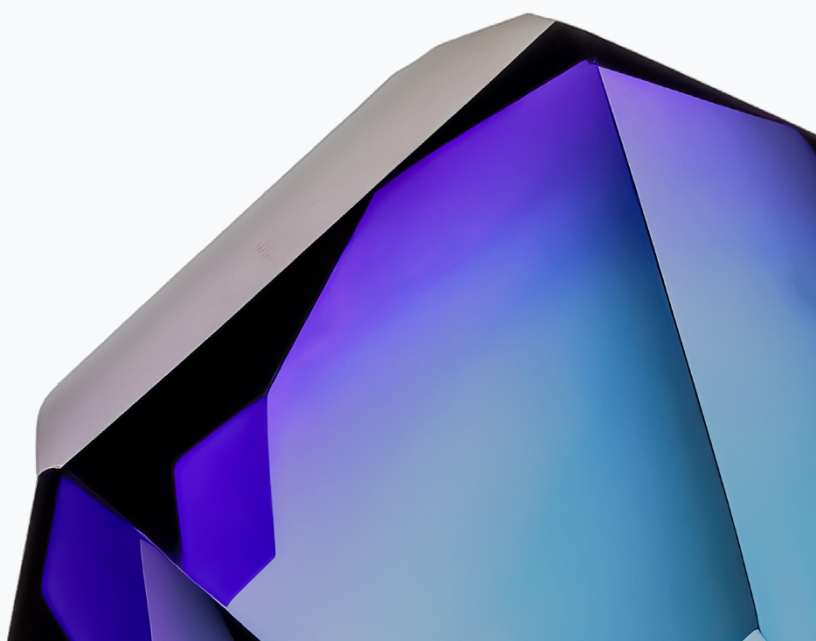


Вступ до агентів

Автори: Алан Блаунт, Антоніо Гуллі, Шубхам Сабу, Майкл Циммерманн та Володимир Вускович



Подяки

Автори

Енріке Чан Майк

Кларк Дерек Іган

Анант Навалгарія

Канчана Патлолла

Джулія Вісінгер

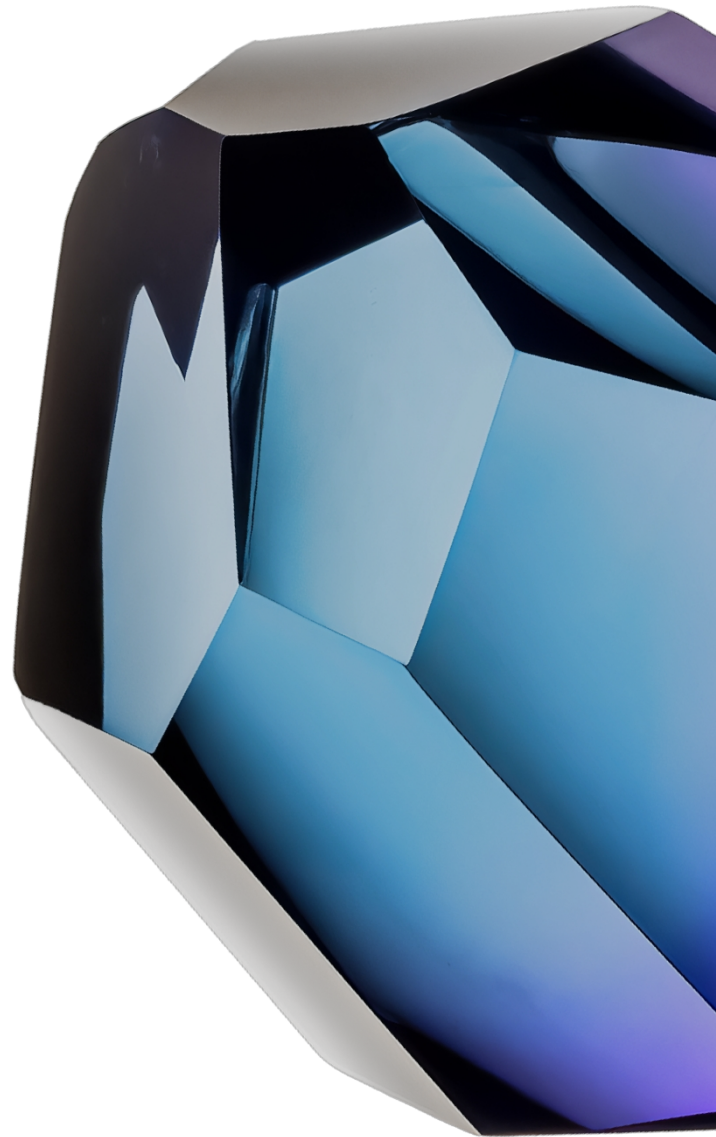
Куратори та редактори

Анант Навалгарія Канчана

Патлолла

Дизайнер

Майкл Ланнінг



Зміст

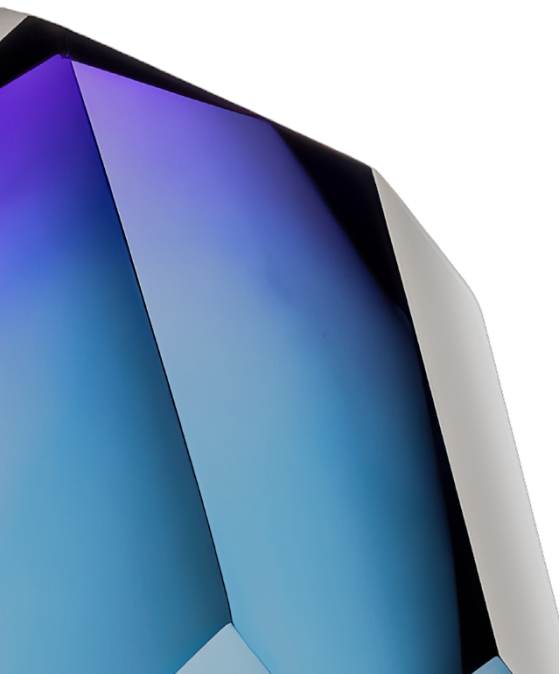
Від прогностного штучного інтелекту до автономних агентів	6
Вступ до штучного інтелекту	8
Процес вирішення проблем агентами	10
Таксономія агентних систем	14
Рівень 0: Основна система міркування	15
Рівень 1: Підключений вирішувач проблем	15
Рівень 2: Стратегічний вирішувач проблем	16
Рівень 3: Співробітницька мультиагентна система	17
Рівень 4: Саморозвиваюча система	18
Основна архітектура агента: модель, інструменти та оркестрування	19
Модель: «Мозок» вашого агента штучного інтелекту	19
Інструменти: «руки» вашого AI-агента	20
Отримання інформації: ґрунтування в реальності	21
Виконання дій: зміна світу	21
Виклик функцій: підключення інструментів до вашого агента	22
.....	
Агенти та гроші	
Рівень оркестрування	22
Основні проектні рішення	23
..... йте, використовуючи знання про сферу діяльності та персонажів	23
..... йте контекстом	24
..... тні системи та шаблони проектування	24
Розробка агентів та послуги	26
Агентно-орієнтований підхід до непередбачуваного	27
Висновок, що має значення: інструментарій успіху, подібний до експерименту А/В	29
Якщо «пройшов/не пройшов»: використання LM Judge	29
Розробка нових метрик: ваше рішення про впровадження або відмову від впровадження	30
Визначення метрик за допомогою OpenTelemetry Traces: відповідь на питання «Чому?»	30

Зміст

Цінуйте відгуки людей: керівництво для вашої автоматизації	31
Сумісність агентів	31
Агенти та люди	32
Агенти та агенти	33
	34

Зміст

Забезпечення єдиного агента: компроміс між довірою та безпекою	34
Ідентичність агента: новий клас принципала	35
Політики обмеження доступу	37
Захист агента ADK	37
Масштабування від одного агента до корпоративного парку	39
Безпека та конфіденційність: зміцнення захисних меж агента	40
Управління агентами: контрольна площа замість розростання	40
Як агенти розвиваються та навчаються	42
Як агенти навчаються та саморозвиваються	43
Моделювання та Agent Gym — наступний рубіж	46
Приклади просунутих агентів	47
Співробітник Google	47
Агент AlphaEvolve	49
Висновок	51
Кінцеві примітки	52





Агенти є природним розвитком мовних моделей, які стали корисними в програмному забезпеченні.

Від прогнозного ШІ до автономних агентів

Штучний інтелект змінюється. Протягом багатьох років основна увага приділялася моделям, які добре справляються з пасивними, дискретними завданнями: відповідають на запитання, перекладають текст або генерують зображення за запитом. Ця парадигма, хоч і є потужною, вимагає постійного керівництва з боку людини на кожному етапі. Зараз ми спостерігаємо зміну парадигми: від штучного інтелекту, який лише прогнозує або створює контент, до нового класу програмного забезпечення, здатного автономно вирішувати проблеми та виконувати завдання.

Ця нова межа побудована навколо агентів ШІ. Агент — це не просто модель ШІ в статичному робочому процесі, це повноцінний додаток, який планує та вживає заходів для досягнення цілей. Він поєднує здатність мовної моделі (LM) **міркувати** з практичною здатністю **діяти**, дозволяючи

йому виконувати складні, багатоетапні завдання, які модель сама по собі не може виконати. Ключовою здатністю є те, що агенти можуть працювати самостійно, визначаючи наступні кроки, необхідні для досягнення мети, без постійного керівництва з боку людини.

Цей документ є першим у серії з п'яти частин і слугує офіційним посібником для розробників, архітекторів та керівників продуктів, які переходять від концептуальних доказів до надійних агентських систем виробничого рівня. Хоча створення простого прототипу є нескладним, забезпечення безпеки, якості та надійності є значним викликом. Цей документ надає вичерпну основу:

- **Основна анатомія:** розкладання агента на три основні компоненти: модель міркування, інструменти для виконання дій та керуючий рівень оркестрування.
- **Таксономія можливостей:** класифікація агентів від простих, пов'язаних між собою засобів вирішення проблем до складних, спільних мультиагентних систем.
- **Архітектурний дизайн:** детальний розгляд практичних аспектів дизайну для кожного компонента, від вибору моделі до впровадження інструментів.
- **Створення для виробництва:** встановлення дисципліни Agent Ops, необхідної для оцінки, налагодження, захисту та масштабування агентських систем від одного екземпляра до цілого парку з корпоративним управлінням.

На основі попередньої [білої книги про агентів](#)¹ та [Agent Companion](#)² ; цей посібник містить основні концепції та стратегічні рамки, необхідні для успішного створення, розгортання та управління новим поколінням інтелектуальних додатків, які можуть міркувати, діяти та спостерігати для досягнення [цілей](#)³ .

Слів недостатньо, щоб описати, як люди взаємодіють з ШІ. Ми схильні антропоморфізувати та використовувати такі людські терміни, як «думати», «розмірковувати» та «знати». Ми ще не маємо слів для позначення «знати з семантичним значенням» проти «знати з високою ймовірністю максимізації функції винагороди». Це два різні типи знання, але результати в 99,Х% випадків однакові.

Вступ до штучного інтелекту

У найпростіших термінах, агент ШІ можна визначити як поєднання моделей, інструментів, рівня оркестрування та служб виконання, які використовують LM у циклі для досягнення мети. Ці чотири елементи формують основну архітектуру будь-якої автономної системи.

- **Модель («мозок»):** Основна мовна модель (LM) або базова модель, яка служить центральним механізмом міркування агента для обробки інформації, оцінки варіантів і прийняття рішень. Тип моделі (загального призначення, точно налаштована або мультимодальна) визначає когнітивні можливості агента. Агентна система є кінцевим куратором вікна контексту вхідних даних LM.
- **Інструменти («руки»):** ці механізми пов'язують міркування агента із зовнішнім світом, дозволяючи виконувати дії, що виходять за межі генерації тексту. Вони включають розширення API, функції коду та сховища даних (наприклад, бази даних або векторні сховища) для доступу до фактичної інформації в режимі реального часу. Агентна система дозволяє LM планувати, які інструменти використовувати, виконувати інструмент і вносити результати інструменту у вікно контексту введення наступного виклику LM.
- **Рівень оркестрування («нервова система»):** керуючий процес, який управляє операційним циклом агента. Він відповідає за планування, пам'ять (стан) та виконання стратегії міркувань. Цей рівень використовує фреймворки підказок та техніки міркувань (такі як

[Chain-of-Thought](#)⁴ або [ReAct](#)⁵) для розбиття складних цілей на етапи та прийняття рішення, коли слід думати, а коли використовувати інструмент. Цей рівень також відповідає за надання агентам пам'яті для «запам'ятовування».

- **Розгортання («тіло і ноги»):** Хоча створення агента на ноутбучі є ефективним для прототипування, саме розгортання у виробництві робить його надійним і доступним сервісом. Це передбачає розміщення агента на безпечному, масштабованому сервері та інтеграцію його з основними виробничими сервісами для моніторингу, реєстрації та управління. Після розгортання користувачі можуть отримати доступ до агента через графічний інтерфейс або програмно за допомогою API Agent-to-Agent (A2A).

Зрештою, створення генеративного агента ШІ — це новий спосіб розробки рішень для виконання завдань. Традиційний розробник діє як «каменярь», точно визначаючи кожен логічний крок. На відміну від цього, розробник агента більше схожий на режисера. Замість того, щоб писати явний код для кожної дії, ви встановлюєте сцену (керівні інструкції та підказки), вибираєте акторський склад (інструменти та API) і надаєте необхідний контекст (дані). Основним завданням стає керівництво цим автономним «актором» для досягнення бажаного результату.

