



РОЗРОБЛЕННЯ МЕТОДУ ВЕРИФІКАЦІЇ РЕЗУЛЬТАТІВ МІГРАЦІЇ УСПАДКОВАНОГО КОРИСТУВАЦЬКОГО АВАР-КОДУ В МУЛЬТИАГЕНТНІЙ СИСТЕМІ ІНТЕЛЕКТУАЛЬНОГО РЕІНЖІНІРИНГУ

Одним із актуальних завдань сучасної ІТ-галузі є інтелектуальний реінжиніринг комп'ютерних систем при переході на нові технологічні платформи. Встановлено, що особливістю переходу з SAP ERP 6.0 на SAP S/4HANA є потреба у міграції успадкованого користувацького АВАР-коду, яка становить до 40 % від загального обсягу проєктних робіт та вимагає автоматизації цього процесу. З'ясовано, що застосування великих мовних моделей у закритому корпоративному середовищі надає широкі можливості щодо автоматизованої трансформації коду, але потребує забезпечення якості мігрованого коду без залучення додаткових зовнішніх сервісів і фахівців. Виявлено два системних обмеження розробленої раніше базової одноагентної архітектури системи інтелектуального реінжинірингу для автоматизованої міграції коду: накопичення помилок для семантично складних перетворень та неможливість верифікації функціональної еквівалентності для 25–40 % фрагментів без тестового покриття. Для усунення зазначених обмежень розроблено архітектуру мультіагентної системи інтелектуального реінжинірингу на основі формалізації завдання міграції користувацького АВАР-коду як марківського процесу прийняття рішень. При цьому інтегральну метрику оцінювання якості мігрованого АВАР-коду (IAQS) розширено від бінарного критерію зупинки ітеративного циклу трансформації коду до потрібної керівної функції на основі адаптації великої мовної моделі-розробника, управління ітеративним циклом та формування структурованого діагностичного зворотного зв'язку, що забезпечило цільову трансформацію коду замість повного його регенерування. Удосконалено метод верифікації результатів міграції користувацького коду, що реалізує триетапну послідовну перевірку якості мігрованого коду за принципом ранньої зупинки (статичний аналіз, семантична валідація методом зворотного перекладу та ранжування за категоріями довіри), що формує замкнений контур якості коду без додаткового залучення фахівців. Оцінено вплив кількості агентів і конфігурацій системи на інтегральну метрику оцінювання якості мігрованого АВАР-коду. Абляційне дослідження на 300 реальних АВАР-фрагментах підтвердило, що запропонована конфігурація мультіагентної системи забезпечує поріг прийнятності якості мігрованого коду, досягаючи $IAQS = 0,845^{±0,031}$ проти $0,718^{±0,047}$ для базової одноагентної архітектури. Встановлено практичну ефективність запропонованих рішень за результатами промислової апробації на 3200 АВАР-об'єктах у межах гібридної стратегії розгортання системи інтелектуального реінжинірингу, що загалом відповідає рівню автоматизації процесу міграції коду на рівні 86,5 % та скороченню трудовитрат приблизно на 80 %. Доведено, що запропоновані рішення придатні до узагальнення на інші ERP-платформи.

Ключові слова: реінжиніринг комп'ютерної системи; генеративний штучний інтелект; велика мовна модель; марківський процес прийняття рішень; інтегральна метрика оцінювання якості коду.

Вступ / Introduction

Постійне вдосконалення корпоративних інформаційних комп'ютерних систем (КС) є ключовим чинником підвищення конкурентоспроможності сучасних підприємств [7]. Особливого значення набуває інтелектуальний реінжиніринг ERP-систем (англ. *Enterprise Resource Planning* – система планування ресурсів підприємства), зокрема – перехід з платформи SAP ERP 6.0 на SAP S/4HANA, зумовлений припиненням підтримки SAP ERP 6.0 у грудні 2027 р. Встановлено, що міграція успадкованого користувацького АВАР-коду становить до 40 % від загального обсягу робіт такого проєкту та є

його найтрудомісткішою складовою [7]. При цьому пропрієтарна мова АВАР практично відсутня у публічних навчальних корпусах загальнодоступних великих мовних моделей (ВММ), а передавання корпоративного коду хмарним сервісам заборонено ліцензійними угодами SAP (абревіатура від *System Applications and Products in Data Processing*).

Застосування генеративного штучного інтелекту (ШІ) та ВММ для автоматизації завдань генерування та трансформації наявного програмного коду досліджують у сучасній літературі [2, 23], проте наявні підходи до розроблення програмного забезпечення орієнтуються

© Copyright 2026 by the author(s)

Поздняков Олег Анатолійович, аспірант, кафедра програмних засобів.

Email: oleg.pozdnyakov.ua@gmail.com; <https://orcid.org/0009-0006-3955-802X>

Пархоменко Анжеліка Володимирівна, канд. техн. наук, доцент, кафедра програмних засобів.

Email: parkhomenko.anzhelika@gmail.com; <https://orcid.org/0000-0002-6008-1610>

Цитування за ДСТУ: Поздняков О. А., Пархоменко А. В. Розроблення методу верифікації результатів міграції успадкованого користувацького АВАР-коду в мультіагентній системі інтелектуального реінжинірингу. Науковий вісник НЛТУ України. 2026, т. 36, № 4. С. 88–97.

Citation APA: Pozdnyakov, O. A., & Parkhomenko, A. V. (2026). Development of a method for verification of the results of legacy user ABAP code migration in a multiagent system of intelligent reengineering. *Scientific Bulletin of UNFU*, 36(4), 88–97.

<https://doi.org/10.36930/40360410>

переважно на генерування нового коду із природно-мовного опису. При цьому перспективним напрямом дослідження є застосування мультиагентних систем (МАС), у яких рольова спеціалізація агентів істотно підвищує якість очікуваних результатів порівняно з неструктурованою взаємодією [9], але наявні фреймворки не адаптовані до завдання міграції наявного коду зі збереженням функціональної еквівалентності. Завдання міграції коду принципово відрізняється від генерування нового коду, оскільки вихідний код вже існує, реалізує задокументовану бізнес-логіку й зазнав декількох років виробничої експлуатації, а збереження його функціональної еквівалентності при адаптації до обмежень цільової платформи є вимогою, відсутньою у наявних фреймворках [1, 6, 9, 19, 22].

Практична апробація базової архітектури системи інтелектуального реінжинірингу [17] виявила два системних обмеження: накопичення помилок для семантично складних перетворень коду через конкуренцію підзадач за контекстне вікно, через що базова конфігурація одноагентної архітектури не досягала встановленого порогу прийнятності інтегральної метрики оцінювання якості мігрованого АВАР-коду ($IAQS = 0,76$), а також неможливість верифікації функціональної еквівалентності для 25-40 % промислових фрагментів коду без тестового покриття [11, 12, 17]. Тому тема дослідження з розроблення архітектури МАС та методу верифікації результатів міграції успадкованого користувацького АВАР-коду є актуальною.

У роботі вирішується актуальне науково-прикладне завдання з галузі комп'ютерних наук: подальший розвиток моделі архітектури МАС та удосконалення методу верифікації результатів автоматизованої міграції успадкованого користувацького АВАР-коду за інтелектуального реінжинірингу корпоративних КС класу SAP ERP для забезпечення якості мігрованого коду та збереження функціональної еквівалентності у закритому корпоративному середовищі.

Об'єкт дослідження – верифікація результатів міграції успадкованого користувацького АВАР-коду.

Предмет дослідження – методи і засоби верифікації результатів міграції успадкованого користувацького АВАР-коду, що базується на архітектурі МАС, які забезпечать формальну основу для керованого ітеративного циклу трансформації програмного коду з гарантованою умовою зупинки.

Мета роботи – розробити архітектуру мультиагентної системи інтелектуального реінжинірингу КС на підставі удосконаленого методу верифікації результатів міграції успадкованого користувацького АВАР-коду, що дасть змогу верифікувати його функціональну еквівалентність без тестового покриття та без залучення зовнішніх фахівців.

