Архітектура персистентної пам'яті для довгострокових автономних місій з ротацією операторів: дворежимне витягування та сигнал корекції

Овчаров В. О.*
LEX AI LLC, Київ, Україна

2026

Анотація

Крос-доменна валідація на трьох незалежних датасетах підтверджує: надлишковість контексту ∼50−60% є системною проблемою автономних агентів, а не артефактом однієї платформи; ротація операторів коштує +136% діалогових раундів (Hedges' g=0,81, 615 репозиторіїв DevGPT); повнота контексту є найсильнішим предиктором потреби в корекціях (r=−0,60, g=1,73, 12 доменів RAGBench). Існуючі системи пам'яті (MemGPT, Mem0, Letta) не підтримують дві ключові вимоги довгострокових місій: безперервність управління при ротації операторів та стійкість при ескалації подій.

Ми пропонуємо чотири внески. Перший — *трирівнева декомпозиція пам'яті* (предметна область / робочий процес / оператор) з дворежимним витягуванням: pull-режим для активних сесій, push-режим для фонового оновлення контексту простоюючих задач. Другий — *сигнал корекції витягування*: корекції оператора, що були б непотрібними за правильного витягування контексту; цей сигнал масштабується з автономністю агента. Третій — *деградований режим* для ескалації подій: при каскадному зростанні загроз підсистема пам'яті переходить з семантичного витягування на тріаж на основі критичності з темпоральною структурою дайджесту — критично для ротації оператора посеред кризи. Четвертий — крос-доменна валідація, що підтверджує універсальність ефектів (DevGPT, RAGBench, SWE-agent).

Архітектура верифікована на платформі з 70+ інструментами та 380M+ записів (304 сесії, медіанна вартість ініціалізації 30 115 токенів, 60% надлишковість). Обговорюється застосовність до систем управління БПЛА та ситуаційних центрів, де ротація операторів є штатною процедурою, а ескалація — штатним режимом бойової роботи.

Ключові слова: персистентна пам'ять, автономні агентні системи, ротація операторів, дворежимне витягування, сигнал корекції, крос-доменна валідація, надлишковість контексту, ескалація подій, деградований режим, БПЛА

1 Вступ: проблема безперервності управління

1.1 Актуальність

Автономні агентні системи — від великих мовних моделей (LLM), що працюють як програмні агенти, до безпілотних літальних апаратів (БПЛА) та наземних роботизованих комплексів —

*Кореспонденція: volodymyr@legal.org.ua

дедалі частіше виконують місії тривалістю від годин до тижнів. У таких системах людина-оператор здійснює нагляд (oversight) за автономним агентом: коригує рішення, змінює пріоритети, підтверджує або відхиляє дії.

При ротації операторів (зміна екіпажу БПЛА, зміна вахти ситуаційного центру, передача управління між змінами) виникає критична проблема: **втрата контексту рішень**. У термінах трирівневої моделі ситуаційної обізнаності [5], ротація руйнує всі три рівні: сприйняття (які елементи обстановки актуальні), розуміння (яке їхнє значення для місії) та проєкцію (як обстановка розвиватиметься). Новий оператор повинен знати:

- які рішення прийнято попередником і чому;
- які корекції внесено та на підставі якої інформації;
- які загрози виявлено, але не усунено;
- який контекст місії залишається актуальним.

Існуючі підходи до пам'яті агентних систем [3, 24, 32] організовані навколо діалогових епізодів і не вирішують проблему зміни операторів. Дослідження нагляду оператора за кількома автономними системами [2] та взаємодії людина–автономія [22] фіксують проблему когнітивного навантаження при передачі управління, але не пропонують архітектури пам'яті для її вирішення. Системи доповненого витягування для програмних агентів [45] витягують фрагменти без провенансу рішень. Моделі з довгим контекстом [34] демонструють деградацію уваги після ∼200K токенів [18].

1.2 Зв'язок зі школою кібернетичного управління

Запропонована робота розвиває принципи школи онтологічно-керованих систем, закладені в [26]: формальна онтологічна структура повинна не лише описувати систему, а *контролювати* її поведінку. Цей принцип послідовно застосовувався на дедалі вищих рівнях обчислювального стеку — від архітектури систем [26] через обробку природномовних текстів [27, 28] до контролю виходу великих мовних моделей [29, 30] та еволюційних кібернетичних систем [31].

