DOI: https://doi.org/10.25140/2411-5363-2026-1(43)-229-245
УДК 004.94:004.8

Владислав Олексійович Прищепа[1], Артем Олександрович Задорожній[2]

[1]аспірант кафедри інформаційних та комп'ютерних систем
Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: vladpryshchepa1@gmail.com. **ORCID:** https://orcid.org/0009-0002-7627-0456

[2]кандидат технічних наук, доцент кафедри інформаційних технологій та програмної інженерії
Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: zaotroy@gmail.com. **ORCID:** https://orcid.org/0000-0002-3424-7293. **ResearcherID:** F-6358-2016

ПРОЕКТУВАННЯ LLM-КЕРОВАНОЇ МЕТАСИСТЕМИ ДЛЯ ДИНАМІЧНОГО АГЕНТНО-ОРІЄНТОВАНОГО МОДЕЛЮВАННЯ

У роботі представлено архітектурне проєктування LLM-керованої метасистеми для агентно-орієнтованого моделювання. Підхід дозволятиме створення агентно-орієнтованих моделей і виконання симуляцій на основі природномовних специфікацій. Запропоновано три рівні використання LLM у метасистемі та шість стратегій інтеграції LLM під час виконання симуляцій. Описано підходи до валідації й оцінювання якості моделей, а також проаналізовано напрями застосування, обмеження та виклики практичного впровадження.

Ключові слова: *агентно-орієнтоване моделювання; велика мовна модель; LLM; метасистема; мова програмування Go; мікросервісна архітектура; Java-технології; якість моделі; валідація симуляції.*

Рис.: 3. Табл.: 2. Бібл.: 23.

Актуальність теми дослідження. Протягом останніх трьох десятиліть агентно-орієнтоване моделювання (Agent-Based Modeling, ABM) еволюціонувало від периферійного методологічного інструменту до визнаної дослідницької практики у широкому спектрі наукових дисциплін [1]. Bonabeau [1] характеризує ABM як підхід, за якого система концептуалізується як колекція автономних об'єктів, здатних до прийняття рішень. Проте традиційні підходи до побудови агентно-орієнтованих моделей потребують значних компетенцій у програмуванні та глибокого розуміння симуляційних фреймворків, що створює суттєві перешкоди для дослідників. Останні можуть володіти експертними знаннями про системи, які підлягають моделюванню, однак відчувають брак технічних навичок для практичної реалізації симуляцій.

Водночас розвиток великих мовних моделей (Large Language Models, LLM) демонструє вражаючі можливості у розумінні природної мови, генерації коду та міркуванні [2; 3]. Наприклад, за даними Raschka [4], GPT-4, який навчений на основі 13 трильйонів токенів, має 1,8 трильйона параметрів, що дозволяє обробляти контексти обсягом до 32 768 токенів. Такі характеристики відкривають перспективи для інтеграції LLM у системи агентно-орієнтованого моделювання та потенційного спрощення доступу до симуляційних технологій.

Незважаючи на це, наявні дослідження зосереджуються переважно на використанні LLM для аналізу результатів симуляцій або генерації окремих компонентів моделей [5; 6], залишаючи поза увагою питання створення повноцінних метасистем, здатних до комплексної інтеграції LLM на всіх етапах симуляційного процесу.

Постановка проблеми. Центральна проблематика дослідження полягає у створенні метасистеми для агентно-орієнтованого моделювання, здатної суттєво спростити процес розробки, виконання та аналізу симуляційних моделей. Насамперед така система має забезпечувати можливість специфікації моделей природною мовою: дослідники повинні мати змогу описувати структуру моделі, типи агентів, їхню поведінку та характеристики середовища у формі діалогу з системою, без необхідності програмування. Це передбачає побудову схем агентів, правил поведінки, механізмів взаємодії та параметрів середовища симуляції, які можуть бути безпосередньо використані ядром агентно-орієнтованого моделювання. Зазначене зумовлює потребу в інтелектуальному шару інтерпретації, здатному узгоджувати семантику природної мови з формальними представленнями моделей.

Наступною складовою проблеми є інтеграція великих мовних моделей безпосередньо в цикл виконання симуляції. У такому підході LLM виступає не лише інструментом попередньої генерації моделі, а й компонентом прийняття рішень агентами в реальному часі, що дає змогу моделювати складні, контекстно залежні та адаптивні форми поведінки, які важко формалізувати за допомогою жорстко заданих правил.

Окремої уваги потребує проблема ефективності та масштабованості. Метасистема має підтримувати симуляції з тисячами й десятками тисяч агентів, поєднуючи LLM-керовану поведінку з традиційними механізмами, а також забезпечувати збереження великих обсягів симуляційних даних і надання інструментів для їх подальшого аналізу, візуалізації та прийняття рішень.

Наявні агентно-орієнтовані фреймворки, такі як NetLogo, MASON та Repast, орієнтовані переважно на ручне програмування моделей і не підтримують природномовну специфікацію, автоматичну генерацію структур моделі та інтеграцію великих мовних моделей у цикл виконання симуляцій. У результаті дослідники змушені поєднувати різнорідні інструменти та підходи, що ускладнює розробку моделей, обмежує їхню адаптивність і знижує відтворюваність результатів.

Таким чином, відсутність цілісної системи, що поєднувала б природномовну специфікацію моделей, автоматизовану побудову агентно-орієнтованих структур, інтелектуальне виконання симуляцій і масштабовану обробку результатів, зумовлює необхідність розробки нових підходів до створення та використання агентно-орієнтованих симуляційних систем.

Аналіз останніх досліджень і публікацій. Bonabeau [1] визначає ABM як підхід, у якому система моделюється як колекція автономних об’єктів, що приймають рішення, названих агентами. Кожен агент індивідуально оцінює свою ситуацію та приймає рішення на основі набору правил. Macal і North [7] деталізують цю концепцію, підкреслюючи, що більшість агентно-орієнтованих моделей складається з численних агентів, специфікованих на різних масштабах, евристик прийняття рішень, правил навчання або адаптивних процесів, топології взаємодії та середовища.

Застосування ABM охоплює різноманітні наукові домени. В епідеміології цей підхід використовується для моделювання поширення захворювань на основі індивідуальних контактів [8]. Екологія застосовує ABM для моделювання взаємодій «хижак-жертва» та конкуренції за ресурси [9]. Соціальні науки звертаються до ABM для вивчення динаміки думок та поширення інформації [10]. Grimm et al. [9] наголошують на важливості патерн-орієнтованого підходу в контексті ABM, що дозволяє зіставляти результати симуляцій з емпіричними патернами.

Архітектура трансформерів, на якій базуються сучасні LLM, використовує механізм уваги (attention mechanism), що дозволяє моделі обробляти відносини між усіма елементами послідовності одночасно [11]. Vaswani et al. [11] продемонстрували революційність цього підходу для обробки природної мови. GPT-4 навчався у два етапи: спочатку на великих наборах текстових даних для передбачення наступного токена, потім застосовувалося навчання з підкріпленням від людського зворотного зв’язку (RLHF) [4].

Розвиток «reasoning» моделей у 2024 році ознаменував новий етап еволюції LLM. Ці моделі навчені генерувати покрокову аналітику перед створенням фінальних відповідей, що забезпечує кращі результати на складних задачах у математиці, програмуванні та логіці [12]. Це особливо релевантно для агентно-орієнтованого моделювання, де прийняття рішень агентами може потребувати складного міркування.

Інтеграція LLM та ABM є відносно новим напрямом досліджень, що переживає активний розвиток. Park et al. [6] представили концепцію «Generative Agents», де LLM керують поведінкою агентів у віртуальному середовищі. Водночас їхня система радше демонструє можливості окремих LLM-керованих агентів у специфічному контексті.

Wu et al. [13] запропонували концепцію Smart Agent-Based Modeling (SABM), яка інтегрує LLM у ABM для моделювання реальних сценаріїв. Дослідники зазначають, що традиційні ABM постають перед труднощами при формалізації природномовних інструкцій та здорового глузду в математичних рівняннях чи правилах. SABM демонструє ефективність через три дослідницькі кейси з відкритим вихідним кодом на GitHub, проте питання масштабованості залишається відкритим.