Для досягнення зазначеної мети визначено такі основні завдання дослідження:

- проаналізувати обмеження наявних мультиагентних підходів до автоматизованої трансформації коду та методів верифікації їх семантичної еквівалентності, що дасть змогу обґрунтувати доцільність розроблення нових архітектурних рішень та засобів верифікації;
- формалізувати завдання міграції успадкованого коду як марківський процес прийняття рішень та розробити архітектуру МАС з трьома спеціалізованими агентами, що забезпечить формальну основу для керованого ітеративного циклу трансформації коду з гарантованою умовою зупинки;

- удосконалити метод верифікації результатів міграції успадкованого коду шляхом інтеграції методу зворотного перекладу як незалежного засобу оцінювання його семантичної еквівалентності, що дасть можливість верифікувати його функціональну еквівалентність без тестового покриття та без залучення зовнішніх фахівців;
- здійснити абляційне дослідження на реальних промислових даних для кількісного підтвердження ефективності кожного архітектурного компонента, що дасть змогу кількісно обґрунтувати внесок кожного з них та визначити оптимальну конфігурацію МАС;
- виконати промислову апробацію запропонованих рішень у межах реального проєкту міграції SAP ERP 6.0 → S/4HANA 2023, щоб підтвердити практичну ефективність запропонованих рішень та їх придатність до реальних промислових кодових баз.

Аналіз останніх досліджень та публікацій. Аналіз сучасної наукової літератури свідчить про зростання інтересу дослідників до автоматизації роботи з програмним кодом засобами генеративного штучного інтелекту – зокрема до генерування коду, мультиагентних підходів та верифікації семантичної еквівалентності. Водночас зазначені напрями здебільшого розвиваються незалежно один від одного, без урахування специфіки трансформації успадкованого коду за умов закритого корпоративного середовища. Зокрема, сучасні дослідження у галузі автоматизації роботи з програмним кодом засобами BMM [3, 21, 23] орієнтовані переважно на генерування нового коду із природно-мовного опису.

У роботах [3, 23] запропоновано стандартизовані бенчмарки HumanEval та CodeGeeX для оцінювання якості BMM у завданнях генерування коду, що підтвердили здатність сучасних BMM до генерування синтаксично коректного коду загальних мов програмування, проте виявили істотне зниження точності для доменно-специфічних задач з обмеженим представленням у публічних корпусах. У роботі [21] показано перспективність мультиагентної координації для вирішення завдань, де жоден окремий агент не має достатньої компетентності. Для АВАР-коду, практично відсутнього у відкритих корпусах, загальні бенчмарки оцінювання якості генерування не відображають реальної якості трансформації, що зумовлює потребу у спеціалізованій метриці та доменному донавчанні BMM.

У роботі [9] запропоновано мультиагентний фреймворк MetaGPT на основі рольової спеціалізації агентів та стандартизованого обміну артефактами. Автори показали, що рольова спеціалізація істотно підвищує якість результатів порівняно з неструктурованою взаємодією. Разом з тим, систему орієнтовано на генерування нового коду із природно-мовного опису: у ній відсутні механізми роботи з вихідним кодом, формалізована метрика функціональної еквівалентності та підтримка її локального розгортання.

Комерційні агентні інструменти [1, 6] для оркестрації завдань розроблення ПЗ демонструють ефективність агентного підходу до вирішення складних багатокрокових завдань. Проте ці рішення функціонують винятково через хмарні API (англ. *Application Programming Interface*), що суперечить ліцензійним вимогам SAP щодо корпоративного коду, і не підтримують доменно-специфічного планування на основі корпоративної документації. Це підтверджує потребу у власній архітектурі з локальним розгортанням.

У роботах [11, 12] розглянуто класичні підходи до верифікації семантичної еквівалентності програмного

коду: формальна верифікація засобами SMT (англ. *Satisfiability Modulo Theories*) розв'язувачів та символічне виконання. Ці методи забезпечують математично строгі гарантії, проте потребують формальних семантичних моделей мови програмування, яких для ABAP практично не існує, та погано масштабуються для промислових кодових баз. У роботі [20] запропоновано метрику CodeBLEU, адаптовану для оцінювання якості згенерованого коду через порівняння AST (англ. *Abstract Syntax Tree*) – піддерев. Виявлена непридатність формальних методів для ABAP зумовлює потребу в альтернативному підході до верифікації – на основі статистичної оцінки семантичної еквівалентності, що реалізовано у запропонованому методі верифікації через метрику CodeBLEU та BT-score.

Зокрема, у роботі [5] застосовано метод зворотного перекладу для аугментації навчальних даних у завданнях машинного перекладу, що підтвердило властивість обертності семантично коректних перетворень. Автори показали, що масштабування методу до 226 млн речень покращує якість перекладу на 4,8 BLEU-пунктів. Основний внесок цієї праці у цьому дослідженні – концептуальна основа для адаптації методу зворотного перекладу до завдання верифікації еквівалентності ABAP-коду.

Отже, проведений аналіз останніх досліджень та публікацій показав, що наявні підходи до автоматизованої міграції програмного коду розглядають здебільшого ізольовано: мультиагентні фреймворки орієнтовані на генерування нового коду; методи формальної верифікації не придатні для ABAP через відсутність формальних семантичних моделей; метод зворотного перекладу не застосовувався для верифікації еквівалентності коду та не інтегрований з механізмом маршрутизації за категоріями довіри.

Виявлена невирішена наукова проблема полягає у відсутності архітектурного рішення, яке одночасно забезпечує формалізований критерій зупинки ітеративного циклу трансформації з контролем якості мігрованого коду, верифікацією функціональної еквівалентності, доменне планування на основі корпоративної документації та локальне розгортання адаптованої BMM без передавання корпоративного коду хмарним сервісам. Тому тема дослідження з розроблення моделі архітектури мультиагентної системи та методу верифікації результатів міграції успадкованого користувацького ABAP-коду є актуальною.

Матеріали та методи дослідження. У дослідженні застосовано такі методи та засоби: теорія марківських процесів прийняття рішень – для формалізації ітеративного завдання міграції коду; методи QLoRA (англ. *Quantized Low-Rank Adaptation* – квантований метод низькорангової адаптації) та ORPO (англ. *Odds Ratio Preference Optimization* – метод вирівнювання за перевагами на основі відношення шансів) – для параметро-ефективної адаптації великих мовних моделей до доменно-специфічного корпусу ABAP; метод зворотного перекладу, адаптований з NLP (англ. *Natural Language Processing* – оброблення природної мови) – для верифікації семантичної еквівалентності мігрованого коду за умов відсутності тестового покриття; інструменти статичного аналізу ABAP-коду (abaplint, SAP ATC – SAP ABAP Test Cockpit з варіантом S4HANA_READINESS) – для синтаксичної та структурної верифікації; метрика CodeBLEU (англ. *Code Bilingual Evaluation Understudy*) – для

кількісного оцінювання семантичної подібності програмних артефактів; методи математичної статистики (парний *t*-критерій, три незалежні запуски, $p < 0,01$) – для підтвердження достовірності результатів.

Результати дослідження та їх обговорення / Research results and their discussion

Метрика оцінювання якості та архітектура MAC. Цільову функцію якості мігрованого коду – метрику IAQS (англ. *Integral ABAP Quality Score*) [15], що поєднує три нормовані показники якості коду відповідно до характеристик стандарту ISO/IEC 25010 [10], у зваженій лінійній комбінації визначають як:

$$IAQS = 0,3 \cdot M_{syn} + 0,4 \cdot M_{func} + 0,3 \cdot M_{sem}, \quad (1)$$

де: M_{syn} – синтаксична коректність (abaplint); M_{func} – функціональна відповідність (pass@1); M_{sem} – семантична еквівалентність (CodeBLEU). Підвищена вага $M_{func} = 0,4$ обґрунтована пріоритетністю збереження бізнес-логіки [15]. Порогове значення $\delta = 0,76$ встановлено на основі апробації базової архітектури: фрагменти з IAQS $\geq 0,76$ підтверджуються коректними у 94 % випадків з ручної верифікації.

Для усунення виявлених обмежень базової архітектури [7] розроблено архітектуру мультиагентної системи (MAC) інтелектуального реінжинірингу КС [16]. Архітектура розробленої MAC (рис. 1) містить чотири взаємопов'язані підсистеми. Підсистема адаптації BMM реалізує триетапну підготовку моделі-розробника: безперервне попереднє навчання на доменному корпусі ABAP – CPT (Continual Pre-Training), параметро-ефективне донавчання методом QLoRA [4] та вирівнювання за перевагами методом ORPO [4 з використанням IAQS як автоматичного сигналу переваги замість ручного анутовування. Результатом є адаптована модель Qwen 2.5 Coder 14B, придатна до її локального розгортання у корпоративному середовищі.

