Досліджено механізми валідації даних у MongoDB на основі JSON Schema, проаналізовано рівні суворості та режими валідації, а також їх вплив на якість і узгодженість даних.

Ключові слова: *MongoDB, валідація даних, JSON Schema.*

Вступ

MongoDB – одна з найпопулярніших документоорієнтованих СУБД, яка відома своєю гнучкою схемою зберігання даних. Вона дозволяє зберігати документи різної структури без попереднього визначення схеми, що спрощує розробку та масштабування. Однак із ростом проєктів виникає потреба контролювати узгодженість даних. Для цього MongoDB має вбудований механізм валідації [1], який поєднує гнучкість документоорієнтованого підходу з надійністю структурованих даних.

Мета роботи – дослідити механізми валідації даних у MongoDB, проаналізувати їх можливості та обмеження, а також визначити практичні підходи до їх застосування при проєктуванні надійних схем даних.

Основний матеріал

MongoDB зберігає дані у документах у форматі BSON (Binary JSON). Кожен документ у колекції є самостійною одиницею і може мати унікальний набір полів. Ця особливість називається «schemaless» підходом – колекція не має фіксованої схеми. Відсутність обов’язкової схеми дає змогу швидко змінювати структуру документів, зберігати різнорідні об’єкти в одній колекції та спрощує ітеративну розробку на ранніх етапах. Однак у виробничих системах це може призводити до потрапляння некоректних або неповних даних, ускладнення підтримки та дебагінгу, а також непередбачуваної поведінки застосунків при зчитуванні документів з різними полями.

Для вирішення цих проблем MongoDB підтримує Schema Validation на основі JSON Schema [2]. Валідацію даних налаштовують за допомогою оператора $jsonSchema при створенні або зміні колекції. Приклад оголошення схеми: поле *name* типу string є обов’язковим і повинно містити щонайменше 3 символи; поле *age* типу int має бути в діапазоні від 0 до 120; поле *email* є обов’язковим рядком і повинно відповідати формату електронної пошти; обов’язкове поле *role* може набувати лише визначених значень (user, admin, moderator); поле *isActive* має логічний тип, а поле *skills* є масивом рядків з мінімум одним елементом; вкладений об’єкт *address* має містити поля місто та країну.

MongoDB підтримує два параметра для налаштування валідації – validationLevel (що перевіряти?) і validationAction (що робити, якщо документ невалідний?).

У параметра validationLevel (визначає – до яких документів застосовується перевірка) є два значення – *“moderate”* перевіряє лише нові документи або ті документи, що вже

відповідають правилам валідації, залишаючи наявні некоректні записи без змін; і значення *"strict"* (за замовчуванням) – застосовує перевірку до всіх операцій запису.

У параметра *validationAction* (визначає реакцію системи) є два значення – *"error"* блокує запис невалідного документа, і *"warn"* лише фіксує попередження в журналі, не перешкоджаючи запису.

Важливою особливістю є підтримка оператора *$jsonSchema* разом з іншими операторами запитів (*$and, $or, $nor* тощо), що дає можливість будувати складні умовні правила валідації, які залежать від значень конкретних полів документа, реалізуючи так звану *умовну (контекстну) валідацію.*

Результати та обговорення

Проведений аналіз показує, що механізм валідації в MongoDB є потужним інструментом, який наближає документоорієнтовану СКБД до реляційних систем за рівнем контролю якості даних, зберігаючи при цьому притаманну їй гнучкість. Валідація на рівні бази даних є надійнішою, ніж перевірка виключно на рівні застосунку, оскільки захищає від некоректних записів незалежно від клієнта.

Разом з тим слід відзначити певні обмеження: 1) JSON Schema в MongoDB не підтримує посилальну цілісність між колекціями – на відміну від зовнішніх ключів у реляційних СКБД; 2) надмірно жорстка схема знижує переваги гнучкого підходу і може ускладнити еволюцію моделі даних; 3) режим "warn" може маскувати проблеми з якістю даних у виробничому середовищі, тому рекомендовано його застосовувати лише на етапі міграції.

Висновки

Валідація даних у MongoDB є ефективним механізмом забезпечення якості та узгодженості даних. Застосування оператора $jsonSchema дозволяє декларативно описати вимоги до структури документів, а гнучке налаштування рівнів і режимів валідації забезпечує баланс між строгістю контролю і збереженням переваг MongoDB.

Таким чином, сучасні проєкти на MongoDB не зобов'язані обирати між гнучкістю і надійністю – грамотне застосування механізмів валідації дозволяє поєднати обидва підходи. Перспективним напрямком подальших досліджень є порівняльний аналіз продуктивності колекцій з валідацією та без неї на великих обсягах даних.

Список використаних джерел

1. MongoDB Documentation: Schema Validation. URL: https://www.mongodb.com/docs/manual/core/schema-validation/ (дата звернення: 05.04.2026).

