Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
https://doi.org/10.36930/40350618
Article received 04.11.2025
Article accepted 22.12.2025
UDC 004.[272+627+75]:005.8
Correspondence author: R. I. Vaskiv roman.i.vaskiv@lpnu.ua
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)

Р. І. Васьків, В. В. Пасічник
Національний університет "Львівська політехніка", м. Львів, Україна

КОНТЕКСТНО-ОРІЄНТОВАНА СИСТЕМА ОРГАНІЗАЦІЇ ДАНИХ ДЛЯ РОЗПОДІЛЕНИХ ПРОЄКТНИХ КОМАНД

Досліджено проблему фрагментованої організації даних у розподілених проєктних командах великих IT-компаній та сервісних організацій, що працюють у кількох часових поясах, юрисдикціях і доменах. Показано, що централізовані архітектури даних (Data Warehouse, Data Lake, Lakehouse), орієнтовані насамперед на звітність, погано відображають багатодоменну структуру таких організацій, створюють організаційні "вузькі місця" та не забезпечують контекстно-орієнтованої підтримки життєвого циклу проєктів. На підставі аналізу концепції мережі Data Mesh обґрунтовано релевантність доменно-орієнтованого підходу до організації даних із використанням дата-продуктів, контрактів на дані (data contracts), платформи самообслуговування та федеративного обчислювального управління, але виявлено розрив на рівні доступу стейкхолдерів до проєктного контексту. Проаналізовано сучасні практики інтеграції великих мовних моделей і розмовної аналітики в BI- та хмарні платформи. Показано, що такі рішення здебільшого залишаються платформно-специфічними та відтворюють схему інтеграцій M×N між агентами та сервісами. Протокол контексту моделі MCP (англ. *Model Context Protocol*) розглянуто як відкритий протокол прикладного рівня, що нормалізує взаємодію LLM-агентів зі зовнішніми ресурсами через примітиви resources, tools, prompts і дає змогу зменшити інтеграційну складність до O(M+N). На цьому підґрунті запропоновано контекстно-орієнтовану архітектурну модель системи організації даних для розподілених проєктних команд у конфігурації "Data Mesh + MCP + LLM-агенти". Запропоновано архітектуру "Data Mesh + MCP + агенти на підставі великих мовних моделей (LLM)", у якій доменні дата-продукти (data products) та контракти на дані (data contracts) системи мережі Data Mesh формально зіставлено з MCP-ресурсами (resources) та інструментами (tools), а доступ до них здійснюється через платформу самообслуговування даними (self-service data platform) з інтегрованою підтримкою LLM-агентів. Аналітична модель інтеграційної складності демонструє перехід від квадратичного зростання витрат типу M×N у класичній схемі до лінійної залежності M+N в архітектурі "Data Mesh + MCP", що створює передумови для підвищення масштабованості інтеграційного шару, зниження інтеграційних витрат і прозорого, контекстно-орієнтованого доступу до проєктних артефактів у розподілених командах. Перспективою подальших досліджень є емпірична верифікація запропонованої архітектури в організаціях із багатодоменними портфелями проєктів та подальший розвиток моделей контекстно-орієнтованої взаємодії LLM-агентів із дата-продуктами та даними про проєктні ризики.

Ключові слова: мережі даних; протокол контексту моделі; великі мовні моделі; агенти на підставі великих мовних моделей; платформа самообслуговування даними; федеративне обчислювальне управління.

Вступ / Introduction

Сучасні розподілені проєктні команди у продуктових IT-компаніях, аутсорсингових сервісних провайдерах, логістичних операторах, фінтех- і e-commerce-екосистемах працюють одночасно в кількох часових поясах, юрисдикціях і культурних контекстах. Їхні проєктні процеси спираються на широкий спектр інструментів: системи управління продуктом і беклогом, платформи для розроблення та експлуатації, системи моніторингу, аналітичні сервіси, корпоративні сховища даних. Унаслідок цього інформація про вимоги, архітектурні рішення, ризики, експлуатаційні показники та бізнес-метрики розподіляється між ізольованими сервісами й командами, формуючи фрагментоване середовище підтримки проєктів. Традиційні централізовані архітектури організації даних на підставі Data Warehouse та Data Lake [5], попри їхню технологічну зрілість, стають щоразу менш придатними для такого середовища: вони погано масштабуються, централізують відповідальність у вузької команди платформи та орієнтовані насамперед на побудову звітності, а не на безперервний супровід життєвого циклу проєктів у розподілених командах.

У відповідь на ці обмеження за останнє десятиріччя сформувалася концепція мережі Data Mesh як децентралізованої системи організації даних [4], яка базується на доменно-орієнтованому володінні даними, трактуванні даних як продукту, інформаційно-технологічній платформі самообслуговування та федеративному управлінні політиками. Мережа Data Mesh розподіляє відповідальність за повний життєвий цикл дата-продуктів між доменними командами, формалізує взаємодію че-

Інформація про авторів:
Васьків Роман Ігорович, аспірант, кафедра інформаційних систем та мереж. **Email:** roman.i.vaskiv@lpnu.ua; https://orcid.org/0000-0002-8549-5035
Пасічник Володимирович, д-р техн. наук, професор, кафедра інформаційних систем та мереж. **Email:** vpasichnyk@gmail.com; https://orcid.org/0000-0002-5231-6395
Цитування за ДСТУ: Васьків Р. І., Пасічник В. В. Контекстно-орієнтована система організації даних для розподілених проєктних команд. Науковий вісник НЛТУ України. 2025, т. 35, № 6. С. 153–168.
Citation APA: Vaskiv, R. I., & Pasichnyk, V. V. (2025). Context-oriented data organization system for distributed project teams. *Scientific Bulletin of UNFU*, 35(6), 153–168. https://doi.org/10.36930/40350618

рез контракти на дані та забезпечує баланс між автономією доменів і глобальною узгодженістю організації. Раніше було показано [24], що така модель є придатною основою для підтримки розподілених проєктних команд, де одночасно функціонують десятки доменів і технічних стеків. Водночас навіть у межах Data Mesh доступ до доменних даних для основних стейкхолдерів, аналітиків чи архітекторів залишається опосередкованим через знання схем, SQL-запити та спеціалізовані інструменти, що обмежує реальний рівень self-service-аналітики для широкого кола учасників проєктів.

Паралельно стрімкий розвиток великих мовних моделей LLM (англ. *Large Language Model*) і агентних підходів відкрив можливість працювати з розподіленими даними та сервісами через запити природною мовою [15]. Однак, більшість наявних інтеграцій LLM із системами організації даних будуються як проміжні рішення: окремі "чат-асистенти" поверх BI-сховищ, кастомні конектори до окремих API, локальні плагіни без єдиної архітектурної рамки, механізмів контексту та узгодженого управління доступами. Протокол контексту моделі MCP (англ. *Model Context Protocol*) запропоновано як відкритий протокол взаємодії між мовними моделями [2], інструментами та зовнішніми ресурсами, надає стандартизований канал, через який інтелектуальні агенти можуть відкривати інструменти, ініціювати дії та працювати з різними джерелами даних без строгого зв'язування з конкретними реалізаціями. Протокол MCP істотно знижує складність інтеграції за схемою інтеграції N×M між моделями та сервісами, але наразі не існує формалізованої моделі того, як його потрібно вбудовувати в архітектуру мережі Data Mesh як цілісної системи організації даних, орієнтованої на підтримку проєктних процесів у розподілених командах.

Нагадаємо, схема інтеграції M×N у контексті мікроелектроніки описує архітектуру, де M – це кількість вихідних сигналів (або рядків), а N – кількість вхідних сигналів (або стовпців) мультиплексора MUX (англ. *Multiplexer*) чи демультиплексора DEMUX (англ. *Demultiplexer*), що означає, що вона дає змогу вибрати один із M входів для N виходів (або навпаки) на одному кристалі, забезпечуючи компактність та ефективність.

Отже, дослідницький розрив полягає у відсутності контекстно-орієнтованої архітектурної моделі, яка поєднувала б принципи мережі Data Mesh із можливостями протоколу MCP та інтелектуальних агентів на підставі моделей LLM для підтримки розподілених проєктних команд. Потрібна система, що, з одного боку, зберігає доменну автономію, взаємодію на підставі дата контрактів та федеративне управління даними, а з іншого – надає учасникам проєктів (керівникам, аналітикам, архітекторам, інженерам) єдиний контекстно-залежний інтерфейс доступу до релевантних дата-продуктів, артефактів і метрик без необхідності прямої роботи зі схемами, запитами та гетерогенними інструментами. У межах такого підходу контекст проєкту (ідентифікатор, домен, роль, стадія життєвого циклу, політики доступу) має бути базовою сутністю, яка супроводжує всі операції з даними й артефактами та інтерпретується як за Data Mesh-рівнем, так і за MCP-інфраструктурою.

Об'єкт дослідження – система організації даних у розподілених проєктних командах, що функціонують у мультидоменних середовищах.

Предмет дослідження – методи і засоби організації даних для розподілених проєктних команд у мультидоменних середовищах на підставі поєднання принципів мережі даних Data Mesh та протоколу контексту моделі (MCP) з використанням агентів на підставі великих мовних моделей (LLM-агентів), що забезпечує контекстно орієнтовану взаємодію з проєктними артефактами та дасть змогу формалізувати підхід до оцінювання інтеграційної складності протокольного шару архітектури.

Мета роботи – розробити контекстно-орієнтовану архітектурну модель системи організації даних для розподілених проєктних команд на підставі синергії мережі даних Data Mesh і протоколу Model Context Protocol з використанням LLM-агентів, що забезпечить адаптивний доступ до доменних дата-продуктів та пов'язаних інформаційних ресурсів відповідно до стадій життєвого циклу проєктів, ролей учасників і контексту завдань та знизить інтеграційну складність забезпечення взаємодії в мультидоменному середовищі.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

- формалізувати вимоги до контекстно-орієнтованої системи організації даних у мультидоменному розподіленому середовищі з урахуванням стадій життєвого циклу проєкту, функціональних ролей учасників і контексту завдань, що забезпечить релевантну підтримку проєктних процесів у режимі реального часу;
- розробити еталонну архітектуру типу "Data Mesh + MCP + LLM-агенти", у якій доменні дата-продукти, контракти на дані та платформа самообслуговування даними доповнюються мережею MCP-серверів і LLM-агентів, що забезпечить контекстно адаптивну маршрутизацію запитів і зменшить навантаження на централізовану дата-інфраструктуру та команду платформи даних;
- побудувати модель подання контексту проєкту як формалізованої структури координат "проєкт – домен – роль – стадія життєвого циклу", яка визначає простір допустимих операцій над даними, що забезпечить контроль доступу та відповідність дій регламентам домену;
- сформулювати прикладні сценарії використання запропонованої архітектури "Data Mesh + MCP + LLM-агенти" в розподілених IT-командах та розробити підхід до оцінювання її впливу на ключові показники ефективності проєктних процесів, що дасть змогу здійснити кількісну валідацію її доцільності та практичної ефективності.

Аналіз останніх досліджень та публікацій. Класичні централізовані архітектури організації даних – сховища даних (Data Warehouse), озера даних (Data Lake) та гібридні моделі типу Data Lakehouse – історично забезпечували підприємствам уніфіковане джерело даних для звітності та аналітики. У таких архітектурах дані з множини операційних систем завантажуються в єдину платформу, де нормалізуються, агрегуються та стають доступними для BI-інструментів і аналітиків. Сучасні варіації на кшталт Lakehouse намагаються об'єднати гнучкість Data Lake з керованістю Data Warehouse, пропонуючи єдиний шар зберігання й оброблення для структурованих та напівструктурованих даних.