Chopra et al. [14] звертаються до фундаментальної проблеми масштабованості LLM-керованих ABM, представляючи методологію LLM архетипів. Їхній підхід балансує поведінкову складність з обчислювальною ефективністю. У дослідженні симулюється 8,4 мільйона агентів, що представляють Нью-Йорк під час пандемії COVID-19, демонструючи можливість застосування LLM до великомасштабних симуляцій. Ця робота підтверджує, що концепція архетипів може бути ключовою для досягнення масштабованості в LLM-керованих ABM, хоча автори визнають певні обмеження у виразності поведінки індивідуальних агентів.

Chen et al. [5] розглядали використання LLM для генерації правил поведінки агентів, однак виконання симуляції відбувалося без участі LLM, що обмежує адаптивність агентів під час виконання. Avola et al. [15] досліджували використання машинного навчання для виявлення аномалій у відеопотоках з безпілотних літальних апаратів.

Виділення недосліджених частин загальної проблеми. Критичний аналіз наукових публікацій свідчить про наявність низки суттєвих прогалин у сучасних підходах до використання великих мовних моделей в агентно-орієнтованому моделюванні. Насамперед у наявних дослідженнях відсутні метасистеми, здатні забезпечити повноцінне створення агентно-орієнтованих моделей на основі природномовної специфікації. Запропоновані підходи, як правило, обмежуються автоматизацією окремих етапів моделювання або потребують значної участі програміста під час формалізації структури моделі та логіки поведінки агентів.

Недостатньо дослідженою проблемою залишається інтеграція великих мовних моделей безпосередньо в цикл виконання симуляції. У більшості попередніх робіт LLM використовуються переважно як інструмент генерації або постобробки, тоді як їх застосування для прийняття рішень агентами в реальному часі під час симуляції практично не досліджувалося. Такий підхід суттєво обмежує можливості моделювання адаптивної поведінки агентів та їх реакції на емерджентні ситуації, що виникають у динамічному середовищі.

Крім того, відсутні систематичні дослідження компромісів між різними підходами до використання LLM у симуляціях (індивідуальний/колективний контроль, повний контроль/тільки рішення). Wu et al. [13] та Chopra et al. [14] запропонували окремі стратегії, проте комплексне порівняння та можливість гнучкого вибору стратегії в межах однієї системи не розглядалися.

Недостатньо розробленими також залишаються архітектурні патерни побудови високопродуктивних симуляційних систем з інтеграцією LLM. Зокрема, потребують подальшого дослідження питання ефективного зберігання та обробки часових рядів, потокової передачі станів симуляції, а також управління життєвим циклом компонентів у масштабованих обчислювальних середовищах.

Нарешті, у наявних роботах приділяється недостатня увага проблемі зберігання великих обсягів симуляційних даних у системах з інтегрованими LLM, особливо з урахуванням часової структури даних та вимог до ефективного доступу для подальшого аналізу, візуалізації та підтримки прийняття рішень.

Мета дослідження. Метою дослідження є проєктування LLM-керованої метасистеми для динамічного агентно-орієнтованого моделювання. Система має забезпечувати природномовну специфікацію симуляційних моделей через діалог з LLM, автоматичну генерацію структур даних моделей (включаючи агентів, правила, взаємодії та середовище), а також механізми оцінювання якості та валідації створених моделей.

Важливим аспектом дослідження є розробка стратегій інтеграції великих мовних моделей у симуляційний процес, спрямованих на дослідження компромісів між рівнем інтелектуальності поведінки агентів і обчислювальною ефективністю системи. Передбачається реалізація декількох стратегій інтеграції LLM, що дозволятиме оцінити їхній вплив на продуктивність симуляцій, масштабованість і якість отриманих результатів.

Окрему увагу в межах дослідження приділено забезпеченню ефективного виконання симуляцій з великою кількістю агентів, підтримці потокової передачі станів симуляції в режимі реального часу, а також організації масштабованого зберігання результатів симуляцій.

Крім того, передбачається розробка інструментів для аналізу результатів з використанням LLM, що дозволить дослідникам отримувати висновки та рекомендації щодо результатів симуляцій на основі зібраних даних.

Виклад основного матеріалу. Запропоноване архітектурне рішення ґрунтується на клієнт-серверній архітектурі з чітким розмежуванням відповідальності між компонентами системи. Для реалізації клієнтської частини планується використання фреймворку Vue, який повинен забезпечити інтерактивний вебінтерфейс для природномовної специфікації моделей, налаштування параметрів симуляцій та візуалізації результатів. Серверна частина системи, на якій зосереджено основну бізнес-логіку, проєктується на основі мікросервісної архітектури [16] з використанням мови програмування Go для основного серверного додатка. Крім того, ядро системи реалізовуватиметься із застосуванням HTTP-фреймворку Echo та механізмів інверсії залежностей на основі Uber Fx [17; 18]. При цьому основний додаток виконуватиме функції створення та управління симуляційними моделями, організації діалогової взаємодії з користувачем, а також безпосереднього виконання агентно-орієнтованих симуляцій.

Взаємодія з великими мовними моделями реалізовуватиметься через окремі адаптерні мікросервіси, які дотримуватимуться уніфікованого контракту обміну даними з основним додатком та зможуть використовувати як локально розгорнуті мовні моделі, так і зовнішні LLM-платформи, зокрема OpenAI та Anthropic. Використання мікросервісного підходу забезпечуватиме ізоляцію ядра системи від специфіки конкретних реалізацій LLM, спрощуватиме масштабування та оновлення компонентів, а також дозволятиме використання різних мов програмування для створення мікросервісів без впливу на основну логіку системи.

Вибір Go для створення центрального серверного застосунку обґрунтовується високою продуктивністю для паралельних обчислень, безпекою типів та ефективним управлінням пам’яттю [19]. Водночас для реалізації LLM-адаптерних мікросервісів можуть використовуватися Java-технології, що забезпечують широкий інструментарій для побудови enterprise-рішень, зокрема розвинену екосистему бібліотек для HTTP-комунікації та обробки JSON. Загалом архітектура відповідає принципам розподілу відповідальності, у межах яких окремі модулі ізольовано реалізують інтеграцію (комунікацію) з LLM-провайдерами, управління діалогами для генерації моделей, а також збереження та виконання симуляційних процесів.

На рис. 1 представлено багаторівневу архітектуру LLM-керованої метасистеми для агентно-орієнтованого моделювання.