2. JSON Schema Specification. URL: https://json-schema.org/specification (дата звернення: 05.04.2026).

Відомості про автора:

Шевченко Тимур Антонович – здобувач вищої освіти першого рівня, кафедра інженерії програмного забезпечення Національного аерокосмічного університету «ХАІ». *Наукові інтереси*: інженерія програмного забезпечення, NoSQL бази даних.

E-mail: t.a.shevchenko@student.khai.edu

УДК 004.42:336

СУЧАСНІ АРХІТЕКТУРИ МОБІЛЬНИХ ЗАСТОСУНКІВ ВІДКРИТИХ БАНКІВСЬКИХ СИСТЕМ

Тарас ШПАРУК
Здобувач вищої освіти першого рівня, кафедра ІПЗ
Наталія ШИБИЦЬКА
к.т.н., доц., доцент кафедри ІПЗ
Національний університет «Київський авіаційний інститут»

Запропоновано підхід до побудови гібридної архітектури мобільного застосунку для відкритих банківських систем, що поєднує інтеграцію з API та імпорт виписок (PDF, CSV, Excel). Реалізовано окремий шар обробки даних для уніфікації інформації в єдину модель транзакцій. Архітектура базується на MVVM і Clean Architecture, забезпечує масштабованість, реактивне оновлення інтерфейсу та можливість розширення без зміни основної логіки.

Ключові слова: *відкритий банкінг, мобільні застосунки, гібридна архітектура мобільного застосунку, API-інтеграція, імпорт виписок, обробка транзакцій, MVVM, Clean Architecture.*

Вступ

Відкритий банкінг передбачає безпечний обмін фінансовими даними між банком та сторонніми додатками за згодою користувача. У більшості країн це реалізовано через стандартизовані API (наприклад, PSD2 у ЄС або Open Banking у Великій Британії) з використанням протоколів авторизації OAuth2 та підтвердження особи. Такий підхід дозволяє додаткам отримувати доступ до балансів і історії транзакцій користувача в реальному часі. Однак на практиці не всі банки підтримують API або сумісні зі стандартами, тому часто використовують імпорт виписок (PDF/CSV/XLSX) як альтернативу. У результаті основною проблемою є відсутність універсального підходу до отримання фінансових даних із різних джерел. Це обмежує можливість побудови єдиного мобільного рішення, яке стабільно працювало б як із банками, що підтримують API, так і з банками, де доступ до даних можливий лише через експортовані виписки.

Ціль роботи

Метою є детальний аналіз та опис сучасних архітектур мобільних застосунків у контексті відкритого банкінгу та розробка концепції гібридної архітектури мобільного застосунку, здатної працювати як із банківськими API, так і з експортованими фінансовими виписками, забезпечуючи цілісність, зручність і ширше охоплення фінансових даних користувача.

Матеріали та методи

Для дослідження використовувалися офіційні технічні рекомендації Android та iOS щодо побудови мобільних застосунків, зокрема архітектурні підходи та бібліотеки ViewModel і LiveData, специфікації протоколу OAuth2 та розширення PKCE, стандарти форматів файлів CSV, XLSX і PDF, стандарт OWASP MASVS щодо безпеки мобільних додатків, а також практичні статті й дослідження у сфері Open Banking. На основі опрацьованих матеріалів

сформовано концептуальне бачення архітектури застосунку та узагальнено практичні підходи до інтеграції фінансових даних.

У межах дослідження розглядаються два основні способи отримання фінансових даних у мобільному застосунку: через API відкритого банкінгу, що передбачає взаємодію з REST API, використання механізмів авторизації OAuth2/PKCE та підтримку оновлення даних, а також шляхом файлового імпорту банківських виписок у форматах CSV, XLSX і PDF з подальшим парсингом, нормалізацією та підготовкою до подальшого використання.

Результати та обговорення

За результатами проведеного аналізу встановлено, що для мобільного застосунку у сфері відкритого банкінгу ключовими вимогами до архітектури мобільного застосунку є безпечність обробки фінансових даних, підтримка кількох джерел їх надходження, відокремлення логіки представлення від бізнес-логіки, а також можливість масштабування при додаванні нових банків, форматів виписок і механізмів синхронізації. Для такої задачі архітектурне рішення має забезпечувати не лише зручну організацію UI-рівня, а й чітке розмежування компонентів, що відповідають за оброблення даних і взаємодію із зовнішніми джерелами. У табл. 1 наведено порівняння підходів до організації коду мобільного застосунку з погляду придатності до інтеграції банківських API та файлового імпорту.