У контексті розподілених організацій ці архітектури мають низку системних обмежень. По-перше, відповідальність за інтеграцію джерел, моделювання схеми та публікацію наборів даних концентрується в невеликої "центральної" команди платформи, що призводить до черг запитів і високого time-to-data для доменних команд [5]. По-друге, модель "одного великого сховища"

погано відображає багатодоменну структуру транснаціональних компаній, у яких проєктні портфелі розбиті за продуктами, ринками, бізнес-напрямами та регіонами: схема даних неминуче стає компромісною, а локальний контекст доменів втрачається. По-третє, централізована архітектура організації даних (Data Warehouse, Data Lakes, Data LakeHouse) орієнтована переважно на аналітичні сценарії (dashboarding, batch-звітність), тоді як проєктні процеси розподілених команд потребують також операційної підтримки – моніторингу прогресу спринтів, ризиків, інцидентів, експлуатаційних метрик – із низькою латентністю та можливістю локальної еволюції артефактів [9].

Низка оглядових робіт і промислових кейсів підкреслюють, що у великих організаціях саме централізована модель стає "вузьким місцем" під час масштабування управління даними: зростання кількості доменів, джерел та аналітичних запитів призводить до організаційної перевантаженості платформи, втрати прозорості й конфліктів між глобальною стандартизацією та локальними потребами доменів [16]. У розподілених проєктних командах це зумовлює збільшення латентності доступу до релевантних даних, формування несанкціонованих дубльованих сховищ на рівні окремих підрозділів і підвищення ризику управлінських помилок, спричинених використанням фрагментованих та непослідовно оновлюваних інформаційних ресурсів.

Концептуальні засади мережі Data Mesh – доменно-орієнтоване володіння даними, трактування даних як продукту (data as a product), платформа самообслуговування даними (self-serve data platform) та федеративне обчислювальне управління (federated computational governance) – системно сформулював й обґрунтував Ж. Деггані як децентралізовану систему організації аналітичних даних, що протиставляється централізованим архітектурам на підставі Data Warehouse та Data Lake [4].

У контексті підтримки розподілених проєктних команд мережу Data Mesh уже розглядали як адекватну архітектурну підставу, оскільки домени часто відповідають продуктам, ринкам або бізнес-напрямам, а дата-продукти відображають ключові артефакти супроводу – беклог, метрики релізів, показники сервісної якості, ризики тощо [24]. Однак, навіть за умови впровадження Data Mesh доступ до цих артефактів для учасників проєктів залишається переважно технічним: потрібні знання схем, володіння SQL або спеціалізованими інструментами, що обмежує реальний рівень self-service для менеджерів проєктів, архітекторів чи представників бізнесу.

У сфері супроводу розподілених проєктних команд модель Data Mesh розглядають як релевантну архітектурну підставу організації даних, оскільки домени природно ізоморфні до продуктів, ринків або бізнес-напрямів, а дата-продукти можуть інтерпретуватися як формалізовані артефакти проєктної діяльності (беклог, метрики релізів, показники сервісної якості, реєстри ризиків тощо). У такій конфігурації архітектури домен відповідає організаційній одиниці, що несе відповідальність за повний життєвий цикл даних у межах конкретного продукту чи сервісу, а дата-продукти забезпечують відтворюваний, контрольований та документований доступ до релевантних проєктних даних [4, 17]. Водночас, навіть за умов впровадження моделі Data Mesh, взаємодія з цими артефактами для більшості стейкхолдерів залишається опосередкованою технічними засобами: необхідні знання схем, володіння мовами запитів і спеціалізованими інструментами аналітики, що знижує фактичний рівень режиму самообслуговування для керівників проєктів, архітекторів та бізнес-представників і обмежує можливість прямої роботи з проєктним контекстом без залучення вузькоспеціалізованих команд з управління даними.

Розвиток великих мовних моделей сформував окремий клас інтерфейсів доступу до даних, у яких запити природною мовою (promts) перетворюються на формальні SQL-вирази або композиції викликів API. Сучасні аналізи text-to-SQL-підходів в епоху моделей LLM демонструють, що комбінування попередньо натренованих мовних моделей із доменною схемною інформацією дає змогу досягати високої точності побудови запитів на низці бенчмарків й істотно знижує бар'єр входження до складних аналітичних систем для нефахівців з баз даних.

На цій основі провайдери корпоративних аналітичних платформ інтегрують генеративні компоненти безпосередньо у BI-середовища, формуючи парадигму розмовної аналітики. Зокрема, Tableau розвиває лінійку агентних сервісів [21], орієнтованих на підтримку повного циклу роботи з даними – від виявлення релевантних джерел до побудови візуалізацій у режимі діалогу [13, 21]. Microsoft позиціонує Copilot in Power BI, як вбудований мовний інтерфейс для побудови звітів, візуалізацій та модифікації моделей даних у відповідь на запити природною мовою бізнес-користувачів [12]. Аналогічно, у хмарних платформах даних з'являються сервіси на кшталт Snowflake Cortex Analyst, які забезпечують формування запитів та інсайтів на підставі моделей LLM поверх корпоративних дата-моделей [13].

Попри значний прогрес, більшість таких рішень мають характер вертикальних, платформо-специфічних інтеграцій. Кожне BI-середовище або дата-платформа формує власний набір конекторів, примусово зв'язуючи конкретну LLM-модель з локальними сховищами, каталогами та інструментами. Це відтворює класичну N×M проблему інтеграції, коли із зростанням кількості моделей і сервісів різко збільшується обсяг інженерної роботи з підтримки окремих конекторів, ускладнюється забезпечення наскрізного управління доступами й аудитом, а також фрагментується керування контекстом на рівні підприємства.

У відповідь на ці обмеження запропоновано Model Context Protocol (MCP) як відкритий протокол, що стандартизує взаємодію між мовними моделями, клієнтськими застосунками та зовнішніми ресурсами. Протокол MCP реалізує клієнт-серверну архітектуру, у якій MCP-клієнт, вбудований у хост-застосунок з моделей LLM, під'єднується до множини MCP-серверів, що репрезентують бази даних, сервіси, інструменти або файли, а обмін відбувається через стандартизовані повідомлення з описом ресурсів та "tools" [2]. Така модель істотно зменшує інженерні витрати на побудову N×M-інтеграцій, оскільки дає змогу різним моделям використовувати однакові MCP-сервери, а різним застосункам – спільний стек серверів контексту.

Ключовою властивістю такого підходу є перетворення класичного завдання інтеграції з типу M×N у M+N: замість розроблення M×N окремих конекторів

для кожної пари "AI-застосунок – джерело даних" достатньо реалізувати M MCP-клієнтів (для різних застосунків) і N MCP-серверів (для різних джерел), після чого будь-яка сумісна мовна модель може використовувати ці сервери без додаткових змін [2]. З погляду корпоративної архітектури протоколу, MCP доцільно розглядати як стандартизований проміжний інтеграційний шар між агентами на підставі моделей LLM та сукупністю платформ управління даними й сервісних систем, подібно до того, як HTTP-API нормалізують взаємодію між вебзастосунками та бекенд-сервісами, а Language Server Protocol уніфікує взаємодію інтегрованих середовищ розроблення з мовними інструментами.

У межах сучасної наукової парадигми активно досліджують потенціал великих мовних моделей (LLM), протоколу Model Context Protocol (MCP) та архітектури Data Mesh для підвищення ефективності роботи з даними в розподілених середовищах. Зокрема, у роботі [5] Dietz, Wider і Harrer презентують Governance AI – інструмент на підставі великих мовних моделей (LLM), що автоматизує оцінювання запитів на доступ до даних відповідно до політик компаній та нормативних актів, формалізуючи процедури перевірки відповідності в умовах федеративних архітектур; однак, попри дотичність до принципів Data Mesh, дослідження не охоплює ані протокол Model Context Protocol (MCP), ані концепцію платформ самообслуговування для доменної взаємодії. У роботі [11] розглянуто розширення Data Mesh на підставі технологій федеративного навчання, акцентуючи увагу на захисті локальних даних у мультидоменних середовищах, проте без механізмів протокольної маршрутизації запитів.

Натомість дослідження [10] спрямоване на формування уніфікованої багатокомпонентної агентної архітектури, у якій Model Context Protocol (MCP) виконує роль стандарту для керування контекстом і координації між агентами, забезпечуючи масштабовану та контекстно-чутливу взаємодію; водночас у роботі не розглядають інтеграції MCP з принципами Data Mesh чи моделлю доменних дата-продуктів. Подібно, у роботі [8] Hou, Zhao, Wang і Wang систематизують можливі загрози безпеці в MCP-середовищах та описують ландшафт протоколу, однак поза контекстом інтеграції з самообслуговуючими платформами чи LLM-агентами. Аналіз [7] Goedegebuure та співавторів представляє систематичний огляд Data Mesh, проте не містить згадок про MCP або контекстно-орієнтовані протоколи. У роботі [15] описують проєкт Gorilla як платформу LLM-доступу до API, що підтверджує потенціал LLM-агентів до роботи з великим числом зовнішніх інтерфейсів, однак позбавлена концептуальної інтеграції з MCP чи Data Mesh.

Отже, проведений аналіз дає змогу зробити висновок, що наразі у наукових джерелах відсутні комплексні дослідження, що безпосередньо поєднують архітектуру системи організації даними Data Mesh із протоколом Model Context Protocol та LLM-агентами в єдину контекстно-орієнтовану модель організації даних у мультидоменних розподілених проєктних командах. Це підтверджує наукову новизну обраної теми та обґрунтовує потребу в реалізації дослідження. Запропонований синтез Data Mesh, MCP та LLM-агентів має підвищити якість роботи з дата-продуктами, забезпечити доменні команди релевантними даними для ухвалення управлінських і бізнес-рішень, а також підтримати формування нових дата-продуктів у межах платформи самообслуговування.

Матеріали та методи дослідження. Матеріалами дослідження слугували сучасні наукові праці та протокольні специфікації, в яких розглянуто систему організації даних Data Mesh, контрактно-орієнтовані підходи до організації даних, протокол Model Context Protocol (MCP) та агентні рішення на підставі великих мовних моделей (LLM); загалом проаналізовано понад 20 закордонних джерел. У роботі використано результати аналізу фундаментальних концепцій доменно-орієнтованого володіння даними та трактування даних як продукту (data as a product), моделей федеративного обчислювального управління (federated computational governance), а також підходів до специфікації протоколів прикладного рівня для доступу до даних і сучасних механізмів інтеграції LLM-агентів зі зовнішніми сервісами. Додатковим емпіричним матеріалом стали узагальнені архітектурні моделі реальних багатодоменних організацій, у яких організація даних базується на виділених доменних командах і розподілених проєктних портфелях із високим ступенем географічної та функціональної розподіленості.

Методологія дослідження поєднує архітектурний і системний аналізи, формальне моделювання інтеграційної складності та сценарне моделювання роботи архітектури в мультидоменному середовищі. Спершу виконано порівняльний аналіз централізованих архітектур даних і Data Mesh за параметрами централізації, автономії доменів та механізмів управління доступом. Далі сформовано інтеграційну модель, у якій архітектура Data Mesh доповнюється протокольним шаром MCP як "контекстною тканиною" між доменними дата-продуктами та платформою самообслуговування даних: LLM-агенти через MCP отримують стандартизований доступ до баз даних, каталогів, дата-контрактів, API та документних сховищ, використовуючи природно-мовні запити, що істотно спрощує доступ до даних для учасників доменних команд та зменшує інтеграційну складність та усуває потребу в прямому технічному зв'язуванні компонентів.

