DOI: 10.32347/2412-9933.2025.63.191-200
УДК 004.8

Поздняков Олег Анатолійович
Аспірант кафедри програмних засобів,
https://orcid.org/0009-0006-3955-802X
Національний університет «Запорізька політехніка», Запоріжжя
Пархоменко Анжеліка Володимирівна
Кандидат технічних наук, доцент, доцент кафедри програмних засобів,
https://orcid.org/0000-0002-6008-1610
Національний університет «Запорізька політехніка», Запоріжжя

ДОСЛІДЖЕННЯ ТА ВИБІР ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ ДЛЯ АВТОМАТИЗАЦІЇ МІГРАЦІЇ АВАР-КОДУ

Анотація. *Досліджено проблему міграції програмного коду при переході зі старої версії системи SAP ERP 6.0 на сучасні ERP-платформи, зокрема SAP S/4HANA. Обґрунтовано необхідність автоматизації процесу міграції з використанням методів та засобів штучного інтелекту. Комплексний аналіз існуючих академічних та промислових тематичних досліджень підтвердив доцільність використання великих мовних моделей для міграції коду. Виявлено низку переваг підходу на основі великих мовних моделей, зокрема здатність обробляти різноманітні сценарії міграції. Виділено ключові ризики та обмеження, поширені в корпоративному середовищі, зокрема: безпеку даних, конфіденційність, високу вартість викликів та обмежені знання моделей щодо предметної області. Розроблено метод вибору великої мовної моделі з відкритим вихідним кодом, яка може безпечно використовуватися в ландшафті підприємства, тим самим зменшуючи ризики, пов'язані з хмарними рішеннями. Метод базується на розробленій системі критеріїв, що включає: розмір моделі, розмір контекстного вікна, тип ліцензії, підтримку навчання з точним налаштуванням, орієнтацію на якість коду (виміряну за допомогою еталонного набору даних), підтримку квантування моделі та підтримку спільноти. Для систематичного ранжування моделей-кандидатів запропоновано використання гібридного методу AHP-TOPSIS. Метод AHP використовувався для перевірки ваг критеріїв, тоді як TOPSIS використовувався для ранжування моделей на основі їх близькості до ідеального рішення. На основі розробленого методу було обрано три великі мовні моделі: Qwen 2.5 Coder 14B, DeepSeek-Coder-V2 16B та Llama 3.1 8B. Розроблено план комплексного тестування обраних моделей для подальших експериментальних досліджень. Наукова новизна роботи полягає в розробці методу вибору великих мовних моделей для задач міграції користувацького коду, що відрізняється від існуючих методів системою критеріїв вибору моделі та застосуванням гібридного підходу AHP-TOPSIS для ранжування моделей-кандидатів. Практична цінність роботи полягає в тому, що розроблений метод дозволяє обрати модель з урахуванням обмежень корпоративного середовища та вимог інформаційної безпеки, що дозволить підвищити рівень автоматизації міграції АВАР-коду під час переходу на нові версії ERP-систем від компанії SAP SE. Розроблений метод може застосовуватися для вибору великих мовних моделей під час модернізації інших великих комп'ютерних систем, розроблених як на мовах програмування з відкритим вихідним кодом, так і на пропрієтарних мовах.*

Ключові слова: ERP-система; міграція АВАР-коду; велика мовна модель; критерії вибору; метод AHP-TOPSIS

Постановка проблеми

Процес цифрової трансформації сучасних підприємств вимагає постійного розвитку на основі регулярного оновлення використовуваних складних комп'ютерних систем. Особливі виклики пов'язані з міграцією користувацького програмного забезпечення із застарілих версій систем, зокрема SAP ERP 6.0, на сучасну ERP-систему SAP S/4HANA [1; 2]. Це завдання набуває особливої актуальності у зв'язку з широким використанням в Україні та світі систем SAP ERP від компанії SAP SE, що є світовим лідером у розробці бізнес-застосунків.

На відміну від інших складних комп'ютерних систем, створених з використанням поширених мов програмування, всі стандартні бізнес-застосунки SAP та весь користувацький код системи підприємства розробляються на основі пропрієтарної

мови ABAP. Користувацький ABAP-код, напрацьований десятиліттями, зазвичай не є стандартизованим, включає застарілі конструкції, специфічні API та відображає унікальні особливості бізнес-процесів підприємства. Ручна адаптація таких рішень для сумісності з новими версіями ERP-системи вимагає значних витрат ресурсів та часу [3].

Тому підприємства потребують впровадження автоматизованих засобів міграції користувацького ABAP-коду з використанням штучного інтелекту (ШІ) як ключового інструменту адаптації до змін [4].

Актуальність теми підтверджується зростанням інтересу до застосування ШІ у задачах підтримки життєвого циклу програмного забезпечення, у тому числі на етапах розробки, міграції та модернізації коду [3].

Ефективна автоматизація процесу міграції коду потребує застосування великих мовних моделей (Large Language Models, LLM), здатних виконувати рефакторинг та переробку коду зі збереженням функціональності та бізнес-логіки [5 – 7]. Застосування LLM дозволяє значно прискорити процеси аналізу, адаптації та переписування застарілих фрагментів програмного коду, особливо в умовах обмеженого доступу до документації та відсутності авторів. При цьому задачі міграції користувацького коду мають специфіку, відмінну від задач генерації нового коду, що вимагає додаткових досліджень.

Незважаючи на наявність стандартних інструментів аналізу коду від компанії SAP (ATC, Custom Code Migration [2]) та відоме використання генеративного ШІ [5 – 7], не існує відкритих рішень, які застосовують можливості LLM для автоматизованої міграції ABAP-коду для ERP-системи від компанії SAP SE.