Підсистема мультиагентного конвеєра реалізує ітеративний цикл трансформації: агент-планувальник виконує аналіз AST вихідного фрагмента та формує покроковий план на основі доповненого генерування із залученням зовнішніх знань RAG (англ. *Retrieval-Augmented Generation*) із бази SAP Notes, що містить настанови з усунення несумісностей при переході на SAP S/4HANA; агент-розробник виконує трансформацію коду; модуль IAQS обчислює значення інтегральної метрики оцінювання якості мігрованого коду та порівнює його з порогом δ ; агент-рецензент на основі результату модуля IAQS формує структурований діагностичний зворотний зв'язок із прив'язкою до конкретних рядків коду. Підсистема верифікації реалізує триетапний каскад з принципом ранньої зупинки: статичний аналіз (abaplint, SAP ATC), семантична валідація методом зворотного перекладу (BT-score) та ранжування за інтегральним показником довіри з маршрутизацією до категорій А, В або С. Замкнений контур якості коду автоматично передає виправлені фрагменти категорії С до корпусів QLoRA та ORPO, забезпечуючи безперервне вдосконалення адаптованої BMM між проектами без залучення фахівців.

Формалізацію завдання трансформації як марківського процесу прийняття рішень MDP [18], так і ключові конструктивні рішення MAC викладено нижче. Для побудови ефективної системи керування на основі навчання з підкріпленням RL (англ. *Reinforcement Learning*)

ning) завдання трансформації представляють як Марківський процес прийняття рішень (MDP), а архітекту-

ру інформаційної системи реалізують у вигляді мульти-агентної системи (MAC).

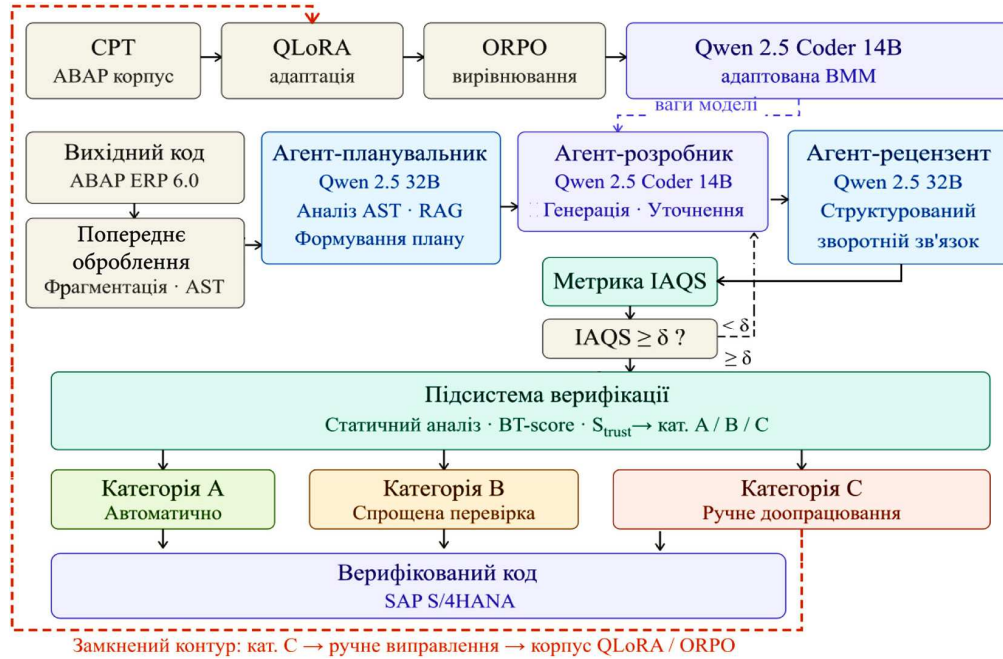


Рис. 1. Архітектура мультиагентної системи інтелектуального реінжинірингу КС / Architecture of the multi-agent system for intelligent reengineering of computer systems

Стан MAC у момент часу t характеризується таким вектором ознак:

$$V_{st} = \langle M_{syn}, M_{func}, M_{sem}, C, \bar{H} \rangle, \quad (2)$$

де: M_{syn} – синтаксична коректність коду (abaplint); M_{func} – функціональна відповідність (pass@1); M_{sem} – семантична еквівалентність (CodeBLEU); C – цикломатична складність вихідного фрагмента; $\bar{H} = [h_j, j = \overline{1, n}]$ – вектор попередніх дій (history ітеративного циклу). Початковий стан s_0 відповідає вихідному коду C_{old} ; цільова множина $S^* = \{s_t \mid IAQS(s_t) \geq \delta\}$ визначає умову прийняття. Стан мультиагентної системи в контексті децентралізованого або частково спостережуваного Марківського процесу прийняття рішень (Dec-POMDP/MMDP) може розглядатись на двох рівнях: глобальному та локальному.

Область значень компонентів вектора: $M_{syn} \in \{0, 1\}$ – бінарний показник синтаксичної коректності; $M_{func} \in [0, 1]$ – частка успішно виконаних тестових випадків (pass@1); $M_{sem} \in [0, 1]$ – семантична подібність за CodeBLEU. Введення C і H до вектора стану зумовлено їхньою специфічною роллю в управлінні ітеративним циклом: цикломатична складність C визначає рівень складності фрагмента (L1: $C \leq 10$; L2: $10 < C \leq 25$; L3: $C > 25$), який обирає агент-планувальник для формування стратегії доменного планування; вектор дій $H = (a_1, \dots, a_{t-1}) \in A^*$ кодує послідовність усіх попередніх ітерацій і дає змогу агенту-рецензенту враховувати, які виправлення вже застосовувались без досягнення δ .

Початковий стан MAC до першої ітерації визначають як: $s_0 = \langle 0, 0, 0, C(C_{old}), \emptyset \rangle$, де нульові значення метрик відображають відсутність оцінки якості адаптованої BMM до першої трансформації коду; $C(C_{old})$ – складність вихідного фрагмента; $\emptyset$ – порожня історія дій. Ітеративний цикл завершується за виконання умови:

$$S^* = \{s_t \mid IAQS(s_t) \geq \delta \vee t = T\}, \quad (3)$$

де $T = 3$ – встановлений ліміт ітерацій, що запобігає нескінченному циклу для фрагментів, де δ не досягне

за заданих параметрів моделі. Нагадаємо, встановлений ліміт ітерацій, який запобігає нескінченному циклу для фрагментів (snippets) у програмуванні та шаблонізаторах, найчастіше називається лімітом ітерацій циклу (англ. *Loop Iteration Limit*), максимальною глибиною рекурсії (maximum recursion depth) або таймаутом виконання (execution timeout) залежно від конкретного середовища.

Простір дій визначають як:

$$A = A_{plan} \cup A_{code} \cup A_{review}, \quad (4)$$

де: $A_{plan} = \{analyze, retrieve, plan\}$ – дії агента-планувальника; $A_{code} = \{generate, refine\}$ – дії агента-розробника; $A_{review} = \{validate, accept, reject\}$ – дії агента-рецензента. Кожна дія відповідає конкретному виходу агента у структурованому артефакті формату JSON (англ. *JavaScript Object Notation*).

Функцію винагороди [2] визначають як:

$$R(s_t, a_t, s_{t+1}) = \Delta IAQS(s_t, s_{t+1}) + R_{bonus}(s_{t+1}), \forall t \in T, \quad (5)$$

де: $R_{bonus} = \beta \cdot 1[IAQS(s_{t+1}) \geq \delta]$; $\beta = 0,5$. Значення $\beta = 0,5$ обрано так, щоб перетин порогу δ забезпечував істотний стимул, не перетворюючись на єдину ціль оптимізації: система продовжує покращувати якість коду навіть нижче δ , що критично для формування репрезентативних трійок переваг ORPO у замкненому контурі якості мігрованого АВАР-коду.