Результати дослідження та їх обговорення / Research results and their discussion

Запропонована методологія дала змогу оцінити властивості архітектури типу "Data Mesh + MCP + LLM-агенти" як цілісної операційної моделі та формально обґрунтувати її здатність забезпечувати контекстно-орієнтований доступ до доменних даних, зменшувати інтеграційну складність і підтримувати масштабування в багатодоменних організаціях.

Оцінювання інтеграційної складності архітектури. У подальшому розглядають велику географічно та організаційно розподілену архітектуру, у якій корпоративний ландшафт даних структуровано відповідно до парадигми моделі Data Mesh (доменні дата-продукти (data products), контракти на дані (data contracts), платформа самообслуговування даними (self-serve data platform)), тоді як доступ до цих ресурсів здійснює множина клієнтів на підставі великих мовних моделей (LLM-застосунки, агентні інтерфейси). Метою формалізованої

моделі є порівняння інтеграційних витрат у класичній схемі прямої інтеграції моделей LLM із доменними сервісами та в цільовій архітектурі типу "Data Mesh + MCP + LLM-агенти".

Щоб оцінити ефективність цієї архітектури, проведено оцінювання інтеграційної складності архітектури. Введено множини клієнтських застосунків на підставі великих мовних моделей (M) і доменних MCP-серверів (N). Модель передбачає параметризацію витрат на окремі інтеграції та протокольні компоненти. Для класичної моделі прямих інтеграцій витрати масштабуються як O(M×N), тоді як для архітектури з протокол MCP – як O(M+N). Отримано формальні вирази для обчислення сумарних витрат, інтеграційного виграшу та його нормованої оцінки.

Сценарне моделювання ґрунтується на прикладі багатодоменної проєктної екосистеми, де агент отримує параметри запиту з реєстру контексту, ідентифікує релевантні домени, звертається до MCP-хостів та виконує операції з протокольно описаними ресурсами. Такий підхід дає змогу формально змоделювати відповідність між архітектурними компонентами та організаційною структурою.

Для забезпечення наукової відтворюваності: зафіксовано ключові припущення (наявність функціональної Data Mesh-інфраструктури, лінійне масштабування, адитивний характер витрат); наведено формальні означення понять (домен, дата-продукт, контракт на дані, протокол MCP-ресурс, протокол MCP-інструмент, моделей LLM-агент); описано алгоритм формування сценаріїв і обчислень; наведено подання інтеграційних витрат і виграшу. Приклад зменшення інтеграційної складності з M×N до M+N завдяки використанню протоколу MCP наведено на рис. 1.

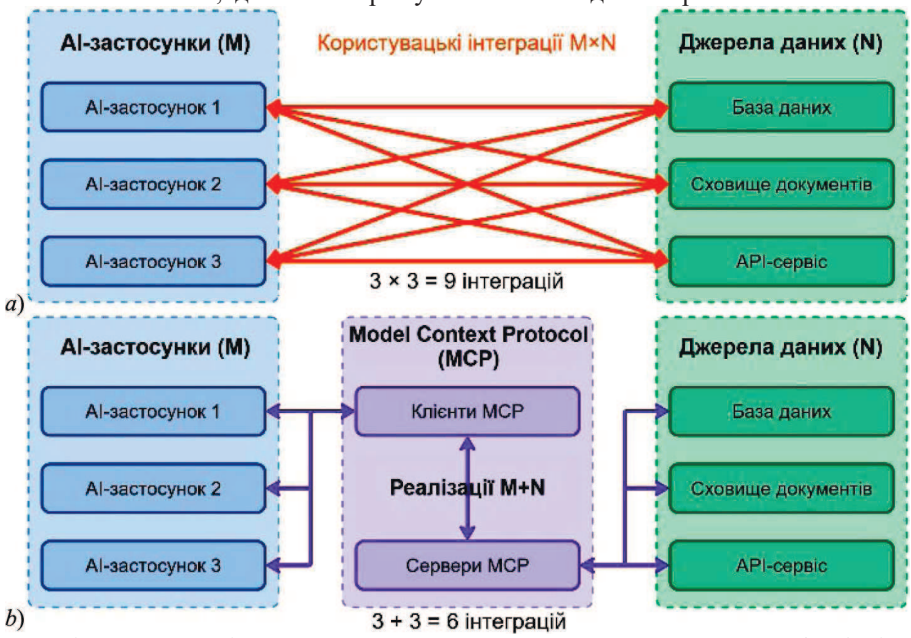

Рис. 1. Зменшення інтеграційної складності з M×N до M+N завдяки використанню протоколу MCP / Reduction of integration complexity from M×N to M+N through the adoption of the MCP protocol: *a*) до впровадження MCP, проблема інтеграції M×N / Before the introduction of MCP, the problem of M×N integration; *b*) після впровадження MCP, рішення інтеграції M+N / after MCP implementation, M+N integration solution [1]

Основні обмеження моделі пов'язані з абстрагуванням від конкретних реалізацій, припущенням лінійності масштабування та відсутністю емпіричної апробації на виробничих даних. Це визначає напрями подальших етапів дослідження, зокрема побудову дослідних прототипів і їх валідацію за умов реальних багатодоменних організацій.

Застосуємо далі такі позначення:

- $M \in N$ – кількість LLM-клієнтів (LLM-застосунків / агентів, що потребують доступу до доменних даних);
- $N \in N$ – кількість доменних MCP-серверів (доменних безпосередніх точок), кожен з яких агрегує один або кілька доменних дата-продуктів і публікує їх через базові примітиви протоколу MCP (resources, tools, prompts). Отже, N інтерпретується як кількість логічних точок інтеграції, а не як сумарна кількість фізичних структур зберігання даних;
- $P \in N$ – кількість проєктів у портфелі. Це сукупність проєктів, програм та інших робіт, об'єднаних для досягнення спільних стратегічних цілей організації; це динамічний показник, що відображає обсяг роботи, необхідний для реалізації бізнес-стратегії, і може змінюватися залежно від пріоритетів компанії;
- $\overline{M}_{proj} \geq 0$ – середня кількість LLM-клієнтів (ролей/інтерфейсів) на один проєкт, тоді на рівні портфеля $M \approx \overline{M}_{proj} P$.

Введемо усереднені витрати на інтеграцію в часовому та ресурсному вимірах:

- $c_{classic} > 0$ – середні сукупні витрати на реалізацію й супровід одного класичного конектора типу "LLM-застосунок – доменний сервіс";
- $c_{client} > 0$ – середні витрати на реалізацію й супровід одного MCP-клієнта на боці LLM-застосунку;
- $c_{server} > 0$ – середні витрати на реалізацію й супровід одного доменного MCP-сервера (кінцевої точки Data Mesh-домену).

Вважатимемо, що: базова Data Mesh-інфраструктура вже розгорнута, а модель описує саме інтеграційний шар; витрати є адитивними, тобто сумарні інтеграційні витрати пропорційні кількості незалежних інтеграційних компонентів; c_{client} та c_{server} трактують як усереднені величини, що не зростають швидше, ніж лінійно за збільшення кількості безпосередніх точок, тобто доменні команди масштабують сервіси так, щоб граничні вит-

рати на ще один MCP-endpoint залишалися приблизно сталими.

У класичній схемі кожна пара "LLM-застосунок – доменний сервіс" потребує окремого конектора. Кількість конекторів:

$$K_{classic}(M,N) = MN ,$$

а сумарні інтеграційні витрати на горизонті життєвого циклу мають такий вигляд:

$$T_{classic}(M,N) = c_{classic}MN .$$

Отже, інтеграційна складність масштабується як $O(MN)$, де додавання нового LLM-застосунку або нового домену потребує оновлення цілої підматриці конекторів.

У цільовій архітектурі "Data Mesh + MCP + LLM-агенти" кожен LLM-застосунок реалізує один MCP-клієнт, а кожен домен Data Mesh експонує свої дата-продукти через один доменний MCP-сервер (кінцева точка MCP). Кількість реалізацій протокольних компонентів становить

$$K_{MCP}(M,N) = M + N ,$$

а сумарні інтеграційні витрати –

$$T_{MCP}(M,N = c_{client}M + c_{server}N .$$

У цій моделі доменні MCP-сервери повторно використовуються всіма моделей LLM-клієнтами, а інтеграційна складність масштабується лінійно відносно кількості клієнтів та доменних безпосередніх точок, тобто як $O(M+N)$, навіть якщо кожен MCP-сервер агрегує тисячі дата-продуктів.

Визначимо інтеграційний виграш (економію інженерних ресурсів) від переходу до архітектури "Data Mesh + MCP", а саме

$$G(M,N) = T_{classic}(M,N) - T_{MCP}(M,N) .$$

Підставляючи до визначення інтеграційного виграшу явні вирази для сумарних інтеграційних витрат у класичній конфігурації $T_{classic}(M,N)$ та у конфігурації з протоколом MCP $T_{MCP}(M,N)$, отримаємо

$$G(M,N) = c_{classic}MN - (c_{client}M + c_{server}N) .$$

Для портфельного рівня, коли $M \approx \overline{M}_{proj}P$, отримаємо апроксимацію

$$G(P,N) \approx c_{classic}\overline{M}_{proj}PN - (c_{client}\overline{M}_{proj}P + c_{server}N) ,$$

що відображає квадратичне зростання інтеграційних витрат за PN у класичній схемі проти лінійного зростання за P і N в MCP-сценарії.

Для порівняння незалежно від абсолютних значень витрат введемо нормований відносний виграш:

$$R(M,N) = \frac{G(M,N)}{T_{classic}(M,N)} = 1 - \frac{c_{client}M + c_{server}N}{c_{classic}MN} .$$

За фіксованих додаткових констант: $c_{classic}$, c_{client}, $c_{server} > 0$, отримаємо

$$\lim_{M,N\to\infty} R(M,N) = 1 ,$$

тобто частка інтеграційних витрат, що підлягає скороченню завдяки переходу до протоколу MCP як протокольного шару поверх мережі Data Mesh, асимптотично прямує до 100% зі зростанням кількості LLM-клієнтів та доменних MCP-безпосередніх точок. Зокрема, у сценаріях, коли один домен мережі Data Mesh акумулює тисячі або десятки тисяч дата-продуктів, агрегованих в обмежену кількість MCP-серверів, модель залишається коректною: масштабування відбувається за кількістю логічних точок інтеграції N, а не за загальною кількістю фізичних об'єктів зберігання.

У підсумку, запропонована модель формалізує перехід від квадратичної інтеграційної складності $O(MN)$ у класичній схемі прямих інтеграцій моделей LLM із системами організації даних до лінійної складності $O(M+N)$ в архітектурі "Data Mesh + MCP + LLM-агенти", що для великих розподілених організацій означає структурне зниження інтеграційних витрат і підвищення масштабованості інтеграційного шару.

Протокол MCP у мережі Data Mesh: інтеграційний шар підтримки розподілених проєктів. З погляду корпоративної архітектури, протокол MCP виступає формалізованим проміжним шаром інтеграції між агентами на підставі моделей LLM та різнорідною сукупністю платформ даних і сервісних систем. Водночас у сучасній літературі з інженерії даних та управління розподіленими проєктними командами практично відсутні формалізовані моделі поєднання протоколу MCP з доменно-орієнтованими системами організації даних на кшталт мережі Data Mesh та з рамками керованої підтримки проєктних процесів. Отже, залишається відкритим питання, як саме протокольно описані ресурси й інструменти протоколу MCP можуть бути системно пов'язані з дата-продуктами, дата-контрактами й процесними артефактами у розподілених проєктних командах, забезпечуючи при цьому формальний, контекстно-орієнтований та керований доступ до них. Саме на заповнення цього розриву спрямована подальша частина роботи.