Крім того, використання хмарних LLM (таких як ChatGPT, Claude, Gemini тощо) для задач міграції користувацького ABAP-коду ускладнене або взагалі неможливе через обмеження, що накладаються корпоративною політикою підприємств щодо інформаційної безпеки та ліцензування. Тому важливо забезпечити локальне розгортання та налаштування LLM на внутрішній інфраструктурі підприємства, що можливо лише для моделей з відкритим кодом та відкритою ліцензією (Apache 2.0, MIT або аналогічними).

Таким чином, актуальним є подальше дослідження особливостей застосування LLM у процесах міграції користувацького ABAP-коду між версіями SAP-систем.

Мета статті

Розробити та практично застосувати метод вибору LLM для автоматизації міграції користувацького ABAP-коду при переході на нову версію ERP системи з урахуванням обмежень корпоративного середовища та вимог інформаційної безпеки.

Аналіз останніх досліджень і публікацій

Останніми роками з'явилося кілька досліджень, що демонструють можливості LLM у завданнях міграції коду.

Одне з найбільших практичних досліджень було проведено в компанії Google групою інженерів, яка використовувала LLM для автоматизації масштабної міграції коду в корпоративному репозиторії [5]. Їх робота описує передачу ідентифікаторів з 32-бітного на 64-бітний по всьому системному коду (понад 500 мільйонів рядків), оновлення застарілих бібліотек модульного тестування JUnit3 до JUnit4, а також заміну бібліотеки Joda-Time на сучасний пакет java.time. Автори називають підхід успішним прикладом для таких ініціатив.

Компанія Amazon також активно розробляє свої рішення в області міграції коду з використанням LLM [6]. У 2023 році компанія представила інструмент Amazon Q – систему генеративного ШІ, яка спрощує оновлення великих проєктів Java з версії 8/11 до версії 17. У звітах Amazon стверджується, що використання ШІ-асистента дозволило заощадити сотні мільйонів доларів за рахунок автоматизації оновлення багатьох застосунків [7].

Окрім проєктів промислової міграції, у 2024–2025 роках в академічних колах активно оцінювались можливості та обмеження LLM у завданні міграції коду. Автори [8] показали, що ChatGPT здатний ефективно виконувати API-міграцію бібліотеки: у їх експерименті модель успішно перенесла застосунок на Python на нову версію бібліотеки SQLAlchemy ORM. Більш того, при правильному формулюванні запиту LLM не тільки фіксувала несумісні виклики, а й модернізувала код, використовуючи нові можливості бібліотеки (асинхронність, типізація) зі збереженням початкової функціональності [8].

Для систематичного вивчення існуючих проблем в роботі [6] було запропоновано бенчмарк CodeMEnv, який містить 922 приклади реальних змін API в популярних бібліотеках Python і Java. Цей пакет перевіряє три аспекти: виявляє застарілі виклики, пояснює зміни та автоматично переписує код для цільової версії. Також були виявлені типові проблеми: моделі краще оновлюють старий код до нової версії, ніж навпаки, і часто допускають логічні помилки, наприклад, посилаючись на зміни на неправильні версії бібліотеки. Тим не менш, сама поява такого орієнтиру стимулює подальші дослідження і дозволяє провести об'єктивне порівняння підходів [6].

Існують також вузькоспеціалізовані дослідження, присвячені міграції коду в конкретних доменах, наприклад, автоматичне оновлення програм на мові Qiskit (фреймворк для квантового програмування) при переході на нову версію бібліотеки [9]. В цій роботі автори проаналізували документацію та створили класифікацію різних типів змін в API (наприклад, коли функції застарівають, переносяться в інші модулі або класи перейменовуються). Потім вони подавали на вхід мовної моделі цю класифікацію змін і застарілий код, який потрібно оновити. Модель використовувала ці дані разом зі своїми внутрішніми знаннями про міграцію коду для пропозиції автоматичного рефакторингу.

На конференції TechEd 2024 [10] було презентовано Generative AI in ABAP Cloud – перший фірмовий LLM від SAP, навчений на спеціалізованому корпусі коду ABAP. Цей сервіс ШІ, який є частиною ініціативи SAP Joule, вже вміє генерувати код, підказувати з документації та навіть створювати модульні тести в ABAP.

У табл. 1 наведено результати аналізу останніх досліджень та публікацій з точки зору сценарію міграції, використовуваної LLM, мови програмування, ключових метрик.

Як показали проведені дослідження, LLM мають низку ключових переваг перед традиційними методами міграції коду (табл. 2). LLM можуть самостійно генерувати зміни в коді на основі опису завдання, виконуючи величезну кількість однотипних правок набагато швидше, ніж людина [5]. На відміну від скриптів AST (Abstract Syntax Tree), які пишуться під конкретний сценарій, LLM здатна адаптуватися до різних типів міграції, різних мов програмування та алгоритмів. Достатньо змінити інструкцію на природній мові, і модель спробує перенести код на нову версію бібліотеки, інший фреймворк або навіть оптимізувати його під кращі практики [5].

Таблиця 1 – ***Результати дослідження щодо використання LLM у проєктах міграції коду***

Сценарій міграції	**Мова**	**Масштаб/ метрика**	**Використовувана LLM**	**Ключові висновки**
32→64-бит ID, JUnit 3→4, Joda-Time→java.time [5]	C++/Java	500 мільйонів рядків; 5,359 файлів	допрацьована версія Google Gemini	74% редагувань, згенерованих LLM; Час виконання проєкту скорочено на 50%
Java 8/11→17 [6; 7]	Java	«Сотні проєктів» (NDA)	ChatGPT-4	AI-помічник прискорює роботу
SQLAlchemy 1.3→2.0 [8]	Python	12 репозиторіїв OSS	ChatGPT-4	chatGPT виправив 83% несумісностей, впровадивши асинхронність/типізацію
922 Рефакторингу API [6]	Python, Java	pass\@1 ≈ 26.5 % (GPT-4)	GPT-Turbo-3.5, GPT-4o-Mini, GPT-4o, Llama-3.1-8B, Qwen2.5-Coder-7B	Виявлені часті логічні помилки, потрібен контроль
v0.39→v1.* [9]	Qiskit-Python	245 прикладів файлів	ChatGPT-4	Таксономія + LLM автоматично усунула 90% застарілих конструкцій
Generative AI in ABAP Cloud [10]	ABAP	У планах	SAP AI	LLM пояснює код, пропонує правильний код Cloud-ABAP (хмарний код та тільки хмарна LLM)