Таблиця 1

Порівняння підходів до організації коду мобільного застосунку

Критерій	MVC	MVVM	MVVM + CA
Розподіл ролей	UI і логіка значною мірою поєднані в Controller, що ускладнює підтримку застосунку	ViewModel відокремлює стан і логіку представлення від UI	UI зосереджений у Presentation Layer, бізнес-логіка – у Domain Layer, робота з джерелами даних – у Data Layer
Робота з даними	Логіка доступу до даних часто змішується з логікою екрана	Частину логіки можна винести у ViewModel, однак вона все ще залежить від структури UI	Робота з API, файлами та локальною БД ізолюється в Data Layer через репозиторії та data source
Підтримка декількох джерел	Ускладнена, оскільки при додаванні нових джерел контролер швидко перевантажується	Можлива, але часто призводить до ускладнення ViewModel	Природна, оскільки API Data Source і File Import Data Source можуть бути об’єднані через єдиний репозиторій
Тестування	Обмежене через високу зв’язаність UI та логіки	Краще, ніж у MVC, оскільки логіку представлення можна тестувати окремо у ViewModel без прямої залежності від UI-компонентів	Найкраще завдяки ізоляції Domain Layer та інтерфейсам доступу до даних

Продовження Табл.1.

Масштабованість	Обмежена, оскільки зі зростанням кількості функцій, екранів і джерел даних контролери швидко перевантажуються, а зв'язаність компонентів зростає	Краща, ніж у MVC, оскільки ViewModel дозволяє краще відокремити логіку представлення, однак при ускладненні роботи з даними навантаження на ViewModel також зростає	Висока, оскільки нові джерела даних, бізнес-сценарії та функціональні модулі можуть додаватися через окремі шари без суттєвої зміни UI-рівня
Складність впровадження	Найнижча, оскільки не потребує складної багатошарової організації та може бути реалізована з мінімальною кількістю компонентів	Помірна, оскільки потребує додаткового виділення ViewModel і організації взаємодії між станом інтерфейсу та даними	Вища на початку, але виправдана для складніших застосунків

На основі порівняння можна зробити висновок, що для задачі автоматизованого обліку фінансових операцій у мобільному застосунку найбільш доцільним є поєднання MVVM та Clean Architecture (далі CA). MVVM забезпечує зручну організацію рівня представлення і стану інтерфейсу, тоді як CA дозволяє відокремити бізнес-логіку та механізми доступу до даних. Саме така комбінація є найбільш придатною для застосунку, що одночасно працює з банківськими API та імпортованими виписками.

На рис. 1 подано спрощену логічну схему гібридної архітектури мобільного застосунку. У межах цієї моделі UI Layer відповідає за відображення даних користувачеві, ViewModel керує станом інтерфейсу та ініціює сценарії оброблення, Domain Layer містить use case для бізнес-логіки, а Data Layer реалізує доступ до кількох джерел фінансових даних. До складу останнього входять Open Banking API Data Source, File Import Data Source, Transactions Repository та Local Database.

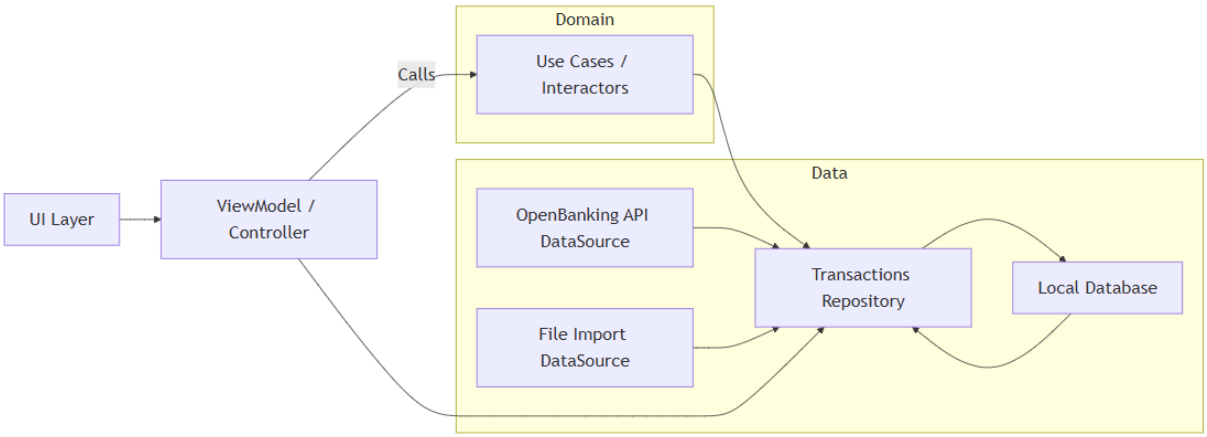


Рис. 1 – Логічна діаграма архітектури мобільного застосунку

Запропонована структура дозволяє організувати роботу застосунку таким чином, що користувацький інтерфейс не залежить безпосередньо від способу отримання даних. При зверненні до історії транзакцій ViewModel викликає відповідний use case, який працює з репозиторієм. Репозиторій, у свою чергу, координує доступ до банківського API, файлового

імпорту та локального сховища, уніфікує отримані записи й передає їх до єдиної моделі транзакцій. Це спрощує розширення функціональності застосунку, оскільки додавання нового банку або нового формату виписки не потребує зміни логіки UI-рівня.