Марківська властивість MDP забезпечується через структурований JSON-артефакт – спільну структуру даних, що передають між агентами, та містить повний контекст поточного стану міграції успадкованого коду:

- код C_{old} та його AST-подання (входить від агента-планувальника);
- класифікацію рівня складності (L1/L2/L3) та покроковий план міграції успадкованого коду з посиланнями на SAP Notes (виходить від агента-планувальника);
- поточний результат міграції успадкованого коду C_{new} (виходить від агента-розробника);
- вектор діагностики агента-рецензента при $IAQS < \delta$: перелік проблемних рядків, класифікація типів помилок, рекомендації з документації SAP;

- значення IAQS та його компонентів M_{syn} , M_{func} , M_{sem} ; лічильник ітерацій t .

Зберігання повного контексту попередніх ітерацій у JSON-артефакті забезпечує марківську властивість MDP: наступна дія залежить винятково від поточного стану MAC, а не від усієї попередньої траєкторії міграції успадкованого коду, що спрощує навчання та управління агентами.

Принциповою відмінністю від базової архітектури [14] є розширення ролі метрики IAQS від бінарного критерію зупинки до потрібної керівної функції. По-

перше – адаптаційна функція: між проєктами IAQS слугує автоматичним сигналом переваги для ORPO-вирівнювання моделі. По-друге – функція управління ітеративним циклом: ітерації тривають до $IAQS \geq \delta = 0,76$ або $T = 3$ ітерацій. По-третє – функція генерування структурованого діагностичного зворотного зв'язку: агент-рецензент формує перелік конкретних помилок із прив'язкою до рядків коду та рекомендаціями з документації SAP, що дає змогу агенту-розробнику виконувати цільове виправлення (табл. 1).

Табл. 1. Відповідність компонентів архітектури MAC базовій архітектурі та новим функціям / Correspondence of MAS Architecture Components to the Baseline Architecture and New Functions

Агент	Базова архітектура	Нова функція в MAC	Модель BMM
Планувальник	Підсистема попереднього оброблення	Адаптаційна: RAG-запит, класифікація L1/L2/L3, покроковий план	Qwen 2.5 32B Instruct
Розробник	Qwen 2.5 Coder 14B (CPT→QLoRA→ORPO)	Цільове виправлення за діагностичним зворотним зв'язком	Qwen 2.5 Coder 14B (адаптована)
Рецензент	Модуль IAQS (бінарний сигнал зупинки)	Структурований зворотний зв'язок: рядки, типи помилок, рекомендації SAP	Qwen 2.5 32B Instruct

Аналіз результатів абляційного дослідження підтверджує, що ітеративний цикл MAC зі структурованим зворотним зв'язком забезпечує стійку збіжність: зниження стандартного відхилення IAQS від 0,047 (B-1, без ітерацій) до 0,031 (MAC) – на 34 % – свідчить про передбачуваність шляху до прийнятного результату. Для фрагментів рівня L1 ліміт $T = 3$ ітерацій використовується рідко; для L3-фрагментів $T = 2-3$ ітерацій є типовою картиною, оскільки цільове виправлення конкретних рядків потребує кількох уточнень – що відображають у вищому середньому $\Delta IAQS$ за ітерацію для цього рівня складності.

Метод верифікації результатів міграції коду. Метрика IAQS керує внутрішнім ітеративним циклом між агентами, однак не визначає рівня ручного контролю, необхідного перед передаванням результату до продуктивної системи SAP S/4HANA. Для вирішення цього завдання розроблено метод верифікації із трьома послідовними етапами та принципом ранньої зупинки: фрагмент, що не пройшов Етап 1, спрямовується до категорії C без виконання ресурсомістких обчислень Етапу 2 [16].

Етап 1. Статичний аналіз. Статичний аналіз є першим етапом верифікації та виконується без запуску коду у виконавчому середовищі SAP. Він містить три послідовні перевірки з різними інструментами, кожна з яких верифікує окремий аспект якості мігрованого коду: синтаксичну коректність, сумісність з платформою SAP S/4HANA та повноту перетворення (рис. 2).

Блок 1. Початок.

Блок 2. Завантаження мігрованого фрагмента коду C_{new} для подальшої перевірки.

Блок 3. Синтаксична валідація засобами abaplint. Мігрований код C_{new} перевіряється АВАР-парсером на відповідність формальній граматиці мови АВАР. Результат є бінарним: $M_{syn} \in \{0, 1\}$. Синтаксична валідація відповідає обчисленню компонента M_{syn} метрики IAQS (1) та є лінійною за часом відносно обсягу вихідного фрагмента.

Блок 4. Перевірка умови: якщо $M_{syn} = 0$ – синтаксично некоректний код не може бути виконаний у жодному середовищі SAP, тому вихідний фрагмент спрямовується безпосередньо до Блоку 8 (категорія C) без подальших перевірок; якщо $M_{syn} \neq 0$ – виконання переходить до Блоку 5.

Блок 5. Перевірка сумісності з платформою SAP S/4HANA засобами SAP ATC з варіантом S4HANA_READINESS. SAP ATC виявляє три типи проблем: звернення до таблиць та об'єктів, видалених або спрощених у S/4HANA відповідно до Simplification Item Catalog; використання АВАР-конструкцій, несумісних з базою даних SAP HANA; порушення рекомендацій продуктивності для in-memoю обчислень. ATC повертає результати перевірки з пріоритетами від 1 (помилка, що блокує виконання) до 3 (інформаційне попередження). Результат: $P_{ATC} \in \{1, 2, 3\}$.

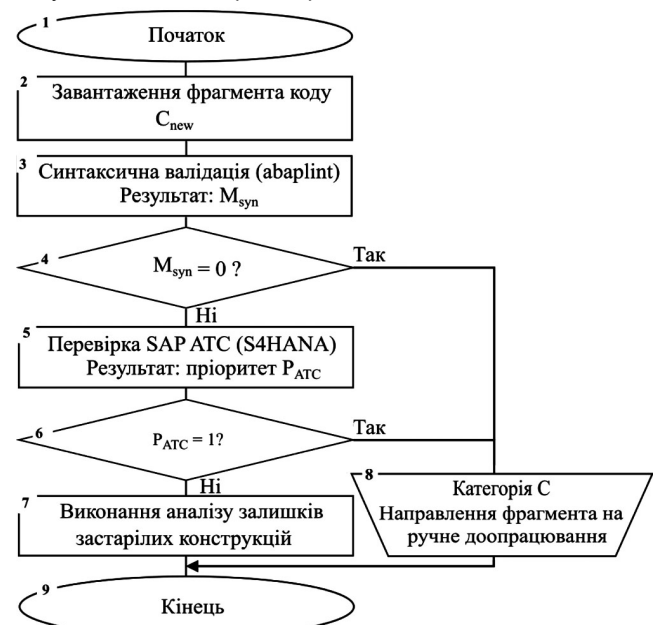


Рис. 2. Схема алгоритму статичного аналізу мігрованого АВАР-коду / Algorithm of static analysis of migrated ABAP code

Блок 6. Перевірка умови: якщо $P_{ATC} = 1$ – критичні помилки несумісності, фрагмент спрямовується до Блоку 8 (категорія C); якщо $P_{ATC} \neq 1$ – результати пріоритетів 2–3 не блокують верифікацію, але враховуються під час ранжування; виконання переходить до Блоку 7.

Блок 7. Аналіз залишків застарілих конструкцій для визначення повноти перетворення ($P_{compl} \in \{0, 1\}$). Застосовується набір регулярних виразів, що ідентифікують типові застарілі шаблони АВАР: оператор MOVE, оголошення заголовкових рядків HEADER LINE, вик-

лики функціональних модулів з каталогу спрощень SAP, прямий доступ до спрощених таблиць (BSEG, BKPF, KONV). Наявність залишків ($P_{compl} = 0$) свідчить про неповне перетворення. Формуються елементи вектора результату першого етапу:

$$V^{(1)}(C_{new}) = \left[v_j^{(1)}(C_{new}) = f(M_{syn}, P_{ATC}, P_{compl}), j = \overline{1, n^{(1)}} \right],$$

де: M_{syn} – синтаксична коректність коду; P_{ATC} – сумісність із платформою; P_{compl} – повнота перетворення. Метод виконується без залучення GPU (Graphics Processing Unit); за оцінками, до 18-22 % фрагментів не проходять Етап 1 та спрямовуються до Блоку 8.