У даній роботі онтологічний контроль застосовується до *підсистеми пам'яті* автономного агента: формальна структура визначає, який контекст подається оператору при ініціалізації сесії, як оновлюється контекст простоюючих місій, та як вимірюється якість витягування через сигнал корекції.

1.3 Постановка задачі

Мета: розробити архітектуру підсистеми пам'яті для автономних агентних систем, що забезпечує:

1. збереження контексту рішень між сесіями операторів;
2. автоматичне оновлення контексту для простоюючих задач;
3. генерацію сигналу корекції для вдосконалення системи навчання.

Об'єкт дослідження: процеси людино-машинної взаємодії в автономних агентних системах з довгостроковими місіями.

Предмет дослідження: методи та моделі забезпечення безперервності контексту управління при ротації операторів.

2 Аналіз існуючих підходів

2.1 Системи пам'яті для діалогових агентів

MemGPT [24] реалізує віртуальне управління контекстом за аналогією з ієрархічною пам'яттю операційних систем. Generative agents [32] використовують епізодичну пам'ять для симульованих агентів. Mem0 [3] забезпечує довгострокову пам'ять для LLM через зовнішнє сховище.

A-MEM [41] розвиває парадигму епізодичної пам'яті у напрямку структурованого знання: кожне спостереження перетворюється на атомарну нотатку за принципом Zettelkasten з явними зв'язками між нотатками, утворюючи граф знань поверх потоку спостережень. Це наближає одиницю пам'яті до структурованого запису, проте зв'язки залишаються асоціативними, а не каузальними — нотатка не фіксує відхилені альтернативи та не прив'язується до конституційного принципу.

Letta [25], комерціалізований наступник MemGPT, вводить два механізми, безпосередньо релевантні цій роботі. По-перше, *контекстні репозиторії* (context repositories) [15] — персистентні версіоновані сховища структурованого контексту, до яких агент звертається між сесіями; вони замінюють плоску архівну модель MemGPT типізованими колекціями з підтримкою запитів. По-друге, *контекстна конституція* (context constitution) [14] — декларативна специфікація того, до якого контексту агент має доступ і з яким пріоритетом, аналогічна політиці витягування, вираженій як конфігурація, а не як код. Ці механізми наближають пам'ять агента до витягування за провенансом рішень: контекстні репозиторії надають субстрат зберігання, а контекстна конституція — рудиментарну політику витягування. Проте одиницею витягування в Letta залишається контекстний блок (текстовий фрагмент з метаданими), а не оперативне рішення з провенансом, альтернативами та конституційним обґрунтуванням.

Reflexion [35] підтримує епізодичну пам'ять вербальних рефлексій — агент зберігає текстові описи минулих помилок для покращення майбутніх дій. Цей механізм паралельний до нашого рівня оператора, де зберігаються структуровані абстракції корекцій. Когнітивна архітектура CoALA [38] декомпозує агента на модулі пам'яті, дії та прийняття рішень — наша трирівнева декомпозиція є конкретною реалізацією цієї рамки для довгострокових місій. Систематичний огляд механізмів пам'яті для LLM-агентів подано у Zhang et al. [46].

Обмеження: усі розглянуті системи — MemGPT, Generative Agents, A-MEM, Mem0, Letta — використовують як одиницю витягування діалоговий обмін, спостереження персонажа або контекстний блок, а не оперативне рішення з повним провенансом.

2.2 Витягування для програмних агентів

Парадигма доповненої генерації (RAG) [16] забезпечує фундамент витягування для мовних моделей. Corrective RAG [42] вводить оцінювач якості витягування з сигналами корекції — найближча до нашого retrieval-correction signal робота, хоча обмежена одиничними запитами без довгострокового контексту місії. RepoCoder [45] та подібні системи витягують фрагменти коду за семантичною подібністю. CodeRAG-Bench [39] забезпечує систематичний бенчмарк для витягування з доповненою генерацією коду, охоплюючи документацію, довідники API та приклади коду як джерела витягування. SWE-bench [12] та SWE-agent [43] оцінюють агентів на реальних задачах з GitHub (issues), демонструючи, що витягування з урахуванням структури репозиторію та контексту задачі підвищує частку успішних розв'язків.

Обмеження: усі зазначені системи витягують *текст коду та ідентифікатори* — без провенансу рішень: чому цей код написано саме так, які альтернативи розглядались, які обмеження враховано.