Сучасні емпіричні дослідження розподілених команд фіксують стійкий набір викликів, пов'язаних із часовою розподіленістю учасників, асинхронною взаємодією, культурною та мовною різнорідністю, а також високою залежністю від цифрових засобів співпраці [19]. Узагальнення кейсів глобального розроблення програмного забезпечення показує, що ключовими детермінантами результативності таких команд є узгодженість комунікацій, прозорість спільного контексту та гарантована доступність актуальних проєктних артефактів – беклогу, специфікацій, рішень з архітектури, реєстрів ризиків, показників продуктивності та якості [22, 25].

Професійні звіти в галузі управління проєктами підтверджують, що фізична колокація учасників поступово втрачає статус визначального чинника успіху. Натомість критичного значення набувають зрілість цифрової екосистеми організації, якість спільних даних і стандартизованість проєктних процесів. Актуальні аналітичні матеріали PMI на підставі серії звітів "Pulse of the Profession" демонструють, що організації, які системно інвестують у цифрові інструменти співпраці, інтегровані сховища проєктних даних та аналітичну підтримку прийняття рішень, досягають вищих показників дотримання термінів, бюджетів і бізнес-цілей у гібридних та розподілених моделях роботи, порівняно з командами, що спираються переважно на традиційні офлайнові практики [18].

Для транснаціональних компаній це формує чіткі вимоги до інформаційних технологій супроводу проєктних процесів. Попри інструментальну фрагментованість (Jira, Azure DevOps, GitHub, Confluence, системи моніторингу, корпоративні фінансові та ERP-платфор-

ми) та розподіл відповідальності між доменами, організація потребує єдиного узгодженого контексту проєкту [22, 23]. Такий контекст має забезпечувати оперативний доступ до релевантної інформації з урахуванням ролі користувача (керівник проєкту, системний архітектор, бізнес-аналітик, інженер тощо), формування відтворюваного ланцюга трасування від стратегічних бізнес-цілей до конкретних робочих елементів, інтегроване подання операційних та аналітичних даних, а також систематизоване виявлення й моніторинг ризиків та міжкомандних залежностей на рівні портфеля проєктів [23].

Синтез результатів аналізу централізованих архітектур організації даних, системи організації даних мережі Data Mesh, а також сучасних практик інтеграції моделей LLM/AI у дата-платформи та специфіки управління розподіленими проєктними командами дає змогу сформулювати системні вимоги до цільової системи організації даних, орієнтованої на підтримку проєктних процесів у географічно, функціонально та часово розподілених середовищах.

По-перше, така система має зберігати ключові переваги парадигми мережі Data Mesh у частині доменної автономії та контрактно-орієнтованої взаємодії між командами. Дані мають залишатися структурованими у вигляді дата-продуктів із чітко визначеними доменними власниками, розгорнутими метаданими, угодами про рівень обслуговування SLA (англ. *Service Level Agreement*), формальними схемами, політиками доступу та явно визначеними data contracts між продюсерами й споживачами; саме така постановка створює передумови для масштабованості та когерентності оброблення даних у багатодоменному середовищі [4, 17].

По-друге, цільова архітектура організації даних має гарантувати глобальну узгодженість через федеративне управління. Політики "як код" (policy-as-code), спільні глосарії та стандартизовані механізми контролю якості мають бути інтегровані в життєвий цикл дата-продуктів, не нівелюючи при цьому гнучкість доменних команд. У сучасних моделях впровадження мережі Data Mesh рівень governance розглядають як сервісний прошарок платформи, що надає доменам шаблони контрактів, механізми перевірки відповідності та інструменти спостережності, а не тільки як централізовану регулятивну інстанцію [4, 17].

По-третє, з огляду на специфіку розподілених проєктних команд, доступ до даних та артефактів повинен бути контекстно орієнтованим: кожна операція над даними має інтерпретуватися в координатах конкретного проєкту, домену, ролі, фази життєвого циклу та релевантного набору політик безпеки й відповідності. Це передбачає явне моделювання проєктного контексту як формалізованої структурованої сутності, що супроводжує запити до дата-продуктів і сервісів та визначає простір допустимих операцій.

По-четверте, self-service-рівень не може обмежуватися тільки data-інженерами та вузькоспеціалізованими аналітиками; він має бути доступним для керівників проєктів, доменних експертів, архітекторів, product owners та інших ключових стейкхолдерів. У цьому контексті агенти на підставі великих мовних моделей (моделей LLM) виступають природним універсальним інтерфейсом до розподілених ресурсів, оскільки дають змогу формулювати запити мовою предметної області (prompts), а не мовою низькорівневих технічних інструкцій. Сучасні дослідження в галузі text-to-SQL та природно-мовних інтерфейсів до баз даних показують, що моделей LLM-орієнтовані підходи здатні істотно знизити бар'єр доступу до складних аналітичних екосистем для нефахівців із даних (наприклад, DB-GPT та інші моделей LLM-засновані системи для взаємодії з СУБД, а також бенчмарки text-to-SQL, побудовані на великих мовних моделях).

По-п'яте, інтеграція LLM-агентів у Data Mesh-екосистему потребує стандартизованого протоколу взаємодії з доменними сервісами, дата-продуктами та проєктними артефактами, який усуває потребу в множині кастомних конекторів і ad-hoc-розширень. З огляду на свою архітектуру Model Context Protocol, запропонований як відкритий стандарт для під'єднання мовних моделей до зовнішніх інструментів, API та джерел даних, можна розглядати як перспективний кандидат на таке технологічне ядро, оскільки описує уніфіковане подання ресурсів та інструментів і стандартизований спосіб їх виклику з боку LLM-агентів [2].

Model Context Protocol (MCP) визначається як відкритий стандарт інтеграції між великими мовними моделями (моделей LLM), клієнтськими застосунками та зовнішніми системами даних і сервісами, що забезпечує модель-агностичний спосіб під'єднання агентів до корпоративної інфраструктури. Базова архітектура протоколу розрізняє три ролі: MCP-хост, MCP-клієнт, MCP-сервер.

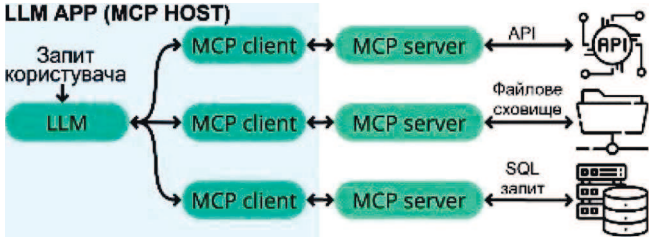

Рис. 2. Архітектура протоколу контексту моделі MCP: взаємодія LLM-застосунку з MCP-клієнтами та MCP-серверами зовнішніх ресурсів / Architecture of the Model Context Protocol (MCP): interaction between the LLM application, MCP clients, and MCP servers exposing external resources [14]

MCP-хост – це середовище виконання агента, у межах якого відбувається взаємодія користувача з моделей LLM. MCP-клієнт працює всередині host-а як реалізація протоколу: встановлює сесії, реєструє під'єднані сервери, маршрутизує запити агентів і нормалізує відповіді. MCP-сервер репрезентує зовнішнє джерело даних або сервіс (систему керування базами даних, файлове сховище, доменний програмний інтерфейс (API), систему керування завданнями) та інкапсулює їхню функціональність у вигляді стандартизованого набору примітивів протоколу MCP (рис. 2).

На рівні обміну даними протоколу MCP використовує JSON-RPC 2.0 як формат запитів, відповідей і нотифікацій, чітко розділяючи рівень протокольних повідомлень від транспортного рівня. Як транспорт офіційно підтримуються, зокрема, локальні stream-з'єднання через STDIO (типовий кейс для CLI/IDE-інструментів) та мережеві варіанти на базі HTTP(S) для віддалених серверів, включно із хмарними оточеннями. Така стратифікація дає змогу реалізовувати єдиний набір протокольних примітивів протоколу MCP у локальних,

гібридних та хмарних конфігураціях без зміни семантики повідомлень.

Архітектура та семантика протоколу MCP. З погляду корпоративної архітектури, протокол MCP функціонує як стандартизований проміжний інтеграційний шар між LLM-агентами та різнорідним ландшафтом дата-платформ і сервісних систем – за роллю аналогічний до HTTP-API у веб-архітектурах або протокол мовного сервера LSP (англ. *Language Server Protocol*) в IDE: еволюція моделей і бекенд-сервісів розв'язується, а інтеграційна зв'язаність у масштабі підприємства зменшується. У подальшій архітектурі "Data Mesh + MCP" саме цей шар виступатиме точкою входу LLM-агентів до доменних дата-продуктів.

Семантика протоколу MCP задається набором базових примітивів, які визначають, які дані агент може спостерігати та які дії виконувати щодо зовнішніх систем. На стороні серверів ключовими є такі примітиви як Resources, Tools та Prompts.

Resources інтерпретуються як формалізовані read-only репрезентації даних, що охоплюють окремі файли, документи, директорії, віртуальні табличні подання та результати виконання запитів. Специфікація протоколу MCP задає стандартизований набір операцій над ресурсами – отримання переліку доступних ресурсів, читання їх вмісту та, за потреби, підписку на зміни. У запропонованій цільовій моделі Data Mesh-ресурси виступають безпосереднім відображенням доменних дата-продуктів, знімків (snapshots) проєктних артефактів і агрегованих операційних та бізнес-метрик.

У специфікації MCP-інструменти розглядають як формально визначені операції виклику зовнішніх сервісів, що виконуються з параметрами та можуть змінювати стан підлеглих систем (зокрема, ініціювати виконання запитів до СУБД, реєструвати елементи в системі керування беклогом, оновлювати записи в реєстрі ризиків чи запускати конвеєри контролю якості даних). Кожен tool задається строго типізованою схемою вхідних параметрів та очікуваною структурою відповіді, а MCP-сервер зобов'язаний явно декларувати модифікації стану даних та зовнішніх систем, які здійснюються під час виконання відповідної операції. Така формалізація створює передумови для безпечного, відтворюваного й керованого використання LLM-агентів у промислових середовищах і узгоджується з архітектурними вимогами протоколу MCP щодо прозорості, контрольованості та інтеграції з механізмами корпоративного governance та аудиту.

Prompts у специфікації протоколу MCP трактуються як формалізовані шаблони взаємодії (prompt templates), що інкапсулюють інструктивну частину, релевантний контекст і параметри користувача у вигляді структурованого запиту до моделей LLM. MCP-сервер може публікувати реєстр таких шаблонів, прив'язаний до типізованих класів завдань (зокрема формування інтегрованого огляду поточного стану проєкту, аналітичної оцінки ризик-профілю релізу, генерування узагальнених управлінських звітів), що забезпечує відтворюваність, стандартизацію та керованість сценаріїв використання агентів у проєктних процесах.