Блок 8. Спрямування фрагмента на ручне доопрацювання (категорія С). Після завершення доопрацювання виконання переходить до Блоку 9.

Блок 9. Кінець.

Статичний аналіз гарантує синтаксичну коректність та сумісність з платформою, однак не здатний повністю виявити втрату семантики – ситуацію, коли мігрований код є синтаксично коректним та SAP S/4HANA-сумісним, але реалізує дещо відмінну бізнес-логіку. Це зумовлює потребу в Етапі 2.

Етап 2. Семантична валідація через зворотний переклад. Статичний аналіз гарантує синтаксичну коректність, але не виявляє семантичних втрат – ситуацію, коли мігрований код компілюється без помилок, але реалізує відмінну бізнес-логіку. Для подолання цієї проблеми розроблено метод семантичної валідації через зворотний переклад (back-translation), який базується на властивості оберненості [5]: якщо $C_{old} \rightarrow C_{new}$ зберегло семантику, то $C_{new} \rightarrow C'_{old}$ повинно відновити код, близький до C_{old} .

Крок 1 – C_{new} подають ВММ з системним контекстом "аналітика зворотної міграції коду" для отримання C'_{old} , тобто відповідатиме вихідному коду.

Крок 2 – $BT\text{-}score = CodeBLEU(C_{old}, C'_{old}) \in [0, 1]$ [16]. Паралельно для фрагментів з тестовим покриттям

виконується динамічне тестування (M_{func}). Елементи вектора показників верифікації визначають як:

$$V^{(2)}(C_{new}) = \left[v_j^{(2)}(C_{new}) = f(BT\text{-}score, M_{func}, M_{sem}), j = \overline{1, n^{(2)}} \right], (6)$$

де M_{sem} – семантична подібність до еталонного SAP S/4HANA-коду (CodeBLEU відносно навчального корпусу).

Етап 3. Ранжування за категоріями довіри. Результат $V^{(2)}(C_{new})$ (BT-score) та агрегована метрика IAQS, яка вже містить результат $V^{(1)}(C_{new})$ через компоненту (синтаксична коректність), інтегруються в інтегральний показник довіри, що визначають як:

$$S_{trust} = \alpha \cdot IAQS + (1 - \alpha) \cdot BT\text{-}score, (7)$$

де коефіцієнт $\alpha = 0,7$ відображає вищу відносну важливість IAQS (яка містить реальне динамічне тестування через M_{func} та синтаксичну валідацію M_{syn}) порівняно з BT-score (що оцінює тільки оберненість перетворення).

Для фрагментів коду без тестового покриття, де BT-score недоступна, $S_{trust} = IAQS$. Це означає, що для фрагментів коду, які не мають тестового покриття і де метрика BT-score недоступна, встановлюється дефолтне значення рівня довіри: $S_{trust} = IAQS$ (де IAQS означає Initial Automated Quality Score або аналогічний базовий показник якості). Ці параметри будуть враховані для подальшого аналізу надійності, розрахунку ризиків або автоматичної генерації тестів для цих ділянок.

Розподіл за категоріями (табл. 2) відображає принцип прогресивного контролю якості мігрованого коду: автоматичне прийняття (категорія А) застосовується тільки за одночасного виконання всіх трьох умов мігрованого коду; спрощена перевірка (категорія В) скорочує ручну трудомісткість у 3-4 рази порівняно з повним ручним доопрацюванням; категорія С слугує джерелом навчальних прикладів для замкненого контур якості мігрованого коду. Такий підхід забезпечує масштабованість рішення на промислові кодові бази зі збереженням контролю якості мігрованого коду.

Табл. 2. Категорії довіри та регламент верифікації / Trust categories and verification workflow

Категорія	Умова	Характеристика	Порядок дій
А – висока	$S_{trust} \geq 0,70$	Синтаксично коректний, S/4HANA-сумісний, семантику збережено	Автоматичне прийняття
В – помірна	$0,40 \leq S_{trust} < 0,70$	Функціональна коректність потребує підтвердження	Спрощена перевірка (~8-15 хв.)
С – низька	$S_{trust} < 0,40$ або зупинка Етапу 1	Критичні помилки або істотна семантична втрата	Ручне доопрацювання; поповнення QLoRA/ORPO (рис. 1)

Замкнений контур якості коду без залучення фахівців. Ключовою відмінністю запропонованої МАС інтелектуального реінжинірингу від наявних мультиагентних підходів [9] є її замкнений контур якості мігрованого коду, що забезпечує безперервне вдосконалення адаптованої ВММ між проектами без залучення фахівців (див. рис. 1). Механізм реалізується так.

Фрагменти категорії С ($S_{trust} < 0,40$) після ручного виправлення експертом автоматично поповнюють два паралельних потоки: (1) пари (C_{old}, C_{new}) поповнюють корпус QLoRA – паралельний корпус доменної адаптації; (2) трійки переваг ($C_{old}, C_{new}, C_{rejected}$) формують корпус ORPO для вирівнювання моделі, де $C_{rejected}$ – відхилений варіант агента-розробника. Трійки формуються тільки для пар, у яких різниця IAQS між прийнятим та відхиленим варіантами є статистично значущою, що містить слабкі нерепрезентативні сигнали переваги. Отже, IAQS виконує роль автоматичного сигналу переваги, замінюючи людину-оцінювача: кожен проєкт збагачує доменний корпус без додаткових трудовитрат.

Абляційне дослідження конфігурацій системи інтелектуального реінжинірингу. Для кількісного підтвердження внеску кожного архітектурного компонента здійснено абляційне дослідження на 300 реальних промислових АВАР-фрагментах трьох рівнів складності: 102 рівня L1, 126 рівня L2, 72 рівня L3. Попередні результати для зведеної метрики IAQS опубліковано у роботі [15]. У цій роботі вони доповнюються розбивкою за рівнями складності та аналізом механізмів, що пояснюють закономірності результатів (табл. 3). Вибірку сформовано методом стратифікованої випадкової вибірки з генеральної сукупності 3200 АВАР-об'єктів: 102 фрагменти рівня L1, 126 – L2, 72 – L3, що відповідає розподілу складності у генеральній сукупності. Для забезпечення відтворюваності всі три незалежних запуски виконано з фіксованим зерном генератора випадкових чисел (seed = 42). Ручну верифікацію фрагментів категорії С виконували два незалежних АВАР-розробники; міжочінюванню узгодженість розраховували за коефіцієнтом Cohen's Kappa ($\kappa > 0,82$), що відповідає

рівню "майже ідеальної" узгодженості за шкалою Landis & Koch. До можливих джерел похибок належать: варіативність виходів ВММ за однакових вхідних даних (контролюється трьома незалежними запусками); суб'єктивність ручної верифікації категорії С (контролюється процедурою міжочінюваної узгодженості); специфічність вибірки для одного машинобудівного підприємства, що може обмежувати узагальнюваність результатів на інші галузі. Чотири конфігурації обрано так, щоб кожна послідовно додавала один механізм IAQS-керування:

- B-1 (Zero-shot): одноагентна трансформація коду без ітеративного циклу та без доменного планування; IAQS виступає тільки підсумковою метрикою оцінювання якості мігрованого коду;
- B-2 CoT (Chain of Thought): той самий агент з ланцюжком думок – часткова структуризація генерування, проте ітерація некерована, зворотний зв'язок відсутній;
- B-3 (Dual-Agent): агент-розробник і агент-рецензент – повний ітеративний цикл зі структурованим зворотним зв'язком; доменне планування відсутнє (без Агента-планувальника);

Табл. 3. Результати абляційного дослідження ($n = 300$, середнє $\pm \sigma$, 3 незалежних запуски, $p < 0,01$ для всіх пар) / Ablation study results ($n = 300$, mean $\pm$ SD, 3 independent runs, $p < 0.01$ for all pairs)

Конфігурація системи інтелектуального реінжинірингу	1-Msyn % ↓	Msem ↑	Mfunc ↑	IAQS ↑	ΔIAQS до B-1
B-1: Zero-shot (QLoRA без ORPO)	15,4 ^{±2,1}	0,761 ^{±0,038}	0,642 ^{±0,045}	0,718 ^{±0,047}	–
B-2: CoT (одноагентна з CoT)	11,8 ^{±1,8}	0,794 ^{±0,032}	0,689 ^{±0,041}	0,762 ^{±0,039}	+0,044 (ітерація)
B-3: Dual-Agent (без планувальника)	7,1 ^{±1,5}	0,851 ^{±0,029}	0,761 ^{±0,037}	0,806 ^{±0,035}	+0,088 (+ зв'язок)
MAC (повна конфігурація)	4,2 ^{±1,1}	0,892 ^{±0,025}	0,803 ^{±0,031}	0,845 ^{±0,031}	+0,127 (+ планув.)