Висновки

У роботі проведено аналіз сучасних підходів до побудови архітектури мобільних застосунків у контексті відкритого банкінгу та визначено ключові вимоги до таких систем: багатоканальність отримання фінансових даних, безпечність і масштабованість. На основі виконаного аналізу запропоновано гібридну архітектуру мобільного застосунку, особливістю якої є поєднання двох способів надходження даних – через банківські API та шляхом імпорту виписок. Структурною основою запропонованої архітектури є поєднання MVVM і CA, у межах якого виокремлено сім основних компонентів: UI, ViewModel, Domain (Use Cases), Repository, API Source, File Source та Local DB. Такий підхід дозволяє уніфікувати оброблення інформації з різних джерел, зменшити залежність від окремих банківських інтерфейсів і створити основу для подальшого розширення застосунку без суттєвої зміни його базової логіки.

Список використаних джерел

1. Hardt D. The OAuth 2.0 Authorization Framework. RFC 6749. 2012. URL: https://datatracker.ietf.org/doc/html/rfc6749 (дата звернення: 07.04.2026).

2. Android Developers. Guide to app architecture. URL: https://developer.android.com/jetpack/guide (дата звернення: 06.04.2026).

3. Apple Inc. View controller programming guide for iOS. URL: https://developer.apple.com/library/archive/featuredarticles/ViewControllerPGforIOS/.

5. Sakimura N., Bradley J., Jones M. Proof Key for Code Exchange by OAuth Public Clients (PKCE). RFC 7636. 2015. URL: https://datatracker.ietf.org/doc/html/rfc7636

6. Shafranovich Y. Common Format and MIME Type for Comma-Separated Values (CSV). RFC 4180. 2005. URL: https://datatracker.ietf.org/doc/html/rfc4180 (дата звернення: 11.04.2026).

Відомості про авторів

Шпарук Тарас Олександрович – здобувач вищої освіти першого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технології Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інженерія програмного забезпечення.

E-mail: taras.shparuk@gmail.com

Шибицька Наталія Миколаївна – доцент кафедри інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси:* інтелектуальні системи навчання та експертні методи оцінювання, методи та моделі штучного інтелекту, нечітка математика та нейронні мережі.

E-mail: shibnatnik@ukr.net

UDC 004.8(045)

ARTIFICIAL INTELLIGENCE IN SOFTWARE ENGINEERING

Oleksandr SHOSTAK
Bachelor's Degree student, Software Engineering Department
National University «Kyiv Aviation Institute»
Scientific supervisor – Olena KOLHANOVA, PhD

This paper examines the real-world effectiveness of generative AI tools in software engineering through a literature review and a personal project case study. While AI accelerates routine coding tasks and is widely adopted, actual time savings remain modest and are often offset by increased code review effort – a phenomenon known as the "Great Toil Shift." The study concludes that AI is a powerful assistant rather than a replacement for human engineers, shifting the developer's role from writing to managing and reviewing code.

Keywords: *artificial intelligence, software engineering, generative AI, LLM, technical debt, software development lifecycle.*

Introduction

Artificial intelligence is entering every part of our lives today. It changes how we work, communicate, and learn. Software engineering is no exception. Generative AI models now help developers write code and build applications. We hear many stories about how these tools will soon do all the programming work. However, we need to look closer and understand if these tools are truly effective in real projects or if they just create new hidden problems.

Purpose of the research

The goal of this research is to analyze the current state of artificial intelligence in software engineering. I want to compare the data from recent scientific studies with my own observations. This research evaluates the efficiency of modern tools and identifies the main problems that developers face when they use these models in real work.

Materials and methods

The methodology of this research relies on a comprehensive and integrated approach to evaluate the impact of artificial intelligence on software engineering. Instead of focusing only on theoretical benchmarks, this study combines a systematic literature review with a practical case study. This method gives a clear picture of how generative models function in real work environments.

The first step involved analyzing recent academic literature and industry reports published in 2026. This review specifically targeted the transition from simple code generation to advanced systems that manage complex tasks. By examining these studies, I analyzed structural changes in the development process and the emergence of new challenges such as cognitive debt [2].

Second, I used a retrospective analysis of my own university project from a course on software modeling. For this project, I applied an extreme automation approach: the artificial intelligence tool was tasked with generating the entire codebase for a .NET backend and a React interface from scratch. My role as a human developer was strictly limited to reviewing the generated logic, fixing architectural errors, and making the system functional. I used this project as a baseline to compare my personal experience with the global data from professional surveys [1].

Results and discussion

The results of this research show that artificial intelligence provides a major boost in speed for routine tasks. According to the developer survey, about 79 percent of programmers now use these tools every day [1]. The biggest advantages are found in the design and implementation phases (see Table 1) [1].

Table 1

Impact of artificial intelligence by development phase [1]

Development phase (SDLC)	Share of positive responses
Design and Implementation	85%
Operation and Maintenance	75%
Requirements Analysis	43%
Testing and Integration	37%
Planning	18%

However, the objective time saved is often less than people expect (see Table 2). While 86 percent of developers are satisfied with their tools, almost half of the workforce saves less than one hour per week [2].