Таблиця 2 – ***Переваги LLM перед традиційними методами міграції коду***

Аспект	**Традиційний підхід**	**Підхід з LLM**	**Перевага LLM**
Виявлення застарілих викликів	Статичний аналіз, регулярні вирази	Перевірка семантичного контексту	Уловлює нетривіальні шаблони
Генерація виправлень	Скрипти трансформації AST	Пряма генерація патчів	Швидше адаптується до нових правил
З урахуванням стилю	Обмежена, ручна	Модель переймає ідіому з корпусу	Код відразу "чистий" і уніфікований
Покриття випадків	Обмежено скриптовими сценаріями	One-shot / few-shot генерація	Охоплює рідкісні, складні випадки
Час до результату	Від тижнів до місяців	Хвилини–години	Скорочує час міграції на 50-90%

LLM враховують ширший контекст коду, ніж прості заміни шаблонів. Вони навчаються на мільйонах прикладів і вміють враховувати семантику: розуміти призначення функцій, угоди про іменування, типи змінних. Завдяки цьому LLM часто роблять міграцію «розумнішою». Наприклад, вони можуть одразу замінити застарілий виклик на рекомендований аналог і налаштувати пов'язані частини коду (параметри, типи, обробку помилок) [6].

Використання готової моделі скорочує час від постановки проблеми до отримання чорнового рішення. Розробнику не потрібно розробляти окремий інструмент міграції, що є трудомістким завданням, а достатньо сформулювати запит до LLM. За умови правильної перевірки та доопрацювання кінцевий процес відбувається набагато швидше, ніж класичним ручним переписуванням коду [5].

LLM, навчені на великих репозиторіях, часто дотримуються ідіоматичних стилів коду та сучасних практик. Генеруючи оновлений код, модель може негайно застосовувати рекомендовані угоди стилів і шаблони проєктування. У результаті, крім мінімальної міграції коду на нову версію, можна одночасно поліпшити якість коду, підвищити читабельність, додати типізацію та тести [5].

Незважаючи на перелічені вище переваги, розробники та дослідники стикаються з низкою суттєвих проблем при використанні LLM для міграції коду. Зокрема, LLM можуть генерувати некоректні зміни. Наприклад, Google повідомляє, що ~20% змін з LLM вимагали повернення до попередньої версії або доопрацювання, оскільки вони були неправильними або зайвими [5].

LLM мають властивість допускати логічні невідповідності: іноді вони посилаються на зміни в неправильній версії бібліотеки або неправильно тлумачать необхідне перетворення [6]. Без перевірки такі помилки можуть порушити працездатність системи. Оскільки застосування LLM не гарантує стовідсотково правильний результат, згенерований код має бути перевірений фахівцями. Це вимагає суттєвих витрат часу, оскільки кожна зміна має бути перевірена: чи проходить код компіляцію та тести, чи не вніс ШІ якісь побічні ефекти.

Більшість сучасних LLM обмежені фіксованою довжиною запиту, що ускладнює роботу з дуже великими файлами або міжмодульними змінами. З іншого боку, проєкти міграції часто включають тисячі файлів одночасно [5].

LLM може не бачити повного контексту проєкту, через що є ризик пропустити пов'язані редагування. Крім того, генерація великого коду є витратною з точки зору обчислень та має високий час відгуку.

LLM навчаються на даних, які можуть не містити найновішої інформації про останні версії бібліотек. Якщо модель не знає про зміни в нещодавно випущеній версії, її пропозиції щодо міграції можуть бути застарілими або неправильними. Без підключення до актуальної документації або додаткового навчання на нових даних, це обмежує застосовність «з коробки».

Запуск загальних LLM або моделей, навчених на великому обсязі домену (мови програмування), який не потрібен для міграції, вимагає значних обчислювальних ресурсів. При міграції великого проєкту може виникнути необхідність «прогону» сотень тисяч рядків через модель, що призводить до значних витрат часу та коштів [5].

Важливими є питання безпеки та конфіденційності, адже надсилання вихідного коду зовнішнім службам LLM може порушити політику інформаційної безпеки компаній у разі закритого коду. Це змушує або використовувати локальні моделі, або анонімізувати/розбивати дані, що ускладнює процес. У контексті розвитку підприємств ця проблема є суттєвою, оскільки не кожна організація готова перенести свій код на хмарний ШІ [11].

Усі описані вище дослідження в галузі міграції коду здійснювалися або планувалися для використання комерційними сервісами LLM, насамперед ChatGPT та аналогами. Такий підхід має декілька принципових обмежень, які є дуже важливими:

– навіть у версіях Team/Enterprise провайдер LLM не може гарантувати стовідсоткову ізоляцію та конфіденційність корпоративних даних, оскільки вся переписка проходить через зовнішню інфраструктуру [11];

– моделі загального призначення, до яких належить ChatGPT, не забезпечують повної конфіденційності домену та потребують додаткового навчання або доповнення RAG для адекватної роботи з вузькоспеціалізованою мовою ABAP [12];

– при великих обсягах коду вартість викликів API GPT-4 (особливо з контекстом понад 32к) стає значною, що серйозно збільшує бюджет складних проєктів міграції (SAP ERP → S/4HANA) [13].