Ви швидко зрозумієте, що найбільша сила LM — її неймовірна гнучкість — є також вашою найбільшою проблемою. Здатність великої мовної моделі робити *що завгодно* ускладнює завдання змусити її надійно і бездоганно виконувати *одну конкретну дію*. Те, що раніше ми називали «інженерією підказок», а тепер називаємо «інженерією контексту», допомагає LM генерувати бажаний результат. Для кожного окремого запит до LM ми вводимо наші інструкції, факти, доступні інструменти для виклику, приклади, історію сеансів, профіль користувача тощо — заповнюючи вікно контексту саме тією інформацією, яка потрібна для отримання необхідних результатів. Агенти — це програмне забезпечення, яке управляє вхідними даними LM для виконання роботи.

Налагодження стає необхідним, коли виникають проблеми. «Agent Ops» по суті переосмислює звичний цикл вимірювання, аналізу та оптимізації системи. За допомогою трасування та журналів ви можете відстежувати «процес мислення» агента, щоб виявити відхилення від запланованого шляху виконання. У міру розвитку моделей та вдосконалення фреймворків роль розробника полягає в наданні

критичні компоненти: досвід у галузі, визначену особистість та безперебійну інтеграцію з інструментами, необхідними для практичного виконання завдань. Важливо пам'ятати, що комплексні оцінки та аналізи часто переважають вплив початкового запиту.

Коли агент точно налаштований з чіткими інструкціями, надійними інструментами та інтегрованим контекстом, що служить пам'яттю, чудовим користувацьким інтерфейсом, здатністю планувати та вирішувати проблеми, а також загальними знаннями про світ, він виходить за межі поняття простої «автоматизації робочого процесу». Він починає функціонувати як спільна одиниця: високоефективний, унікально адаптивний та надзвичайно здібний новий член вашої команди.

По суті, агент — це система, призначена для курації контекстного вікна. Це є безперервним циклом складання контексту, підказки моделі, спостереження за результатом, а потім повторного складання контексту для наступного кроку. Контекст може включати системні інструкції, введення користувача, історію сеансів, довгострокові спогади, базові знання з авторитетних джерел, які інструменти можна використовувати, та результати вже задіяних інструментів. Це складне управління увагою моделі дозволяє її розумовим здібностям вирішувати проблеми в нових обставинах і досягати цілей.

Агентний процес вирішення проблем

Ми визначили агента штучного інтелекту як повну, орієнтовану на ціль програму, яка інтегрує модель міркування, інструменти для дій та керуючий рівень координації. Версія shofi — це «LM у циклі з інструментами для досягнення мети».

Але як насправді *працює* ця система? Що робить агент від моменту отримання запиту до моменту надання результату?

В основі своєї роботи агент діє за безперервним циклічним процесом для досягнення своїх цілей. Хоча цей цикл може бути дуже складним, його можна розбити на п'ять основних етапів, як детально описано в книзі «Агентний дизайн систем»: ⁶

- 1. Отримати завдання:** Процес ініціюється конкретною метою високого рівня. Це завдання надається користувачем (наприклад, «Організувати поїздку моєї команди на майбутню конференцію») або автоматичним тригером (наприклад, «Надійшов новий квиток клієнта з високим пріоритетом»).
- 2. Сканувати ситуацію:** агент сприймає своє оточення, щоб зібрати контекст. Це передбачає доступ рівня оркестрування до доступних ресурсів: «Що сказано в запиті користувача?», «Яка інформація є в моїй терміновій пам'яті? Чи я вже намагався виконати це завдання? Чи дав мені користувач вказівки минулого тижня?», «До чого я маю доступ за допомогою своїх інструментів, таких як календарі, бази даних або API?».
- 3. Продумайте все:** це основний цикл «мислення» агента, що керується моделлю міркування. Агент аналізує **місію** (крок 1) у порівнянні зі **сценою** (крок 2) і розробляє план. Це не одна думка, а часто ланцюжок міркувань: «Щоб забронювати поїздку, мені спочатку потрібно знати, хто входить до команди. Я скористаюся інструментом `get_team_roster`. Потім мені потрібно буде перевірити їхню доступність за допомогою `calendar_api`».
- 4. Вжиття заходів:** Рівень оркестрування виконує перший конкретний крок плану. Він вибирає і викликає відповідний інструмент — викликає API, запускає функцію коду або запитує базу даних. Це агент, який *діє* у світі за межами власного внутрішнього міркування.
- 5. Спостерігати та повторювати:** агент спостерігає за *результатом* своїх дій. Інструмент `get_team_roster` повертає список із п'яти імен. Ця нова інформація додається до контексту або «пам'яті» агента. Потім цикл повторюється, повертаючись до кроку 3: «Тепер, коли я маю список, мій наступний крок — перевірити календар для цих п'яти осіб. Я використаю `calendar_api`».

Цей цикл «Думати, діяти, спостерігати» продовжується — керується **рівнем оркестрування**, обґрунтовується **моделлю** та виконується **інструментами**, доки внутрішній план агента не буде завершений і початкова **місія** не буде виконана.

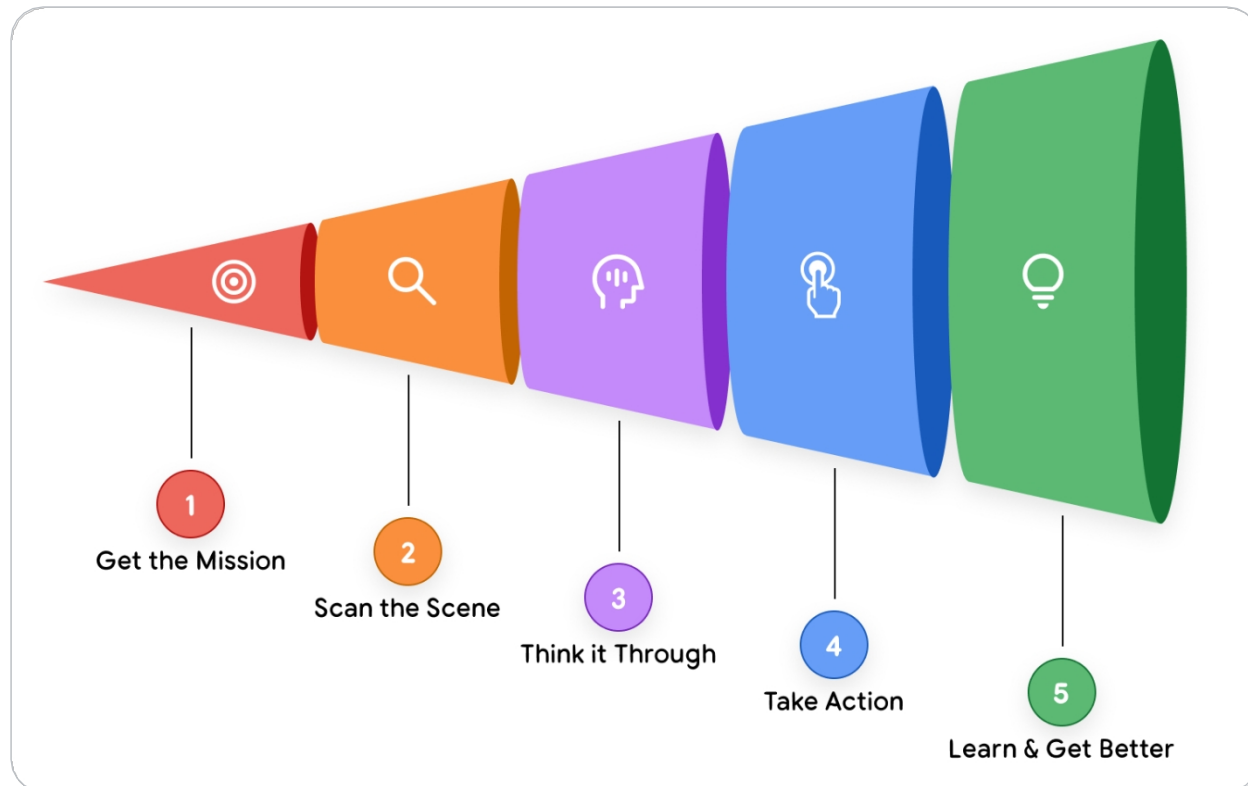


Рисунок 1: Процес вирішення проблем агентною ШІ

Давайте розглянемо реальний приклад того, як агент служби підтримки клієнтів буде працювати в цьому 5-кроковому циклі:

Уявіть, що користувач запитує: «Де моє замовлення № 12345?»

Замість того, щоб діяти відразу, агент спочатку переходить до фази **«Продумай все»**, щоб розробити повну стратегію. Він міркує:

«Користувач хоче дізнатися про статус доставки. Щоб надати повну відповідь, мені потрібен багатетапний план:

- 1. **Ідентифікація:** спочатку я повинен знайти замовлення в нашій внутрішній базі даних, щоб підтвердити його існування та отримати детальну інформацію.*
- 2. **Відстеження:** з цих деталей мені потрібно витягти номер відстеження перевізника. Потім я запитую API зовнішнього перевізника про поточний статус.*
- 3. **Відповіді:** Нарешті, я повинен синтезувати зібрану інформацію в чітку, корисну відповідь для користувача».*

Маючи на увазі цей **багатетапний план**, агент починає його виконання.

На першому етапі «Дії» він виконує перший крок свого плану, викликаючи інструмент `find_order("12345")`. Він спостерігає за результатом — повним записом замовлення, включаючи номер для відстеження «ZYX987».

Рівень оркестрування агента розпізнає, що перший рафі його плану виконано, і негайно переходить до другого. Він діє, викликаючи інструмент `get_shipping_status("ZYX987")`. Він спостерігає новий результат: «Видано для доставки».

Нарешті, успішно виконавши етапи збору даних свого плану, агент переходить до кроку «Reprofi». Він розуміє, що має всі необхідні компоненти, планує остаточне повідомлення і діє, генеруючи відповідь: «Ваше замовлення № 12345 «Видано для доставки»!

Таксономія агентних систем

Розуміння 5-крокового операційного циклу є першою частиною головоломки. Друга частина полягає у визнанні того, що цей цикл можна масштабувати за складністю для створення різних класів агентів. Для архітектора або керівника продукту ключовим початковим рішенням є визначення, *який* тип агента потрібно створити.

Ми можемо класифікувати агентні системи на кілька широких рівнів, кожен з яких базується на можливостях попереднього.



Рисунок 2: Агентна система в 5 кроків

Рівень 0: Основна система міркування

Перш ніж ми зможемо отримати агента, ми повинні оснастити його «мозком» у його найпростішій формі: самим механізмом міркування. У цій конфігурації мовна модель (LM) працює ізольовано, реагуючи виключно на основі своїх величезних попередньо навчених знань, без будь-яких інструментів, пам'яті або взаємодії з реальним середовищем.

Її сила полягає в цьому всебічному навчанні, що дозволяє їй пояснювати усталені поняття та планувати підхід до вирішення проблеми з великою глибиною. Компромісом є повна відсутність усвідомлення в режимі реального часу; вона функціонально «сліпа» до будь-яких подій або фактів, що виходять за межі її навчальних даних.

Наприклад, вона може пояснити правила професійного бейсболу та повну історію команди «Нью-Йорк Янкіз». Але якщо ви запитаете: «Яким був кінцевий рахунок гри «Янкіз» вчора ввечері?», вона не зможе відповісти. Ця гра є конкретною подією в реальному світі, яка відбулася *після* збору даних для навчання, тому ця інформація просто не існує в її знаннях.

Рівень 1: Підключений вирішувач проблем

На цьому рівні механізм міркування стає функціональним агентом, підключаючись до зовнішніх інструментів і використовуючи їх — це компонент «Руки» нашої архітектури. Його вирішення проблем більше не обмежується статичними, заздалегідь навченими знаннями.

Використовуючи 5-кроковий цикл, агент тепер може відповісти на наше попереднє запитання. З огляду на «Місію»: «Яким був остаточний рахунок гри «Янкіз» вчора ввечері?», його крок «Думати» розпізнає це як потребу в даних у реальному часі. Потім його крок «Діяти» викликає інструмент, такий як Google Search API, із відповідною датою та пошуковими термінами. Він «Спостерігає» за результатом пошуку (наприклад, «Янкі виграли 5-3») і синтезує цей факт у остаточну відповідь.

Ця фундаментальна здатність взаємодіяти зі світом — чи то за допомогою інструменту пошуку для отримання результату, фінансового API для отримання поточної ціни акцій, чи то бази даних через Retrieval-Augmented Generation (RAG) — є основною здатністю агента рівня 1.

Рівень 2: Стратегічний вирішувач проблем

Рівень 2 означає значне розширення можливостей, перехід від виконання простих завдань до стратегічного планування складних, багатоаспектних цілей. Ключовою навичкою, яка з'являється на цьому рівні, є **контекстна інженерія**: здатність агента активно відбирати, упаковувати та керувати найбільш релевантною інформацією для кожного етапу свого плану.

Точність агента залежить від сфокусованого, високоякісного контексту. Контекстна інженерія курує обмежену увагу моделі, щоб запобігти перевантаженню та забезпечити ефективну роботу.

Наприклад, розглянемо «Місію»: **«Знайти хорошу кав'ярню на півдорозі між моїм офісом за адресою 1600 Amphitheatre Parkway, Маунтін-В'ю, та офісом мого клієнта за адресою 1 Market St, Сан-Франциско».**

Агент рівня 2 почне створювати план:

- 1. Подумає:** «Спочатку я повинен знайти середину шляху».
 - **Діяти:** запустити інструмент **«Карти»** з обома адресами.
 - **Спостерігати:** «Півдорозі — це Millbrae, Каліфорнія».
- 2. Подумайте:** «Тепер я повинен знайти кав'ярні в Millbrae. Користувач попросив «хороші», тому я буду шукати заклади з рейтингом 4 зірки або вище».