2.3 Записи архітектурних рішень

Практика формального документування архітектурних рішень має тривалу історію в програмній інженерії. Формат Architecture Decision Record (ADR), запропонований Найґардом [21], визначає легковагий шаблон у форматі Markdown — назва, статус, контекст, рішення, наслідки — що зберігається безпосередньо разом із кодовою базою. Стандарт ISO/IEC/IEEE 42010 [11] формалізує опис архітектури на організаційному рівні, надаючи нормативну рамку для документування архітектурних точок зору та відповідностей. Y-формат Цьорнера [48] структурує рішення у вигляді шаблонного висловлювання: «В контексті [ситуація], зіткнувшись із [проблемою], ми вирішили [варіант], щоб досягти [якість], приймаючи [компроміс]». Zimmermann et al. [47] розширює практику управління ADR до міжпроєктного керівництва, додаючи моделювання простору проблем та систематичне відстеження залежностей між рішеннями.

ADR фіксують *провенанс рішень* — саме той контент, що відсутній у витягуванні code-RAG. Проте механізм витягування ADR залишається файловим: рішення знаходять переглядом директорії за назвою та датою, а не семантичним запитом. Відсутні індекс вбудовувань (embedding index), гібридне витягування, перерангування за релевантністю до поточної задачі. Крім того, ведення ADR потребує ручного авторства: у високошвидкісному робочому процесі з продуктивністю 14,7 PR/день кураторське навантаження на підтримку повного реєстру ADR стає заборонно високим.

За військовою аналогією, ADR подібні до журналу рішень командира — цінний артефакт, але пошук у ньому здійснюється за датою та підрозділом, а не за семантичною релевантністю до поточної оперативної ситуації.

2.4 Моделі з довгим контекстом

Gemini 1.5 Pro [34] підтримує контекст до 1M+ токенів. Проте дослідження [10, 17, 18] демонструють деградацію уваги після ∼200K токенів: модель «губить» інформацію в середині довгого контексту. Плоске завантаження всього контексту місії не масштабується з тривалістю місії.

2.5 Системи управління знаннями в ситуаційних центрах

Онтологічно-керовані системи підтримки рішень [26] забезпечують формальне представлення знань для ситуаційної обізнаності. Palagin et al. [30] демонструють ефективність інтеграції нейромережевих та онтолінгвістичних парадигм. В Інституті кібернетики НАН України розвиваються суміжні напрямки: теорія конфліктно-керованих процесів [4] для формалізації задач переслідування та ухилення (застосовна до автономного управління БПЛА), оптимізація маршрутів команд БПЛА з динамічними депо [9], гібридні нейромережі для інтелектуального управління літальними апаратами [44]. Проте жодна з існуючих систем не поєднує онтологічну структуру з дворежимним витягуванням для забезпечення безперервності управління при ротації операторів.

Синтез. Жодна з п'яти ліній робіт — діалогова пам'ять, витягування для програмних агентів, реєстри архітектурних рішень, моделі з довгим контекстом, онтологічно-керовані системи — не розглядає провенанс рішення як першокласну одиницю пам'яті із семантичним витягуванням. Жодна не забезпечує примітиву повільного циклу оновлення (*slow-loop refresh*), що підтримує актуальність пам'яті про неактивні задачі на горизонтах у кілька тижнів — інтервалі, характерному для ротації операторів у штабних процесах. Архітектура, описана в цій роботі, займає перетин цих ліній: вона запозичує структуру провенансу з ADR, семантичне витягування з Code-

RAG, персистентність із систем діалогової пам'яті та селективне завантаження контексту з RAG — додаючи дворежимне витягування (*dual-mode retrieval*) як нову архітектурну примітиву, спеціально спроектовану для забезпечення безперервності управління при зміні операторів.

3 Формалізація проблеми

3.1 Три характеристики довгострокових місій

Довгострокова автономна місія характеризується трьома вимірюваними властивостями, верифікованими на пілотному датасеті [23]:

Горизонт. Пов'язані рішення охоплюють дні — тижні. У верифікаційному розгортанні: 105 днів активної роботи з медіанним інтервалом між комітами менше 2 годин, але архітектурними потоками тривалістю 4–6 тижнів.

Композиційність. Рішення A тижня 2 обмежує рішення B тижня 6. Вибір схеми бази даних визначив контракти 29 інструментів, побудованих впродовж наступних п'яти тижнів.

Персистентність. Стан місії є спільним між сесіями операторів. Агент не зберігає історію діалогу — він повертається до зміненого стану місії кожної нової сесії.