Виклад основного матеріалу

На основі проведених досліджень авторами було виконано класифікацію проблемних ситуацій при міграції коду, а також виявлено існуючі методи та практики, які дозволяють подолати їх за умови ефективного використання LLM (табл. 3).

Аналіз показав, що суттєву частину проблем (витік конфіденційного коду через зовнішні API; обмежена доменна експертиза універсальних моделей; висока вартість масштабного доступу до комерційних LLM) можливо усунути за рахунок локального розгортання LLM з відкритим вихідним

кодом, яку потрібно навчати на мові ABAP, шаблонах міграції та розширювати шаром RAG (Retrieval-Augmented Generation).

Для цього потрібно розробити метод вибору LLM з відкритим вихідним кодом, яка відповідає технічним та ліцензійним вимогам щодо міграції ABAP-коду на нові версії систем SAP. Метод включає наступні етапи:

– розробка системи критеріїв вибору;

– формування початкового набору релевантних моделей;

– формування остаточного набору моделей.

На основі порівняльного аналізу відкритих моделей LLM (їх архітектур [14; 15], ліцензій, метрики HumanEval [16], обсягу контексту та доступності SFT/PEFT (Supervised Fine-Tuning/Full-Parameter Fine-Tuning) адаптації [17], підтримки квантування [18]), специфіки завдання міграції ABAP-коду на SAP S/4HANA, а також виявлених проблем та шляхів їх вирішення, було розроблено систему критеріїв вибору LLM (табл. 4). Оскільки знання мови ABAP не значиться серед мов у документації в жодній із проаналізованих моделей (ця мова зустрічається рідко у відкритих репозиторіях, її частка у навчальних даних мінімальна), тому до таблиці не потрапив критерій, що характеризує рівень знання мови ABAP.

На наступному етапі, на основі розробленої системи критеріїв, потрібно проаналізувати існуючі LLM та сформувати початковий набір релевантних моделей (табл. 5). При аналізі обраних моделей було виявлено, що деякі критерії мають однакові значення для всіх розглянутих альтернатив. Зокрема, критерії «Підтримка навчання з точним налаштуванням (SFT/PEFT)» та «Підтримка квантування моделі» показали однакові результати для всіх семи моделей: усі підтримують методи PEFT (зокрема LoRA) для ефективного перенавчання та квантування до 4-8 біт без значної втрати якості. Оскільки ці критерії не дозволяють диференціювати альтернативи, вони отримають однакову нормалізовану оцінку при подальших розрахунках.

На наступному етапі необхідно виконати вибір найкращої LLM, що передбачає розв'язання задачі багатокритеріального прийняття рішень (Multiple-Criteria Decision-Making, MCDM). При цьому необхідно враховувати одразу кілька параметрів моделі, іноді суперечливих.

Таблиця 3 – ***Класифікація проблемних ситуацій при міграції коду***

Класифікаційна ознака	**Проблема**	**Прояв**	**Методи/практики подолання**
Технічні обмеження LLM	Неточність генерації	Некоректні API, пропуск контексту	Поєднання «пошук → LLM → тести»; покрокова схема *detect–explain–fix* [5; 6]
	Обмежений контекст	Великі файли не вміщуються в контекстне вікно	Підбір моделі з великим контекстним вікном; модуляція; RAG з семантичним пошуком [5; 6]
	Старіння знань	Модель не знає змін у бібліотеці/мові	RAG; перенавчання на нових даних [5;6]
Ресурсні обмеження	Вартість ресурсу	Висока ціна за запитами	Використання opensource LLM; підбір оптимального розміру моделі; квантування моделі 2-bit..8-bit [8;9]
Людський фактор	Довіра розробників	Скептицизм, перевантаження рев'ювера коду	Пояснювальні підказки, інтерактивні інструменти міграції, людина в циклі, семантичний аналіз отриманого коду [5]
Безпека та конфіденційність	Безпека та конфіденційність	Можливі витоки даних, порушення ліцензійних прав	Встановлення локальної LLM [6]
Доменна специфіка	Неповна доменна область («знання» мови міграції)	Набори даних, що використовуються для викладання LLM малого обсягу та якості	Локальна модель LLM з перенавчанням для цільового домену (мова, бібліотека) [6]

Таблиця 4 – ***Критерії вибору LLM***

Критерій	Вага	Опис
Розмір моделі	0,12	До 16 млрд параметрів (щоб модель могла працювати на одному графічному процесорі з 24 ГБ VRAM, наприклад NVIDIA RTX 3090)
Розмір контекстного вікна	0,2	Щонайменше 32 000 токенів (32K) для обробки великих обсягів коду та пов'язаної документації
Тип ліцензії	0,1	Дозволені тільки відкриті ліцензії, що дозволяють комерційне використання, а саме Apache 2.0, MIT, Llama 2 Community License або подібні
Підтримка навчання з точним налаштуванням (SFT/PEFT)	0,15	Для адаптації моделі до конкретних даних (у нашому випадку - коду ABAP) необхідна підтримка методів перенавчання з ефективністю параметрів (PEFT, наприклад, LoRA)
Орієнтація на якість коду (Бенчмарк HumanEval)	0,32	Моделі або універсальні, але навчені на значному обсязі вихідного коду, або спеціально донавчені для генерації та розуміння коду. Особлива увага приділяється підтримці різних мов програмування
Підтримка квантування моделі	0,05	Оцінюється, наскільки добре модель піддається квантуванню (8-біт, 4-біт тощо) без значної втрати якості та з підвищенням швидкості
Підтримка спільноти	0,06	Наскільки активно модель обговорюється та вдосконалюється спільнотою

Таблиця 5 – ***Зведена таблиця обраних моделей***

Модель	Ліцензія	Контекстне вікно	Human Eval, %	Розмір, B	Спільнота*
Qwen 2.5 Coder 14B [19]	Apache 2 (1)	128 000	89.6	14	3
DeepSeek-Coder-V2 [20]	MIT (1)	128 000	90.2	16	3
Gemma 3 12B [21]	Gemma license (0.7)	128 000	45.7	12	2
CodeLlama 13B [22]	Llama 2 (1)	100 000	50.6	13	3
Llama 3.1 8B [23]	Llama 3 (1)	128 000	72.6	8	3
Mistral NeMo 12B [24]	Apache 2 (1)	128 000	71.3	12	2
Phi-3 14B [25]	MIT (1)	128 000	59.1	14	2
*Спільнота: 3 = Сильна (>10 K зірок/завантажень), 2 = Середня.					