Табл. 4. IAQS за рівнями складності фрагментів успадкованого коду / IAQS by complexity levels of legacy code fragments

Конфігурація системи інтелектуального реінжинірингу	L1 – низька ($n=102$)	L2 – середня ($n=126$)	L3 – висока ($n=72$)	Δ MAC – B-1
B-1 (Zero-shot)	0,812 ^{±0,035}	0,711 ^{±0,042}	0,587 ^{±0,058}	–
B-2 (CoT)	0,841 ^{±0,030}	0,754 ^{±0,037}	0,649 ^{±0,051}	+0,029/+0,043/+0,062
B-3 (Dual-Agent)	0,869 ^{±0,027}	0,802 ^{±0,033}	0,718 ^{±0,044}	+0,057/+0,091/+0,131
MAC (повна)	0,895 ^{±0,022}	0,847 ^{±0,028}	0,772 ^{±0,039}	+0,083/+0,136/+0,185

По-перше, внесок структурованого зворотного зв'язку зростає нелінійно зі складністю. Різниця між B-2 і B-3 для L1 становить 0,028 IAQS, для L3 – вже 0,069. Для простого L1-коду агент-розробник виправляє однотипні помилки навіть без точної адресації. Для L3-коду з локалізованими помилками у конкретних рядках (наприклад, некоректне звернення до спрощеної таблиці BSEG у рядках 12-15 80-рядкового фрагмента) тільки структурований зв'язок дає змогу виконати цільове виправлення замість повного регенерування.

По-друге, B-3 для рівня L3 не досягає порогу $\delta = 0,76$ (IAQS = 0,718), тоді як MAC його перевищує (0,772). Різниця 0,054 ізольовано відображає внесок агента-планувальника для складного коду: попереднє доменне планування на основі RAG-запиту до бази SAP Notes є необхідною умовою для L3-перетворень із міжмодульними залежностями. Без нього агент-розробник систематично припускається класів помилок 1 і 2 (розд. "Промислова апробація"), які структурований зворотний зв'язок не може усунути, оскільки вони зумовлені відсутністю контексту, а не деталями реалізації.

По-третє, для L1-коду різниця між B-3 і MAC мінімальна (+0,026). Це означає, що для масових простих перетворень рівня L1 залучення агента-планувальника є надлишковим, що підтверджує доцільність гібридної стратегії апробації: базова архітектура – для L1–L2, MAC – для L3.

- MAC (повна): всі три механізми активні – доменне планування (RAG + SAP Notes), управління ітерацією та структурований зворотний зв'язок.

Зіставлення приростів між конфігураціями системи інтелектуального реінжинірингу дає змогу ізольовано оцінити внесок кожного механізму: керованість ітерацією додає +0,044 IAQS (B-1→B-2); структурований зворотний зв'язок – ще +0,044 (B-2→B-3); доменне планування агента-планувальника – ще +0,039 (B-3→MAC). Приблизно рівний внесок кожного механізму (~0,039-0,044 IAQS) свідчить про збалансованість архітектури, де жоден компонент не є домінуючим у загальному вимірі. MAC є єдиною конфігурацією, що перевищує поріг прийнятності $\delta = 0,76$ (IAQS = 0,845 проти 0,718 для B-1), а стандартне відхилення зменшується від 0,047 до 0,031 (–34 %), що свідчить про підвищення стабільності.

Для встановлення умов мігрованого коду, за яких кожен механізм є найзначущішим, виконано поділ результатів за рівнями складності вихідних фрагментів спадкованого коду (табл. 4).

Промислова апробація системи інтелектуального реінжинірингу. Промислову апробацію системи інтелектуального реінжинірингу виконано у межах реального проекту переходу SAP ERP 6.0 → S/4HANA 2023 великого машинобудівного підприємства. Загальний обсяг: 3200 АВАР-об'єктів (звіти – 46 %, функціональні модулі – 31 %, класи – 16 %, програми – 7 %). Фрагменти рівнів L1–L2 (~80 %) обробляла базова архітектура [14], рівня L3 (~20 %) – запропонована MAC (табл. 5).

Табл. 5. Результати промислової апробації (3200 АВАР-об'єктів): розподіл за категоріями та економія трудовитрат / Industrial approbation results (3200 ABA objects): category distribution and labour cost savings

Категорія	Кількість об'єктів, шт. (%)	Тривалість ручної міграції коду	Зі системою / Економія
А – висока довіра	2 112 (66,0)	38-210 хв	Не потрібно / ~100 %
В – помірна довіра	736 (23,0)	38-85 хв	8-15 хв / 75-82 %
С – низька довіра	352 (11,0)	85-210 хв	45-90 хв / 55-57 %
Зважена середня	3200 (100)	–	Ефект. авт. 86,5 % / ~80 %

Аналіз 352 фрагментів категорії С виявив чотири домінуючих класів помилок. Клас 1 (38 %): прямий доступ до спрощених таблиць (BSEG, VKPF, KONV) – система формувала технічно коректний, але функціональ-

но нееквівалентний замінник через нестачу контексту бізнес-логіки. Клас помилок 2 (29 %): вкладені SELECT у LOOP AT – конструкції вимагають концептуальної перебудови алгоритму, яку агент-розробник виконував некоректно без попереднього доменного планування, що підтверджує необхідність агента-планувальника для L3. Клас 3 (22 %): застаріле оброблення виключень CALL FUNCTION...EXCEPTIONS – система коректно визначала застарілий механізм, але некоректно відображала його на класовий механізм виключень ABAP, генеруючи код з логічними розривами. Клас 4 (11 %): міжмодульні залежності – фрагменти, функціональність яких залежить від стану інших об'єктів поза межами поточного контексту. Цей клас помилок є пріоритетним напрямом подальших досліджень.

Усі 352 виправлених фрагменти передано до замкненого контура міграного коду: 289 пар поповнили корпус QLoRA; 241 трійка переваг – корпус ORPO. Порівняно з базовою архітектурою [2], що забезпечує ефективну автоматизацію 72,3 % для об'єктів рівня L1–L2, запропонована MAC досягла 86,5 % на розширеній вибірці з 20 % складних об'єктів L3, що підтверджує доцільність гібридної стратегії.

Обговорення результатів дослідження. У роботі [19] автори проаналізували мультиагентну систему ChatDev для комунікативної взаємодії спеціалізованих агентів у завданнях розроблення програмного забезпечення та показали, що структурований діалог між агентами різних ролей знижує частоту логічних помилок у згенерованому коді. Проте розглянута ними система орієнтована винятково на генерування нового коду з природно-мовного опису й не містить механізму збереження функціональної еквівалентності з вихідним кодом. На відміну від цього, у нашому дослідженні аналогічний принцип структурованої взаємодії реалізовано через JSON-артефакт, що передають між агентами, й містить повний контекст міграції коду, враховуючи діагностичний зворотний зв'язок з прив'язкою до конкретних рядків коду та посиланнями на документацію SAP.

У роботі [22] автори запропонували фреймворк AutoGen для побудови мультиагентних LLM-застосунків через конwersaційну взаємодію та показали масштабованість підходу для різних класів завдань автоматизації розроблення програмного забезпечення. Проте в їх роботі взаємодія агентів організована як довільний діалог без формалізованого критерію якості результату. На відміну від цього, у нашому дослідженні взаємодія агентів формалізована як марківський процес прийняття рішень з визначеним простором станів, дій та функцією винагороди на основі IAQS, що забезпечує передбачувану збіжність – підтверджену зниженням стандартного відхилення від 0,047 до 0,031 (–34 %).