На боці клієнта специфікація протоколу MCP додатково визначає примітиви кореневих областей робочого простору (roots), процедур вибірки відповіді моделі (sampling) та еліцитації контексту (elicitation). Примітив roots задає множину робочих директорій, з якими конкретний MCP-сервер має права взаємодіяти; у разі файлових серверів ці області, зазвичай, описуються у разі URI зі схемою file:// і використовуються клієнтом для координації доступу серверів до локальних ресурсів, фіксуючи межі простору ресурсів, що можуть бути ними оброблені. При цьому механізми автентифікації й авторизації залишаються на рівні операційної системи та корпоративної інфраструктури керування ідентичністю та доступом (IAM), а roots тільки надають протокольний засіб формального оголошення дозволених робочих областей. Примітив sampling формалізує спосіб, у який клієнт ініціює запити до мовної моделі з урахуванням доступних інструментів, політик виконання та режимів потокової відповіді. Примітив elicitation описує діалогові схеми збирання додаткового контексту від користувача (уточнення вимог, параметрів завдання, вибір сценарію) перед ініціюванням викликів tools або формуванням фінальної відповіді агента.

Сукупно примітиви протоколу MCP формують чітко визначений, технологічно нейтральний інтерфейс взаємодії між LLM-агентами та Data Mesh-екосистемою в межах запропонованої моделі, формалізований у вигляді протокольного контракту: доменні дата-продукти відображаються на resources, операції, задані data contracts, моделюються як tools, а типові сценарії взаємодії проєктних ролей – як prompts. Така відповідність створює основу для формально визначеної, контекстно-орієнтованої інтеграції протоколу MCP в архітектуру організації даних розподілених проєктних команд.

Як цільове рішення пропонують архітектуру організації даних для розподілених проєктних команд, у якій принципи мережі Data Mesh поєднано з протокольною моделлю MCP. Отже, протокол MCP виступає не окремим інтеграційним "додатком", а керованим операційним фронтендом мережі Data Mesh. На рівні даних підтримується типовий для мережі Data Mesh спосіб організації: доменні дата-продукти, закріплені за окремими product-, market- та region-domains, визначаються формальними схемами, SLA, політиками доступу та data contracts, що регламентують взаємодію між продуцентами та споживачами. Принципова відмінність запропонованої моделі полягає в тому, що кожний доменний дата-продукт має формалізоване відображення у примітивах протоколу MCP: подання та агрегати даних тільки для читання (read-only) інтерпретуються як ресурси (resources), операції, визначені дата-контрактами (зокрема оновлення реєстрів ризиків, модифікація беклогу, ініціювання пайплайнів контролю якості даних), задаються як інструменти (tools), а типові сценарії взаємодії проєктних ролей (огляд статусу релізу, аналіз впливу затримок, формування підсумкового executive-звіту) – як шаблони запитів/підказок (prompts). Сукупно це формує формально визначений, технологічно нейтральний інтерфейс між LLM-агентами та Data Mesh-екосистемою, у якому доменна автономія зберігається, але доступ до дата-продуктів і артефактів стає контекстно-орієнтованим і уніфікованим.

На рівні компонентної архітектури інтеграція реалізується через виокремлення двох ключових шарів: доменних серверів протоколу MCP (доменні MCP-серве-

ри) та федеративного шару MCP-хост (federated MCP-хост layer). Доменний MCP-сервер виконує функцію адаптера між сервісом доменного дата-продукту (англ. *Domain Data Product Service* – сервісом, що реалізує операції створення, читання, модифікації та видалення даних (CRUD), аналітичні подання, показники якості та метадані конкретного дата-продукту) та специфікацією протоколу MCP. Операції доступу тільки для читання й агрегування даних формально інтерпретуються та оголошуються як ресурси (resources), операції зміни стану – як інструменти (tools), а повторювані аналітичні завдання предметної області – як шаблони запитів/підказок (prompts).

Федеративний MCP-хост-рівень інтегрується з користувацькими інтерфейсами (веб-портали, інтегровані середовища розроблення, корпоративні чат-клієнти), реалізує клієнт протоколу MCP (MCP-клієнт), під'єднує множину доменних серверів та взаємодіє з корпоративною інфраструктурою керування ідентичностями (Identity and Access Management, IAM) і рушієм реалізації політик як коду (policy-as-code engine). На цьому рівні відбувається застосування політик доступу, маршрутизація запитів агентів до відповідних доменів, протокольне журналювання викликів до ресурсів та інструментів (resources/tools) і однозначне зіставлення кожної операції з конкретним проєктним контекстом.

Окремою складовою частиною запропонованої архітектури є запровадження централізованого реєстру проєктного контексту (англ. *Project Context Registry*), який фіксує опис проєкту у формі формалізованої структурованої сутності. Такий реєстр містить, зокрема, унікальний ідентифікатор проєкту, перелік доменів відповідальності, стадію життєвого циклу, рольову модель учасників, а також посилання на пов'язані доменні дата-продукти та проєктні артефакти (беклоги, реєстри ризиків, журнали інцидентів, показники продуктивності й якості). Project Context Registry виконує роль канонічного джерела істини щодо контексту проєкту, відокремлюючи його моделювання від технічних деталей конкретних платформ даних.

У запропонованій моделі агенти на підставі великих мовних моделей не звертаються до доменних дата-продуктів безпосередньо; натомість вони спершу отримують із реєстру формалізований проєктний контекст, на його основі визначають релевантні домени, обирають відповідні MCP-сервери та тільки після цього ініціюють виклики до конкретних ресурсів (resources) та інструментів (tools). Така послідовність взаємодії забезпечує контекстно-орієнтований доступ, у якому кожна операція інтерпретується в координатах "проєкт – домен – роль – стадія життєвого циклу", і водночас задає формальний механізм обмеження простору допустимих дій. Це, водночас, є критично важливим для керованого використання LLM-агентів у мультидоменному середовищі розподілених організацій, де потрібно поєднати автономію доменів із строгими вимогами до безпеки, відповідності політикам та простежуваності операцій.

Запропонована в роботі архітектурна модель організації даних розглядає поєднання Data Mesh, Model Context Protocol (MCP) та агентів на підставі великих мовних моделей (LLM-агентів) не як сукупність окремих технологічних інновацій, а як цілісну операційну рамку системи організації даних для розподілених проєктних команд. Ключовий внесок полягає у формалізації зв'язку між доменними дата-продуктами (data products), їхніми контрактами на дані (data contracts) та інтелектуальними агентами через протокольно визначені інтерфейси, у межах яких контекст проєкту (project context) виступає сутністю першого класу (first-class entity) й задає простір допустимих операцій над даними. У такій постановці архітектура організації даних не є черговою варіацією сховища даних чи платформи бізнес-аналітики (business intelligence, BI-платформи), а пропонує альтернативну модель корпоративної архітектури даних (enterprise data architecture) для організацій, що функціонують у мультидоменному, географічно та часово розподілених середовищах.

Порівняно із класичними централізованими архітектурами даних (Data Warehouse, Data Lake, Lakehouse), запропонований підхід відходить від єдиної централізованої платформи як основної точки інтеграції та контролю. Централізовані рішення залишаються ефективними для побудови єдиного джерела істини та пакетної аналітики (batch analytics), проте їхня організаційна модель неминуче формує вузьке місце масштабування у великих багатодоменних портфелях проєктів. Мережа Data Mesh, кристалізувавши принципи доменно-орієнтованого володіння даними (domain-oriented ownership), трактування даних як продукту (data as a product), федеративного обчислювального управління (federated computational governance) та платформи самообслуговування даними (self-serve data platform), уже пропонує децентралізовану альтернативу. Водночас навіть зріла реалізація мережі Data Mesh часто залишає нерозв'язаною "останню милю" між доменними дата-продуктами (data products) та ключовими проєктними ролями, які змушені взаємодіяти з даними через низькорівневі технічні інтерфейси (low-level technical interfaces), а не через контекстно усвідомлений, предметно орієнтований шар доступу до даних.

Відмінність від платформно-специфічних інтеграцій моделей LLM із BI- та хмарними середовищами (наприклад, Copilot in Power BI / Fabric, Tableau Agent, Snowflake Cortex Analyst) полягає в тому, що запропонована архітектура системи організації даних "Data Mesh + MCP +LLM-агенти" не фіксує агентів усередині одного стека технологій (technology stack). Продукти класу розмовної бізнес-аналітики (conversational business intelligence) демонструють високу функціональність у межах конкретної платформи, але жорстко прив'язують контекст, політики доступу, семантику даних і можливості агентів до локальної моделі даних. У запропонованій моделі "Data Mesh + MCP + LLM-агенти" великі мовні моделі розглядають як технологічно нейтральний інтерфейсний та інтеграційний шар над сукупністю доменних дата-продуктів (data products), незалежно від того, чи реалізовані вони у форматі lakehouse, колонкового сховища, потокової платформи оброблення даних (stream processing platform) або спеціалізованих сервісів. Структурування екосистеми відбувається не навколо окремого вендорного середовища, а навколо доменно-визначених дата-продуктів (data products) та їхніх контрактів на дані (data contracts).

Ще виразніший контраст спостерігають щодо поширеної практики впровадження моделей LLM-чат-асистентів без формалізованого протокольного шару. У та-

ких рішеннях інтеграція з корпоративними системами реалізується як набір індивідуально розроблених конекторів (custom connectors) до програмних інтерфейсів застосунків (API), файлових сховищ чи окремих сервісів; логіка доступу, безпеки та управління (governance) строго зашита у програмний код й інженерію підказок (prompt engineering). Це відтворює класичну проблему M×N: для M агентів і N джерел потрібна підтримка M×N зв'язків, що обмежує масштабованість й ускладнює аудит. Протокол MCP як відкритий протокол прикладного рівня (application-layer protocol) на підставі JSON-RPC 2.0 задає модель M+N, у якій LLM-застосунки реалізують M клієнтів, а доменні системи – N серверів, і будь-який сумісний агент може працювати з будь-яким оголошеним MCP-сервером без додаткових з'єднувачів. Через примітиви ресурсів (resources), інструментів (tools) і шаблонів підказок (prompts) протокол MCP формалізує контракт між агентами та доменними дата-продуктами (data products): ресурси репрезентують доступні подання даних, інструменти – дозволені операції зі зміни стану або запуску конвеєрів оброблення даних (data pipelines), шаблони підказок – стандартизовані сценарії взаємодії. У поєднанні з мережею Data Mesh це дає змогу явно прив'язати доменні дата-продукти (data products) та контракти на дані (data contracts) до протоколу MCP-ресурсів і протоколу MCP-інструментів, тим самим переводячи інтеграцію із площини ситуативних рішень у площину протокольно визначених інтерфейсів, у які вбудовано федеративне обчислювальне управління (federated computational governance) через політики доступу, що застосовуються на рівні MCP-хост.

У цьому контексті платформа самообслуговування даними (self-serve data platform) набуває додаткового, принципово нового виміру. У класичному вимірі Data Mesh платформа самообслуговування даними (self-serve data platform) переважно розглядається як інфраструктурно-інструментальний шар для доменних команд даних: каталоги даних, конвеєри оброблення (data pipelines), стандартизовані сервіси зберігання та аналітичного оброблення. Запропонована архітектура "Data Mesh + MCP + LLM-агенти" зміщує фокус до платформної конфігурації типу agent-aware self-serve data platform, у межах якої платформа самообслуговування даними постає не тільки набором технічних сервісів, а керованою "точкою входу" до Data Mesh. MCP-хост, під'єднаний до множини доменних MCP-серверів, у поєднанні з реєстром контексту проєкту (Project Context Registry) та агентами на підставі великих мовних моделей (LLM-агенти), формує операційний фронтенд, вбудований у стандартну траєкторію доступу до доменних дата-продуктів (data products). Для проєктних ролей, зокрема керівників проєктів (project managers), архітекторів програмних та інформаційних систем (software/system architects), бізнес-аналітиків (business analysts), режим самообслуговування вже не зводиться до самостійного конструювання запитів чи інформаційних панелей (dashboards), а реалізується як керований, протокольно визначений інтерфейс, у якому агенти, спираючись на контекст проєкту (project context), автономно відшукують релевантні доменні дата-продукти (data products), викликають узгоджені інструменти (tools) та повертають узагальнені результати у формі, придатній до операційного використання. За такої конфігурації платформа самообслуговування даними (self-serve data platform) еволюціонує з інструментального шару для інженерів з даних (data engineers) до "agent-aware" платформи, у межах якої агенти на підставі великих мовних моделей (LLM-агенти) виступають регульованими посередниками між користувачами, доменними даними та політиками управління даними (data governance policies).