- **Дія:** Викличте інструмент `google_places` із запитом `query="coffee shop in Millbrae, CA", min_rating=4.0`. (Це контекстна інженерія — вона автоматично створила новий, цілеспрямований пошуковий запит на основі результату попереднього кроку).
- **Спостерігайте:** «Пошук повертає результати «Millbrae Coffee» та «The Daily Grind».

3. Подумайте: «Я синтезую ці результати і представлю їх користувачеві».

Таке стратегічне планування також дозволяє надавати проактивну допомогу, наприклад, агент, який читає довгий електронний лист із підтвердженням рейсу, аналізує ключовий контекст (номер рейсу, дату) і додає його до вашого календаря.

Рівень 3: Спільна мультиагентна система

На найвищому рівні парадигма повністю змінюється. Ми відходимо від створення єдиного, всемогутнього «суперагента» і рухаємося в бік «команди фахівців», що працюють у концепті, моделі, яка безпосередньо відображає людську організацію. Колективна сила системи полягає в цьому розподілі праці.

Тут агенти розглядають інших агентів як інструменти. Уявіть собі агента «Керівник проекту», який отримує «Завдання»: «Запустити наші нові навушники «Solaris».

Агент-керівник проекту не виконує всю роботу самостійно. Він діє, створюючи нові місії для своєї команди спеціалізованих агентів, подібно до того, як це відбувається в реальному житті:

1. **Делегати до MarketResearchAgent:** «Проаналізуйте ціни конкурентів на навушники з функцією шумозаглушення. До завтра надайте підсумковий документ».
2. **Делегує MarketingAgent:** «Складіть три версії прес-релізу, використовуючи специфікацію продукту «Solaris» як контекст».

- 3. Доручення WebDevAgent:** «Створіть HTML-код нової сторінки продукту на основі доданих макетів дизайну».

Ця модель співпраці, хоча і обмежена можливостями сучасних LM, є передовою технологією автоматизації складних бізнес-процесів від початку до кінця.

Рівень 4: Саморозвиваюча система

Рівень 4 представляє собою глибокий стрибок від делегування до автономного створення та адаптації. На цьому рівні агентна система може виявляти прогалини у своїх власних можливостях і динамічно створювати нові інструменти або навіть нових агентів для їх заповнення. Вона переходить від використання фіксованого набору ресурсів до їх активного розширення.

На нашому прикладі агент «Керівник проекту», якому доручено запуск «Solaris», може усвідомити, що йому потрібно стежити за настройками в соціальних мережах, але в його команді немає такого інструменту або агента.
Його команді.

- 1. Думати (мета-міркування):** «Я повинен відстежувати обговорення «Solaris» у соціальних мережах, але у мене немає таких можливостей».
- 2. Діяти (автономне створення):** замість того, щоб зазнати невдачі, він викликає високорівневий інструмент AgentCreator з новою місією: «Створити нового агента, який відстежує соціальні медіа на наявність ключових слів «навушники Solaris», виконує аналіз настроїв і щодня подає звіт».
- 3. Спостерігайте:** новий спеціалізований SentimentAnalysisAgent створюється, тестується і додається до команди на льоту, готовий до виконання початкового завдання.

Такий рівень автономності, коли система може динамічно розширювати власні можливості, перетворює команду агентів на організацію, яка дійсно навчається та розвивається.

Архітектура основного агента: модель, інструменти та оркестрування

Ми знаємо, що робить агент і як він може масштабуватися. Але як його насправді *створити*? Перехід від концепції до коду полягає в конкретному архітектурному дизайні трьох основних компонентів.

Модель: «мозок» вашого AI-агента

LM є розумовим ядром вашого агента, і його вибір є критично важливим архітектурним рішенням, яке визначає когнітивні здібності, експлуатаційні витрати та швидкість роботи вашого агента. Однак розглядати цей вибір як просту справу вибору моделі з найвищим балом тестування є типовим шляхом до невдачі. Успіх агента в виробничому середовищі рідко визначається загальними академічними тестами.

Реальний успіх вимагає моделі, яка виділяється за **основними характеристиками агента**: чудовим **мисленням** для вирішення складних, багатоетапних проблем та надійним **використанням інструментів** для взаємодії зі [cBіTOM](#)⁷.

Щоб зробити це добре, визначте бізнес-проблему, а потім протестуйте моделі за показниками, які безпосередньо відповідають цьому результату. Якщо ваш агент повинен писати код, протестуйте його на вашій приватній базі коду. Якщо він обробляє страхові вимоги, оцініть його здатність витягувати інформацію з ваших конкретних форматів документів. Потім цей аналіз необхідно зіставити з практичними аспектами вартості та затримки. «Найкраща» модель — це та, яка знаходиться на оптимальному перетині якості, швидкості та ціни для *вашого* [конкретного завдання](#)⁸.

Ви можете вибрати більше ніж одну модель, «команду фахівців». Не варто розбивати горіх кувалдою. Надійна архітектура агента може використовувати передову модель, таку як **Gemini 2.5 Pro**, для важкої роботи з початкового планування та складного міркування, але потім інтелектуально направляти простіші завдання з великим обсягом, такі як класифікація намірів користувачів або узагальнення тексту, до набагато швидшої та економічно ефективнішої моделі, такої як **Gemini 2.5 Flash**. Маршрутизація моделі може бути автоматичною або жорстко запрограмованою, але є ключовою стратегією для оптимізації як [продуктивності, так і вартості](#)⁽⁹⁾.

Той самий принцип застосовується до обробки різних типів даних. Хоча вбудована мультимодальна модель, така як [Gemini live mode](#)⁽¹⁰⁾, пропонує спрощений шлях до обробки зображень та аудіо, альтернативою є використання спеціалізованих інструментів, таких як [Cloud Vision API](#)¹¹ або [Speech-to-Text API](#)¹².

У цій моделі світ спочатку перетворюється на текст, який потім передається до моделі, що працює виключно з мовою, для обробки. Це додає гнучкості та дозволяє використовувати найкращі компоненти, але також значно ускладнює процес.

Нарешті, сфера штучного інтелекту перебуває в стані постійної, швидкої еволюції. Модель, яку ви обираєте сьогодні, через півроку буде застарілою. Підхід «налаштував і забув» є неприйнятним. Щоб підготуватися до цієї реальності, потрібно інвестувати в гнучку операційну структуру — [практику](#) «Agent Ops»⁽¹³⁾. Завдяки надійному конвеєру CI/CD, який постійно оцінює нові моделі відповідно до ваших ключових бізнес-показників, ви можете зменшити ризики та прискорити оновлення, гарантуючи, що ваш агент завжди працює на базі найкращого доступного інтелекту, без необхідності повного перегляду архітектури.

Інструменти: «руки» вашого AI-агента

Якщо модель є мозком агента, то інструменти — це руки, які пов'язують його міркування з реальністю. Вони дозволяють агенту вийти за межі статичних навчальних даних, щоб отримувати інформацію в режимі реального часу та діяти у світі. Надійний інтерфейс інструменту — це цикл із трьох етапів: визначення того, що може робити інструмент, його запуск та спостереження за результатом.

Ось декілька основних типів інструментів, які розробники надають своїм агентам. Більш детальну інформацію можна знайти в технічному документі, присвяченому інструментам для агентів, що входить до цієї серії.

Отримання інформації: ґрунтування в реальності

Найбільш фундаментальним інструментом є можливість доступу до актуальної інформації.

Технологія RAG (Retrieval-Augmented Generation) надає агенту «бібліотечну картку» для запиту зовнішніх знань, які часто зберігаються у **векторних базах даних** або **графіках знань**, починаючи від внутрішніх документів компанії і закінчуючи веб-знаннями через Google Search. Для структурованих даних інструменти **Natural Language to SQL (NL2SQL)** дозволяють агенту запитувати бази даних, щоб відповісти на аналітичні питання, такі як «Які наші продукти були найбільш продаваними минулого кварталу?». Шукаючи інформацію перед тим, як говорити — чи то в документі, чи то в базі даних — агент ґрунтується на фактах, що значно зменшуючи галюцинації.

Виконання дій: зміна світу

Справжня сила агентів розкривається, коли вони переходять від читання інформації до активних дій. Об'єднуючи існуючі **API** та функції коду в інструменти, агент може надсилати електронні листи, планувати зустрічі або оновлювати записи про клієнтів у ServiceNow. Для більш динамічних завдань агент може **писати та виконувати код на льоту**. У безпечній пісочниці він може генерувати SQL-запит або скрипт Python для вирішення складної проблеми або виконання обчислення, перетворюючись із досвідченого помічника на [автономного виконавця](#)⁽¹⁴⁾.

Сюди також входять інструменти для взаємодії з людьми. Агент може використовувати інструмент **Human in the Loop (HITL)** для призупинення робочого процесу та запиту підтвердження (наприклад, `ask_for_confirmation()`) або запиту конкретної інформації з інтерфейсу користувача (наприклад, `ask_for_date_input()`), забезпечуючи участь людини в прийнятті критичних рішень. HITL може бути реалізований за допомогою SMS-повідомлень та завдання в базі даних.

Виклик функцій: підключення інструментів до вашого агента

Щоб агент міг надійно виконувати «виклик функцій» і використовувати інструменти, йому потрібні чіткі інструкції, безпечні з'єднання та [оркестрування](#)¹⁵. Це забезпечують давно існуючі стандарти, такі як специфікація **OpenAPI**, що надає агенту структурований контракт, який описує призначення інструменту, необхідні параметри та очікуваний відгук. Ця схема дозволяє моделі щоразу генерувати правильний виклик функції та інтерпретувати відгук API. Для спрощення пошуку та підключення до інструментів популярними стали відкриті стандарти, такі як **Model Context Protocol (MCP)**, оскільки вони є більш [зручними](#)¹⁶. Крім того, деякі моделі мають вбудовані інструменти, наприклад Gemini з вбудованим Google Search, де виклик функції відбувається як частина [самого](#) виклику LM¹⁷.

Рівень оркестрування

Якщо модель є мозком агента, а інструменти — його руками, то рівень оркестрування є центральною нервовою системою, яка їх з'єднує. Це двигун, який запускає цикл «Думай, дій, Спостерігай», стан машини, що керує поведінкою агента, і місце, де ретельно розроблена логіка розробника оживає. Цей рівень — не просто трубопровід; це диригент усієї симфонії агента, який вирішує, коли модель повинна міркувати, який інструмент повинен діяти і як результати цієї дії повинні впливати на наступний рух.

Основні проектні рішення

Перше архітектурне рішення — визначення ступеня автономності агента. Вибір існує в широкому діапазоні. З одного боку, ви маєте детерміновані, передбачувані робочі процеси, які викликають LM як інструмент для виконання конкретного завдання — невелику кількість штучного інтелекту для доповнення існуючого процесу. З іншого боку, ви маєте LM у ролі водія, який динамічно адаптується, планує та виконує завдання для досягнення мети.

Паралельним вибором є метод реалізації. Безкодові конструктори пропонують швидкість і доступність, даючи можливість бізнес-користувачам автоматизувати структуровані завдання і швидко створювати прості агенти. Для більш складних, критично важливих систем, фреймворки, орієнтовані на код, такі як [Agent Development Kit \(ADK\)](#) від Google¹⁸, забезпечують глибокий контроль, можливість налаштування та інтеграцію, необхідні інженерам.

Незалежно від підходу, необхідна фреймворк виробничого рівня. Він повинен бути **відкритим**, що дозволяє підключати будь-яку модель або інструмент, щоб запобігти залежності від постачальника. Він повинен забезпечувати точний контроль, дозволяючи гібридний підхід, де недетермінований розум LM регулюється жорстко запрограмованими бізнес-правилами. Найважливіше, що фреймворк повинен бути побудований для **спостережності**. Коли агент поводить несподівано, ви не можете просто поставити точку зупинки в «думці» моделі. Надійний фреймворк генерує детальні траси та журнали, виявляючи всю траєкторію міркувань: внутрішній монолог моделі, інструмент, який вона вибрала, параметри, які вона згенерувала, та результат, який вона спостерігала.

Надавайте інструкції з використанням знань про домен та персонажа

У цих рамках найпотужнішим важелем розробника є навчання агента з використанням знань у певній галузі та чіткої персони. Це досягається за допомогою системного запиту або набору основних інструкцій. Це не просто проста команда, це конституція агента.

Тут ви кажете йому: «Ти — корисний агент служби підтримки клієнтів Acme Corp...» і надаєте обмеження, бажану схему виводу, правила взаємодії, конкретний тон голосу та чіткі вказівки щодо того, коли і чому він повинен використовувати свої інструменти. Кілька прикладів сценаріїв в інструкціях зазвичай є дуже ефективними.

Доповнення контекстом

«Пам'ять» агента організована у вікні контексту LM під час виконання. Для більш повного занурення в тему дивіться білу книгу, присвячену пам'яті агента, в цій серії.

Короткочасна пам'ять — це активний «блокнот» агента, який зберігає історію поточної розмови. Вона відстежує послідовність пар (дія, спостереження) з поточного циклу, надаючи безпосередній контекст, необхідний моделі для прийняття рішення про подальші дії. Це може бути реалізовано як абстракції, такі як стан, афіакти, сесії або потоки.

Довгострокова пам'ять забезпечує стійкість між сесіями. З архітектурної точки зору це майже завжди реалізується як інший спеціалізований інструмент — система RAG, підключена до векторної бази даних або пошукової системи. Оркестратор надає агенту можливість попередньо завантажувати та активно запитувати власну історію, дозволяючи йому «запам'ятовувати» уподобання користувача або результат подібного завдання кілька тижнів тому для справді персоналізованого та безперервного [досвіду](#).¹⁹

Багатоагентні системи та дизайн-патерни

У міру зростання складності завдань створення єдиного, всемогутнього «суперагента» стає неефективним. Більш ефективним рішенням є застосування підходу «команди фахівців», який відображає людську організацію. Це є основою **мультиагентної системи**: складний процес є

розділені на окремі підзадачі, і кожна з них призначається спеціальному спеціалізованому агенту ШІ. Такий розподіл праці дозволяє кожному агенту бути простішим, більш цілеспрямованим і легшим у створенні, тестуванні та обслуговуванні, що ідеально підходить для динамічних або довготривалих бізнес-процесів.