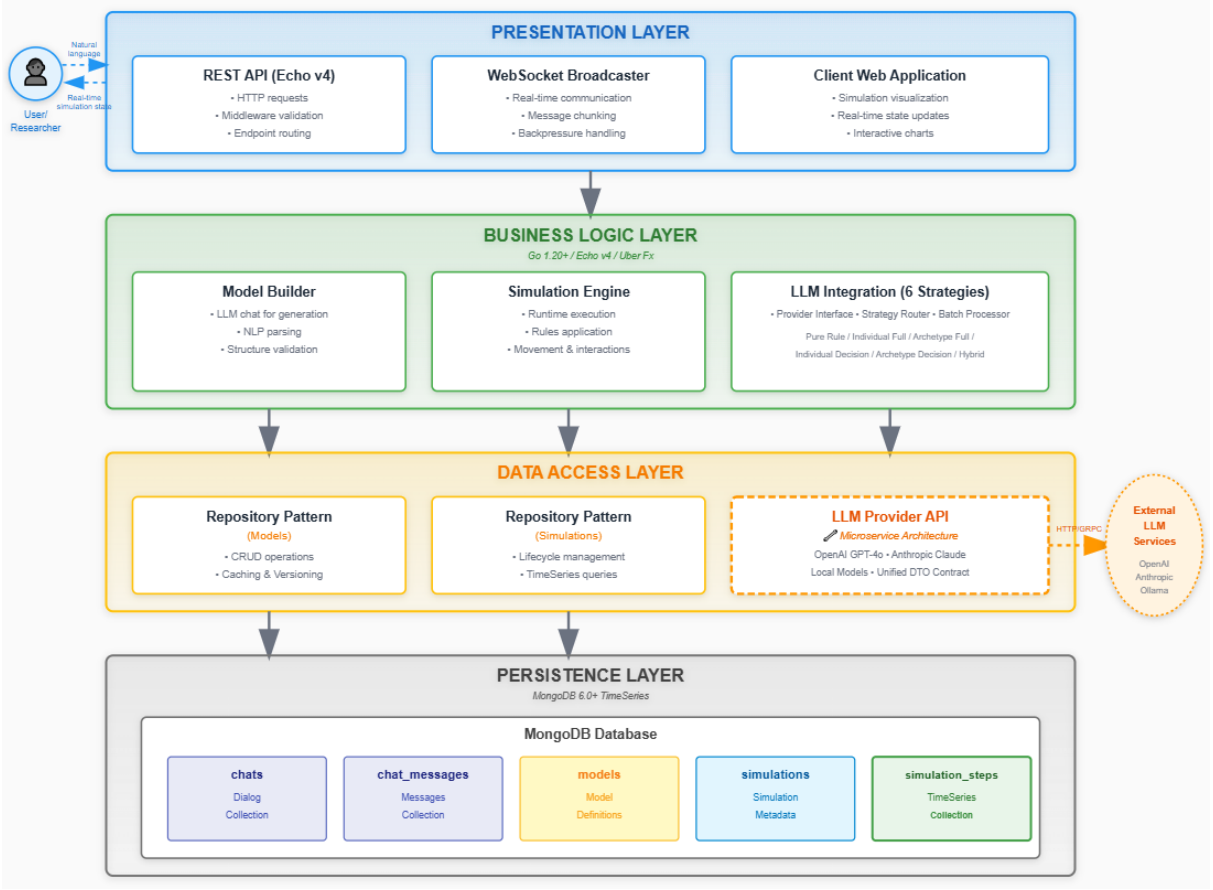

Рис. 1. Архітектура LLM-керованої метасистеми для динамічного агентно-орієнтованого моделювання

Джерело: розроблено авторами.

Технологічний стек. Вибір технологічного стеку зумовлений специфічними вимогами до продуктивності, масштабованості та ефективності розробки. У таблиці 1 міститься перелік основних технологічних рішень та коротке обґрунтування їхнього використання.

Таблиця 1 – Технологічний стек програмної реалізації системи

Компонент	*Технологія*	*Обґрунтування*
Мова	Go 1.20+	Продуктивність, конкурентність, безпека типів [19]
Web Framework	Echo v4	Легковаговий HTTP-маршрутизатор
DI	Uber/fx	Модульність, управління життєвим циклом [17]
Логування	Uber Zap	Структуроване, високопродуктивне
БД	MongoDB	Гнучка схема, колекції часових рядів [20]
WebSocket	gorilla/websocket	Комунікація в реальному часі [21]
Frontend / UI	Vue 3	Реактивний інтерфейс, компонентний підхід, швидкий розвиток

Джерело: розроблено авторами.

Архітектура взаємодії з великими мовними моделями (LLMs). Система побудована за принципом розділення відповідальності між основним серверним додатком та окремими адаптерними сервісами для взаємодії з LLM. Основний додаток відповідає за створення моделей на основі діалогів з користувачем, виконання симуляцій, управління станом агентів та збереження результатів. Взаємодія з великими мовними моделями делегується окремим мікросервісам (LLM-адаптерам), які інкапсулюють специфіку роботи з конкретними провайдерами.

Для забезпечення незалежності основного застосунку від конкретних реалізацій великих мовних моделей (LLM) було визначено уніфікований контракт обміну даними у вигляді DTO (Data Transfer Objects). Цей контракт формалізує базові операції взаємодії з LLM, зокрема підтримку діалогових сесій зі структурованими відповідями, виконання одноразових запитів із можливістю мультимодального аналізу зображень, а також завдання логічного міркування, що потребують покрокового аналітичного опрацювання. Основний застосунок взаємодіє з LLM-адаптерами виключно через визначений контракт, що забезпечує слабке зв'язування компонентів системи та дозволяє змінювати або розширювати набір провайдерів LLM без внесення змін до ядра системи.

Зазначена особливість створює передумови для реалізації LLM-адаптерних сервісів із використанням різних мов програмування відповідно до специфіки задачі та доступного програмного інструментарію. Такий підхід забезпечує технологічну гнучкість і дозволяє обирати оптимальний програмний стек для інтеграції з кожним окремим провайдером. При цьому адаптерні сервіси можуть інкапсулювати взаємодію як з локально розгорнутими мовними моделями, так і з віддаленими сервісами, доступ до яких здійснюється через програмні інтерфейси, із застосуванням найбільш доцільних засобів реалізації.

Рівні використання LLM. Інтеграція LLM реалізуватиметься на трьох концептуально різних рівнях, кожен з яких відповідає окремій фазі життєвого циклу симуляції. На рис. 2 наведено візуалізацію трьох рівнів інтеграції LLM у метасистемі.

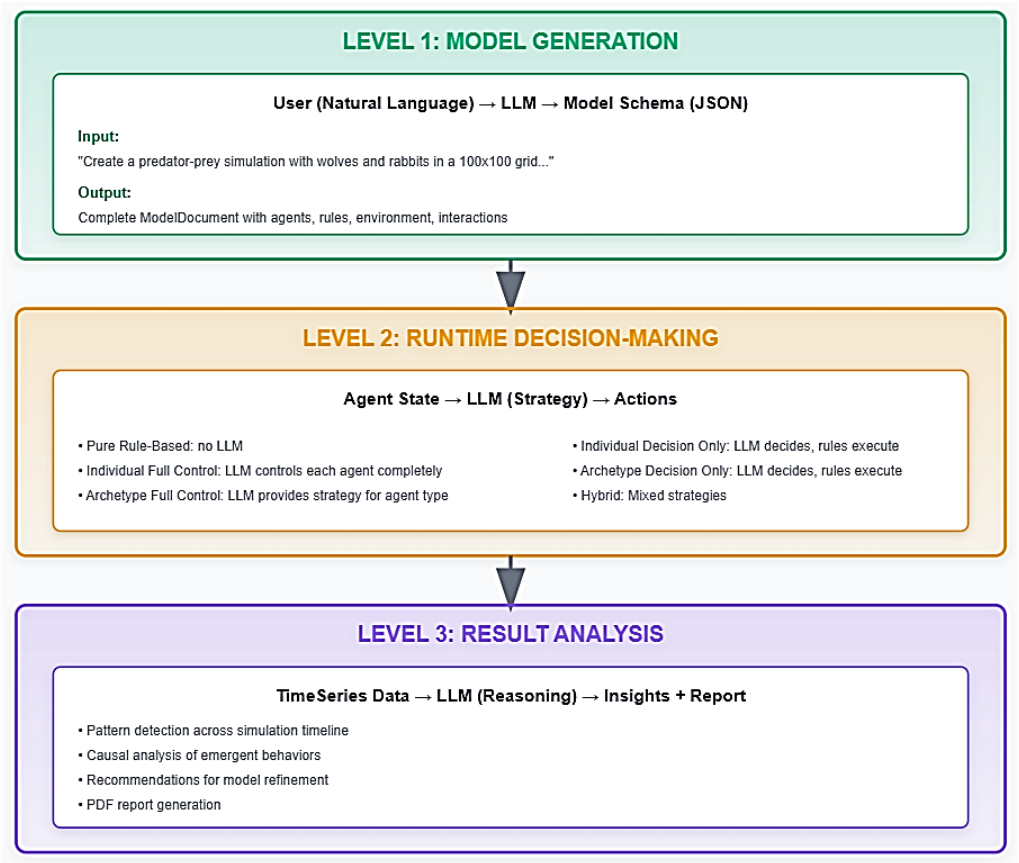

Рис. 2. Три рівні інтеграції LLM у метасистемі

Джерело: розроблено авторами.

Рівень 1 (Генерація моделей). Дослідник у діалозі з LLM описує природною мовою бажану симуляцію. LLM аналізує опис та генерує повну специфікацію моделі у структурованому форматі. Цей процес включає визначення агентів з їхніми властивостями, пра-

вила поведінки з умовами та діями, типи взаємодій між агентами та параметри середовища. Система також підтримуватиме аналіз зображень для визначення просторової структури середовища на основі візуальних схем, що дозволятиме дослідникам передавати топологічну інформацію без формальних описів.