Концептуально важливим є трактування контексту проєкту (project context) як сутності першого класу. У класичних реалізаціях Data Mesh домени та доменні дата-продукти (data products) є базовими об'єктами моделювання, тоді як проєкти відображаються переважно через процеси управління роботами. У запропонованій моделі запроваджено реєстр контексту проєкту (Project Context Registry), який формалізує проєкт як структуровану одиницю, що містить ідентифікатор, перелік релевантних доменів, фазу життєвого циклу, рольову модель, посилання на артефакти та ключові показники діяльності (performance metrics). Саме контекст проєкту (project context) визначає, які MCP-сервери можуть бути під'єднані, які ресурси (resources) є доступними для читання, які інструменти (tools) дозволено викликати та як саме результати мають бути інтерпретовані для конкретної ролі. Така інверсія координат зміщує акцент із суто інженерного подання даних до операційної підтримки життєвого циклу проєктів і відкриває шлях до агентно-орієнтованої аналітики (agentic analytics), у межах якої аналітичні дії агентів прив'язані до формалізованих контекстів і політик управління.

У цій конфігурації агенти на підставі великих мовних моделей (LLM-агенти) перестають бути факультативною "чат-надбудовою" над окремими сховищами чи інформаційними панелями (dashboards). Протокольні контракти протоколу MCP та контекст проєкту (project context) визначають їхні повноваження, обмеження та зони відповідальності: агенти працюють винятково в межах дозволених ресурсів (resources) та інструментів (tools), кожна операція реєструється в журналі подій з прив'язкою до проєктного та рольового контексту, а федеративне обчислювальне управління (federated computational governance) фіксує допустимі сценарії використання для різних категорій користувачів. У термінах корпоративної архітектури (enterprise architecture) агенти на підставі великих мовних моделей (LLM-агенти) набувають статусу керованих, відтворюваних і таких, що підлягають аудиту, компонентів екосистеми на підставі Data Mesh, інтегрованих у платформу самообслуговування даними (self-serve data platform) як її логічний інтерфейсний шар.

Позиціювання запропонованої архітектури "Data Mesh + MCP + LLM-агенти" щодо альтернативних підходів можна узагальнити у вигляді таблиці. Така архітектура окреслює траєкторію переходу від централізованих сховищ даних і вузько платформо-орієнтованих інтеграцій моделей LLM до контекстно-орієнтованої, протокольно керованої мережі Data Mesh, у межах якої LLM-агенти функціонують як керовані учасники розподіленої екосистеми, а платформа самообслуговування даними (self-serve data platform) виступає орієнтованим на агентів (agent-aware) фронтендом до доменних дата-продуктів (data products). Для великих розподілених організацій це створює можливість одночасно знизити ін-

тераційні витрати, пришвидшити доступ до релевантного контексту проєкту (project context) та переосмислити організацію проєктної діяльності як сукупність формально визначених, керованих даними (data-driven) взаємодій між доменами, людьми та інтелектуальними агентами.

Таблиця. Порівняльна характеристика підходів до організації даних та інтеграції LLM-агентів / Comparative overview of data-organization approaches and LLM-agent integration models

Вимір	Централізовані архітектури даних (DWH / Data Lake / Lakehouse)	Платформно-специфічні інтеграції моделей LLM / BI	"Data Mesh + MCP + LLM-агенти"
Рівень централізації та доменної автономії	Висока централізація; предметні домени невиразного подання в єдиній глобальній моделі даних.	Часткова децентралізація; контекст і моделі даних обмежені однією аналітичною або хмарною платформою.	Доменна автономія (domain-oriented ownership) у межах Data Mesh із уніфікованими протокольними інтерфейсами доступу
Інтеграція мовних моделей та агентів	Зовнішні програмні боти й окремі конектори до сховища; інтеграція виконується на рівні індивідуальних рішень.	Вбудовані агенти в межах конкретної BI- чи хмарної платформи; тісна прив'язка до внутрішньої семантичної моделі даних.	Model Context Protocol як універсальний протокол прикладного рівня для інтеграції LLM-агентів із доменними сервісами та дата-продуктами (data products)
Платформа самообслуговування даними (self-serve data platform)	Переважно інструментальний шар для аналітиків (BI-звіти, запити, конектори до сховища).	Режим самообслуговування в межах однієї платформи; інструменти зосереджені навколо вендорного середовища.	Платформа самообслуговування даними (self-serve data platform), орієнтована на взаємодію з агентами ("agent-aware"), яка виступає операційною точкою входу до Data Mesh для проєктних ролей.
Управління контекстом (context management)	Корпоративний контекст (enterprise context) задається на рівні централізованого сховища й глобальної моделі даних.	Платформний контекст (platform context), обмежений межами конкретної BI / cloud-платформи.	Формалізований контекст проєкту (project context) як сутність першого класу; визначає під'єднання MCP-серверів, доступні ресурси та інтерпретацію результатів.
Управління даними та аудит (data governance & audit)	Централізоване управління даними (data governance) й аудит на рівні єдиної платформи зберігання.	Управління даними в межах платформи; механізми аудиту поширюються тільки на її сервіси.	Федеративне обчислювальне управління (federated computational governance), інтегроване в протокол MCP-ресурси та протокол MCP-інструменти (resources / tools) із протокольним журналюванням викликів.
Масштабованість інтеграцій	Переважно точкові (ad hoc) інтеграції; схема зв'язків типу M×N (багато-до-багатьох) між системами та інтерфейсами.	Інтеграції, строго прив'язані до постачальника (vendor lock-in) в межах одного технологічного стека.	Зменшення складності інтеграцій до моделі M+N: обмежена кількість MCP-клієнтів і MCP-серверів замість множини парних зв'язків.

Запропоновану архітектуру "Data Mesh + MCP + LLM-агенти" подано у вигляді функціональної моделі (рис. 3), яка формалізує механізм протокольної інтеграції Model Context Protocol (MCP) у платформу самообслуговування даними в межах Data Mesh [24]. Така інтеграція створює передумови для реалізації agent-aware середовища, в якому користувачі взаємодіють з доменними дата-продуктами через LLM-агентів, здатних інтерпретувати контекст запиту, ініціювати виклики до протоколу MCP-ресурсів та повертати результати у формі, релевантній проєктній ролі та поточному сценарію.

З погляду архітектурної стратифікації, модель розширює класичну чотиришарову структуру Data Mesh (Governance, Self-Serve Platform, Domain Layer, Core Infrastructure) новим шаром інтерактивної інтеграції типу протокол MCP + LLM. Цей шар виконує роль семантичного інтерфейсу між користувачем і доменною логікою, забезпечуючи протокольно визначену маршрутизацію запитів через MCP-хост до відповідних MCP-серверів, що відповідають за доступ до контрактів, ресурсів та інструментів оброблення даних.

Як наслідок, реалізується високорівнева модель обчислювального самообслуговування, в межах якої LLM-агенти виступають керованими, відтворюваними й аудитованими посередниками, здатними ініціювати інтелектуальну взаємодію з розподіленими джерелами даних без потреби у глибоких технічних знаннях з боку безпосередніх учасників доменних команд.

Запропоновану функціональну архітектуру "Data Mesh + MCP + LLM-агенти" показано на прикладі міжнародної логістичної компанії з розподіленими проєктними командами (рис. 3), яка формалізує взаємодію між учасниками розподілених проєктних команд і доменними дата-продуктами в екосистемі мережі Data Mesh через протокольне посередництво протоколу контексту моделі (MCP). Протокол MCP виступає як стандартизований проміжний інтерфейсний шар прикладного рівня, що забезпечує узгоджену взаємодію між LLM-застосунками (MCP-хост) та доменними сервісами (MCP-серверами), поданими через єдині абстракції ресурсів (resources), інструментів (tools) і шаблонів підказок (prompts). Самі MCP-клієнти забезпечують масштабоване під'єднання до розподілених систем – зокрема, REST API, баз даних, файлових сховищ та реалізують модель розділення відповідальності між агентами, логікою доступу і джерелами даних.

Завдяки використанню протоколу JSON-RPC 2.0, архітектура протоколу MCP трансформує традиційну складність інтеграцій типу M×N (M агентів до N сервісів) у лінійну модель M+N, де кожен новий LLM-застосунок або дата-сервіс приєднується до системи без потреби в написанні спеціалізованих конекторів. Це забезпечує високу масштабованість, інтероперабельність і незалежність від вендора. Окрім цього, протокол MCP дає змогу формалізувати контракти між LLM-агентами та дата-продуктами, забезпечуючи дотримання політик

доступу, вимог SLA та принципів федеративного обчислювального управління (federated computational governance) на рівні виклику кожного ресурсу.

У запропонованій моделі обчислювального самообслуговування платформа (Self-Serve Platform) постає не тільки як інструментальний рівень для інженерів даних, а як "agent-aware"-середовище з підтримкою контекстно орієнтованої аналітики. LLM-агенти набувають статусу повноцінних, керованих суб'єктів взаємодії з мережею Data Mesh, здатних опрацьовувати запити користувачів неінженерного профілю (PM, BA, аналітиків), знаходити релевантні дата-продукти, виконувати дозволені трансформації та надавати результати користувачам в уніфікованому форматі.