Архітектори можуть покладатися на перевірені шаблони проектування агентів, хоча можливості агентів і, отже, шаблони швидко еволюціонують ⁽²⁰⁾. Для динамічних або нелінійних завдань необхідний шаблон **координатора**. Він вводить агента-«менеджера», який аналізує складний запит, сегментує основне завдання і інтелектуально направляє кожне підзавдання відповідному спеціалізованому агенту (наприклад, досліднику, письменнику або програмісту). Потім координатор агрегує відповіді від кожного спеціаліста, щоб сформулювати остаточну, вичерпну відповідь.

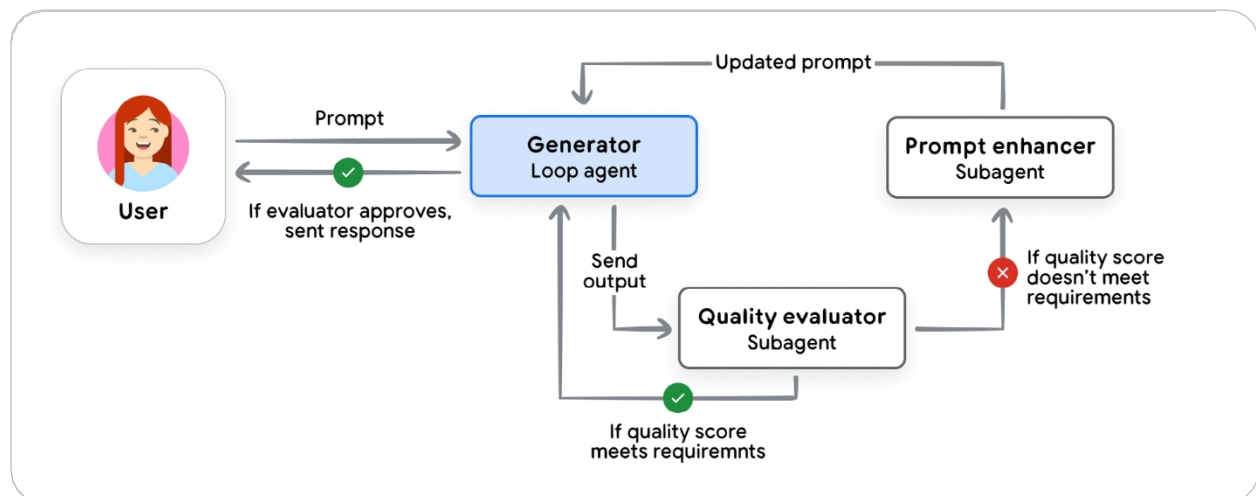


Рисунок 3: Шаблон «ітеративного вдосконалення» з

<https://cloud.google.com/architecture/choose-design-pattern-agentic-ai-system>

Для більш лінійних робочих процесів краще підходить **послідовний** патерн, який діє як цифрова конвеєрна лінія, де вихідні дані одного агента стають прямими вхідними даними для наступного. Інші ключові патерни зосереджуються на якості та безпеці. Патерн **ітеративного вдосконалення** створює цикл зворотного зв'язку, використовуючи агента-«генератора» для створення контенту та агента-«критика» для його оцінки відповідно до

стандартами якості. Для завдань з високими ставками критично важливим є шаблон **«Людина в циклі» (HITL)**, який створює навмисну паузу в робочому процесі, щоб отримати схвалення від людини, перш ніж агент вчинить значну дію.

Розгортання агента та послуги

Після створення локального агента його потрібно розгорнути на сервері, де він буде працювати постійно і де його зможуть використовувати інші люди та агенти. Продовжуючи нашу аналогію, розгортання та послуги будуть тілом і ногами нашого агента. Для ефективної роботи агент потребує декількох послуг, історії сеансів, збереження пам'яті тощо. Як розробник агента, ви також будете відповідати за те, що ви реєструєте, та які заходи безпеки ви вживаєте для забезпечення конфіденційності даних, їхнього зберігання та дотримання нормативних вимог. Усі ці послуги входять до сфери застосування під час розгортання агентів у виробництві.

На щастя, розробники агентів можуть покладатися на десятиліття досвіду в області інфраструктури хостингу додатків. Адже агенти — це нова форма програмного забезпечення, і до них застосовуються багато тих самих принципів. Розробники можуть покладатися на спеціально розроблені, специфічні для агентів опції розгортання, такі як **Vefiex AI Agent Engine**, який підтримує середовище виконання та все інше в одній [платформі](#)²¹. Для розробників програмного забезпечення, які хочуть більш безпосередньо контролювати свої стеки додатків або розгорнути агенти в існуючій інфраструктурі DevOps, будь-який агент і більшість послуг агентів можна додати до контейнера Docker і розгорнути на стандартних для галузі середовищах виконання, таких як [Cloud Run або GKE](#)²².

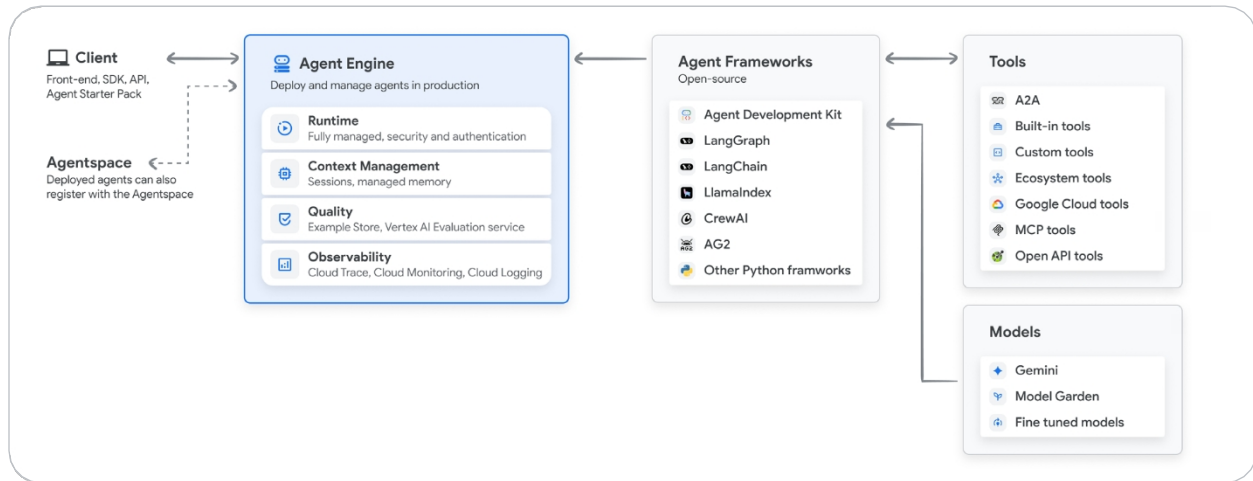


Рисунок 4: Конструктор Vefiex AI Agent з

<https://cloud.google.com/vefiex-ai/generative-ai/docs/agent-engine/overview>

Якщо ви не є розробником програмного забезпечення та експертом DevOps, процес розгортання вашого першого агента може здатися складним. Багато фреймворків агентів спрощують цей процес за допомогою команди **розгортання** або спеціальної платформи для розгортання агента, і їх слід використовувати для попереднього вивчення та впровадження. Перехід до безпечного та готового до виробництва середовища зазвичай вимагає більших часових витрат та застосування найкращих практик, включаючи CI/CD та автоматизоване тестування [ваших агентів](#)²³.

Agent Ops: структурований підхід до непередбачуваного

Під час створення перших агентів ви будете вручну тестувати їхню поведінку знову і знову. Чи працює додана функція? Чи не спричинило виправлення помилки іншу проблему? Тестування є нормальним процесом у розробці програмного забезпечення, але у випадку з генеративною ШІ воно працює інакше.

Перехід від традиційного детермінованого програмного забезпечення до стохастичних агентських систем вимагає нової філософії роботи. Традиційні тести програмних модулів могли просто перевіряти, **чи відповідає результат очікуваному**, але це не працює, коли відповідь агента є ймовірнісною за своєю конструкцією. Крім того, оскільки мова є складною, зазвичай для оцінки «якості» потрібен LM — щоб відповідь агента відповідала всім вимогам, не містила нічого зайвого і мала правильний тон.

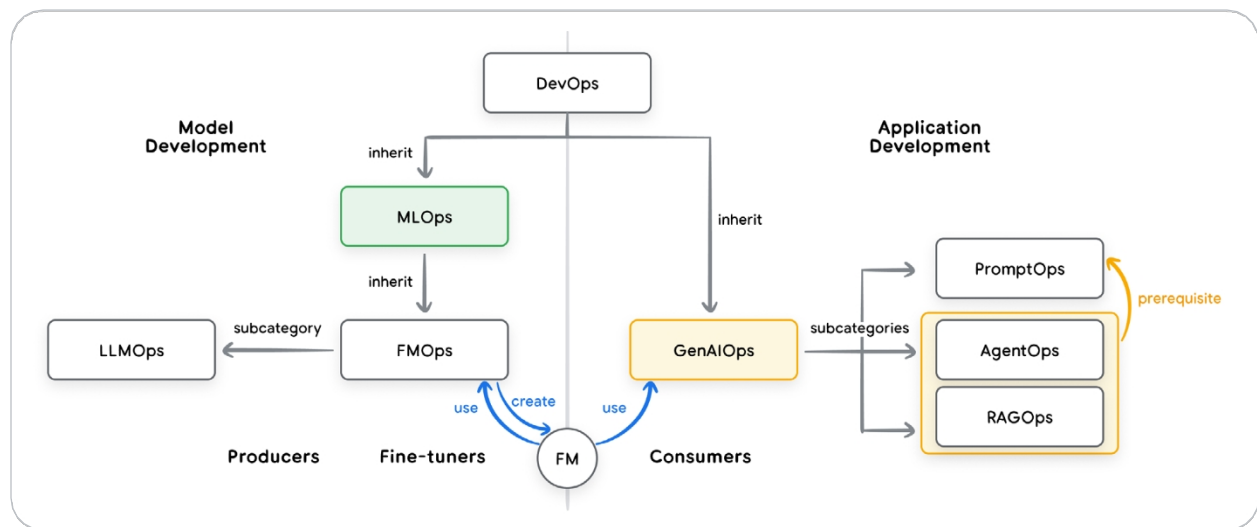


Рисунок 5: Взаємозв'язки між операційними доменами DevOps, MLOps та GenAIOps з <https://medium.com/@sokratis.kafiakis/genai-in-production-mlops-or-genaiops-25691c9becd0>

Agent Ops — це дисциплінований, структурований підхід до управління цією новою реальністю. Це природна еволюція DevOps та MLOps, адаптована до унікальних викликів, пов'язаних із створенням, розгортанням та управлінням агентами ШІ, що перетворює непередбачуваність із недоліку на керовану, вимірювану та надійну функцію.²⁴ Для більш повного занурення в тему дивіться білу книгу, присвячену якості агентів, у цій серії.

Вимірюйте те, що має значення: інструментарій успіху, подібний до експерименту A/B

Перш ніж вдосконалювати свого агента, ви повинні визначити, що означає «краще» в контексті вашого бізнесу. Сформулюйте свою стратегію спостережності як A/B-тест і запитайте себе: які ключові показники ефективності (KPI) підтверджують, що агент приносить користь? Ці показники повинні виходити за межі технічної правильності та вимірювати реальний вплив: рівень досягнення цілей, показники задоволеності користувачів, затримка виконання завдань, операційні витрати на взаємодію та, що найважливіше, вплив на бізнес-цілі, такі як дохід, конверсія або утримання клієнтів.

Цей загальний огляд стане орієнтиром для подальшого тестування, направить вас на шлях розробки, орієнтованої на метрики, і дозволить розрахувати рентабельність інвестицій.

Якість замість «пройшов/не пройшов»: використання LM Judge

Бізнес-метрики не дають відповіді на питання, чи правильно працює агент. Оскільки простий прохід/непрохід неможливий, ми переходимо до оцінки якості за допомогою «LM як судді». Це передбачає використання потужної моделі для оцінки результатів роботи агента за заздалегідь визначеною шкалою: чи дав він правильну відповідь? Чи була відповідь обґрунтованою? Чи дотримувався він інструкцій? Ця автоматизована оцінка, що проводиться на основі золотого набору даних підказок, забезпечує послідовну оцінку якості.

Створення наборів даних для оцінки, які включають ідеальні (або «золоті») питання та правильні відповіді, може бути нудним процесом. Щоб їх створити, слід взяти зразки сценаріїв з існуючих виробничих або розробницьких взаємодій з агентом. Набір даних повинен охоплювати весь спектр випадків використання, з якими, як ви очікуєте, будуть мати справу ваші користувачі, а також кілька несподіваних випадків. Хоча інвестиції в оцінку швидко окупаються, результати оцінки завжди повинні бути

перевірятися експертом у цій галузі, перш ніж їх можна буде визнати дійсними. Все частіше курація та підтримка цих оцінок стає ключовою відповідальністю менеджерів продуктів за підтримки експертів у цій галузі.

Розробка на основі метрик: ваше рішення про впровадження або відмову від нього

Після автоматизації десятків сценаріїв оцінки та встановлення надійних показників якості ви можете впевнено тестувати зміни у вашому агенті розробки. Процес простий: запустіть нову версію на всьому наборі даних оцінки та безпосередньо порівняйте її показники з існуючою виробничою версією. Ця надійна система виключає припущення, забезпечуючи впевненість у кожному розгортанні. Хоча автоматизовані оцінки є критично важливими, не забувайте про інші важливі фактори, такі як затримка, вартість та рівень успішності завдань. Для максимальної безпеки використовуйте розгортання A/B, щоб повільно впроваджувати нові версії та порівнювати ці реальні виробничі показники з вашими оцінками моделювання.