Рівень 2 (Виконання симуляції). Під час виконання LLM інтегрується безпосередньо в цикл прийняття рішень агентів. Залежно від обраної стратегії, LLM зможе повністю контролювати поведінку кожного агента індивідуально, надавати загальні стратегії для архетипів або тільки приймати високорівневі рішення, делегуючи виконання системі правил. Це дозволятиме балансувати між виразністю поведінки та обчислювальною вартістю.

Рівень 3 (Аналіз результатів). LLM з можливостями міркування аналізуватиме результати після завершення симуляції. Система виявлятиме патерни, зв'язки та емерджентні явища, які можуть бути неочевидними при традиційному статистичному аналізі. На основі аналізу буде виконуватися генерація структурованих звітів із висновками та рекомендаціями щодо результатів симуляції.

Стратегії інтеграції LLM під час виконання симуляцій. Архітектурою системи передбачено шість стратегій керування поведінкою агентів під час виконання, які дозволяють динамічно визначати рівень участі великих мовних моделей у прийнятті рішень на кожному кроці симуляції. Визначені стратегії формують континуум від повністю детермінованої поведінки на основі правил до повноцінного LLM-керованого інтелекту, надаючи дослідникам можливість калібрувати баланс між автономністю агентів, обчислювальною ефективністю та вартістю API-викликів залежно від специфіки експерименту. У таблиці 2 наведено класифікацію стратегій інтеграції великих мовних моделей (LLM) в агентно-орієнтовану модель під час виконання симуляцій, яка відображає залежність між кількістю викликів LLM та рівнем інтелектуальності агентів, що дозволяє обирати відповідну стратегію залежно від складності поведінки агентів і обчислювальних обмежень системи.

Таблиця 2 – Стратегії інтеграції LLM під час виконання симуляцій

№	Стратегія	Кількість викликів LLM	Рівень інтелектуальності
1	Правило-орієнтована	0	Низький – LLM не використовується, поведінка агентів повністю визначається системою правил
2	Повний індивідуальний контроль	N × F	Найвищий – LLM приймає рішення та генерує конкретні дії для кожного агента на основі повного контексту
3	Повний контроль на рівні архетипу	A × F	Високий – LLM формує єдину стратегію для кожного архетипу
4	Лише індивідуальне прийняття рішень	N × F	Середній – LLM визначає високорівневі рішення для кожного агента, виконання делегується системі правил
5	Прийняття рішень на рівні архетипу	A × F	Середній – LLM визначає високорівневі рішення для архетипів, виконання делегується системі правил
6	Гібридна стратегія	Змішана	Змінний – налаштовується окремо для кожного типу агентів

де N – кількість агентів у моделі;
A – кількість архетипів агентів;
F – частота звернень до великої мовної моделі під час симуляції.
Джерело: розроблено авторами.

Запропонована таксономія стратегій розвиває ідеї Smart ABM, представлені Wu et al. [13], щодо інтеграції LLM в агентно-орієнтовану модель, та концепцію LLM архетипів від Chopra et al. [14], що використовує узагальнені профілі для групування агентів зі схожими характеристиками. На відміну від попередніх робіт, в архітектурі системи передбачено гра-

нульовану градацію стратегій із можливістю їх гібридної комбінації в межах однієї симуляції: різні групи агентів можуть керуватися різними стратегіями одночасно, що відкриває нові можливості для експериментів із гетерогенними популяціями та дослідження впливу рівня когнітивної складності окремих агентів на динаміку системи загалом.

Стратегія 1: Правило-орієнтована. LLM не використовується під час виконання симуляції. Агенти дотримуються виключно визначених правил, що забезпечує детермінованість, високу швидкість та повну відтворюваність результатів. Ця стратегія доцільна для тестування базової логіки моделі, валідації структури симуляції або коли поведінка агентів може бути повністю специфікована через формальні правила.

Стратегія 2: Повний індивідуальний контроль. LLM ухвалює всі рішення для кожного агента індивідуально, забезпечуючи повну автономію. Кожен агент отримує персоналізовані інструкції на основі свого унікального стану, позиції та локального контексту. LLM аналізує властивості агента, найближче оточення, історію з пам'яті та генерує специфічні дії.

Для оптимізації може застосовуватися пакетування LLM викликів [22], коли запити для кількох агентів групуються в один API-виклик. Це зменшує загальну латентність, хоча вартість залишається пропорційною кількості агентів. Стратегія найбільш доцільна для малих популяцій з високою складністю поведінки, де кожен агент має унікальні характеристики або контекст.

Стратегія 3: Повний контроль на рівні архетипу. LLM надаватиме єдину стратегію для всіх агентів одного типу, здійснюючи один виклик на архетип за крок симуляції. LLM аналізує агреговану статистику для всіх агентів типу, включаючи середні значення властивостей, розподіл у просторі, доступність ресурсів. Генеруватиметься загальна поведінкова стратегія, яка потім застосовуватиметься до всіх агентів цього архетипу, можливо, з варіаціями на основі локального контексту.

Вартість використання цієї стратегії є середньою та пропорційною кількості архетипів, а не агентів. Вона може бути ефективною для середніх та великих популяцій з чіткими ролями, де агенти одного типу мають подібну поведінку, але популяція достатньо велика, щоб індивідуальний контроль був непрактичним.

Стратегія 4: Лише індивідуальне прийняття рішень. LLM прийматиме високорівневі рішення для кожного агента, тоді як система правил виконуватиме їх. Це гібридний підхід, що поєднуватиме інтелект LLM з примусовим виконанням правил. Наприклад, агент отримує від LLM рішення на кшталт «шукати їжу» або «уникати хижака», а система правил транслює ці абстрактні рішення в конкретні дії (зміна напрямку руху, модифікація властивостей).

Такий підхід дозволятиме LLM зосередитися на складних аспектах прийняття рішень, делегуючи виконання ефективним правилам. Доцільний для ситуацій, де рішення є складними та контекстно залежними, але виконання може бути стандартизоване.

Стратегія 5: Прийняття рішень на рівні архетипу. Рішення високого рівня прийматися на рівні архетипу, що є найбільш ефективним підходом для масових популяцій. При цьому LLM визначає загальну стратегію для типу агентів (наприклад, «агресивний пошук»), а система правил адаптує цю стратегію до індивідуальних обставин кожного агента.

Вартість цієї стратегії є найнижчою серед стратегій з LLM-інтеграцією, пропорційна лише кількості архетипів. Стратегія дозволяє масштабувати симуляції, зберігаючи певний рівень інтелектуальної поведінки при мінімальних витратах на API-виклики.

Стратегія 6: Гібридна. Різні стратегії застосовуються для різних архетипів у межах однієї симуляції, забезпечуючи максимальну гнучкість. Налаштування стратегії здійснюється окремо для кожного типу агентів. Наприклад, лідери можуть використовувати

«Повний індивідуальний контроль» для складного прийняття рішень, основна популяція – «Прийняття рішень на рівні архетипу» для ефективності, а фонові агенти – «Правило-орієнтовану» для мінімізації вартості.

Це дозволятиме оптимізувати розподіл обчислювальних ресурсів, вкладаючи більше «інтелекту» в критичні ролі та зменшуючи вартість для менш важливих агентів. Гібридна стратегія може бути особливо корисною для мультирольових симуляцій з ієрархією або спеціалізацією, де різні типи агентів мають якісно різну важливість.

Варто зазначити, що стратегії 2 і 4 (а також 3 і 5) характеризуються однаковою кількістю викликів LLM, проте відрізняються обсягом відповідальності, делегованої моделі в кожному виклику. У стратегіях повного контролю (2 і 3) LLM приймає рішення та генерує конкретні дії агента на основі комплексного аналізу контексту, що потребує ширшого вхідного контексту та повертає більш деталізовану відповідь. Натомість у стратегіях прийняття рішень (4 і 5) LLM лише обирає стратегію поведінки під час прийняття визначених рішень, тоді як система правил транслює її в конкретні дії – це зменшує складність запиту та обсяг відповіді, однак обмежує адаптивність поведінки агентів. Таким чином, при еквівалентній кількості звернень до LLM стратегії повного контролю забезпечують вищий рівень інтелектуальності агентів завдяки більшій деталізації та контекстуальній глибині кожного виклику, тоді як стратегії прийняття рішень є обчислювально ефективнішими на рівні окремого запиту.