Table 2

Distribution of weekly time savings among developers [2]

Category of weekly time savings	Share of surveyed developers
From 1 to 30 minutes	29%
From 31 to 60 minutes	23%
More than 2 hours	18%
General satisfaction with tools	86%

This gap between speed and efficiency is explained by the Great Toil Shift [2]. The time saved on writing code is now spent on reviewing it. In my own project, I saw the same result. The AI built basic folders and data models very fast. However, when the tool generated complex logic, I had to spend hours finding small errors. This creates cognitive debt because the developer loses a deep understanding of the project structure [2].

Technical benchmarks in 2026 also show that while models like Gemini 3.1 and GPT 5.4 have high scores, they still require strict control in production (see Table 3) [2].

Table 3

Efficacy indicators for frontier models in 2026 [2]

Model designation	Developer organization	SWE bench score	Release date
Gemini 3.1 Pro High	Google	84.17%	February 18 2026
GPT 5.4 High	OpenAI	81.05%	March 4 2026
Claude 4.6 Opus	Anthropic	80.8%	February 4 2026

Conclusions

In conclusion, artificial intelligence is a great assistant but it is not a replacement for a human engineer. The technology has evolved rapidly since last year and it is now much more integrated into the development process. However, my research shows that the human role is still critical. We are moving from a world where we write code to a world where we manage and review code. Developers must use these tools with a clear understanding of the risks to maintain high quality and security in software engineering.

References

1. Gurgul V., Gubela R., Lessmann S. The State of Generative AI in Software Development: Insights from Literature and a Developer Survey. *arXiv preprint*. 2026. URL: https://arxiv.org/abs/2603.16975 (accessed: 10.04.2026).

2. Maes S. H. Evaluating the Efficacy of Artificial Intelligence in Software Engineering: A Post-February 2026 Analysis. *Zenodo*. 2026. URL: https://doi.org/10.5281/zenodo.19103818 (accessed: 10.04.2026).

Information about the author

Oleksandr Shostak – Bachelor's Degree student of the Software Engineering Department, Faculty of Computer Science and Technologies, National University “Kyiv Aviation Institute”. *Research interests:* software engineering, artificial intelligence.

E-mail: 8073577@stud.kai.edu.ua

УДК 004:378:005.8:004.9 (045)

СТУДІЯ АНАЛІТИКИ І ДИЗАЙНУ: ІНТЕЛЕКТУАЛІЗАЦІЯ І ШІ-ПІДТРИМКА РІШЕНЬ АНАЛІТИКИ І ДИЗАЙНУ ЦИФРОВИХ B2B ПРОЄКТІВ

Артем ЯКОВЕНКО
Здобувач вищої освіти другого рівня, кафедра ІПЗ
Національний університет «Київський авіаційний інститут»
Науковий керівник – Володимир ТАЛАЛАЄВ, доцент, к.т.н.

У роботі розглядається підхід до інтелектуалізації процесів прийняття рішень у цифрових B2B-проєктах на основі концепції «студії аналітики». Основою є побудова інтегрованого середовища, що поєднує аналітичні методи та засоби штучного інтелекту. У межах дослідження використано підхід AI Opportunity Mapping для виявлення та пріоритезації напрямів застосування ШІ, а також сформовано онтологічну модель, що забезпечує системну підтримку управлінських рішень в бізнес-системах.

***Ключові слова**: штучний інтелект, AI Opportunity Mapping, управлінські рішення, B2B-проєкти, інтелектуалізація.*

Вступ

Сучасна цифрова трансформація B2B-проєктів зумовлює необхідність переходу від традиційної аналітики до інтелектуалізованих систем підтримки прийняття рішень. У межах науково-дослідного проєкту «Студія аналітики і дизайну» формується інтегроване середовище, яке поєднує інструменти збору, обробки та інтерпретації даних із методами штучного інтелекту.

Однак у більшості організацій використання ШІ має фрагментарний характер, що призводить до неузгодженості аналітичних рішень, низької ефективності їх впровадження та складності масштабування [1]. Це обумовлює необхідність створення системного підходу до інтеграції ШІ в аналітичну діяльність.

У цьому контексті особливого значення набуває орієнтація на управлінські задачі та інформаційні потоки, що забезпечують їх підтримку. Такий підхід дозволяє перейти від технологічно-орієнтованої до проблемно-орієнтованої моделі використання ШІ, забезпечуючи узгодженість між бізнес-цілями та аналітичними інструментами.

Ціль роботи

Метою дослідження є розробка концепції «Студії аналітики і дизайну» як середовища інтелектуалізації процесів прийняття рішень у цифрових B2B-проєктах, а також створення онтологічної моделі для ідентифікації точок застосування ШІ.

Це передбачає: аналіз вузлів прийняття рішень; оцінку доступних даних; визначення доцільності використання ШІ; підбір відповідних аналітичних і AI-рішень.

Матеріали та методи

Методологічною основою дослідження є поєднання системного аналізу, концептуального моделювання та принципів model-driven підходу.