Налагодження за допомогою OpenTelemetry Traces: відповідь на питання «Чому?»

Коли ваші показники падають або користувач повідомляє про помилку, вам потрібно зрозуміти «чому». Трасування OpenTelemetry — це високоточна покрокова реєстрація всього шляху виконання агента (траєкторії), що дозволяє налагоджувати [кроки агента](#).²⁵ За допомогою трасування ви можете побачити точний запит, надісланий до моделі, внутрішнє міркування моделі (якщо доступне), конкретний інструмент, який вона вирішила викликати, точні параметри, які вона згенерувала для цього інструменту, та необроблені дані, які повернулися як спостереження. Траси можуть бути складними, коли ви дивитеся на них вперше, але вони надають деталі, необхідні для діагностики та усунення першопричини будь-якої проблеми. Важливі деталі траси можуть бути перетворені на показники, але перегляд трас в першу чергу призначений для налагодження, а не

огляди продуктивності. Дані трасування можна безперешкодно збирати на таких платформах, як **Google Cloud Trace**, які візуалізують і здійснюють пошук серед величезних обсягів трасування, спрощуючи аналіз першопричин.

Цінуйте відгуки людей: керуйте своєю автоматизацією

Відгуки людей — це не прикрість, з якою потрібно боротися, а найцінніший і найбагатший на дані ресурс, який ви маєте для вдосконалення свого агента. Коли користувач подає звіт про помилку або натискає кнопку «не подобається», він робить вам подарунок: новий, реальний крайній випадок, який пропустили ваші автоматизовані сценарії оцінки. Збір і агрегація цих даних є критично важливими; коли ви бачите статистично значущу кількість подібних повідомлень про помилки або падіння показників, ви повинні пов'язати ці випадки з вашою аналітичною платформою, щоб отримати інформацію та запустити алефіси для операційних проблем. Ефективний процес Agent Ops «замикає цикл», фіксуючи цей відгук, відтворюючи проблему та перетворюючи цей конкретний сценарій на новий постійний тестовий випадок у вашому наборі даних оцінки. Це гарантує, що ви не тільки виправите помилку, але й захистите систему від повторення всієї цієї категорії помилок у майбутньому.

Сумісність агентів

Після створення високоякісних агентів ви хочете мати можливість з'єднати їх з користувачами та іншими агентами. У нашій аналогії з тілом людини це буде обличчя агента. Існує різниця між підключенням до агентів та підключенням агентів до даних і API; [агенти не є інструментами](#)²⁶. Припустимо, ви вже підключили інструменти до своїх агентів, тепер давайте розглянемо, як ви можете інтегрувати своїх агентів у ширшу екосистему.

Агенти та люди

Найпоширенішою формою взаємодії агента з людиною є взаємодія через користувацький інтерфейс. У найпростішому вигляді це чат-бот, де користувач вводить запит, а агент, що виконує функції серверної служби, обробляє його і повертає блок тексту. Більш просунуті агенти можуть надавати структуровані дані, такі як JSON, для забезпечення багатого та динамічного інтерфейсу.

Людина в

(HITL) включають уточнення намірів, розширення цілей, підтвердження та запити на уточнення.

Використання комп'ютера — це категорія інструментів, де LM бере під контроль користувацький інтерфейс, часто з взаємодією та наглядом людини. Агент, що підтримує використання комп'ютера, може вирішити, що найкращим наступним кроком буде перехід на нову сторінку, виділення певної кнопки або попереднє заповнення форми відповідною [інформацією](#)²⁷.

Замість того, щоб агент використовував інтерфейс від імені користувача, LM може змінювати інтерфейс користувача відповідно до потреб моменту. Це можна зробити за допомогою інструментів, які контролюють інтерфейс користувача ([MCP UI](#))²⁸, або спеціалізованих систем обміну повідомленнями інтерфейсу користувача, які можуть синхронізувати стан клієнта з агентом ([AG UI](#))²⁹, і навіть генерації індивідуальних інтерфейсів ([A2UI](#))³⁰.

Звичайно, взаємодія людини не обмежується екранами та клавіатурами. Сучасні агенти долають текстові бар'єри та переходять до мультимодального спілкування в режимі реального часу з «живим режимом», створюючи більш природне, схоже на людське спілкування. Такі технології, як [Gemini Live API](#)³¹, забезпечують двосторонній потік, дозволяючи користувачеві розмовляти з агентом і переривати його, як у природній розмові.

Ця можливість кардинально змінює характер співпраці між агентом і людиною. Маючи доступ до камери та мікрофона пристрою, агент може бачити те, що бачить користувач, і чути те, що він говорить, відповідаючи генерованою мовою з затримкою, що імітує людську розмову.

Це відкриває широкий спектр можливостей використання, які просто неможливі з текстом, від технічного фахівця, який отримує інструкції в режимі hands-free під час ремонту обладнання, до покупця, який отримує поради щодо стилю в режимі реального часу. Це робить агента більш інтуїтивним і доступним помічником.

Агенти та агенти

Так само, як агенти повинні зв'язуватися з людьми, вони також повинні зв'язуватися між собою. У міру розширення використання штучного інтелекту в підприємстві різні команди будуть створювати різних спеціалізованих агентів. Без загального стандарту для їх з'єднання потрібно було б створити заплутану мережу крихких, індивідуальних інтеграцій API, які неможливо підтримувати. Основна проблема має два аспекти: виявлення (як мій агент знаходить інших агентів і дізнається, що вони можуть робити?) та комунікація (як ми можемо гарантувати, що вони розмовляють однією мовою?).

Протокол Agent2Agent (A2A) — це відкритий стандарт, розроблений для вирішення цієї проблеми. Він діє як універсальний рукопис для агентної економіки. A2A дозволяє будь-якому агенту опублікувати цифрову «візитну картку», відому як Agent Card. Цей простий файл JSON повідомляє про можливості агента, його кінцеву точку мережі та облікові дані безпеки, необхідні для взаємодії з ним.

Це робить пошук простим і стандартизованим. На відміну від MCP, який зосереджується на вирішенні транзакційних запитів, комунікація Agent 2 Agent зазвичай призначена для додаткового вирішення проблем.

Після виявлення агенти спілкуються за допомогою архітектури, орієнтованої на завдання. Замість простого запиту-відповіді, взаємодії формуються як асинхронні «завдання». Клієнтський агент надсилає запит на завдання серверному агенту, який потім може надавати потокові оновлення, працюючи над проблемою через довготривале з'єднання. Цей надійний, стандартизований протокол комунікації є останньою частиною пазлу, що дозволяє створити спільні багатоагентні системи рівня 3, які представляють собою передову автоматизацію. A2A перетворює сукупність ізольованих агентів на справжню, взаємодіючу екосистему.

Агенти та гроші

Оскільки агенти штучного інтелекту виконують за нас все більше завдань, деякі з них пов'язані з купівлею або продажем, веденням переговорів або сприянням у здійсненні транзакцій. Сучасна мережа створена для людей, які натискають кнопку «купити», і відповідальність лежить на людині. Якщо автономний агент натискає «купити», це створює кризу довіри — якщо щось піде не так, хто буде винен? Це складні питання авторизації, автентичності та відповідальності. Щоб розкрити справжню економіку агентів, нам потрібні нові стандарти, які дозволять агентам безпечно та надійно здійснювати транзакції від імені своїх користувачів.

Ця нова галузь ще далеко не сформована, але два ключові протоколи прокладають їй шлях. **Agent Payments Protocol (AP2)** — це відкритий протокол, розроблений як остаточна мова для агентської комерції. Він розширює протоколи, такі як A2A, шляхом введення криптографічно підписаних цифрових «мандатів». Вони виступають як перевірений доказ намірів користувача, створюючи незаперечний аудиторський слід для кожної транзакції. Це дозволяє агенту безпечно переглядати, вести переговори та здійснювати транзакції в глобальному масштабі на основі делегованих повноважень від користувача. Доповненням до цього є **x402**, відкритий протокол інтернет-платежів, який використовує стандартний код статусу HTTP 402 «Необхідна оплата». Він забезпечує безперешкодні мікроплатежі між машинами, дозволяючи агенту оплачувати такі речі, як доступ до API або цифровий контент, на основі оплати за використання, без необхідності створення складних облікових записів або передплат. Разом ці протоколи створюють фундаментальний рівень довіри для агентного вебу.

Забезпечення безпеки єдиного агента: компроміс між довірою та безпекою

Коли ви створюєте свого першого агента ШІ, ви відразу стикаєтеся з фундаментальною суперечністю: компроміс між корисністю та безпекою. Щоб агент був корисним, ви повинні надати йому повноваження — автономію у прийнятті рішень та інструменти для виконання таких дій, як надсилання електронних листів або запити до баз даних. Однак кожна крапля повноважень, яку ви надаєте, несе відповідний ризик. Основними проблемами безпеки є **несанкціоновані дії** — ненавмисні або шкідливі дії —

^{та}**розкриття конфіденційних даних.** Ви хочете надати своєму агенту достатньо свободи, щоб він міг виконувати свою роботу, але й достатньо контролю, щоб він не потрапив у небажану ситуацію, особливо якщо це стосується незворотних дій або приватних даних вашої компанії.³²

Щоб впоратися з цим, ви не можете покладатися виключно на судження моделі ШІ, оскільки її можна маніпулювати за допомогою таких технік, як [введення підказки](#)³³. Натомість найкращим підходом є гібридний [підхід глибокої оборони](#)⁽³⁴⁾. Перший рівень складається з **традиційних детермінованих** захисних бар'єрів — набору жорстко запрограмованих правил, які діють як точка безпеки поза межами міркувань моделі. Це може бути механізм політики, який блокує будь-яку покупку на суму понад 100 доларів або вимагає явного підтвердження користувача, перш ніж агент зможе взаємодіяти із зовнішнім API. Цей рівень забезпечує передбачувані, піддаються аудиту жорсткі обмеження повноважень агента.

Другий рівень використовує **захист на основі міркувань**, використовуючи ШІ для забезпечення безпеки ШІ. Це передбачає навчання моделі бути більш стійкою до атак (суперницьке навчання) та використання менших, спеціалізованих «моделей охорони», які діють як аналітики безпеки. Ці моделі можуть перевіряти запропонований агент планом *перед* його виконанням, позначаючи потенційно ризиковані або такі, що порушують політику, кроки для перегляду. Ця гібридна модель, що поєднує жорстку чіткість коду з контекстуальною обізнаністю ШІ, створює надійну систему безпеки навіть для одного агента, гарантуючи, що його потужність завжди відповідає його призначенню.

Ідентичність агента: новий клас суб'єкта

У традиційній моделі безпеки є людські користувачі, які можуть використовувати OAuth або SSO, і є служби, які використовують IAM або облікові записи служб. Агенти додають третю категорію принципалів. Агент — це не просто шматок коду; це автономний актор, новий тип *принципала*, який вимагає власної перевіреної ідентичності. Так само, як співробітникам видають ідентифікаційні картки, кожному агенту на платформі повинен бути виданий безпечний, перевірений «цифровий паспорт». Цей агент

Ідентичність відрізняється від ідентичності користувача, який її викликав, та розробника, який її створив. Це фундаментальна зміна в підході до управління ідентичністю та доступом (IAM) в підприємстві.

Перевірка кожної ідентичності та контроль доступу для всіх них є основою безпеки агента. Як тільки агент має криптографічно перевірену ідентичність (часто за допомогою стандартів, таких як [SPIFFE](#)³⁵), йому можуть бути надані власні специфічні дозволи з мінімальними привілеями. [SalesAgent](#) отримує доступ для читання/запису до CRM, тоді як [HRonboardingAgent](#) отримує явну відмову. Такий детальний контроль є критично важливим. Він гарантує, що навіть якщо один агент буде скомпрометований або поведеться несподівано, потенційний радіус ураження буде обмеженим. Без конструкції ідентичності агента агенти не можуть працювати від імені людей з обмеженими делегованими повноваженнями.

Основна суть	Аутентифікація / Верифікація	Примітки
Користувачі	Автентифіковані за допомогою OAuth або SSO	Люди, які мають повну автономію та відповідальність за свої дії
Агенти (нова категорія принципів)	Перевірені за допомогою SPIFFE	Агенти мають делеговані повноваження і діють від імені користувачів
Сервісні облікові записи	Інтегровані в IAM	Додатки та контейнери, повністю детерміновані, не відповідальні за дії

Таблиця 1: Неповний приклад різних категорій учасників для аутентифікації

Політики обмеження доступу

Політика є формою авторизації (AuthZ), відмінною від автентифікації (AuthN). Зазвичай політики обмежують можливості суб'єкта; наприклад, «Користувачі з відділу маркетингу можуть отримати доступ лише до цих 27 кінцевих точок API і не можуть виконувати команди DELETE». Під час розробки агентів нам потрібно застосовувати дозволи до агентів, їхніх інструментів, інших внутрішніх агентів, контексту, яким вони можуть ділитися, та віддалених агентів. Подумайте про це так: якщо ви додаєте всі API, дані, інструменти та агенти до своєї системи, то ви повинні обмежити доступ до підмножини лише тих можливостей, які необхідні для виконання їхніх завдань. Це рекомендований підхід: застосування принципу мінімальних привілеїв, залишаючись [контекстуально релевантним](#).⁽³⁶⁾

Захист агента ADK

Після встановлення основних принципів ідентичності та політики, забезпечення безпеки агента, створеного за допомогою Agent Development Kit (ADK), стає практичним завданням із застосування цих концепцій через код і [конфігурацію](#).³⁷

Як описано вище, цей процес вимагає чіткого визначення ідентичностей: облікового запису користувача (наприклад, OAuth), облікового запису служби (для виконання коду), ідентичності агента (для використання делегованих повноважень). Після завершення автентифікації наступний рівень захисту передбачає встановлення політик для обмеження доступу до служб. Це часто робиться на рівні управління API, разом з управлінням, що підтримує служби MCP та A2A.