Модель даних. Схема даних базуватиметься на п’яти основних колекціях MongoDB, кожна з яких відповідатиме за певний аспект функціональності системи. Колекція «chats» зберігатиме історію діалогів для генерації моделей, тоді як «chat_messages» міститиме окремі повідомлення в цих діалогах. Визначення агентно-орієнтованих моделей зберігатиметься в колекції «models», метадані та стан виконання симуляцій в «simulations», а результати симуляцій – у спеціалізованій колекції часових рядів «simulation_steps».

Для зберігання історії виконання симуляцій пропонується використати спеціалізоване сховище на основі колекцій часових рядів MongoDB [20]. Цей тип колекцій забезпечує ефективне стиснення даних завдяки технології блокової обробки (block processing), що критично важливо для довготривалих симуляцій з великою кількістю агентів [20]. Використання стандартних колекцій MongoDB для зберігання таких обсягів даних може призводити до надмірного споживання дискового простору та поступової деградації продуктивності діапазонних (range) запитів, що ускладнюватиме подальший аналіз даних.

При цьому колекції часових рядів MongoDB можуть розглядатися як компромісне рішення, яке дозволяє уникнути надмірної складності інфраструктури; водночас у разі подальшого розвитку системи допустимим є перехід до спеціалізованих баз даних часових рядів, зокрема InfluxDB, Prometheus або Amazon Timestream.

Модель виконання симуляції. Виконання симуляції організовуватиметься як послідовність дискретних кроків, на кожному з яких система буде оновлювати стан всіх агентів та середовища. Цикл виконання окремого кроку реалізовуватиметься як послідовність фаз. Спочатку визначатиметься та виконуватиметься відповідна стратегія LLM. Потім для кожного агента застосовуватимуться специфічні правила згідно з їх пріоритетами. Умови правил оцінюватимуться в порядку пріоритету, і як тільки умова задовольнятиметься, виконуватимуться відповідні дії.

На наступному етапі оброблятимуться взаємодії між агентами та між ними й середовищем, а результати прийнятих рішень застосовуватимуться до просторового стану агентів шляхом оновлення їхніх позицій і швидкостей.

Під час виконання система підтримуватиме повний стан симуляції в пам'яті, включаючи колекцію агентів з властивостями та позиціями, стан середовища та індекси для швидкого доступу до агентів. Після кожного кроку система оновлюватиме пам'ять агентів та зберігатиме стан у базу даних.

Система пам'яті агентів. Кожен агент матиме обмежену пам'ять подій, що дозволятиме LLM приймати рішення на основі історичного контексту. Без пам'яті кожне рішення базуватиметься виключно на поточному стані. Пам'ять зберігатиме різні типи подій: прийняті рішення (як власні, так і отримані від LLM), взаємодії з іншими агентами (конфлікти, кооперація, обмін ресурсами), виконання правил (які правила спрацювали та чому), та відповіді від LLM (для контролю контексту).

Кожна подія міститиме дані про крок симуляції, тип події, стислий опис та детальну інформацію. Структура подій буде гнучкою та зможе включати додаткові метадані.

Архітектура потокової передачі для візуалізації симуляцій. Реалізація потокової передачі є необхідною для створення клієнтського вебдодатка, який забезпечуватиме візуалізацію процесу симуляцій у режимі реального часу, надаючи користувачам можливість спостерігати за динамікою агентів та їх взаємодіями безпосередньо під час виконання моделі. Для реалізації потокової передачі стану симуляції пропонується застосувати протокол WebSocket [21], який дозволяє двосторонню комунікацію в реальному часі. Це усуває необхідність постійного опитування (polling) сервера клієнтами, знижуючи латентність та навантаження на мережу. Архітектура базується на концепції «кімнат симуляції» [23], де кожна активна симуляція має власну кімнату з підключеними клієнтами.

На рисунку 3 представлено архітектуру потокової передачі стану симуляції.

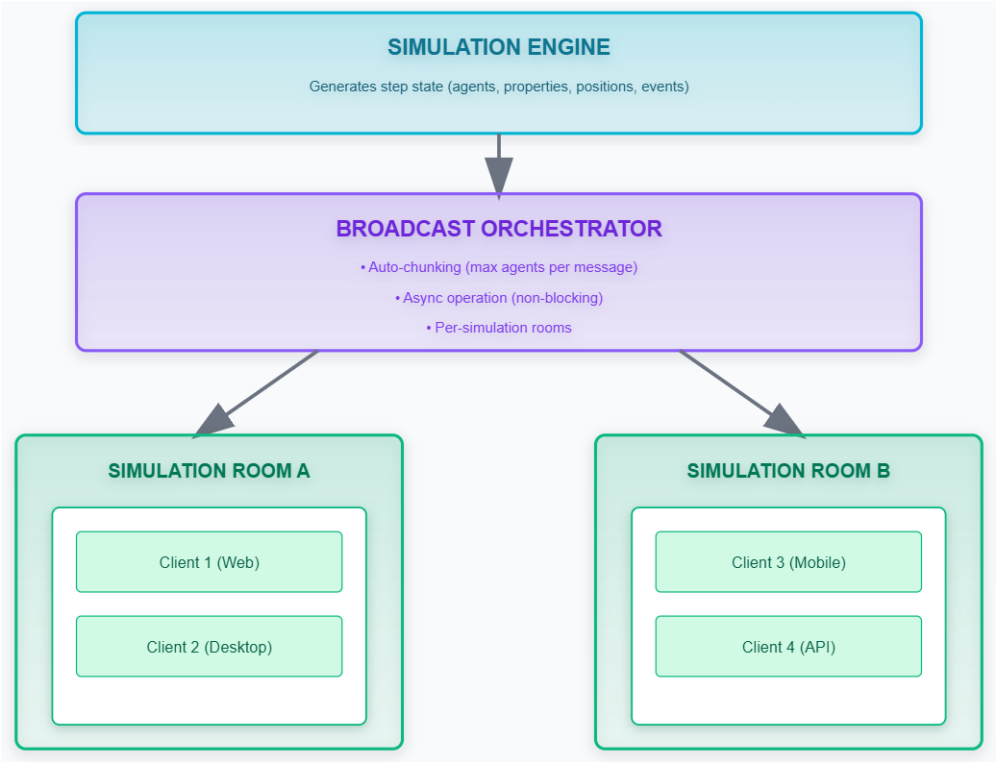

Рис. 3. Архітектура потокової передачі стану симуляції

Джерело: розроблено авторами.

Оркестратор метасистеми координуватиме розсилку оновлень та управлятиме життєвим циклом з'єднань, створюючи для кожної симуляції окрему кімнату, до якої будуть підключатися клієнти-підписники. Після кожного кроку симуляції оркестратор отримуватиме стан симуляції та розсилатиме його всім активним підписникам кімнати.

Трансляція WebSocket-повідомлень виконуватиметься у фонових горутинах без блокування основного циклу симуляції. Для великих кроків з тисячами агентів стан автоматично розділятиметься на фрагменти (chunks) для запобігання перевантаженню каналів. Клієнт отримуватиме дані про порядок фрагментів та матиме можливість відповідно їх обробити.

Аналіз результатів симуляцій. Після завершення симуляції система надаватиме можливість аналізу результатів з використанням LLM у режимі міркування (reasoning). На відміну від традиційного статистичного аналізу, який обмежується обчисленням агрегатних метрик, застосування великих мовних моделей дозволятиме виявляти складні патерни, каузальні зв’язки та емерджентні явища, що виникають із взаємодії агентів під час симуляцій.

Використання LLM у режимі міркування дозволятиме не лише фіксувати кількісні показники, а й інтерпретувати їх у контексті поведінки агентів, виявляти приховані закономірності та встановлювати причинно-наслідкові зв'язки між подіями в системі.

За результатами аналізу система забезпечуватиме можливість генерації структурованих PDF-звітів для документування результатів. Такі звіти інтегруватимуть як кількісні метрики, так і результати аналізу, отримані за допомогою LLM.