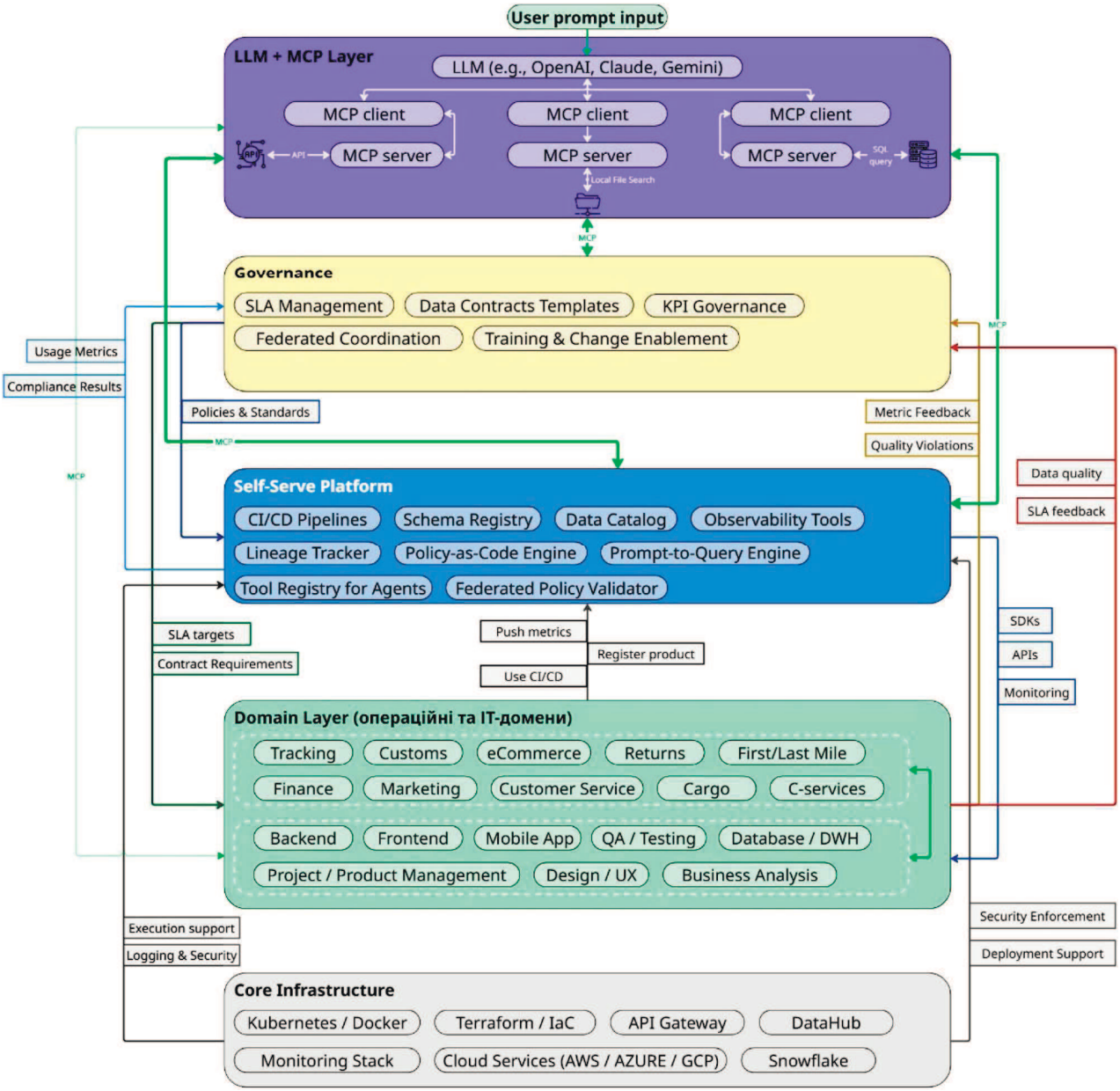

Рис. 3. Архітектура мережі Data Mesh із протокольною інтеграцією протоколу контексту моделі (MCP): підтримка LLM-агентів у контекстно орієнтованій роботі з даними в розподілених проєктних командах / Data Mesh Architecture with Protocol-Based Integration via Model Context Protocol (MCP): Enabling Context-Aware Data Interaction by LLM Agents in Distributed Project Teams [24]

Ключовою інновацією є запровадження реєстру контексту проєкту (англ. Project Context Registry) як сутності першого класу. Це, по суті, база даних або документ, що фіксує всі важливі характеристики, умови, зацікавлені сторони та обмеження проєкту в певний момент часу, допомагаючи зрозуміти його "як і чому" він існує, його цілі, вплив, ресурси та взаємозв'язки з організацією та зовнішнім середовищем, що є критичним для ефективного управління проєктами (УП). Саме контекст проєкту визначає, які доменні MCP-сервери доступні, які ресурси та інструменти дозволено викликати, а також у який спосіб інтерпретувати результати з урахуванням ролі користувача.

Отже, запропонована архітектура мережі "Data Mesh + MCP + LLM-агенти" забезпечує:

- зниження технічного бар'єру для взаємодії з даними;
- уніфікований контроль й аудит усіх викликів;
- підвищення масштабованості інтеграційних рішень;
- формалізацію контексту та ролей у межах аналітичного процесу.

На відміну від класичних централізованих підходів до роботи з даними, де інтеграції реалізуються в ручному режимі або в межах конкретних вендорних платформ, "Data Mesh + MCP + LLM" формує відкриту, протокольно керовану архітектуру. Вона дає змогу розподіленим командам швидко адаптуватися до змін у

проєктному середовищі, будувати масштабовані дата-продукти, а також реалізовувати інтерактивну взаємодію з даними без втрати якості управління, безпеки та узгодженості з політиками організації.

Обговорення результатів дослідження. У роботі [5] автори аналізують архітектурні принципи Data Mesh як децентралізованого підходу до управління даними, зосередженого на доменній відповідальності, даних як продукті та інфраструктурі самообслуговування даними. Отримані в цьому дослідженні результати узгоджуються з цими засадами, однак розширюють їх через введення протокольного шару MCP та LLM-агентів, які трансформують мережу дата-продуктів у контекстно орієнтоване середовище підтримки проєктів, а не тільки аналітичну інфраструктуру.

У праці [9] запропоновано багатошарову Big Data-архітектуру для великих організацій, що охоплює отримання, зберігання, аналітику, машинне навчання та безпеку, а Data Mesh розглядають як один із напрямів еволюції таких рішень. Порівняно з цим підходом проведене дослідження зосереджується не на повному стеку великих даних, а на формалізованому інтеграційному шарі між Data Mesh та інтелектуальними агентами. Це дає змогу перейти від просто масштабованої архітектури даних до контекстно керованої підтримки розподілених проєктних команд.

У роботі [16] архітектуру Data Mesh подано як сучасну хмароорієнтовану модель організації даних, у якій ключову увагу приділяють тришаровій імплементації та організаційним передумовам, зокрема федеративному управлінню (federated governance), продуктовому мисленню щодо даних (data product thinking) та інституційній готовності. Отримані результати цього дослідження загалом узгоджуються з висновками авторів про потребу у збалансованих механізмах федеративного контролю, проте розширюють їх завдяки формалізації протокольних інтерфейсів доступу до дата-продуктів на підставі Model Context Protocol (MCP) та введенню реєстру проєктного контексту як сутності першого класу для підтримки роботи розподілених проєктних команд.

У роботі [4] системно сформульовано чотири принципи Data Mesh і продемонстровано, що централізовані архітектури даних не відповідають складності сучасних організацій. Наше дослідження спирається на ці принципи як на базову архітектурну рамку, але розвиває їх через операційне поєднання Data Mesh із протоколом MCP та LLM-агентами. Це забезпечує перехід від суто декларативного трактування “даних як продукту” до протокольно формалізованої взаємодії агентів із дата-продуктами в мультидоменному середовищі, що уможливлює їхнє контекстно кероване використання та інтеграцію.

У роботі [17] зосереджено увагу на практичних аспектах реалізації Data Mesh через контракти на дані, які пов'язують дата-продукти в єдину мережу. Запропонована в нашому дослідженні архітектура розвиває цю логіку, трактуючи протокольний шар MCP як "контекстну тканину" поверх Data Mesh: контракти на дані (data contracts) слугують формальною основою для інструментів протоколу Model Context Protocol (MCP-tools), а агенти на підставі великих мовних моделей (LLM-agents) отримують контрольований доступ до доменних сервісів через стандартизовані примітиви ресурсів та операцій (MCP-resources / MCP-tools), усуваючи потребу в множинних нестандартизованих інтеграційних конекторах (ad hoc connectors).

У роботі [7] подано систематичний огляд літератури стосовно Data Mesh і узагальнено практичні трактування доменної відповідальності, платформи самообслуговування даними та федеративного управління, а також окреслено спектр відкритих наукових питань. Наші результати частково закривають виявлені прогалини, оскільки пропонують формальну модель інтеграційного шару "Data Mesh + MCP + LLM-агенти" та описують, як доменні дата-продукти можна протокольно відобразити на MCP-ресурси й MCP-інструменти в контексті розподілених проєктних команд.

У дослідженні [19] показано, що для локаційно незалежних організацій критичними стають узгоджений спільний контекст, прозорість артефактів і зріла цифрова інфраструктура співпраці. Наші результати конкретизують ці висновки на рівні архітектури даних: запропонована модель контекстно-орієнтованої системи організації даних із реєстром проєктного контексту та LLM-агентами над Data Mesh фактично реалізує описану потребу в "єдиному операційному просторі" для географічно й часово розподілених команд.

У публікації [2] Model Context Protocol (MCP) подано як відкритий стандарт, що забезпечує уніфікований доступ AI-агентів до різнорідних систем даних та замінює фрагментовані інтеграції моделлю M+N. Наше дослідження розвиває цей підхід у двох напрямах: по-перше, пропонує формальну математичну модель інтеграційної складності MCP у контексті Data Mesh, по-друге, розглядає MCP як керований операційний фронтенд платформи самообслуговування даних для розподілених проєктів, а не як ізольований інтеграційний механізм.

У роботі [15] продемонстровано, що спеціалізовані LLM-моделі на кшталт Gorilla здатні підвищувати точність використання зовнішніх інструментів і API, зменшуючи частоту галюцинацій та помилкових викликів. Наше дослідження перегукується з цими результатами, але переносить акцент з рівня окремих API-запитів на рівень корпоративної архітектури: LLM-агенти працюють не з довільним набором API, а з формально описаними MCP-ресурсами й MCP-інструментами, що походять із доменно-структурованої Data Mesh. Це підвищує керованість і відтворюваність їхньої поведінки.

У роботі [6] розглянуто застосування генеративних моделей до автоматизації федеративного управління в межах системи організації даних Data Mesh, зокрема до оцінювання запитів на доступ до даних з урахуванням політик і регуляцій. Отримані нами результати узгоджуються з цією логікою, оскільки архітектура "Data Mesh + MCP + LLM-агенти" інтегрує федеративне обчислювальне управління безпосередньо в протокольні інтерфейси: виклики MCP-ресурсів та MCP-інструментів здійснюють у межах формалізованого проєктного контексту та політик доступу, що створює основу для подальшої автоматизації процедур федеративного управління.

У роботі [8] автори аналізують шар MCP з погляду архітектури та безпеки, описують життєвий цикл MCP-серверів і пропонують таксономію загроз. Наше дослідження є комплементарним: ми фокусуємося не на безпековій класифікації MCP, а на його ролі як інтегра-

ційного шару в Data Mesh для підтримки розподілених проєктних команд. Водночас виявлена авторами потреба у чітких межах достовірності та формалізованих контрактах узгоджується із введенням проєктного контексту та доменних MCP-серверів як контрольованих точок доступу.

Обговорення результатів дослідження показало, що наявні роботи, які стосуються: Data Mesh, MCP, LLM-агентів і розподілених проєктних команд в організаціях окремо описують децентралізовані архітектури даних, стандартизовані протоколи інтеграції чи нові форми агентної взаємодії. Водночас вони не пропонують цілісної контекстно орієнтованої моделі системи організації даних для розподілених проєктних команд на підставі поєднання Data Mesh, MCP та LLM-агентів із формалізованим проєктним контекстом і кількісною оцінкою інтеграційної складності. Запропонована в цій роботі модель "Data Mesh + MCP + LLM-агенти" заповнює виявлений дослідницький розрив. Результати дослідження можуть бути використані для побудови високонавантажених платформ самообслуговування даних у транснаціональних організаціях, де потрібна контекстно керована взаємодія між доменами, проєктними артефактами та агентними сервісами. Запропонована архітектура "Data Mesh + MCP + LLM-агенти" забезпечує формалізоване управління доступом, автоматизоване створення й еволюцію дата-продуктів та істотно знижує операційні витрати на підтримку інтеграційного шару, підвищуючи оперативність та обґрунтованість управлінських рішень у розподілених проєктних командах.

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

Наукова новизна отриманих результатів дослідження – розроблено модель контекстно-орієнтованої організації доступу до доменних даних у розподілених проєктних командах, яка формалізує протокольний доступ агентів на підставі великих мовних моделей (LLM agents) до доменних дата-продуктів (data products) та контрактів на дані (data contracts) у межах децентралізованої архітектури Data Mesh шляхом їх відображення на ресурси й інструменти протоколу Model Context Protocol (MCP resources / MCP tools), із використанням формалізованого проєктного контексту як проміжного керівного шару доступу та маршрутизації, що забезпечує керованість і відтворюваність взаємодії в мультидоменному середовищі та зменшує інтеграційну складність доступу до даних.