Наступний рівень полягає у вбудовуванні захисних механізмів у ваші інструменти, моделі та субагенти для забезпечення дотримання політик. Це гарантує, що незалежно від причин LM або зловмисних підказок, власна логіка інструменту відмовлятиметься виконувати небезпечні або несумісні з політикою дії. Такий підхід забезпечує передбачувану та піддається аудиту базову безпеку, перетворюючи абстрактні політики безпеки на [конкретний, надійний код](#).³⁸

Для більш динамічної безпеки, яка може адаптуватися до поведінки агента під час виконання, ADK надає **функції зворотного виклику та плагіни**. Функція `before_tool_callback` дозволяє перевіряти параметри виклику інструменту перед його запуском, порівнюючи їх з поточним станом агента, щоб запобігти невідповідним діям. Для більш багаторазового використання політик ви можете створювати плагіни. Поширеним зразком є «Gemini as a Judge» ⁽³⁹⁾, який використовує швидку та недорогу модель, таку як Gemini Flash-Lite або вашу власну налагоджену модель Gemma, для перевірки вхідних даних користувача та вихідних даних агента на предмет швидких вставлень або шкідливого вмісту в режимі реального часу.

Для організацій, які віддають перевагу повністю керованому рішенню корпоративного рівня для цих динамічних перевірок, **Model Armor** може бути інтегрований як додаткова послуга. Model Armor діє як спеціалізований рівень безпеки, який перевіряє запити та відповіді на наявність широкого спектру загроз, включаючи введення запитів, спроби джейлбрейку, витік конфіденційних даних (PII) та [шкідливі URL-адреси](#)⁴⁰. Передаючи ці складні завдання безпеки спеціалізованій службі, розробники можуть забезпечити постійний надійний захист без необхідності самостійно створювати та підтримувати ці захисні механізми. Цей гібридний підхід в ADK, що поєднує надійну ідентифікацію, детерміновану логіку в інструменті, динамічні захисні механізми на базі штучного інтелекту та додаткові керовані служби, такі як Model Armor, дозволяє створити єдиний агент, який є одночасно потужним і надійним.

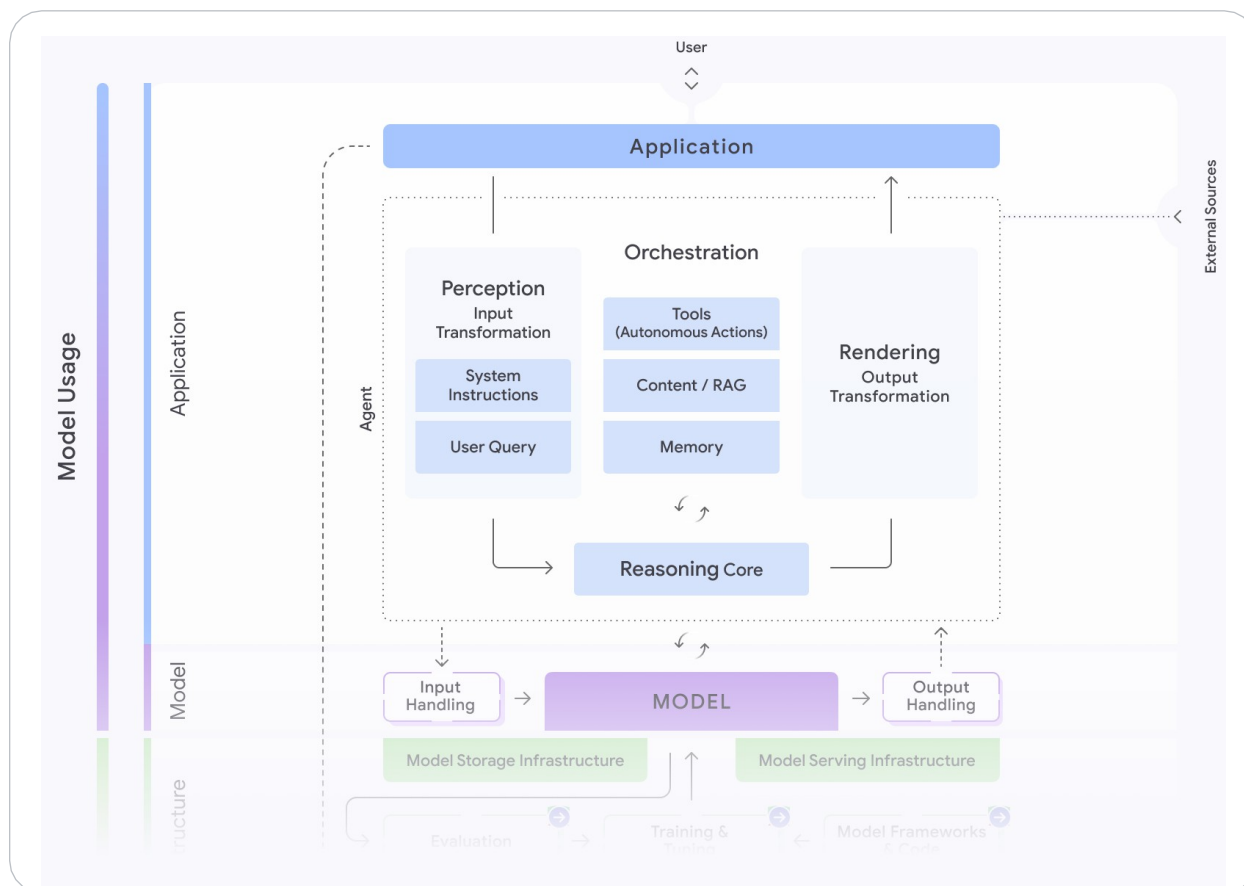


Рисунок 6: Безпека та агенти з <https://saif.google/focus-on-agents>

Масштабування від одного агента до корпоративного парку

Успіх виробництва одного агента ШІ — це тріумф. Масштабування до парку з сотень агентів — це виклик для архітектури. Якщо ви створюєте одного або двох агентів, ваші турботи переважно стосуються безпеки. Якщо ви створюєте багато агентів, ви повинні розробити системи, здатні обробляти набагато більше. Подібно до розростання API, коли агенти та інструменти поширюються по всій організації,

вони створюють нову, складну мережу взаємодій, потоків даних і потенційних вразливостей безпеки. Управління цією складністю вимагає вищого рівня управління, що інтегрує всі ваші ідентичності та політики і переносить їх у центральну площину контролю.

Безпека та конфіденційність: зміцнення агентської межі

Платформа корпоративного рівня повинна вирішувати унікальні проблеми безпеки та конфіденційності, властиві генеративній ШІ, навіть якщо працює лише один агент. Сам агент стає новим вектором атаки. Зловмисники можуть спробувати **ввести команди**, щоб перехопити інструкції агента, або **отруїти дані**, щоб пошкодити інформацію, яку він використовує для навчання або RAG. Крім того, агент із недостатніми обмеженнями може ненавмисно виточити конфіденційні дані клієнтів або власницьку інформацію у своїх відповідях.

Надійна платформа забезпечує стратегію глибокого захисту для зменшення цих ризиків. Вона працює з даними, гарантуючи, що конфіденційна інформація підприємства ніколи не використовується для навчання базових моделей і захищена такими засобами контролю, як VPC Service Controls. Вона вимагає фільтрування вхідних і вихідних даних, діючи як брандмауер для запитів і відповідей. Нарешті, платформа повинна пропонувати договірні захисні заходи, такі як відшкодування інтелектуальної власності як для даних навчання, так і для згенерованих вихідних даних, надаючи підприємствам юридичну та технічну впевненість, необхідну для розгортання агентів у виробництві.

Управління агентами: контрольна площа замість розростання

У міру поширення агентів та їхніх інструментів в організації вони створюють нову, складну мережу взаємодій та потенційних вразливостей, що часто називають «розростанням агентів». Для управління цим процесом необхідно вийти за межі захисту окремих агентів і впровадити архітектурний підхід вищого рівня: центральний шлюз, який служить контрольною площиною для всієї діяльності агентів.

Уявіть собі метушливий мегаполіс з тисячами автономних транспортних засобів — користувачів, агентів та інструментів — які рухаються з певною метою. Без світлофорів, номерних знаків та центральної системи управління панував би хаос. Підхід на основі шлюзу створює таку систему управління, встановлюючи обов'язкову точку входу для всього трафіку агентів, включаючи запити користувача до агента або взаємодію з користувальницьким інтерфейсом

, виклики агента до інструменту (через MCP), співпрацю агента з агентом (через A2A) та прямі запити на висновок до LM. Знаходячись на цьому критичному перехресті, організація може перевіряти, маршрутизувати, контролювати та керувати кожною взаємодією.

Ця площа контролю виконує дві основні взаємопов'язані функції:

- 1. Забезпечення дотримання політик під час виконання:** вона діє як архітектурна точка контролю для впровадження безпеки. Вона обробляє аутентифікацію («Чи знаю я, хто цей актор?») та авторизацію («Чи мають вони дозвіл на це?»). Централізація забезпечення дотримання забезпечує «єдине вікно» для спостереження, створюючи загальні журнали, метрики та траси для кожної транзакції. Це перетворює сплетіння розрізнених агентів та робочих процесів на прозору та піддану аудиту систему.
- 2. Централізоване управління:** для ефективного виконання політик шлюз потребує джерела достовірної інформації. Це забезпечується центральним реєстром — корпоративним магазином додатків для агентів та інструментів. Цей реєстр дозволяє розробникам знаходити та повторно використовувати існуючі ресурси, запобігаючи надмірній роботі, а адміністраторам — отримувати повний перелік ресурсів. Що ще важливіше, він забезпечує формальний життєвий цикл агентів та інструментів, дозволяючи проводити перевірки безпеки перед публікацією, версіями та створенням детальних політик, які визначають, які бізнес-підрозділи можуть отримати доступ до яких агентів.

Поєднуючи шлюз виконання з центральним реєстром управління, організація перетворює ризик хаотичного розростання на керовану, безпечну та ефективну екосистему.

Вартість та надійність: інфраструктурна основа

Зрештою, агенти корпоративного рівня повинні бути як надійними, так і економічно ефективними. Агент, який часто виходить з ладу або надає повільні результати, має негативний ROI. І навпаки, агент, який є надто дорогим, не може масштабуватися для задоволення потреб бізнесу. Базова повинна бути спроектована таким чином, щоб управляти цим компромісом, забезпечуючи безпеку та відповідність нормативним вимогам і суверенітету даних.

У деяких випадках необхідною функцією є масштабування до нуля, коли трафік до певного агента або підфункції є нерегулярним. Для критично важливих робочих навантажень, чутливих до затримок, платформа повинна пропонувати виділену, гарантовану потужність, таку як [Provisioned Throughput](#)⁴¹ для послуг LM або 99,9% угод про рівень обслуговування (SLA) для середовищ виконання, таких як [Cloud Run](#)⁴². Це забезпечує передбачувану продуктивність, гарантуючи, що ваші найважливіші агенти завжди реагують, навіть під великим навантаженням. Забезпечуючи такий спектр інфраструктурних опцій у поєднанні з комплексним моніторингом як витрат, так і продуктивності, ви створюєте остаточну, необхідну основу для масштабування агентів ШІ від перспективної інновації до основного, надійного компонента підприємства.

Як агенти розвиваються та навчаються

Агенти, розгорнуті в реальному світі, працюють у динамічних середовищах, де політики, технології та формати даних постійно змінюються. Без здатності адаптуватися продуктивність агента з часом погіршуватиметься — процес, який часто називають «старінням» — що призведе до втрати корисності та довіри. Ручне оновлення великого парку агентів, щоб йти в ногу з цими змінами, є неекономічним і повільним. Більш масштабованим рішенням є розробка агентів, які можуть самостійно навчатися та розвиватися, покращуючи свою якість роботи з [мінімальними інженерними зусиллями](#) ⁽⁴³⁾.

Як агенти навчаються та самостійно розвиваються

Подібно до людей, агенти навчаються на основі досвіду та зовнішніх сигналів. Цей процес навчання живиться кількома джерелами інформації:

- **Досвід виконання:** агенти навчаються на основі афіфактів виконання, таких як журнали сеансів, траси та пам'ять, які фіксують успіхи, невдачі, взаємодію інструментів та траєкторії прийняття рішень траєкторій. Важливо, що це включає зворотний зв'язок Human-in-the-Loop (HITL), який забезпечує авторитетні виправлення та рекомендації.
- **Зовнішні сигнали:** Навчання також стимулюється новими зовнішніми документами, такими як оновлені політики підприємства, державні регуляторні рекомендації або критика від інших агентів.

Ця інформація потім використовується для оптимізації майбутньої поведінки агента. Замість того, щоб просто підсумовувати минулі взаємодії, просунуті системи створюють узагальнені афіаки для керівництва майбутніми завданнями. Найуспішніші техніки адаптації поділяються на дві категорії:

- **Покращена контекстна інженерія:** система постійно вдосконалює свої підказки, приклади з декількома спробами та інформацію, яку вона отримує з пам'яті. Оптимізуючи контекст, що надається LM для кожного завдання, вона збільшує ймовірність успіху.
- **Оптимізація та створення інструментів:** Агент може виявляти прогалини у своїх можливостях і вживати заходів для їх усунення. Це може включати отримання доступу до нового інструменту, створення нового інструменту на льоту (наприклад, скрипту Python) або модифікацію існуючого інструменту (наприклад, оновлення схеми API).