Потенційні напрями застосування. Спроектована метасистема має широкий спектр потенційних застосувань у наукових та прикладних доменах, де агентно-орієнтоване моделювання традиційно вимагає значних технічних компетенцій. Природномовна специфікація моделей та гнучкі стратегії інтеграції LLM роблять систему особливо цінною для міждисциплінарних досліджень, де експерти зможуть безпосередньо формалізувати свої знання без залучення програмістів.

Одним із базових напрямів застосування має стати освіта, оскільки метасистема знижує поріг входу та скорочує час створення моделей. Завдяки цьому студенти різних спеціальностей, у тому числі нетехнічних, зможуть експериментувати з моделями у своїх предметних областях без необхідності глибокого занурення в технічні аспекти реалізації. Це може значно прискорити засвоєння концепцій складних систем та емерджентної поведінки.

Управління надзвичайними ситуаціями становить критично важливу область застосування. Під час кризових подій (природні катастрофи, техногенні аварії, пандемії) оперативне прийняття рішень вимагає швидкого моделювання можливих сценаріїв розвитку ситуації. Традиційні підходи до створення симуляцій є занадто повільними для реагування в реальному часі. Метасистема надаватиме можливість фахівцям служб надзвичайних ситуацій описати поточну обстановку природною мовою («евакуація 50 000 мешканців прибережної зони при загрозі циклону»), а LLM автоматично згенерує модель з різними типами агентів (цивільні особи з різною мобільністю, рятувальники, транспортні засоби), правилами поведінки (паніка, допомога іншим, пошук укриття) та інфраструктурними обмеженнями.

Концепція Smart City представляє ще один перспективний напрям застосування. Урбанізація створює складні виклики для планування міст, управління транспортними потоками, оптимізації енергоспоживання та реагування на міські кризи. Як продемонстрували Chopra et al. [14], LLM-керовані ABM можуть симулювати мільйони мешканців з реалістичною поведінкою. Проєктована метасистема розширюватиме ці можливості, дозволяючи муніципальним планувальникам самостійно створювати та модифікувати симуляції без програмування.

Конкретні сценарії Smart City застосувань можуть включати: моделювання впливу нової транспортної інфраструктури (метро, велодоріжки, BRT системи) на мобільність мешканців; оптимізацію розташування публічних сервісів (лікарні, школи, парки) з урахуванням демографічних патернів; аналіз поширення епідемій у міському середовищі з урахуванням транспортних потоків та соціальних контактів.

Існує багато інших перспективних напрямів застосування, зокрема епідеміологічне моделювання, економічні та фінансові системи, екологія й соціальні науки, де природномовна специфікація може суттєво прискорити розробку ABM. Система дозволятиме швидко описувати інтервенції та характеристики популяцій для аналізу поширення інфекцій, задавати структуру ринків і поведінкові особливості учасників для відтворення колективних ефектів, формалізувати екологічні взаємодії та адаптацію видів до змін середовища, а також моделювати соціальну динаміку (дезінформація, громадська думка, протести, міграція) з урахуванням контекстної комунікації агентів.

Варто підкреслити, що універсальність запропонованої метасистеми є складною метою та неминуче передбачатиме компроміси. У зв'язку з цим повне покриття всіх доменів і сценаріїв не може бути гарантоване, адже окремі сценарії використання вимагатимуть доменно-специфічної адаптації або розширення функціональності та залишатимуться поза межами застосовності метасистеми.

Оцінювання якості та валідація моделей. Автоматична генерація агентно-орієнтованих моделей на основі природномовних описів ставить перед розробниками нетривіальне завдання забезпечення їхньої коректності. На відміну від традиційного підходу, де дослідник безпосередньо контролює кожен аспект специфікації, LLM-керована генерація вносить елемент невизначеності: модель може бути синтаксично коректною, проте семантично не відповідати намірам користувача. Це зумовлює необхідність багаторівневої валідації на етапі реалізації системи.

Першим кроком є структурна валідація, що охоплює перевірку синтаксичної коректності згенерованих специфікацій агентів, правил та взаємодій. На цьому етапі виявляються типові помилки генерації: невідповідність типів даних, відсутність обов'язкових атрибутів, порушення логічної узгодженості між компонентами моделі. Структурна валідація реалізується як автоматизований тестовий етап і дозволяє відсіяти некоректні моделі ще до спроби їх виконання.

Значно складнішим є питання поведінкової валідації. Grimm et al. [9] обґрунтували важливість патерн-орієнтованого підходу для забезпечення реалістичності агентно-орієнтованих моделей, де результати симуляцій зіставляються з емпіричними патернами реальних систем. У межах розробки метасистеми цей підхід передбачає створення референтних тестових наборів із заздалегідь відомими очікуваними патернами – характерними розподілами популяцій, динамікою взаємодій, просторовим групуванням агентів. Такі набори уможливлюють регресійне тестування під час модифікації алгоритмів генерації та виявлення систематичних відхилень у поведінці згенерованих моделей.

Окремої уваги заслуговує семантична валідація – оцінка відповідності згенерованої моделі початковому природномовному опису. Цей рівень валідації є найбільш проблематичним, оскільки потребує розуміння намірів дослідника та експертних оцінок предметних спеціалістів.

Для кількісного оцінювання якості доцільно використовувати набір метрик: повноту специфікації, що відображає охоплення моделлю всіх аспектів природномовного запиту; узгодженість поведінки під час повторних запусків з ідентичними параметрами; каліброваність результатів щодо емпіричних даних або експертних очікувань; а також збереження реалістичності при масштабуванні кількості агентів.

Обмеження та виклики запропонованого підходу. Незважаючи на описані можливості, запропонована архітектура має низку обмежень, що визначають межі її застосовності.

Система проєктується для дискретних агентно-орієнтованих моделей і не призначена для континуальних процесів, що вимагають диференціальних рівнянь. Гібридні моделі, де деякі компоненти описуються ODEs (ordinary differential equations), а інші – агентами, потребують нетривіальної адаптації архітектури. Крім того, симуляції з

критичними вимогами до продуктивності реального часу (наприклад, системи з апаратним включенням у контур (HIL) для робототехніки) можуть зазнавати негативного впливу від непередбачуваної латентності LLM API-викликів, що робить систему непридатною для застосувань із жорсткими вимогами реального часу.

Фізично-орієнтовані симуляції зі складною просторовою динамікою (гідродинаміка, деформація матеріалів) також виходять за межі можливостей системи, оскільки вона не інтегрує спеціалізовані чисельні методи. Симуляції, що потребують суворої верифікації та сертифікації (аерокосмічна галузь, ядерна енергетика), можуть бути обмежені недетермінованістю LLM та труднощами формальної валідації згенерованих моделей.

Окремим суттєвим викликом є економіка використання комерційних LLM API. Вартість викликів формується на основі кількості оброблених токенів, причому генерація відповідей (вихідні токени) зазвичай коштує у три-чотири рази дорожче за обробку запитів. Для ілюстрації масштабу проблеми розглянемо симуляцію з 1000 агентів протягом 10 000 кроків при використанні стратегії «Повний індивідуальний контроль» з частотою LLM-викликів кожні 10 кроків. За таких параметрів система генерує близько мільйона запитів, кожен з яких передбачає обробку кількасот токенів. Сукупний обсяг може сягати сотень мільйонів токенів, що при поточних тарифах провідних комерційних моделей означатиме витрати значних обсягів коштів за одну симуляцію. Для дослідницьких проєктів такі витрати швидко стають неприйнятними.

Стратегія на основі архетипів частково пом’якшує цю проблему, зменшуючи кількість викликів пропорційно до числа типів агентів замість загальної популяції. Однак для моделей з десятками різних архетипів економія може виявитися недостатньою. Альтернативою є використання локально розгорнутих моделей (Llama 3, Mistral та інші), проте їхня якість міркування та виконання запитів поступається передовим комерційним рішенням. Це може негативно позначитися на коректності згенерованих моделей та адекватності поведінки агентів. Крім того, розгортання локальних моделей вимагає значних обчислювальних ресурсів (GPU з 24+ GB VRAM для ефективного інференсу), що переносить вартість з операційних витрат на капітальні.