Для формалізації зв'язків між контекстом рішень, даними, аналітичними методами та управлінськими діями використано модель C-R-I-A (Context-Resources-Intelligence-Action).

Застосовано таксономію управлінських рішень із поділом на стратегічний, тактичний та операційний рівні. Проведено аналіз інформаційних потоків з точки зору їх походження, якості та доступності.

Оцінювання даних здійснюється за критеріями повноти, релевантності та надійності в рамках model-driven підходу.

Процес інтелектуалізації реалізовано як послідовність етапів:

– ідентифікація вузлів прийняття рішень;
– формалізація джерел даних;
– оцінка потенціалу автоматизації та інтелектуалізації;
– вибір аналітичних і AI-патернів.

У межах дослідження також використано підхід AI Opportunity Mapping [1], який дозволяє системно виявляти можливості застосування ШІ шляхом аналізу бізнес-процесів і точок прийняття рішень. Даний підхід забезпечує пріоритезацію AI-ініціатив на основі їх бізнес-цінності та складності реалізації.

Забезпечено узгодження типу задачі з класом моделей (прогнозування, класифікація, оптимізація), що підвищує відповідність між бізнес-потребами та аналітичними рішеннями.

Результати та обговорення

У результаті дослідження сформовано онтологічну модель студії аналітики, яка відображає ключові сутності та взаємозв'язки між процесами прийняття рішень, даними та аналітичними інструментами.

Запропонована модель дозволяє:

– структурувати аналітичну діяльність;
– формалізувати процес інтелектуалізації;
– забезпечити повторюваність підходу в різних предметних областях.

У якості центрального інструменту запропоновано використання AI Opportunity Map, яка забезпечує візуалізацію:

– вузлів прийняття рішень;
– доступних джерел даних;
– потенційних AI-рішень.

Застосування цієї карти дозволяє:

– ідентифікувати пріоритетні напрями впровадження ШІ;
– оцінити їх вплив на ключові показники ефективності (KPI);
– обґрунтувати доцільність інвестицій у аналітичні рішення.

Таким чином, AI Opportunity Map виступає інструментом інтеграції аналітики та ШІ в єдину систему підтримки управлінських рішень.

Висновки

Запропонована концепція студії аналітики забезпечує системний підхід до інтелектуалізації цифрових B2B-проєктів. Розроблена онтологічна модель дозволяє узгодити бізнес-цілі, дані та аналітичні інструменти в межах єдиного середовища.

Використання AI Opportunity Map підвищує прозорість процесу вибору AI-рішень та дозволяє оцінити їхній вплив на KPI ще на етапі планування. Це створює надійне підґрунтя для прийняття обґрунтованих управлінських рішень і знижує ризики неефективного впровадження ШІ.

Список використаних джерел

1. From FOMO to Focus. *INNOQ*. URL: https://www.innoq.com/en/blog/2025/11/ai-opportunity-mapping/ (дата звернення: 08.04.2026).

Відомості про автора:

Яковенко Артем Олексійович – здобувач вищої освіти другого рівня, кафедра інженерії програмного забезпечення факультету комп'ютерних наук та технологій Національного університету «Київський авіаційний інститут». *Наукові інтереси*: бізнес-аналітика, інтелектуалізація.

E-mail: artemlg1065@gmail.com

UDC 004.622 (045)

MINING METADATA FROM SOURCE CODE REPOSITORIES

Viktor YAKUBIV
Second-year PhD student,
Taras Shevchenko National University of Kyiv
Scientific supervisor – Oleksandr PROVOTAR, Professor, Doctor of P-M. Sciences

Metadata extraction from source code repositories constitutes a critical phase of the mining software repositories (MSR) process. High-quality metadata enables dataset filtering, characterises diversity, and supports the identification of collection deficiencie – all of which are essential to the curation of reliable source code datasets. Building upon a previously proposed data mining approach, this paper addresses the subsequent data analysis phase of the MSR pipeline by presenting a method for extracting metadata from source code files and their containing repositories. The proposed approach is designed around the UNIX philosophy of composable steps and draws upon three complementary data sources: the file system node, the source code file contents, and the parent repository context. The resulting metadata can augment curated source code datasets, serving as a foundation for further analysis, research, or targeted subset selection.

Keywords: *metadata extraction, source code analysis.*

Introduction

Metadata plays a crucial role in determining the quality of source code datasets. It enables users to select and filter files according to properties of interest, serves as a quality indicator, characterises dataset diversity, and allows curators to identify deficiencies. The extraction of such properties – referred to as data analysis – constitutes a distinct, concluding phase of the mining software repositories (MSR) process [1]. The overall MSR process comprises four consecutive phases: (1) study design, (2) repository identification, (3) data mining, and (4) data analysis.

In our previous work, we presented an approach addressing phases 2 and 3, namely the mining of source code from GitHub [2]. In this paper, we build upon that approach and its design principles to propose a method for metadata extraction from source code files and repositories – addressing phase 4. Specifically, given a collection of source code repositories and files produced by the mining phase, we present an approach to labelling these artefacts with metadata derived from both the source code files themselves and their containing repositories.