Додаткові методи оптимізації, такі як динамічна реконфігурація багатоагентних шаблонів проектування або використання підкріплювального навчання на основі зворотного зв'язку від людини (RLHF), є активними напрямками досліджень.

Приклад: вивчення нових правил відповідності

Розглянемо корпоративного агента, що працює в суворо регульованій галузі, такій як фінанси або біологічні науки. Його завданням є створення репофісів, які повинні відповідати правилам конфіденційності та нормативним вимогам (наприклад, GDPR).

Це можна реалізувати за допомогою багатоагентного робочого процесу:

1. **Агент запитів** отримує необроблені дані у відповідь на запит користувача.
2. **Агент репофінгу** синтезує ці дані в проект репофі.
3. **Агент-критик**, озброєний відомими правилами відповідності, перевіряє репофі. Якщо він виявляє неоднозначність або потребує остаточного затвердження, він передає справу до людського доменного експерта.
4. **Агент навчання** спостерігає за всією взаємодією, приділяючи особливу увагу коригувальним відгукам від експерта-людини. Потім він узагальнює ці відгуки в нові, придатні для повторного використання правила (наприклад, оновлені правила для агента-критика або уточнені контексти для агента-репофі).

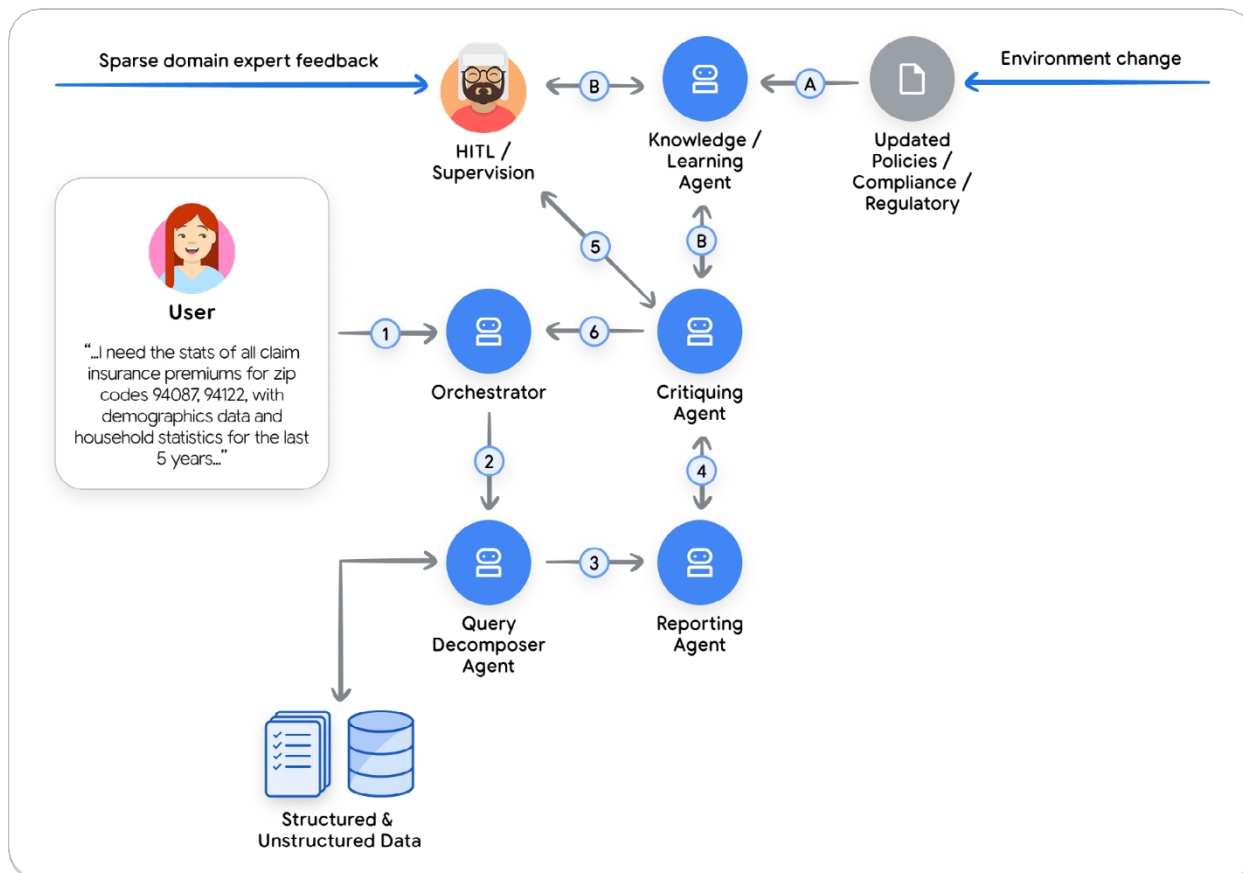


Рисунок 7: Приклад багатоагентного робочого процесу для правил відповідності

Наприклад, якщо людина експерта вказує, що статистичні дані про домогосподарства повинні бути анонімізовані, навчальний агент записує це виправлення. Наступного разу, коли буде згенеровано подібний герофі, агент-критик автоматично застосує це нове правило, зменшивши необхідність втручання людини. Цей цикл критики, зворотного зв'язку від людини та узагальнення дозволяє системі самостійно адаптуватися до мінливих вимог щодо дотримання норм⁴⁴.

Моделювання та Agent Gym — наступний рубіж

Представлений нами шаблон проектування можна класифікувати як вбудоване навчання, де агенти повинні навчатися за допомогою ресурсів та шаблону проектування, з якими вони були розроблені. Зараз досліджуються більш

Зараз досліджуються більш просунуті підходи, де існує спеціальна платформа, розроблена для оптимізації мультиагентної системи в офлайн-процесах за допомогою просунутих інструментів та можливостей, які не є частиною мультиагентного середовища виконання. Ключовими атрибутами такого [Agent Gym](#)⁴⁵ є:

1. Він не знаходиться на шляху виконання. Це автономна позавиробнича платформа, тому вона може використовувати будь-яку модель LM, офлайн-інструменти, хмарні додатки тощо
2. Він пропонує середовище моделювання, тому агент може «тренуватися» на нових даних і навчатися. Це середовище моделювання чудово підходить для «методу проб і помилок» з багатьма шляхами оптимізації.
3. Він може викликати просунуті генератори синтетичних даних, які направляють симуляцію, щоб вона була якомога реалістичнішою, і перевіряють агента під тиском (це може включати просунуті техніки, такі як «червона команда», динамічна оцінка та сімейство критичних агентів).
4. Арсенал інструментів оптимізації не є фіксованим, і він може приймати нові інструменти (через відкриті протоколи, такі як MCP або A2A), або в більш просунутому налаштуванні - вивчати нові концепції та створювати інструменти навколо них.
5. Нарешті, навіть такі конструкції, як Agent Gym, можуть бути не в змозі подолати крайній випадок *sefiain* (через добре відому проблему «племінних знань» в підприємстві). У таких випадках ми бачимо, що Agent Gym здатний підключатися до людської структури доменних експертів і консультуватися з ними щодо правильного набору результатів, щоб скерувати наступний набір оптимізацій.

Приклади просунутих агентів

Google Co-Scientist

Co-Scientist — це просунутий агент штучного інтелекту, призначений для роботи в якості візуального дослідницького співробітника, що прискорює наукові відкриття шляхом систематичного дослідження складних проблемних просторів. Він дозволяє дослідникам визначити мету, закріпити агента в певних публічних і власних джерелах знань, а потім генерувати і оцінювати ландшафт нових гіпотез.

Щоб досягти цього, Co-Scientist створює цілу екосистему агентів, які співпрацюють між собою.

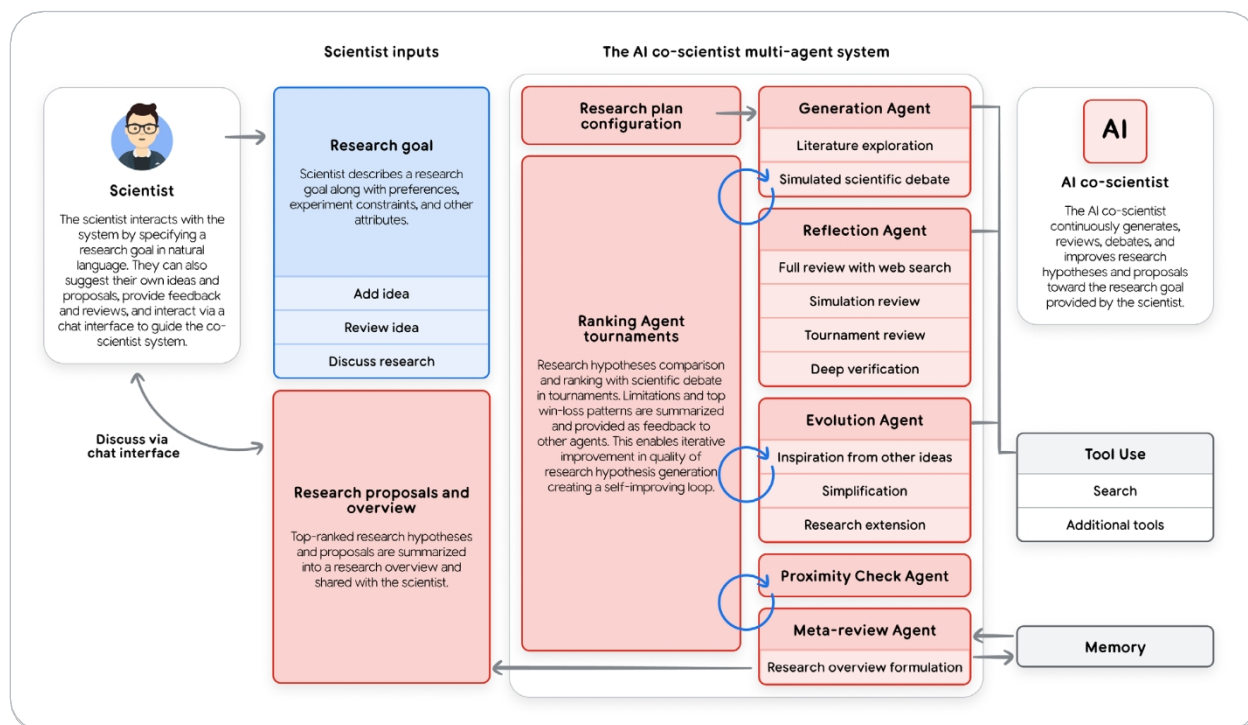


Рисунок 8: Система проектування AI co-scientist

Уявіть собі цю систему як менеджера дослідницького проекту. ШІ спочатку бере широку дослідницьку мету і створює детальний план проекту. Потім агент «Супервізор» діє як менеджер, делегуючи завдання команді спеціалізованих агентів і розподіляючи ресурси, такі як обчислювальна потужність. Така структура забезпечує легке масштабування проекту і вдосконалення методів роботи команди в процесі досягнення кінцевої мети.

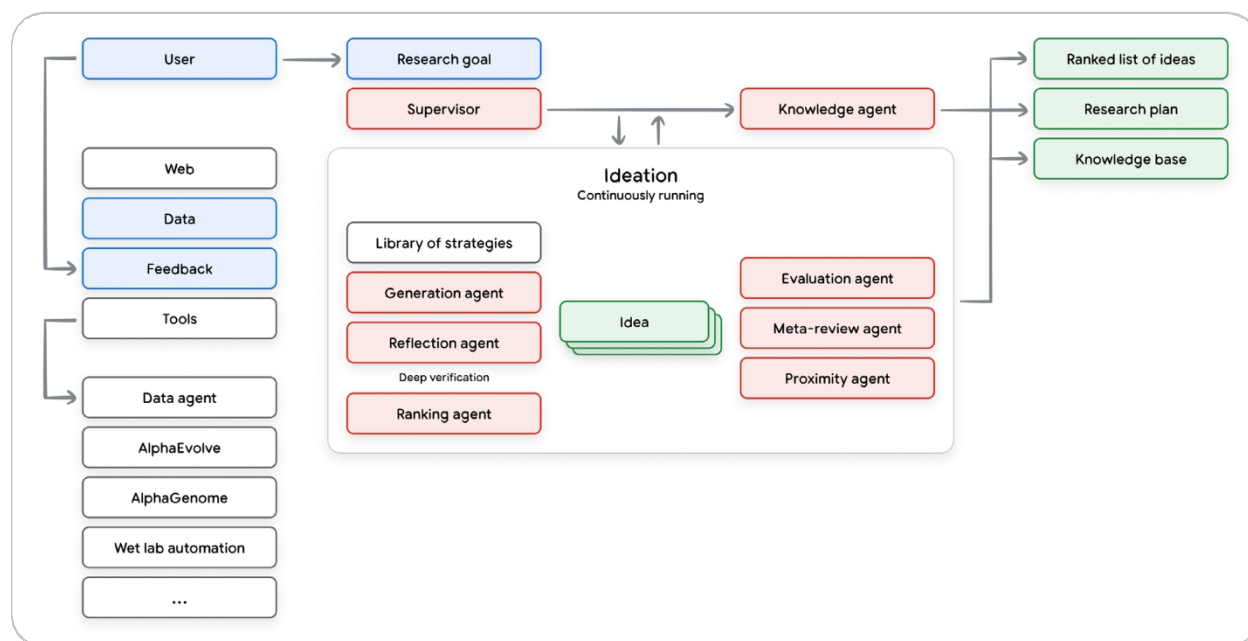


Рисунок 9: Робочий процес мультиагента-співдослідника

Різні агенти працюють годинами, а то й днями, і постійно вдосконалюють сформовані гіпотези, запускаючи цикли та метацикли, які покращують не тільки сформовані ідеї, але й спосіб, у який ми оцінюємо та створюємо нові ідеї.