3.2 Проблема ініціалізації сесії

Кожна нова сесія оператора починається із завантаження контексту. Вимірювання на 304 сесіях: медіана вартості ініціалізації — 30 115 вхідних токенів, з яких ∼59% — надлишковий контекст. Медіана *коефіцієнта надлишковості* (файли, зчитані при ініціалізації, але жодного разу не використані під час сесії) — 60%.

При ротації операторів проблема загострюється: новий оператор не має *жодного* контексту рішень попередника — лише стан місії без обґрунтувань.

3.3 Формальна постановка

Нехай T — кількість токенів ініціалізації, K — кількість зчитувань файлів/записів, W — коефіцієнт надлишковості.

- **Pull-режим** зменшує K та T при ініціалізації: замість плоского завантаження — цільове витягування за описом задачі.
- **Push-режим** підтримує актуальність контексту: для простоюючих задач/секторів фонове оновлення забезпечує, що при відновленні роботи контекст вже актуальний.
- **Метрика якості:** частка корекцій оператора, спричинених недостатнім контекстом (retrieval-correction signal).

Цільові показники: $T' \leq 10\,000$ токенів, $W' \leq 20\%$.

4 Архітектура трирівневої пам'яті

Підсистема пам'яті декомпозується на три рівні з різною семантикою витягування та субстратом зберігання. Рисунок 1 показує загальну архітектуру з обома режимами витягування.

4.1 Предметний рівень (Domain Layer)

Предметний рівень містить оперативні дані домену.