Related research

Metadata mining – encompassing metadata analysis, extraction, and labelling – is a central concern in the MSR field, with its specific form varying by application area. Prominent applications include source code labelling to enhance software development tool capabilities and correlating projects by patterns to predict defects.

Sulír and Porubän [3] propose a classification framework for labelling tools that characterises possible data sources, targeted data subsets, presentation methods, and means of storing extracted data. They further analyse the challenges of incorporating existing software analysis tools into an integrated development environment. Nagappan et al. [4] mine metrics from source code and correlate these with project change history to predict software defects. Similarly, Poncin et al. [5] propose a

framework for analysing code repository history alongside other software development process events, enabling these to be matched against one another to identify process-level patterns.

The present work differs from the above in that it does not target a specific application domain. Rather, the proposed approach is designed to be generic: whilst the concrete metadata extracted will depend on the study in which the approach is applied, the underlying extraction process and its guiding design principles remain consistent across use cases.

File analysis approach

For a given source code file, metadata can be mined from three sources: (S1) file system node referencing the file, (S2) contents of the source code file, and (S3) the parent repository that contains the file. These three sources and the process for extracting data are illustrated in Figure 1. The left column of the figure comprises three data sources, each represented by a cylinder. The central column lists the analysis steps, and the right column presents the corresponding outputs. Each output is depicted as a rectangle with a wavy bottom edge, denoting a file. Several outputs additionally contain nested rectangles illustrating the properties stored within the respective file.

The file system node (FS node, S1) is the starting point for the data extraction. The most informative FS node property is the file name when it follows established naming conventions. The most widely adopted convention is the extension suffixes, which denote a file type and its subtypes. For example, the *.c* suffix indicates a C-language implementation file, whereas *.h* indicates a C-compatible header file. Similarly, *.js* denotes a generic JavaScript source file, while *.mjs* is a source with ECMAScript modules, and *.cjs* is a source with CommonJS modules. Because JavaScript programmes are distributed over the network as source code, it is common practice to produce an optimised artefact that reduces network packet size; this process, known as minification, typically yields artefacts carrying the *.min.js* suffix, which denotes a minified JavaScript source. In the first extraction step (F1), the file name is received as input (I1); the type and subtype properties (P1.2) and the byte file size (P1.3) are extracted and stored as output O1.

In the second step (F2), the file contents (S2) are parsed into an abstract syntax tree (AST, F2.1), which validates whether the extracted file type (P1.2) conforms to the expected syntax. A feature map (P2.1) is then derived from the AST (F2.2). For C++ source files, such a map may record usage of the <*debugging*> library or unnamed placeholder variables standardised in C++26. Additionally, metrics (P2.2) can be computed from the source file (F2.3), ranging from simple measures – such as total lines of code or non-whitespace line counts (with or without comments) – to structural complexity metrics. Lastly, content-based metadata can be extracted (F2.4), including author names (P2.3), copyright notices (P2.4), and creation dates (P2.5), where present in source code comments. The output of this step O2 comprises the aforementioned properties P2.1–5.

In the third step (F3), a hash value is computed from the raw binary file content and stored as output O3. This hash serves as the uniqueness metric for the source code file. Computing file content hash enables file deduplication, which is particularly useful when analysing repositories that store multiple copies of a single file – a pattern common in repositories containing per-version software artefacts. F3 is independent of all other steps and may therefore execute in parallel.