Не менш важливим є ризик хибної інтерпретації природномовних описів. Попри вражаючі можливості сучасних LLM у розумінні природної мови, вони залишаються схильними до помилок, особливо в доменно-специфічних контекстах. Амбівалентні або недостатньо деталізовані описи можуть призвести до генерації моделей, що не відповідають намірам дослідника. Наприклад, формулювання «агенти конкурують за ресурси» допускає інтерпретацію як прямої агресивної конкуренції, так і непрямої – через споживання обмежених ресурсів, що веде до якісно різних динамік системи.

Також проблематичним є феномен «впевненої некомпетентності» LLM: модель може згенерувати синтаксично бездоганну, проте семантично абсурдну специфікацію, якщо опис виходить за межі її тренувальних даних. Для нішевих наукових доменів зі специфічною термінологією цей ризик є особливо високим і потребує ретельної експертної валідації результатів генерації.

Висновки. У статті представлено проєктування LLM-керованої метасистеми для динамічного агентно-орієнтованого моделювання. Архітектура базується на клієнт-серверному підході з мікросервісною організацією серверної частини, що забезпечуватиме модульність, масштабованість та гнучкість системи.

Запропоновано трирівневу концепцію інтеграції великих мовних моделей у агентно-орієнтоване моделювання, яка охоплює генерацію моделей на основі природномовної специфікації, керування поведінкою агентів під час виконання симуляції та аналіз результатів із застосуванням LLM у режимі міркування.

Концептуалізовано шість стратегій інтеграції LLM під час виконання симуляцій: від повністю правило-орієнтованої до повного індивідуального контролю, з проміжними варіантами на рівні архетипів та гібридним підходом. Стратегії формують континуум з різними компромісами між інтелектуальністю поведінки агентів, обчислювальною вартістю та масштабованістю системи.

Обґрунтовано архітектурні рішення для проєктування високопродуктивної симуляційної системи. Спроєктовано мікросервісну архітектуру, де основний серверний додаток буде реалізовано на основі мови Go із використанням фреймворку Echo та механізму інверсії залежностей Uber Fx. Клієнтська частина передбачає застосування Vue.js для природномовної специфікації моделей, налаштування параметрів симуляцій та візуалізації результатів. Взаємодія з великими мовними моделями передбачається через окремі адаптерні мікросервіси, що дотримуються уніфікованого контракту обміну даними з основним додатком і можуть використовувати як локальні, так і зовнішні LLM-платформи. Спроєктовано структуру зберігання симуляційних даних у колекціях часових рядів та запропоновано архітектуру потокової передачі стану симуляції на основі протоколу WebSocket.

Визначено підходи до оцінювання якості та валідації автоматично згенерованих моделей на трьох рівнях: структурна, поведінкова та семантична валідація.

Практичне значення отриманих результатів полягає у потенційному зниженні бар'єра входу в агентно-орієнтоване моделювання для дослідників без глибоких програмістських навичок. Визначено напрями застосування системи в освіті, управлінні надзвичайними ситуаціями, концепції Smart City, епідеміологічному моделюванні, економічних та екологічних дослідженнях.

Виявлено обмеження підходу, зокрема непридатність для континуальних процесів з диференціальними рівняннями, систем реального часу з жорсткими обмеженнями та фізично-орієнтованих симуляцій. Також визначено економічні виклики використання комерційних LLM API та ризики хибної інтерпретації природномовних описів.

Перспективи подальших досліджень насамперед пов'язані з реалізацією спроєктованої метасистеми та її експериментальною перевіркою на прикладних сценаріях. Важливим є прискорення та масштабування симуляцій, зокрема через розробку окремого сервісу виконання симуляцій, що дозволятиме підвищувати продуктивність шляхом горизонтального масштабування, зменшувати час обчислень та підтримувати більші популяції агентів. Значним напрямом є також інтеграція з локальними великими мовними моделями для зменшення залежності від комерційних API та розробка або донавчання спеціалізованих моделей для задач агентно-орієнтованого моделювання з метою забезпечення стабільності поведінки агентів і відтворюваності результатів.

Спроектована система відкриває нові можливості для інтеграції сучасних LLM у агентно-орієнтоване моделювання, потенційно трансформуючи процес створення та аналізу симуляцій у широкому спектрі наукових дисциплін.

Заява про використання генеративного ШІ та технологій на основі ШІ в процесі написання тексту статті

Під час написання цього матеріалу автори використовували ChatGPT та Claude AI для покращення читабельності, виправлення стилістичних і граматичних помилок, а також для форматування використаних джерел. Після використання цих інструментів автори переглянули та відредагували зміст за потреби і взяли на себе повну відповідальність за зміст публікації.

Список використаних джерел

1. Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences, 99*(Supplement 3), 7280–7287. https://doi.org/10.1073/pnas.082080899.

2. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., … Zaremba, W. (2023). *GPT-4 technical report*. arXiv. https://arxiv.org/abs/2303.08774.

3. OpenAI. (2024). *Hello GPT-4o*. https://openai.com/index/hello-gpt-4o /.

4. Raschka, S. (2025). *The state of LLMs 2025: Progress, progress, and predictions*. Sebastian Raschka's Magazine. https://magazine.sebastianraschka.com/p/state-of-llms-2025

5. Chen, W., et al. (2024). *Large language models for generating rules, yay or nay?* arXiv. https://arxiv.org/abs/2406.06835.

6. Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 1–22*. https://doi.org/10.1145/3586183.3606763.

7. Macal, C. M., & North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation, 4*(3), 151–162. https://doi.org/10.1057/jos.2010.3

8. Ferguson, N. M., et al. (2020). *Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand*. Imperial College London. https://doi.org/10.25561/77482.

9. Grimm, V. (2005). Pattern-Oriented modeling of agent-based complex systems: Lessons from ecology. *Science, 310*(5750), 987–991. https://doi.org/10.1126/science.1116681

10. Castellano, C., Fortunato, S., & Loreto, V. (2009). Statistical physics of social dynamics. *Reviews of Modern Physics, 81*(2), 591-646. https://doi.org/10.1103/RevModPhys.81.591

11. Vaswani, A., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

12. OpenAI. (2024). *Learning to reason with LLMs*. https://openai.com/index/learning-to-reason-with-llms/.

13. Wu, Z., et al. (2023). *Smart agent-based modeling: On the use of large language models in computer simulations*. arXiv. https://arxiv.org/abs/2311.06330.

14. Chopra, A., Kumar, S., Giray-Kuru, N., Raskar, R., & Quera-Bofarull, A. (2024). On the limits of agency in agent-based models. *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. arXiv. https://arxiv.org/abs/2409.10568.

15. Avola, D., Cinque, L., Di Mambro, A., Diko, A., Fagioli, A., Foresti, G. L., Marini, M. R., Mecca, A., & Pannone, D. (2022). Low-altitude aerial video surveillance via one-class SVM anomaly detection from textural features in UAV images. *Information*, 13(1), 2. https://doi.org/10.3390/info13010002

16. Newman, S. (2021). *Building microservices: Designing fine-grained systems* (2nd ed.). O'Reilly Media.

17. Uber Technologies. (2024). *Fx: A dependency injection based application framework for Go* [Computer software]. GitHub. https://github.com/uber-go/fx.

18. Code-B. (2024). *The best programming languages for microservices in 2024*. Code-B Blog. https://code-b.dev/blog/best-languages-for-microservices.

19. Cortex. (2023). *A guide to Golang microservices*. Cortex Blog. https://www.cortex.io/post/golang-microservices.

20. Palmer, R. (2024). *Supercharging time series collections: Key enhancements in MongoDB 8.0 with block processing*. MongoDB. https://www.mongodb.com/company/blog/technical/key-enhancements-mongodb-8-0-block-processing.

21. Pimentel, V., & Nickerson, B. G. (2012). Communicating and displaying real-time data with WebSocket. *IEEE Internet Computing*, 16(4), 45–53. https://doi.org/10.1109/MIC.2012.64.

22. Yu, G., Jeong, J. S., Kim, G., Kim, S., & Chun, B. (2022). Orca: A distributed serving system for transformer-based generative models. *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI '22)*, 521–538.

23. Fette, I., & Melnikov, A. (2011). *The WebSocket protocol* (RFC 6455). Internet Engineering Task Force (IETF). https://www.rfc-editor.org/rfc/rfc6455.