Аналіз показав, що для вирішення цього завдання існують відомі формальні методи, серед яких можна відзначити такі:

– проста техніка багатокритеріальної оцінки – SMART (Simple Multi-Attribute Rating Technique) [26];

– метод, що реалізує ідею зваженої суми – WSM (Weighted Sum Model) або SAW (Simple Additive Weighting) [27];

– ієрархічний аналітичний процес Сааті, який дозволяє розраховувати вагові коефіцієнти критеріїв і оцінювати альтернативи на основі парних порівнянь – AHP (Analytic Hierarchy Process) [28];

– метод, заснований на пошуку альтернативи, максимально наближеної до гіпотетичного «ідеального» рішення і максимально віддаленої від «найгіршого» рішення – TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) [29].

У науковому дослідженні запропоновано застосувати гібридний підхід AHP-TOPSIS [30], використовуючи AHP для перевірки вагових коефіцієнтів критеріїв, а також TOPSIS для побудови підсумкового рейтингу альтернатив.

Відповідно до методу AHP було сформовано матрицю попарних порівнянь $A_{ij}=w_i/w_j$. Після нормалізації стовпців отримано відповідну нормалізовану матрицю. Перевірка узгодженості ваг показала CR (Consistency Ratio) = 0, що значно нижче граничного значення 0,1 (запропонованого Сааті), і свідчить про високу узгодженість суджень експертів та дозволяє використовувати отримані ваги в подальших розрахунках.

Відповідно до алгоритму TOPSIS, на першому кроці була сформована векторно-нормалізована матриця рішень, що дозволяє усунути масштабні відмінності між показниками. Далі ця матриця була помножена на вектор вагомостей критеріїв, що дало зважену нормовану матрицю. На цій основі визначаються ідеальні та антиідеальні вектори, що представляють відповідно найкращі та найгірші можливі значення для кожного критерію. Після обчислення евклідових відстаней від кожної

альтернативи до заданих векторів, було отримано коефіцієнт відносної близькості до ідеалу, що дозволило сформувати кінцеву матрицю TOPSIS (табл. 6).

Таблиця 6 – ***TOPSIS: дистанції та підсумковий рейтинг***

Модель	d^+ (до ідеалу)	d^- (к анті)	C	Ранг
Qwen 2.5 Coder	0.021	0.079	0.789	1
DeepSeek-Coder-V2	0.028	0.080	0.740	2
Llama 3.1	0.030	0.059	0.659	3
Mistral NeMo	0.036	0.051	0.581	4
Phi-3	0.058	0.032	0.353	5
Gemma 3	0.079	0.022	0.218	6
CodeLlama	0.072	0.020	0.216	7

За результатами оцінки AHP-TOPSIS, модель Qwen 2.5 Coder 14B зайняла перше місце в рейтингу завдяки оптимальному поєднанню характеристик. На другій позиції розташувалася модель DeepSeek-Coder-V2 16B, а на третій – Llama 3.1 8B.

Для експериментального підтвердження критерію якості коду (HumanEval) і підбору фінальної моделі для додаткового навчання мові ABAP розроблено план комплексного тестування обраних моделей:

– тести для мови Python з використанням стандарту HumanEval для перевірки результатів з опублікованими даними моделі. Це дозволить встановити базовий рівень якості та порівняти з існуючими рішеннями;

– експерименти з ABAP, які включатимуть генерацію та міграцію користувацьких програм з ERP на S/4HANA з подальшою оцінкою через SAP ATC та abaplint. Такий підхід забезпечить практичну перевірку функціональності в реальних умовах використання;

– бенчмарк продуктивності, що передбачає вимірювання пропускної здатності в токенах за секунду, споживання відеопам'яті та часу відгуку системи. Ці метрики критично важливі для оцінки практичної придатності рішення в корпоративному середовищі;

– порівняльний аудит якості коду через експертну перевірку читабельності та відповідності документації SAP. Це забезпечить комплексну оцінку не лише технічної коректності, але й практичної придатності згенерованого коду для подальшого використання та підтримки.

За результатами цих тестів буде прийнято остаточне рішення про те, яку модель взяти за основу для подальшого додаткового навчання PEFT мові програмування ABAP.

Висновки

Проведено дослідження особливостей використання LLM для автоматизованої міграції коду при оновленні складних ERP-систем. Визначено ключові переваги, обмеження та проблеми використання LLM, а також проаналізовано методи та практики, які дозволяють подолати виявлені проблемні ситуації. На основі проведеного аналізу було розроблено метод вибору LLM з відкритим вихідним кодом для автоматизації процесу міграції ABAP-коду при переході на нові версії ERP-систем від компанії SAP SE. За результатами багатокритеріального ранжування було обрано три моделі: Qwen 2.5 Coder 14B, DeepSeek-Coder-V2 16B і Llama 3.1 8B, які мають пройти комплексне тестування з метою остаточного визначення моделі для подальшого навчання.

Наукова новизна дослідження полягає в розробці методу вибору LLM для задач міграції користувацького коду, що відрізняється від існуючих системою критеріїв вибору LLM та застосуванням гібридного підходу AHP-TOPSIS.