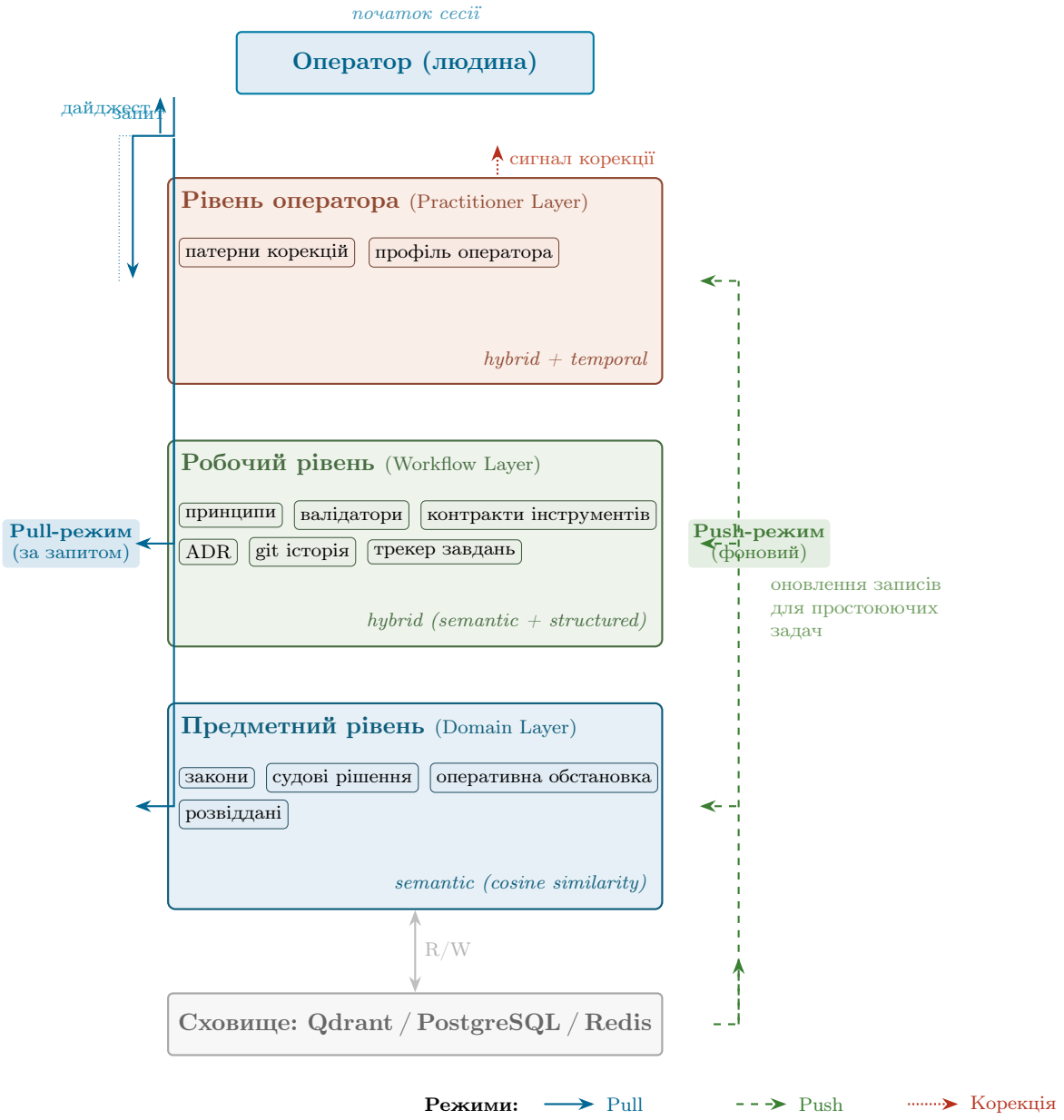

Рис. 1: Трирівнева архітектура персистентної пам'яті з дворежимним витягуванням. Pull-режим (суцільна лінія) ініціюється запитом оператора на початку сесії та паралельно звертається до всіх трьох рівнів. Push-режим (пунктирна лінія) фоново оновлює контекст простоюючих задач. Сигнал корекції (точкова лінія) повертається від рівня оператора при виявленні пропущеного релевантного контексту.

Табл. 1: Зіставлення предметного рівня: юридичний AI та управління БПЛА.

Юридичний AI (верифікація)	**Управління БПЛА (застосування)**
Закони, судові рішення, регламенти	Оперативна обстановка, картографія, розвіддані
Законодавчі зміни, судова практика	Зміни обстановки, нові загрози
70+ MCP-інструментів пошуку	Сенсори, системи спостереження
Семантичне витягування з метаданими (юрисдикція, дата, тип)	Семантичне витягування з метаданими (сектор, тип загрози, час)

Витягування: семантична подібність (cosine similarity) з опціональною фільтрацією за метаданими. Архітектурна зміна порівняно з прямими викликами інструментів: маршрутизація через єдину підсистему пам'яті забезпечує уніфіковане логування всіх запитів, вимірювання частоти промахів та єдиний формат відповіді.

4.2 Робочий рівень (Workflow Layer)

Робочий рівень містить контекст рішень — провенанс, відсутній як у вихідному коді, так і в предметних даних.

Табл. 2: Шість джерел робочого рівня та їхні військові аналоги.

#	**Оригінал**	**Військовий аналог**
1	Реєстр принципів (порушення, що були скориговані)	Журнал порушень ROE та виправлених помилок
2	Визначення валідаторів	Правила перевірки дотримання ROE/SOP
3	Контракти інструментів (70+)	Контракти систем озброєння та сенсорів
4	Записи архітектурних рішень (ADR)	Рішення командира з обґрунтуванням
5	Історія git з семантичним збагаченням	Хронологія дій місії
6	Стан трекера завдань	Стан завдань/цілей місії

Витягування: гібридне — семантична подібність + структуровані фільтри (компонент, тип місії, часовий діапазон).

4.3 Рівень оператора (Practitioner Layer)

Рівень оператора містить патерни рішень конкретного оператора, витягнуті з *pipeline* корекцій [23].

Ключова властивість: цей рівень ніколи не зберігає сирий контент — тільки структуровані абстракції: клас корекції, семантична категорія, афектований компонент, результат (прийнято/відхилено/модифіковано) та одноречennеве резюме.

Застосування при ротації: новий оператор отримує профіль попередника — типові корекції, зони підвищеної уваги, рішення місії з обґрунтуваннями — у компактному дайджесті при ініціалізації сесії.

Витягування: гібридне з фільтрацією за часом та результатом.

Фаза 0: міст промпт–коміт. Перед розгортанням повноцінного рівня оператора встановлюється мінімальний колектор даних. Хук `UserPromptSubmit` у Claude Code створює orphan-коміт у bare-репозиторії git для кожного промпту оператора. Кожний запис містить: текст промпту, мітку часу, ідентифікатор сесії, репозиторій, гілку та режим дозволів.

Протягом 4–6 тижнів пасивного збору формується природномовний корпус, що забезпечує вхідні дані для трьох задач:

- розмежувальні мітки (disambiguation labels) — класифікація типу наміру оператора;
- розподіли частот тем (topic-frequency distributions) — виявлення домінантних робочих контекстів;
- патерни перемикання між проєктами (cross-project switching patterns) — моделювання переходів оператора між репозиторіями та завданнями.