Агент AlphaEvolve

Іншим прикладом просунутої системи агентів є AlphaEvolve, агент штучного інтелекту, який відкриває та оптимізує алгоритми для складних задач у математиці та інформатиці.

AlphaEvolve працює, поєднуючи творче генерування коду наших мовних моделей Gemini з автоматизованою системою оцінки. Він використовує еволюційний процес: штучний інтелект генерує потенційні рішення, оцінювач оцінює їх, а найперспективніші ідеї використовуються як натхнення для наступного покоління коду.

Цей підхід вже привів до значних проривів, серед яких:

- Покращення ефективності центрів обробки даних Google, проектування мікросхем та навчання штучного інтелекту.
- Відкриття швидших алгоритмів множення матриць.
- Пошук нових рішень відкритих математичних проблем.

AlphaEvolve чудово справляється з проблемами, де перевірити якість рішення набагато простіше, ніж його знайти.

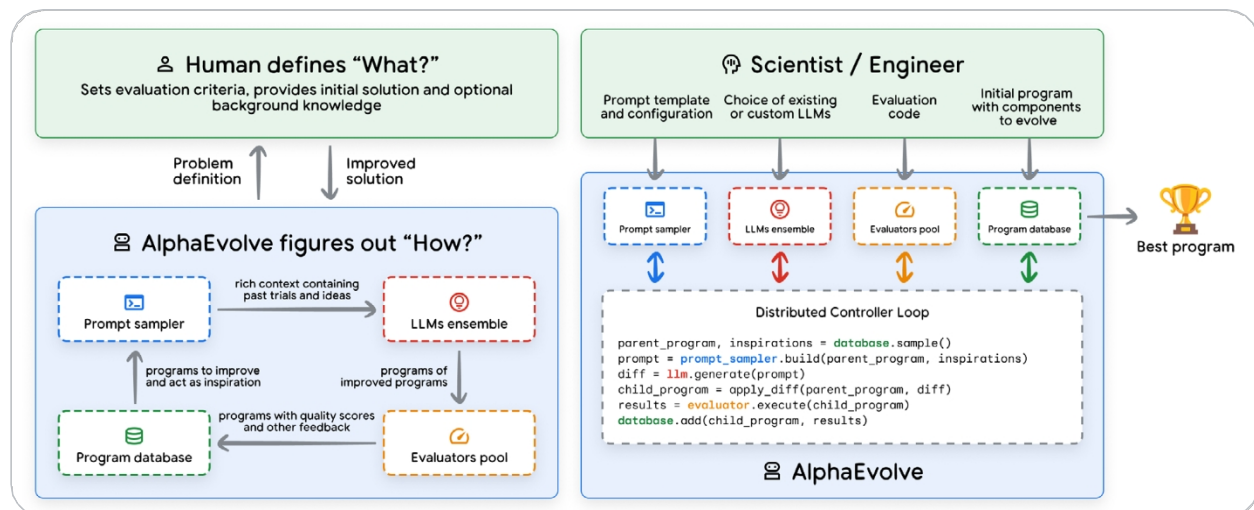
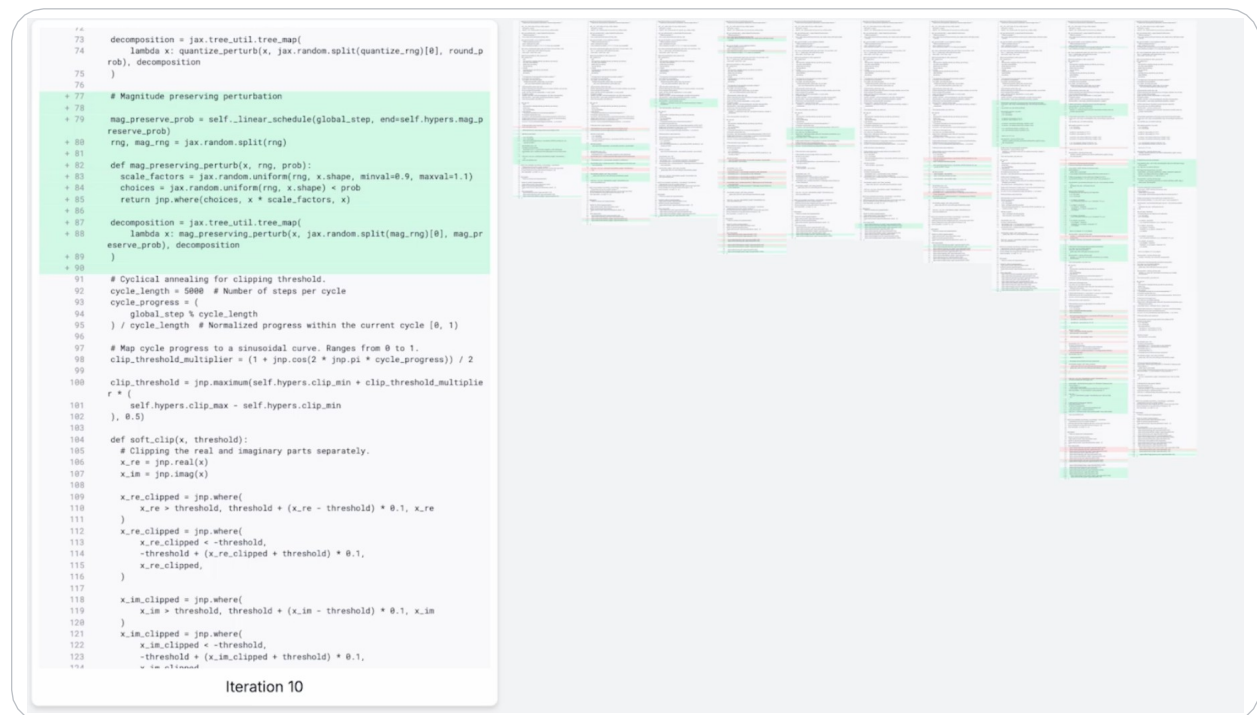


Рисунок 10: Система проектування Alpha Evolve

AlphaEvolve розроблена для глибокого, ітеративного партнерства між людьми та штучним інтелектом. Ця співпраця працює двома основними способами:

- **Прозорі рішення:** ШІ генерує рішення у вигляді коду, зрозумілого для людини. Така прозорість дозволяє користувачам розуміти логіку, отримувати інформацію, довіряти результатам і безпосередньо модифікувати код відповідно до своїх потреб.
- **Експертне керівництво:** людський досвід є необхідним для визначення проблеми. Користувачі керують ШІ, вдосконалюючи метрики оцінки та направляючи дослідження, що запобігає використанню системою ненавмисних прогалин у визначенні проблеми. Цей інтерактивний цикл гарантує, що кінцеві рішення будуть як потужними, так і практичними.

Результатом роботи агента є постійне вдосконалення коду, який продовжує покращувати показники, визначені людиною.



Висновок

Генеративні агенти штучного інтелекту знаменують собою важливу еволюцію, перетворюючи штучний інтелект з пасивного інструменту для створення контенту на активний, автономний інструмент для вирішення проблем. Цей документ надає формальну основу для розуміння та побудови таких систем, виходячи за межі прототипу та створюючи надійну архітектуру виробничого рівня.

Ми розклали агента на три основні компоненти: **модель** міркування («мозок»), **інструменти** для дій («руки») та керуючий **рівень оркестрування** («нервова система»). Саме безперебійна інтеграція цих компонентів, що працюють у безперервному циклі «мислити, діяти, спостерігати», розкриває справжній потенціал агента. Класифікуючи агентні системи — від рівня 1 «Підключений вирішувач проблем» до рівня 3 «Співпрацююча мультиагентна система» — архітектори та керівники продуктів тепер можуть стратегічно визначати свої амбіції відповідно до складності поставленого завдання.

Головне завдання і можливість полягають у новій парадигмі розробника. Ми більше не є просто «каменярами», які визначають явну логіку; ми є «архітекторами» і «директорами», які повинні керувати, обмежувати і налагоджувати автономну суть. Гнучкість, яка робить LM настільки потужними, є також джерелом їх ненадійності. Тому успіх полягає не лише в початковому запиті, а й у суворості інженерії, застосованій до всієї системи: у надійних контрактах на інструменти, стійкій обробці помилок, витонченому управлінні контекстом та всебічній оцінці.

Принципи та архітектурні шаблони, описані тут, слугують основою для проектування. Вони є орієнтирами для навігації в цій новій сфері програмного забезпечення, що дозволяє нам створювати не просто «автоматизацію робочих процесів», а справді співпрацюючих, здатних та адаптивних нових членів наших команд. У міру дозрівання цієї технології, такий дисциплінований архітектурний підхід стане вирішальним фактором у використанні всього потенціалу агентного ШІ.

Кінцеві примітки

1. Джулія Вісінгер, Патрік Марлоу та ін. 2024 «Агенти». Доступно за адресою: <https://www.kaggle.com/whitepaper-agents>.
2. Антоніо Гুলлі, Лаві Нігам та ін. 2025 «Agents Companion». Доступно за адресою: <https://www.kaggle.com/whitepaper-agent-companion>.
3. Шунью Яо, Й. та ін., 2022, «ReAct: Синергія міркування та дії в мовних моделях». Доступно за адресою: <https://arxiv.org/abs/2210.03629>.
4. Wei, J., Wang, X. et al., 2023, «Chain-of-Thought Prompting Elicits Reasoning in Large Language Models» (Ланцюжок думок стимулює міркування у великих мовних моделях). Доступно за адресою: <https://arxiv.org/pdf/2201.11903.pdf>.
5. Shunyu Yao, Y. et al., 2022, «ReAct: Синергія міркування та дії в мовних моделях». Доступно за адресою: <https://arxiv.org/abs/2210.03629>.
6. <https://www.amazon.com/Agentic-Design-Paflerns-Hands-Intelligent/dp/3032014018>
7. Shunyu Yao, et. al., 2024, «τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains» (τ-bench: еталон для взаємодії інструмент-агент-користувач у реальних сферах). Доступно за адресою: <https://arxiv.org/abs/2406.12045>.
8. <https://afiificialanalysis.ai/guide>
9. <https://cloud.google.com/vefiex-ai/generative-ai/docs/model-reference/vefiex-ai-model-optimizer>
10. <https://gemini.google/overview/gemini-live/>
11. <https://cloud.google.com/vision?e=48754805&hl=en>
12. <https://cloud.google.com/speech-to-text?e=48754805&hl=en>
13. <https://medium.com/google-cloud/genaiops-operationalize-generative-ai-a-практичний-посібник-d5bedaa59d78>
14. <https://cloud.google.com/vefiex-ai/generative-ai/docs/agent-engine/code-execution/overview>
15. <https://ai.google.dev/gemini-api/docs/function-calling>
16. <https://github.com/modelcontextprotocol/>
17. <https://ai.google.dev/gemini-api/docs/google-search>

18. <https://google.github.io/adk-docs/>
19. <https://google.github.io/adk-docs/sessions/memory/>
20. <https://cloud.google.com/architecture/choose-design-pattern-agent-ai-system>
21. <https://cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/overview>
22. <https://cloud.google.com/kubernetes-engine/docs/concepts/gke-and-cloud-run>
23. <https://github.com/GoogleCloudPlatform/agent-starter-pack>
24. Сократіс Кафіакіс, 2024, «GenAI у виробництві: MLOps чи GenAIOps?». Доступно за адресою: <https://medium.com/google-cloud/genai-in-production-mlops-or-genaiops-25691c9becd0>.
25. Гуан'я Лю, Суджай Соломон, березень 2025 р. «Спостережуваність агентів ШІ — еволюція стандартів та найкращих практик». Доступно за адресою: <https://opentelemetry.io/blog/2025/ai-agent-observability/>.
26. <https://discuss.google.dev/t/agents-are-not-tools/192812>
27. Дам'єн Массон та ін., 2024, «DirectGPT: інтерфейс прямого маніпулювання для взаємодії з великими мовними моделями». Доступно за адресою: <https://arxiv.org/abs/2310.03691>.
28. MCP UI — це система керування інтерфейсом користувача за допомогою інструментів MCP <https://mcpui.dev/>.
29. AG UI — це протокол управління інтерфейсом користувача за допомогою передачі подій та, за бажанням, спільного стану <https://ag-ui.com/>.
30. A2UI — протокол генерації інтерфейсу користувача за допомогою структурованого виводу та передачі повідомлень A2A <https://github.com/google/A2UI>.
31. <https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash-live-api>.
32. <https://saif.google/focus-on-agents>.
33. <https://simonwillison.net/series/prompt-injection/>.
34. <https://storage.googleapis.com/gweb-research2023-media/pubtools/1018686.pdf>.
35. <https://spiffe.io/>.
36. <https://openreview.net/pdf?id=I9rATNB8Y>.
37. <https://google.github.io/adk-docs/safety/>.

- 38. <https://google.github.io/adk-docs/callbacks/design-patterns-and-best-practices/#guardrails-policy-enforcement>
- 39. ТТТТ
- 40. <https://cloud.google.com/security-command-center/docs/model-armor-overview>
- 41. <https://cloud.google.com/vertex-ai/generative-ai/docs/provisioned-throughput/overview>
- 42. <https://cloud.google.com/run/sla>
- 43. <https://github.com/CharlesQ9/Self-Evolving-Agents>
- 44. Juraj Gottweis, et. al., 2025, «Прискорення наукових проривів за допомогою штучного інтелекту як співдослідника». Доступно за адресою: <https://research.google/blog/accelerating-scientific-breakthroughs-with-an-ai-co-scientist/>.
- 45. Deepak Nathani та ін., 2025, «MLGym: нова структура та еталон для просування агентів досліджень у галузі штучного інтелекту», доступно за адресою: <https://arxiv.org/abs/2502.14499>.