Практична цінність дослідження полягає в тому, що розроблений метод забезпечує систематизований вибір LLM для задач міграції коду, зважаючи на критичні корпоративні обмеження та високі вимоги інформаційної безпеки. Це безпосередньо призводить до підвищення рівня автоматизації міграції ABAP-коду при переході на нові версії ERP-систем SAP SE. Крім того, методологія вибору LLM є трансферабельною та може бути адаптована для модернізації інших складних комп’ютерних систем, незалежно від використання ними відкритих або пропрієтарних мов програмування.

Подальші дослідження будуть зосереджені на декількох ключових напрямках:

– перенавчання обраної моделі на мові ABAP та шаблонах міграції;

– побудова надбудови RAG для покращення функціональних можливостей системи;

– інтеграція моделі в ландшафт SAP, включаючи такі компоненти, як SAP ATC, SAP ADT, SAP Custom Code Migration, а також інструменти з відкритим вихідним кодом ABAPLint та ABAPOpenChecks;

– удосконалення методики тонкого налаштування для підвищення ефективності всього процесу міграції.

Список літератури

1. SAP S/4HANA cloud is a leader in 2024 Gartner® magic quadrant™ for cloud ERP for service-centric enterprises and 2024 Gartner® magic quadrant™ for cloud ERP for product-centric enterprises. URL: https://news.sap.com/2024/11/sap-a-leader-2024-gartner-magic-quadrant-cloud-erp-for-service-centric-enterprises-product-centric-enterprises/ (дата звернення: 17.07.2025).
2. Hardy P. Migrating custom code to SAP S/4HANA. Boston: Rheinwerk Publishing Inc, 2020. P. 333. ISBN 978-1-4932-1994-0.
3. Поздняков О. А., Пархоменко А. В. Міграція користувацького програмного забезпечення при інтелектуальному реінжинірингу складних комп'ютерних систем. *Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: XII Міжнародна НПК*, Запоріжжя, 10-12 грудня 2024: тези доповідей. Запоріжжя: НУЗП, 2024. С. 493–495.
4. Бушуєв Д. А., Лобок Є. А. Систематичний підхід у створенні інноваційних проєктів і продуктів на основі застосувань штучного інтелекту. *Управління розвитком складних систем*. Київ, 2025. № 61. С. 35 – 41, dx.doi.org\ 10.32347/2412-9933.2025.61.35-41.
5. Migrating code at scale with LLMs at Google / Ziftci C. el at. 2025. 12p. (Preprint. arXiv.2504.09691). URL: https://doi.org/10.48550/arXiv.2504.09691 (дата звернення: 17.07.2025).
6. CODEMENV: Benchmarking large language models on code migration. Cheng K. el at. 2025. 25p. (Preprint. arXiv.2506.00894) URL: https://doi.org/10.48550/arXiv.2506.00894 (дата звернення: 17.07.2025).
7. Amazon Q developer just reached a $260 million dollar milestone | AWS DevOps & developer productivity blog. URL: https://aws.amazon.com/blogs/devops/amazon-q-developer-just-reached-a-260-million-dollar-milestone/ (дата звернення: 17.07.2025).
8. Almeida A., Xavier L., Valente M. T. Automatic library migration using large language models: first results. Barcelona Spain: ACM, 2024. P. 427-433. DOI: 10.1145/3674805.3690746.
9. Automatic Qiskit code refactoring using large language models / Suárez J. M. el at. 2025. 7p. (Preprint. arXiv.2506.14535) URL: https://doi.org/10.48550/arXiv.2506.14535 (дата звернення: 17.07.2025).
10. SAP build's next leap: generative AI and ABAP. URL: https://news.sap.com/2024/10/sap-build-generative-ai-abap/ (дата звернення: 17.07.2025).
11. Is chatGPT safe for business? 8 security risks & compliance guide 2025. URL: https://www.metomic.io/resource-centre/is-chatgpt-a-security-risk-to-your-business (дата звернення: 17.07.2025).
12. Don't stop pretraining: adapt language models to domains and tasks / Gururangan S. el at. 2020. 19p. (Preprint. arXiv.2004.10964) URL: https://doi.org/10.48550/arXiv.2004.10964 (дата звернення: 17.07.2025).
13. How much does GPT-4 cost? URL: https://help.openai.com/en/articles/7127956-how-much-does-gpt-4-cost (дата звернення: 17.07.2025).
14. Omidvar Tehrani B., Ishaani M., Anubhai A. Evaluating human-AI partnership for LLM-based code migration. Honolulu HI USA: ACM, 2024. P. 1-8. DOI: 10.1145/3613905.3650896.
15. A survey on large language model (LLM) security and privacy: the good, the bad, and the ugly / Yao Y. el at. *High-Confidence Computing*. 2024. Vol. 4, № 2. P. 100211. DOI: 10.1016/j.hcc.2024.100211.
16. Evaluating large language models trained on code / Chen M. el at. 2021. 8p. (Preprint. (Preprint. arXiv.2107.03374) URL: https://doi.org/10.48550/arXiv.2107.03374 (дата звернення: 17.07.2025).
17. Lang J., Guo Z., Huang S. A comprehensive study on quantization techniques for large language models. 2024. 8p. (Preprint. arXiv.2411.02530) URL: https://doi.org/10.48550/arXiv.2411.02530 (дата звернення: 17.07.2025).
18. Wang L., Chen S., Jiang L., Pan S., Cai R., Yang S., Yang F. Parameter-efficient fine-tuning in large models: a survey of methodologies. 2025. 35p. (Preprint. arXiv.2410.19878) URL: https://doi.org/10.48550/arXiv.2410.19878 (дата звернення: 17.07.2025).
19. QwenLM/Qwen2.5-Coder. URL: https://github.com/QwenLM/Qwen2.5-Coder (дата звернення: 17.07.2025).
20. Deepseek-ai/DeepSeek-Coder-V2. URL: https://github.com/deepseek-ai/DeepSeek-Coder-V2 (дата звернення: 17.07.2025).
21. Gemma 3. Google DeepMind. URL: https://deepmind.google/models/gemma/gemma-3/ (дата звернення: 17.07.2025).
22. meta-llama/codellama. URL: https://github.com/meta-llama/codellama (дата звернення: 17.07.2025).
23. Llama 3.1| Meta Llama. URL: https://www.llama.com/llama3_1/ (дата звернення: 17.07.2025).
24. Mistral NeMo | Mistral AI. URL: https://mistral.ai/news/mistral-nemo (дата звернення: 17.07.2025).
25. microsoft/Phi-3-medium-128k-instruct URL: https://huggingface.co/microsoft/Phi-3-medium-128k-instruct (дата звернення: 17.07.2025).
26. Edwards W., Barron F. H. SMARTS and SMARTER: improved simple methods for multiattribute utility measurement. *Organizational behavior and human decision processes*. 1994. Vol. 60, № 3. P. 306–325. DOI: 10.1006/obhd.1994.1087.
27. Mateo J. R. S. C. Weighted sum method and weighted product method. *Multi criteria analysis in the renewable energy industry*. London: Springer, 2012. P. 19–22. ISBN 978-1-4471-2345-3.
28. Saaty T. L., Vargas L. G. Models, methods, concepts & applications of the analytic hierarchy process. New York: Springer, 2012. P.346. ISBN 978-1-4614-3597-6.
29. Pandey V., Komal, Dincer H. A review on TOPSIS method and its extensions for different applications with recent development. *Soft Computing*. 2023. Vol. 27, 23. P. 18011–18039. DOI: 10.1007/s00500-023-09011-0.
30. Sivalingam C., Subramaniam S. K. Cobot selection using hybrid AHP-TOPSIS based multi-criteria decision making technique for fuel filter assembly process. Heliyon. 2024. Vol. 10, № 4. DOI: 10.1016/j.heliyon.2024.e26374.