References

1. Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences, 99*(suppl_3), 7280-7287. https://doi.org/10.1073/pnas.082080899.

2. Achiam, J., et al. (2023). GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*. https://arxiv.org/abs/2303.08774.

3. OpenAI. (2024). Hello GPT-4o. https://openai.com/index/hello-gpt-4o/.

4. Raschka, S. (2025). The State Of LLMs 2025: Progress, Progress, and Predictions. *Sebastian Raschka's Magazine*. https://magazine.sebastianraschka.com/p/state-of-llms-2025.

5. Chen, W., et al. (2024). Large language models for generating rules, yay or nay? *arXiv preprint arXiv:2406.06835*. https://arxiv.org/abs/2406.06835.

6. Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 1-22. https://doi.org/10.1145/3586183.3606763.

7. Macal, C. M., & North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation*, 4(3), 151-162. https://doi.org/10.1057/jos.2010.3.

8. Ferguson, N. M., et al. (2020). Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand. Imperial College London. https://doi.org/10.25561/77482.

9. Grimm, V., et al. (2005). Pattern-oriented modeling of agent-based complex systems: lessons from ecology. *Science, 310*(5750), 987-991. https://doi.org/10.1126/science.1116681.

10. Castellano, C., Fortunato, S., & Loreto, V. (2009). Statistical physics of social dynamics. *Reviews of Modern Physics, 81*(2), 591-646. https://doi.org/10.1103/RevModPhys.81.591.

11. Vaswani, A., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 5998-6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

12. OpenAI. (2024). Learning to Reason with LLMs. OpenAI Blog. https://openai.com/index/learning-to-reason-with-llms/.

13. Wu, Z., et al. (2023). Smart Agent-Based Modeling: On the Use of Large Language Models in Computer Simulations. *arXiv preprint arXiv:2311.06330*. https://arxiv.org/abs/2311.06330.

14. Chopra, A., Kumar, S., Giray-Kuru, N., Raskar, R., & Quera-Bofarull, A. (2024). On the limits of agency in agent-based models. *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025). arXiv preprint arXiv:2409.10568*. https://arxiv.org/abs/2409.10568.

15. Avola, D., Cinque, L., Di Mambro, A., Diko, A., Fagioli, A., Foresti, G. L., Marini, M. R., Mecca, A., & Pannone, D. (2022). Low-Altitude Aerial Video Surveillance via One-Class SVM Anomaly Detection from Textural Features in UAV Images. *Information, 13*(1), 2. https://doi.org/10.3390/info13010002.

16. Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems* (2nd ed.). O'Reilly Media.

17. Uber Technologies. (2024). Fx: A dependency injection based application framework for Go. GitHub. https://github.com/uber-go/fx.

18. Code-B. (2024). The Best Programming Languages for Microservices in 2024. Code-B Blog. https://code-b.dev/blog/best-languages-for-microservices.

19. Cortex. (2023). A guide to Golang microservices. Cortex Blog. https://www.cortex.io/post/golang-microservices.

20. Palmer, R. (2024). Supercharging time series collections: Key enhancements in MongoDB 8.0 with block processing. MongoDB. https://www.mongodb.com/company/blog/technical/key-enhancements-mongodb-8-0-block-processing.

21. Pimentel, V., & Nickerson, B. G. (2012). Communicating and Displaying Real-Time Data with WebSocket. *IEEE Internet Computing, 16*(4), 45-53. https://doi.org/10.1109/MIC.2012.64.

22. Yu, G., Jeong, J. S., Kim, G., Kim, S., & Chun, B. (2022). Orca: A Distributed Serving System for Transformer-Based Generative Models. *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI '22)*, 521-538.

23. Fette, I., & Melnikov, A. (2011). The WebSocket Protocol. RFC 6455, Internet Engineering Task Force (IETF). https://www.rfc-editor.org/rfc/rfc6455.

Дата першого надходження статті до видання: 13.12.2025
Дата прийняття статті до друку після рецензування: 02.01.2026

UDC 004.94:004.8

Vladyslav Pryshchepa[1], Artem Zadorozhnii[2]

[1]PhD Student of the Department of Information and Computer Systems
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: vladpryshchepa1@gmail.com. **ORCID:** https://orcid.org/0009-0002-7627-0456

[2]PhD inTechnical Sciences, Associate Professor of the Department of Information Technology and Software Engineering
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: zaotroy@gmail.com. **ORCID:** https://orcid.org/0000-0002-3424-7293. **ResearcherID:** F-6358-2016

DESIGN OF LLM-DRIVEN META-SYSTEM FOR DYNAMIC AGENT-ORIENTED MODELING

Over the past three decades, agent-based modeling (ABM) has evolved from a peripheral methodological tool into a recognized research practice across a wide spectrum of scientific disciplines. However, traditional approaches to building agent-based models require significant programming competencies and a deep understanding of simulation frameworks, creating substantial barriers for domain researchers who possess expert knowledge about the systems being modeled but lack the technical skills for practical implementation. The development of large language models (LLMs) has demonstrated impressive capabilities in natural language understanding, code generation, and reasoning, opening perspectives for integrating LLMs into agent-based modeling systems and potentially democratizing access to simulation technologies.

The central problem addressed in this research is the creation of a meta-system for agent-based modeling capable of significantly simplifying the development, execution, and analysis of simulation models of complex socio-technical and natural systems. Such a system must enable natural language model specification, LLM integration directly into the simulation execution cycle, and support simulations with thousands of agents while combining LLM-driven behavior with computationally efficient mechanisms. Existing ABM frameworks such as NetLogo, MASON, and Repast are primarily oriented toward manual model programming and do not support natural language specification, automatic model structure generation, or LLM integration into the simulation execution cycle.

The objective of this research is to design an LLM-driven meta-system for dynamic agent-based modeling that provides natural language specification of simulation models through dialogue with LLM, automatic generation of model data structures including agents, rules, interactions, and environment, as well as mechanisms for quality assessment and validation of created models. An important aspect is the development of LLM integration strategies aimed at exploring trade-offs between the level of agent behavioral intelligence and system computational efficiency.

The proposed architectural solution will be based on a client-server architecture with clear separation of responsibilities between system components. The server part is designed based on microservice architecture using the Go programming language with the Echo HTTP framework and Uber Fx dependency injection mechanisms. LLM interaction will be implemented through separate adapter microservices that follow a unified data exchange contract and can use both locally deployed language models and external LLM platforms. The architecture provides three conceptually different levels of LLM integration: model generation through natural language dialogue, simulation execution with LLM-controlled agent decision-making, and results analysis using LLM in reasoning mode. Six strategies for controlling agent behavior during simulation runtime have been conceptualized, ranging from fully rule-based execution to full individual LLM control, with intermediate variants including archetype-level control and hybrid approaches. MongoDB time-series collections will be used for storing simulation history, leveraging block processing technology for efficient data compression. WebSocket protocol will enable real-time simulation state streaming for visualization.

The research proposes approaches to quality assessment and validation of automatically generated models at three levels: structural validation covering syntactic correctness verification, behavioral validation based on pattern-oriented modeling principles for comparing simulation results with empirical patterns, and semantic validation assessing correspondence between the generated model and the original natural language description. Potential application domains have been identified including education, emergency management, Smart City concept, epidemiological modeling, economic and ecological research. Limitations of the approach have been identified, including unsuitability for continuous processes requiring differential equations, hard real-time applications, physics-oriented simulations, as well as economic challenges when using commercial LLM APIs and risks of misinterpreting natural language descriptions. The designed system opens new possibilities for integrating modern LLMs into agent-based modeling, potentially transforming the process of creating and analyzing simulations across a wide range of scientific disciplines.

Keywords: *agent-based modeling; large language model; LLM; meta-system; Go programming language; microservice architecture; Java integration technologies; model quality; simulation validation.*

Fig.: 3. Table: 2. References: 23.

Прищепа В. О., Задорожній А. О. Проектування LLM-керованої метасистеми для динамічного агентно-орієнтованого моделювання. *Технічні науки та технології*, 2026. № 1(43). С. 229-245. DOI: https://doi.org/10.25140/2411-5363-2026-1(43)-229-245.