Практична значущість результатів дослідження – можна використати для формування методологічної та архітектурної основи контекстно-орієнтованої системи організації даних у розподілених проєктних командах, яку можна застосувати під час проєктування й розвитку платформ самообслуговування даних і систем федеративного обчислювального управління в мультидоменних середовищах, оскільки запропонований синтез принципів мережі даних Data Mesh, протоколу Model Context Protocol та агентів на підставі великих мовних моделей забезпечує формалізований, контекстно адаптивний доступ до доменних дата-продуктів і проєктних артефактів відповідно до ролей учасників, стадій життєвого циклу проєктів і контексту завдань, створює передумови для зменшення інтеграційної складності протокольного шару без порушення доменної автономії та механізмів федеративного управління даними, а також сприяє підвищенню ефективності підтримки управлінських і проєктних рішень упродовж життєвого циклу проєктів.

Висновки / Conclusions

Розроблено контекстно-орієнтовану архітектурну модель системи організації даних для розподілених проєктних команд на підставі синергії мережі даних Data Mesh і протоколу Model Context Protocol з використанням LLM-агентів, що забезпечило адаптивний доступ до доменних дата-продуктів та пов'язаних інформаційних ресурсів відповідно до стадій життєвого циклу проєктів, ролей учасників і контексту завдань та знизило інтеграційну складність забезпечення взаємодії в мультидоменному середовищі. За результатами проведеного дослідження можна зробити такі основні висновки.

1. Розроблено контекстно-орієнтовану архітектурну модель системи організації даних для розподілених проєктних команд на підставі синергії принципів Data Mesh, протоколу Model Context Protocol (MCP) та агентів на базі великих мовних моделей (LLM), що забезпечує адаптивний доступ до доменних дата-продуктів, формалізоване керування проєктним контекстом і зниження інтеграційної складності в мультидоменному середовищі.
2. Запропоновано концептуальну контекстно-орієнтовану модель системи організації даних для розподілених проєктних команд у багатодоменному середовищі. Показано, що традиційні централізовані архітектури даних не забезпечують належної підтримки життєвого циклу проєктів і не усувають фрагментованості проєктного контексту.
3. На підставі принципів Data Mesh та відкритого протоколу контексту моделі (MCP) побудовано архітектурну схему "Data Mesh + MCP + агенти на підставі великих мовних моделей". У її межах доменні дата-продукти та контракти на дані концептуально відображаються на ресурси й інструменти протоколів MCP, а платформа самообслуговування даними трактується як агентно-орієнтований фронтенд доступу до цих об'єктів. Одним із ключових елементів моделі є реєстр контексту проєкту, що формально задає координати "проєкт – домен – роль – стадія життєвого циклу" та використовується для обмеження простору допустимих операцій над даними.
4. Запропонована математична модель інтеграційної складності формалізує перехід від квадратичної залежності витрат типу M×N у класичній схемі прямих інтеграцій до лінійної залежності M+N в архітектурі "Data Mesh + MCP", що створює передумови для підвищення масштабованості інтеграційного шару та зменшення навантаження на централізовані команди даних.
5. Практичне значення отриманих результатів полягає в окресленні цілісної архітектурної рамки, придатної до подальшої інженерної конкретизації у транснаціональних організаціях із багатодоменними портфелями проєктів. Перспективи подальших досліджень пов'язані з побудовою прототипів, емпіричною валідацією впливу моделі на доступ до проєктного контексту та розробленням ризик-орієнтованих сценаріїв використання агентів.

References

1. Amazon Web Services. (2024). Unlocking the Power of Model Context Protocol (MCP) on AWS. URL: https://aws.amazon.com/ru/blogs/machine-learning/unlocking-the-power-of-model-context-protocol-mcp-on-aws/

2. Anthropic. (2024). Introducing the Model Context Protocol (MCP). Announcements. URL: https://www.anthropic.com/news/model-context-protocol
3. Chang, Y., Li, Z., Pham, H. A., & Saju, G. A. (202). Intelligent Agent Planning for Optimizing Parallel MRI Reconstruction via A Large Language Model. 2024 46th *Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (EMBC), Orlando, FL, USA, 4, pp. 1–4. https://doi.org/10.1109/EMBC53108.2024.10782629
4. Dehghani, Z. (2022). Data mesh: Delivering data-driven value at scale. O'Reilly Media. URL: https://www.amazon.com/Data-Mesh-Delivering-Data-Driven-Value/dp/1492092398
5. Dibouliya, A., & Jotwani, V. (2023). Review on Data Mesh Architecture and its Impact. *Journal of Harbin Engineering University*, 44(7), 2353–2363. URL: https://harbinengineeringjournal.com/index.php/journal/article/view/809
6. Dietz, L. W., Wider, A., & Harrer, S. (2025). Automating Data Governance with Generative AI. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, *8*(1), 760–771. https://doi.org/10.1609/aies.v8i1.36587
7. Goedegebuure, A., Kumara, I., Driessen, S., Van Den Heuvel, W.-J., Monsieur, G., Tamburri, D. A., & Di Nucci, D. (2024). Data Mesh: A systematic gray literature review. *ACM Computing Surveys*, 57(1), 1–36. https://doi.org/10.1145/3687301
8. Hou, X., Zhao, Y., Wang, S., & Wang, H. (2025). Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. https://doi.org/10.48550/arXiv.2503.23278
9. Ismail, F. N., Sengupta, A., & Amarasoma, S. (2025, May). Big data architecture for large organizations. https://doi.org/10.13140/RG.2.2.17898.43204
10. Krishnan, N. K. (2025). Advancing Multi-Agent Systems Through Model Context Protocol. *Architecture, Implementation, and Applications*. https://doi.org/10.48550/arXiv.2504.21030
11. Li, H., & Toor, S. (2024). Empowering Data Mesh with federated learning. In *2024 IEEE International Conference on Big Data* (*BigData*), 2340–2342. IEEE. https://doi.org/10.1109/BigData62323.2024.10825390
12. Microsoft. (2025, December). Integrate Copilot in Power BI reports and models. *Microsoft Learn*. Retrieved. URL: https://learn.microsoft.com/en-us/power-bi/create-reports/copilot-integration
13. Milligan, J. N. (2025). Learning Tableau 2025 (6th ed.). O'Reilly Media. URL: https://www.oreilly.com/library/view/learning-tableau-2025/9781835886786/
14. Model Context Protocol. (2025). Architecture overview. URL: https://modelcontextprotocol.io/docs/learn/architecture
15. Patil, S. G., Zhang, T., Wang, X., & Gonzalez, J. E. (2023). Gorilla: Large Language Model Connected with Massive APIs. *arXiv*. https://doi.org/10.48550/arXiv.2305.15334
16. Pattnaik, R. (2025, January 31). Data mesh: A modern approach to scalable cloud data architecture. *International Journal of Computer Engineering and Technology*, 16(1), 1645–1658. https://doi.org/10.34218/IJCET_16_01_121
17. Perrin, S., & Broda, T. (2024). Implementing Data Mesh: How to design, build, and implement a data mesh using data contracts. O'Reilly Media. URL: https://www.amazon.com/Implementing-Data-Mesh-Implement-Contracts/dp/1663754160
18. Project Management Institute. (2023). Global project management job trends 2023. *Project Management Institute*. URL: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/career-central/global-project-management-job-trends-2023.pdf
19. Rhymer, J. (2022). Location-independent organizations: Designing collaboration across space and time. *Administrative Science Quarterly*, 68(1). https://doi.org/10.1177/00018392221129175
20. Snowflake. (2023, November 28). Snowflake Cortex Analyst: Behind the scenes. *Snowflake Engineering Blog*. URL: https://www.snowflake.com/en/engineering-blog/snowflake-cortex-analyst-behind-the-scenes/
21. Tableau. (2024, April 11). Agentic analytics: A new paradigm for business intelligence. *Tableau Blog*. URL: https://www.tableau.com/blog/agentic-analytics-new-paradigm-for-business-intelligence
22. Vaskiv, R. I., & Veretennikova, N. V. (2024). Information and Communication Tools for Effective Functioning of Distributed Project Teams. *Bulletin of the Lviv Polytechnic National University. Information Systems and Networks*, 15, 357–369. https://doi.org/10.23939/sisn2024.15.357
23. Vaskiv, R. I., & Veretennikova, N. V. (2025). Chaos Engineering methodology and the resilience of distributed IT teams. *Computer-Integrated Technologies: Education, Science, Production*, 58, 197–211. https://doi.org/10.36910/6775-2524-0560-2025-58-24
24. Vaskiv, R. I., & Veretennikova, N. V. (2025). Data Mesh – an Innovative Data Organization System for Distributed Project Teams. Bulletin of the Lviv Polytechnic National University. *Information Systems and Networks*, 18(1), 162–177. https://doi.org/10.23939/sisn2025.18.1.162
25. Vaskiv, R. I., & Veretennikova, N. V. (2025). Integrated Risk Management Model in Distributed IT Teams. *Bulletin of the Lviv Polytechnic National University. Information Systems and Networks*, 17, 214–225. https://doi.org/10.23939/sisn2025.17.214

R. I. Vaskiv, V. V. Pasichnyk

Lviv Polytechnic National University, Lviv, Ukraine

CONTEXT-ORIENTED DATA ORGANIZATION SYSTEM FOR DISTRIBUTED PROJECT TEAMS

The article addresses the problem of fragmented data organization within distributed project teams of large IT and service enterprises operating across multiple time zones, jurisdictions, and domains. It is shown that centralized data architectures (Data Warehouse, Data Lake, Lakehouse), primarily oriented toward reporting, poorly reflect the multidomain nature of such organizations, create organizational bottlenecks, and fail to support context-aware project lifecycle management. Based on an analysis of the Data Mesh paradigm, the paper substantiates the relevance of a domain-oriented approach to data organization using data products, data contracts, self-service data platforms, and federated computational governance. However, it identifies a systemic gap in stakeholder access to project context. The study reviews modern practices of integrating large language models (LLMs) and conversational analytics into BI and cloud platforms. It demonstrates that such solutions remain platform-specific and reproduce an M×N integration model between agents and services. The Model Context Protocol (MCP) is explored as an open application-level protocol that standardizes LLM-agent interaction with external resources through primitives such as resources, tools, and prompts, reducing integration complexity to O(M+N). On this basis, the paper proposes a context-oriented architectural model for data organization in distributed project teams configured as "Data Mesh + MCP + LLM agents". The architecture "Data Mesh + MCP + Large Language Model (LLM)–based agents" is proposed, in which domain-specific data products and data contracts defined within the Data Mesh framework are formally mapped to MCP resources and tools. An access to these assets is provided through a self-service data platform with integrated LLM-agent support. The analytical model of integration complexity demonstrates a transition from quadratic M×N scaling in classical integration scenarios to linear M+N scaling in the proposed "Data Mesh + MCP" configuration. This architectural shift creates prerequisites for improving the scalability of the integration layer, reducing integration overhead, and enabling transparent, context-oriented access to project artifacts in distributed teams. Future research will focus on empirical validation of the proposed architecture in organizations with multidomain project portfolios and on extending models for context-aware interaction between LLM agents, data products, and project risk datasets.

Keywords: data mesh; model context protocol (MCP); large language models (LLM); agents based on large language models; self-service data platform; federated computational governance.