Стаття надійшла до редколегії 08.09.2025

Pozdnyakov Oleg
Postgraduate Student of Software Tools Department,
https://orcid.org/0009-0006-3955-802X
National University "Zaporizhzhia Polytechnic", Zaporizhzhia
Parkhomenko Anzhelika
PhD (Eng.), associate professor of Software Tools Department,
https://orcid.org/0000-0002-6008-1610
National University "Zaporizhzhia Polytechnic", Zaporizhzhia

RESEARCH AND SELECTION OF LARGE LEARNING MODES FOR AUTOMATION OF ABAP-CODE MIGRATION

Abstract. *The paper studies the problem of software code migration during the transition from the old version of the SAP ERP 6.0 system to modern ERP platforms, in particular SAP S/4HANA. The necessity of automation of the migration process using artificial intelligence methods and tools is substantiated. A comprehensive analysis of existing academic and industrial case studies confirmed the feasibility of using large language models for code migration. A number of advantages of the approach based on large language models are revealed, in particular, the ability to handle various migration scenarios. The key risks and limitations common in the enterprise environment are highlighted, including data security, privacy, high cost of API calls and limited knowledge of models regarding the subject area. A method is developed for selecting a large open source language model that can be safely used in the enterprise landscape, thereby mitigating the risks associated with cloud solutions. The method is based on the developed system of criteria, including: model size, context window size, license type, support for fine-tuning training, focus on code quality (measured using a reference dataset), support for model quantization and community support. The hybrid AHP-TOPSIS method is proposed for systematic ranking of candidate models. The AHP method was used to check the weights of the criteria, while TOPSIS was used to rank the models based on their proximity to the ideal solution. Based on the developed method, three large language models were selected: Qwen 2.5 Coder 14B, DeepSeek-Coder-V2 16B and Llama 3.1 8B. A comprehensive testing plan for the selected models was developed for further experimental studies. The scientific novelty of the work lies in the development of a method for selecting large language models for custom code migration tasks, which differs from the existing system of model selection criteria and the use of a hybrid AHP-TOPSIS approach for ranking candidate models. The practical value of the work is that the developed method allows you to select a model taking into account the constraints of the corporate environment and information security requirements, which will increase the level of automation in the migration of AVAP when switching to new versions of ERP systems from SAP SE. The developed method can be applied to the selection of large language models when upgrading any large computer systems developed in both open source and proprietary programming languages.*

Keywords: ERP system; migration of ABAP-code; large language model; selection criteria; method AHP-TOPSIS