У роботі [2] автори дослідили верифіковане трансплювання коду засобами BMM із формальними гарантіями коректності та показали здатність LLM-підходів генерувати верифікований код за наявності повного тестового покриття. Ці результати підтверджують перспективність застосування BMM у завданнях трансформації коду. Проте їх підхід потребує формальних специфікацій та тестового покриття, яких немає для 25–40 % промислових ABAP-фрагментів. У нашому дослідженні цю проблему вирішено методом зворотного перекладу як незалежним засобом верифікації семантичної еквівалентності без формальних специфікацій – що дало змогу верифікувати 100 % промислових фрагментів.

У роботі [11] автори розглянули формальні методи верифікації семантичної еквівалентності програмного коду на основі SMT-розв'язувачів та довели, що ці методи забезпечують математично строгі гарантії, проте потребують формальних семантичних моделей мови та погано масштабуються через експоненційне зростання простору станів. У нашому дослідженні запропоновано практично реалізований компроміс: триетапний каскад верифікації з принципом ранньої зупинки, що не потребує формальних семантичних моделей ABAP та дає змогу верифікувати 100 % промислових фрагментів без обмежень на обсяг кодової бази.

У роботі [5] автори запропонували метод зворотного перекладу для аугментації навчальних даних у завданнях машинного перекладу та показали, що семантично коректним перетворенням характерна властивість оберненості. У нашому дослідженні цей метод вперше адаптовано для верифікації еквівалентності програмного коду: $BT\text{-score} = CodeBLEU(C_{old}, C'_{old})$ вимірює оберненість самого перетворення коду, а не подібність природно-мовних послідовностей, що принципово відрізняє запропонований підхід від оригінального застосування [5], та дає змогу верифікувати семантичну еквівалентність без тестового покриття.

У роботі [13] автори проаналізували підхід на основі model-driven engineering для автоматизованої трансляції програмного коду між мовами програмування та довели ефективність формалізованого підходу до трансформації при переході між технологічними платформами. На відміну від цього підходу, де трансформацію описано як перетворення між мовними специфікаціями, у нашому дослідженні завдання міграції коду формалізовано як марківський процес прийняття рішень, що дає змогу керувати ітеративним циклом та гарантувати умову зупинки на основі інтегральної метрики якості IAQS.

У роботі [7] автори систематизували сучасні стратегії модернізації програмного забезпечення та встановили, що гібридні підходи, які поєднують автоматизовану трансформацію з вибірковою ручною верифікацією, є найефективнішими для масштабних проєктів міграції коду корпоративних систем. Ці результати безпосередньо підтверджують доцільність запропонованої у нашому дослідженні гібридної стратегії розгортання системи інтелектуального реінжинірингу КС: базова архітектура – для фрагментів рівнів L1–L2 та MAC – для семантично складних фрагментів рівня L3, що забезпечило ефективну автоматизацію на рівні 86,5 % зі збереженням контролю якості міграного коду.

Отже, внаслідок обговорення наявних результатів у досліджуваній галузі, підтверджено доцільність проведення саме цього дослідження, оскільки наявні у сучасній науковій літературі підходи до автоматизованої міграції програмного коду розглядають мультиагентну трансформацію коду [14, 20], верифіковане трансплювання [2], формальні методи верифікації [10] та метод зворотного перекладу [5] ізольовано, і жоден з них не забезпечує одночасно формалізованого критерію зупинки ітеративного циклу з контролем якості міграного коду, верифікації функціональної еквівалентності без тестового покриття та локального розгортання адаптованої BMM без передавання корпоративного коду зовнішнім сервісам, що реалізовано у запропонованому рішенні та підтверджено промисловою апробацією на 3200 реальних ABAP-об'єктах.

Отже, внаслідок виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження – отримала подальший розвиток модель архітектури мультиагентної системи інтелектуального реінжинірингу КС, яка, на відміну від наявних моделей, враховує удосконалений метод верифікації результатів міграції успадкованого користувацького АВАР-коду, що дає змогу визначити його функціональну еквівалентність без тестового покриття та без залучення зовнішніх фахівців.

Практична значущість результатів дослідження – впроваджено у вигляді локально розгорнутого сумісного з ліцензійними вимогами SAP програмного комплексу та апробовано на 3200 реальних АВАР-об'єктах у межах промислового проєкту міграції коду SAP ERP 6.0 → S/4HANA, що підтвердило ефективну автоматизацію процесу трансформації коду на рівні 86,5 % за скорочення трудовитрат приблизно на 80 %, відповідно запропоновані архітектурні принципи придатні до узагальнення на інші ERP-платформи (Oracle E-Business Suite, Microsoft Dynamics та ін.).

Висновки / Conclusions

Вирішено актуальну науково-прикладну задачу розроблення моделі архітектури мультиагентної системи та методу верифікації результатів для автоматизованої міграції успадкованого користувацького АВАР-коду при реінжинірингу корпоративних систем класу SAP ERP за умови забезпечення якості мігрованого коду та збереження функціональної еквівалентності у закритому корпоративному середовищі. За результатами проведеного дослідження можна зробити такі основні висновки.

1. Проаналізовано обмеження наявних мультиагентних підходів (MetaGPT, ChatDev, AutoGen) та методів верифікації семантичної еквівалентності (SMT-верифікація, формальні моделі). Встановлено, що наявні фреймворки не забезпечують збереження функціональної еквівалентності з вихідним кодом, не мають формалізованого критерію прийнятності та не підтримують локального розгортання адаптованої ВММ, що є обов'язковою умовою під час роботи з корпоративним кодом під ліцензійними обмеженнями SAP.
2. Набула подальшого розвитку модель архітектури мультиагентної системи інтелектуального реінжинірингу шляхом формалізації завдання міграції коду як марківського процесу прийняття рішень та введення трьох спеціалізованих агентів (планувальника, розробника та рецензента), взаємодія яких забезпечується через структурований JSON-артефакт. Метрику IAQS розширено від бінарного критерію зупинки до потрібного керованого параметра – адаптації моделі, управління ітераціями та генерування структурованого діагностичного зворотного зв'язку, що, на відміну від наявних підходів, дає змогу здійснювати цільове виправлення семантично складних фрагментів рівня L3 без повного регенерування користувацького коду.
3. Удосконалено метод верифікації результатів автоматизованої міграції користувацького коду шляхом побудови триетапного каскаду з принципом ранньої зупинки, у якому метод зворотного перекладу застосовується як незалежний засіб оцінювання семантичної еквівалентності, а відхилені фрагменти автоматично формують навчальну вибірку для подальшого донавчання адаптованої ВММ. Це, на відміну від наявних підходів, не потребує залучення фахівців та забезпечує безперервне вдосконалення моделі в замкненому контурі якості мігрованого коду.

4. Абляційне дослідження на 300 реальних промислових АВАР-фрагментах підтвердило, що запропонована MAC є єдиною з чотирьох конфігурацій систем, що перевищує поріг прийнятності $\delta = 0,76$ ($IAQS = 0,845^{+0,031}$ проти $0,718^{+0,047}$ для одноагентної базової лінії). Перевага MAC зростає пропорційно до складності перетворень: $\Delta IAQS = +0,083$ – для рівня L1 та $+0,185$ – для рівня L3, що кількісно обґрунтовує доцільність гібридної стратегії: базова архітектура – для L1–L2, MAC – для L3.
5. Промислова апробація на 3200 АВАР-об'єктах реального проєкту переходу SAP ERP 6.0 → S/4HANA 2023 підтвердила практичну ефективність запропонованих рішень: категорія А (автоматичне прийняття) – 66,0 %, В (спрощена перевірка) – 23,0 %, С (ручне доопрацювання) – 11,0 %; ефективна автоматизація – 86,5 %, скорочення трудовитрат – ~ 80 %. Аналіз 352 фрагментів категорії С виявив чотири домінуючі класи помилок; усі виправлені фрагменти автоматично поповнили корпуси QLoRA/ORPO через замкнений контур якості коду.
6. Запропоновані архітектурні рішення реалізовано у вигляді локально розгорнутого програмного комплексу, сумісного з ліцензійними вимогами SAP. Перспективами подальших досліджень є розширення контексту агента-планувальника до рівня програмного модуля для підтримки міжмодульних залежностей (клас помилок 4), адаптивне калібрування порогу δ залежно від типу SAP-модуля, а також узагальнення розроблених архітектурних принципів на інші ERP-платформи (Oracle E-Business Suite, Microsoft Dynamics).