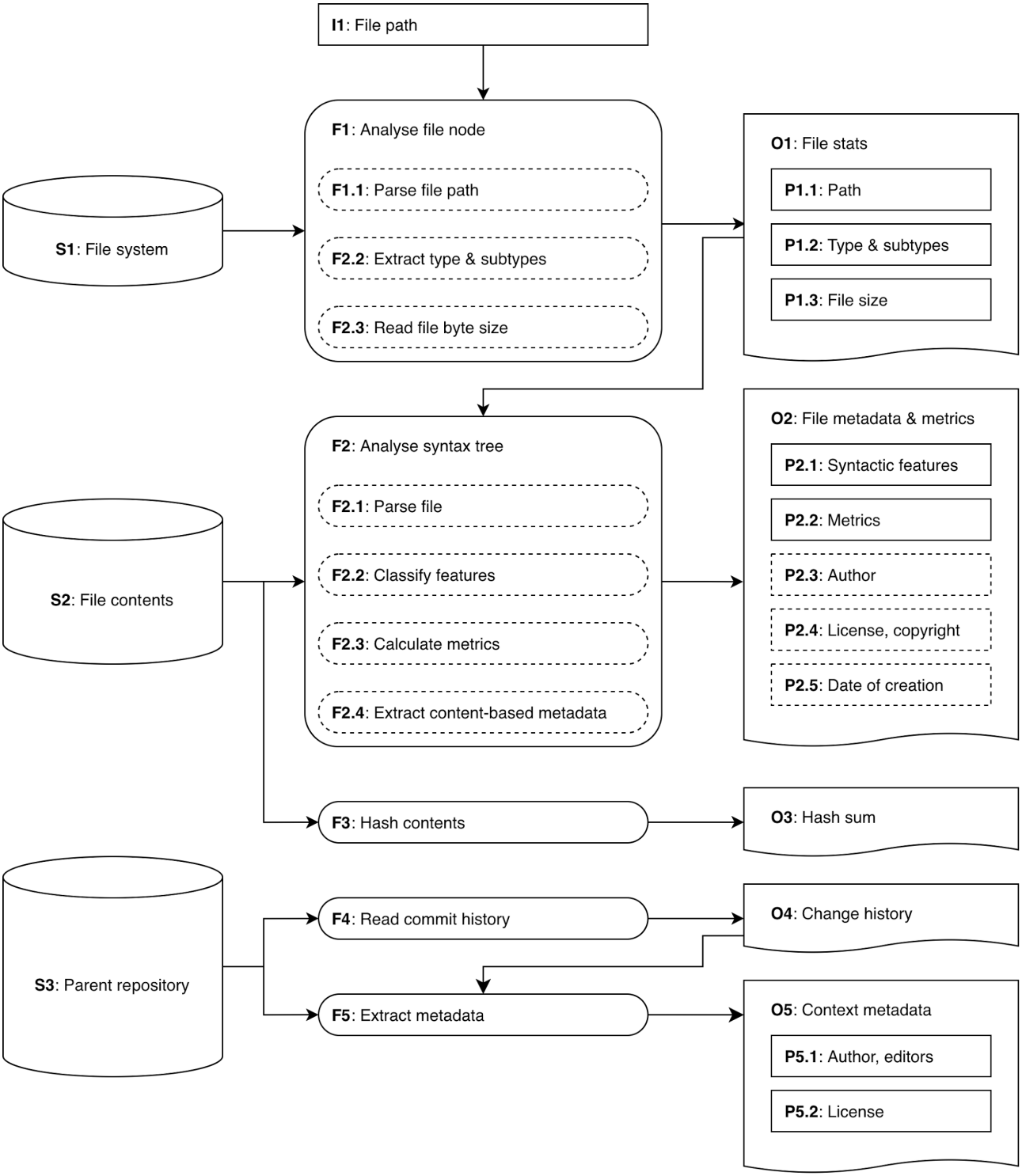


Figure 1 – The metadata extraction approach

The final data source is the context (S3) – the parent repository containing the file. It is analysed on the last two steps: F4 and F5. In the fourth step (F4), the file history is extracted and stored as output O4 in a format suitable for subsequent analysis. The fifth step (F5) aggregates and extracts additional metadata from this history. It is common for individual files to omit licence information, yet the repository may contain a *LICENSE* file applicable to all constituent files (P5.2). Furthermore, the repository history records the first appearance of each file, subsequent modifications (such as feature additions and bug fixes), and the associated authors and contributors (P5.1). These properties are stored as the output O5.

Conclusion

This paper has presented a systematic approach to metadata extraction from source code files and repositories, addressing the data analysis phase of the MSR process. Drawing on three complementary data sources – the file system node, the file contents, and the parent repository – the approach extracts a range of properties including file type and subtype, code metrics, content-based attributes such as licensing and authorship, and the change history. By extending our prior work on repository mining, the proposed method completes a coherent, end-to-end pipeline for constructing richly labelled source code datasets.

References

1. Teaching Mining Software Repositories / Z. Codabux et al. *Handbook on Teaching Empirical Software Engineering* / eds.: D. Mendez et al. Cham: Springer Nature Switzerland AG, 2024. Pp. 399–448. DOI: 10.1007/978-3-031-71769-7.

2. V. Yakubiv, D. Terletskyi. Mining source code from GitHub repositories // *Proceedings of LV National Technical Conference of Vinnytsia National Technical University*. Vinnytsia, 2026.

3. M. Sulír, J. Porubän. Labeling source code with metadata: a survey and taxonomy // *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems*. 2017. Pp. 721–729. DOI: 10.15439/2017F229.

4. N. Nagappan, T. Ball, A. Zeller. Mining Metrics to Predict Component Failures // *ICSE'06: Proceedings of the 28th International Conference on Software Engineering*. ACM, 2006. Pp. 452–461. DOI: 10.1145/1134285.1134349.

5. W. Poncin, A. Serebrenik, M. van den Brand. Process Mining Software Repositories // *Proceedings of 2011 15th European Conference on Software Maintenance and Reengineering*. IEEE, 2011. Pp. 5–14. DOI: 10.1109/CSMR.2011.5.

Information about the author:

Viktor Yakubiv – second-year PhD student at the Faculty of Computer Science and Cybernetics at Taras Shevchenko National University of Kyiv. *Research interests:* metadata extraction, source code analysis.

E-mail: yakubiv@knu.ua