References

1. SAP S/4HANA cloud is a leader in 2024 gartner® magic quadrant™ for cloud ERP for service-centric enterprises and 2024 gartner® magic quadrant™ for cloud ERP for product-centric enterprises. (2024). https://news.sap.com/2024/11/sap-a-leader-2024-gartner-magic-quadrant-cloud-erp-for-service-centric-enterprises-product-centric-enterprises/).
2. Hardy, P. (2020). *Migrating custom code to SAP S/4HANA*. Rheinwerk Publishing Inc. 978-1-4932-1994-0.
3. Pozdnyakov, O. A., & Parkhomenko, A. V. (2024). Custom code migration in intelligent reengineering of complex computer systems. *Modern problems and achievements in radio engineering, telecommunications and information technologies: XII International Scientific and Practical Conference*, Zaporizhzhia, December 10-12, 2024: Abstracts (pp. 493–495). NUZP.
4. Bushuiev, D. A., & Lobok, Y. A. (2025). A systematic approach to creating innovative projects and products based on artificial intelligence applications. *Management of Development of Complex Systems*, 61, 35–41. https://dx.doi.org/10.32347/2412-9933.2025.61.35-41.
5. Ziftci, C., Nikolov, S., Sjövall, A., Kim, B., Codecasa, D., & Kim, M. (2025). *Migrating code at scale with LLMs at Google*. ArXiv. https://doi.org/10.48550/arXiv.2504.09691.
6. Cheng, K., Shen, X., Yang, Y., Wang, T., Cao, Y., Ali, M. A., Wang, H., Hu, L., & Wang, D. (2025). *CODEMENV: Benchmarking large language models on code migration*. ArXiv. https://doi.org/10.48550/arXiv.2506.00894.
7. Amazon Web Services. (2024, August 2). Amazon Q Developer just reached a $260 million dollar milestone. https://aws.amazon.com/blogs/devops/amazon-q-developer-just-reached-a-260-million-dollar-milestone/.
8. Almeida, A., Xavier, L., & Valente, M. T. (2024). Automatic library migration using large language models: First results. *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '24)* (pp. 427-433). Association for Computing Machinery. https://doi.org/10.1145/3674805.3690746.
9. Suárez, J. M., Bibbó, L. M., Bogado, J., & Fernandez, A. (2025). *Automatic Qiskit code refactoring using large language models*. ArXiv. https://doi.org/10.48550/arXiv.2506.14535.
10. Chirapurath, J. G. (2024). SAP build's next leap: generative AI and ABAP. https://news.sap.com/2024/10/sap-build-generative-ai-abap/.
11. Metomic. (n.d.). Is ChatGPT a security risk to your business? Metomic. https://www.metomic.io/resource-centre/is-chatgpt-a-security-risk-to-your-business.

12. Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., & Smith, N. A. (2020). *Don't stop pretraining: adapt language models to domains and tasks*. ArXiv. https://doi.org/10.48550/arXiv.2004.10964.

13. OpenAI. (n.d.). How much does GPT-4 cost? https://help.openai.com/en/articles/7127956-how-much-does-gpt-4-cost.

14. Omidvar Tehrani, B., Ishaani, M., & Anubhai, A. (2024). Evaluating human-AI partnership for LLM-based code migration. *Extended abstracts of the CHI Conference on Human Factors in Computing Systems* (pp. 1–8). ACM. https://doi.org/10.1145/3613905.3650896.

15. Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., & Zhang, Y. (2024). A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4 (2), 100211. https://doi.org/10.1016/j.hcc.2024.100211.

16. Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde de Oliveira Pinto, H., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Zaremba, W. (2021). *Evaluating large language models trained on code*. ArXiv. https://doi.org/10.48550/arXiv.2107.03374.

17. Lang, J., Guo, Z., & Huang, S. (2024). *A comprehensive study on quantization techniques for large language models*. ArXiv. https://doi.org/10.48550/arXiv.2411.02530.

18. Wang, L., Chen, S., Jiang, L., Pan, S., Cai, R., Yang, S., & Yang, F. (2024). *Parameter-efficient fine-tuning in large models: A survey of methodologies*. ArXiv. https://doi.org/10.48550/arXiv.2410.19878.

19. QwenLM. (n.d.). Qwen2.5-Coder [Source code]. GitHub. https://github.com/QwenLM/Qwen2.5-Coder.

20. Deepseek-ai. (n.d.). DeepSeek-Coder-V2 [Source code]. GitHub. https://github.com/deepseek-ai/DeepSeek-Coder-V2.

21. DeepMind. (n.d.).). Gemma 3 [Model card]. DeepMind. https://deepmind.google/models/gemma/gemma-3/.

22. Meta. (n.d.). CodeLlama [Source code]. GitHub. https://github.com/meta-llama/codellama.

23. Meta. (n.d.). LLaMA 3.1 [Model page]. https://www.llama.com/llama3_1/.

24. Mistral AI. (n.d.). Mistral Nemo [News page]. https://mistral.ai/news/mistral-nemo.

25. Microsoft. (n.d.). Phi-3 medium 128k instruct [Model card]. Hugging Face. https://huggingface.co/microsoft/Phi-3-medium-128k-instruct.

26. Edwards, W., & Barron, F. H. (1994). SMARTs and SMARTER: Improved simple methods for multiattribute utility measurement. *Organizational Behavior and Human Decision Processes*, 60(3), 306–325. https://doi.org/10.1006/obhd.1994.1087.

27. Mateo, J. R. S. C. (2012). Weighted sum method and weighted product method. *Multi criteria analysis in the renewable energy industry* (pp. 19–22). Springer London. https://doi.org/10.1007/978-1-4471-2346-0_4.

28. Saaty, T. L., & Vargas, L. G. (2012). *Models, methods, concepts & applications of the analytic hierarchy process* (2nd ed.). Springer.

29. Pandey, V., Komal, & Dincer, H. (2023). A review on TOPSIS method and its extensions for different applications with recent development. *Soft Computing*, 27 (23), 18011–18039. https://doi.org/10.1007/s00500-023-09011-0.

30. Sivalingam, C., & Subramaniam, S. K. (2024). Cobot selection using hybrid AHP-TOPSIS based multi-criteria decision making technique for fuel filter assembly process. *Heliyon*, 10(4), Article e26374. https://doi.org/10.1016/j.heliyon.2024.e26374.

Посилання на публікацію

APA Pozdnyakov, O., & Parkhomenko, A. (2025). Research and selection of Large Learning Models for automation of ABAP-code migration. *Management of Development of Complex Systems*, 63, 191–200, dx.doi.org\10.32347/2412-9933.2025.63.191-200.

ДСТУ Поздняков О. А., Пархоменко А. В. Дослідження та вибір великих мовних моделей для автоматизації міграції ABAP-коду. *Управління розвитком складних систем.* Київ, 2025. № 63. С. 191 – 200, dx.doi.org\10.32347/2412-9933.2025.63.191-200.