Подяка. Роботу виконано у межах науково-дослідної теми "Інформаційні технології інтелектуального комп'ютерингу" (державний реєстраційний номер 0124U004188) кафедри програмних засобів Національного університету "Запорізька політехніка".

Заява про використання інструментів штучного інтелекту. Під час підготовки цього рукопису автори застосовували інструменти штучного інтелекту та програмне забезпечення для оброблення природної мови (ChatGPT, Copilot та Grammarly) винятково задля граматичного редагування, коригування орфографії та стилістичної адаптації складних синтаксичних конструкцій. Автори здійснювали критичний аналіз і подальше редагування згенерованого тексту та несуть повну відповідальність за достовірність, науковий зміст і остаточну редакцію публікації.

References

1. Anthropic. (2025). Claude Code: Build effective agents. URL: <https://www.anthropic.com/engineering/building-effective-agents>
2. Bhatia, S., Qiu, J., Hasabnis, N., Seshia, S. A., & Cheung, A. (2024). Verified code transpilation with LLMs. *Advances in Neural Information Processing Systems* 37 (NeurIPS 2024), 1–18. <https://doi.org/10.52202/079017-1310>
3. Chen, M., et al. (2021). Evaluating large language models trained on code. arXiv. <https://doi.org/10.48550/arXiv.2107.03374>
4. Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems*, arXiv. 10088–10115. <https://doi.org/10.48550/arXiv.2305.14314>
5. Edunov, S., Ott, M., Auli, M., & Grangier D. (2018). Understanding back-translation at scale. arXiv. <https://doi.org/10.48550/arXiv.1808.09381>
6. Google. (2026). Antigravity 2.0. URL: <https://antigravity.google>
7. Guez Assuncao, W. K., Marchezan, L., Arkoh, L., Egyed, A., & Ramler, R. (2025). Contemporary software modernization: Strategies, driving forces, and research opportunities. *ACM Transactions on Software Engineering and Methodology*, 34(5), article ID 142, 1–35. <https://doi.org/10.1145/3708527>

8. Hardy, P. (2020). Migrating custom code to SAP S/4HANA. Boston: Rheinwerk Publishing. URL: https://www.sap-press.com/migrating-custom-code-to-sap-s4hana_5131
9. Hong, S., et al. (2024). MetaGPT: Meta programming for a multi-agent collaborative framework. *Proceedings of ICLR 2024*, Vienna. <https://doi.org/10.48550/arXiv.2308.00352>
10. ISO/IEC 25010:2023. (2023). Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – Product quality model. Geneva: ISO. URL: <https://www.iso.org/standard/78176.html>
11. Jhala, R., & Majumdar, R. (2009). Software model checking. *ACM Computing Surveys*, 41(4), article number 21. <https://doi.org/10.1145/1592434.1592438>
12. King, J. (1976). Symbolic execution and program testing. *Communications of the ACM*, 19(7), 385–394. <https://doi.org/10.1145/360248.360252>
13. Lano, K., & Siala, H. (2024). Using model-driven engineering to automate software language translation. *Automated Software Engineering*, 31(1), article number 20. <https://doi.org/10.1007/s10515-024-00419-y>
14. Pozdnyakov, O. A., & Parkhomenko, A. V. (2025). Migration of custom code to new versions of complex computer systems using intelligent reengineering methods and models. *Scientific Works of DonNTU. Series: Informatics, Cybernetics and Computer Engineering*, 2(41), 86–98. <https://doi.org/10.31474/1996-1588-2025-2-41-86-98>
15. Pozdnyakov, O. A., & Parkhomenko, A. V. (2026). Evaluation and quality assurance of migrated ABAP code. *Radio Electronics, Computer Science, Control*, 1, 80–89. <https://doi.org/10.15588/1607-3274-2026-1-8>
16. Pozdnyakov, O. A., & Parkhomenko, A. V. (2026). Verification model for migrated custom ABAP code based on back-translation and trust category ranking. *Modern Information Technologies and Artificial Intelligence Systems: Proceedings of the II International Scientific-Practical Conference (Kharkiv – Yaremche, 27–28 April 2026)*. Kharkiv: NURE, 79–81. URL: <https://mitais.com.ua/index.html>; https://drive.google.com/file/d/1GHfO8OKdWKtEEef_dhGUPe8y_WeVtOXL/view
17. Pozdnyakov, O., & Parkhomenko, A. (2026). Multi-agent architecture for legacy custom code migration. *Computer Modeling and Intelligent Systems (CMIS-2026): Proceedings of the IX International Workshop (5 May 2026)*, Zaporizhzhia, Ukraine, 236–251 (CEUR Workshop Proceedings, vol. 4220). URL: <https://ceur-ws.org/Vol-4220/paper19.pdf>
18. Puterman, M. L. (1994). Markov Decision Processes. New York: Wiley. <https://doi.org/10.1002/9780470316887>
19. Qian, C., et al. (2024). ChatDev: Communicative agents for software development. *Proceedings of ACL 2024, Bangkok*. 15174–15186. <https://doi.org/10.48550/arXiv.2307.07924>
20. Ren, S., et al. (2020). CodeBLEU: A method for automatic evaluation of code synthesis. arXiv. <https://doi.org/10.48550/arXiv.2009.10297>
21. Talebirad, Y., & Nadiri, A. (2023). Multi-agent collaboration: Harnessing the power of intelligent LLM agents. arXiv. <https://doi.org/10.48550/arXiv.2306.03314>
22. Wu, Q., et al. (2024). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *Proceedings of COLM 2024, Philadelphia*. <https://doi.org/10.48550/arXiv.2308.08155>
23. Zheng, Q., et al. (2023). CodeGeeX: A pre-trained model for code generation with multilingual benchmarking on HumanEval-X. KDD 23, Long Beach CA USA. <https://doi.org/10.1145/3580305.3599790>

O. A. Pozdnyakov, A. V. Parkhomenko

National University "Zaporizhzhia Polytechnic", Zaporizhzhia, Ukraine

DEVELOPMENT OF A METHOD FOR VERIFICATION OF THE RESULTS OF LEGACY USER ABAP CODE MIGRATION IN A MULTIAGENT SYSTEM OF INTELLIGENT REENGINEERING

Migrating custom ABAP code when moving from SAP ERP 6.0 to SAP S/4HANA is fundamentally different from generating new code. The migrated code must maintain functional equivalence with legacy business logic while simultaneously complying with the limitations of the target platform. However, these requirements cannot be met by existing frameworks focused on generating new code based on natural language descriptions. The study has allowed to formalize the code migration problem as a Markov decision process and to develop the architecture of an intelligent reengineering multi-agent system that includes three agents: Planner, Developer, and Reviewer, whose interaction is ensured through a structured JSON artefact. The work also expanded the integrated code quality assessment metric from a binary stopping criterion to a triple control function that regulates verification method adaptation, iteration management, and structured diagnostic feedback, ensuring targeted code transformation instead of full regeneration. As a result, a three-stage sequential verification method with an early-stopping principle (static analysis, semantic evaluation using reverse translation, and ranking by confidence categories) was developed, forming a closed quality assurance cycle without the involvement of external specialists. Ablative research on 300 fragments of real industrial ABAP code confirmed that the proposed multi-agent configuration of the intelligent reengineering system is the only one of the four tested that exceeds the quality threshold, achieving an integrated quality indicator of 0.845 compared to 0.718 for the single-agent baseline architecture. At the same time, the benefit of a multi-agent system increases proportionally to the complexity of the code transformation, confirming the feasibility of a hybrid deployment strategy of the intelligent reengineering system. Industrial testing on 3,200 ABAP objects as part of a real-world SAP ERP migration project confirmed an 86.5 % effective automation of the ABAP code migration process, while simultaneously reducing labour costs by approximately 80 %. As a result, the proposed architectural principles are suitable for generalization to other enterprise ERP platforms.

Keywords: reengineering of computer system; generative artificial intelligence; large language model; Markov decision process; an integrated metric for assessing code quality.