Цей корпус є сировинним матеріалом для embedding-pipeline рівня оператора: з нього витягуються структуровані абстракції, описані вище, без збереження сирого контенту промптів у довготривалій пам'яті.

Продуктивність: затримка захоплення становить 15 мс на один промпт — непомітна в межах 5-секундного таймауту хуків Claude Code. Фаза 0 розгорнута з 9 травня 2026 р.

Попередні результати аналізу корпусу подано у Розділі 7.3.

Військова аналогія. Фаза 0 аналогічна пасивному збору радіоелектронної розвідки (SIGINT): накопичення даних без втручання в оперативну діяльність, формування базового розуміння патернів поведінки оператора до розгортання активних систем аналізу та адаптації.

5 Дворежимне витягування

5.1 Pull-режим (активні сесії)

При вході нового оператора система виконує запит до всіх трьох рівнів паралельно. Опис поточної задачі/сектора вбудовується у вектор (embedding), зіставляється з проіндексованим сховищем пам'яті, і k найрелевантніших записів збираються у компактний дайджест. Оператор починає роботу з фокусованим контекстом ($K' \ll K$, $T' \ll T$) і може витягувати додаткові записи на вимогу.

5.2 Push-режим (фонове оновлення)

Для простоюючих задач та секторів місії: планований фоновий процес відстежує активність і оновлює записи пам'яті пропорційно до швидкості релевантних змін. При відновленні роботи в секторі pull-запит повертає *актуальний* контекст без потреби повторного збору інформації.

Частота оновлення визначається через бал терміновості $u \in [0, 1]$ (Рівняння 1), що є зваженою сумою трьох насичуваних сигналів активності:

$$u(\text{task}) = w_c \frac{n_c}{n_c + \kappa_c} + w_m \frac{n_m}{n_m + \kappa_m} + w_t \frac{\Delta t}{\Delta t + \kappa_t}, \qquad w_c + w_m + w_t = 1, \tag{1}$$

де n_c — кількість комітів, що торкаються задачі від останнього оновлення; n_m — кількість коментарів до задачі за той самий період; Δt — час від останнього pull-запиту (в годинах); $\kappa_c, \kappa_m, \kappa_t > 0$ — константи напівнасичення. Функція $x/(x+\kappa)$ — стандартна насичувана нелінійність: поодинокі події суттєво впливають на бал, подальше зростання має спадну граничну віддачу.

Інтервал між оновленнями обчислюється лінійною інтерполяцією:

$$\tau(\text{task}) = T_{\max} - (T_{\max} - T_{\min}) \cdot u(\text{task}), \tag{2}$$

де $T_{\min} = 24$ год та $T_{\max} = 168$ год (7 діб). При $u = 0$ інтервал дорівнює $T_{\max}$ (мінімальна частота для задач із позначкою `LONG-TERM`); при $u \to 1$ інтервал скорочується до $T_{\min}$.

Кожне оновлення породжує три артефакти:

- **Виконавче резюме змін** — генерується мовною моделлю на основі дифів релевантних комітів та коментарів до задачі; містить стислий виклад усіх суттєвих подій від попереднього оновлення.
- **Дельта лінеажу інструментів** — перелік нових, модифікованих та виведених з експлуатації інструментів із дифами параметричних схем, що дозволяє оператору при поверненні одразу бачити зміни в доступному арсеналі засобів.
- **Відкриті питання** — нерозв'язані альтернативи та точки прийняття рішень, позначені сумаризатором як такі, що потребують втручання оператора при відновленні роботи над задачею.

За військовою аналогією, цей механізм відповідає автоматизованій підготовці брифінгу передачі зміни: система формує зведення навіть тоді, коли жоден оператор не перебуває на бойовому чергуванні, — щоб наступна зміна при заступанні отримала актуальну обстановку без затримки на повторний збір інформації.

Критично для ротації: між змінами операторів push-режим оновлює контекст місії новими подіями, рішеннями інших секторів, змінами обстановки — так що наступна зміна починає з актуального стану.

5.3 Чому необхідні обидва режими

- **Pull без push:** після тривалої паузи контекст застарілий — оператор отримує рішення тижневої давності.
- **Push без pull:** надлишковий контекст при кожній сесії — оператор завалений інформацією з усіх секторів.
- **Комбінація:** push підтримує актуальність, pull фокусує витягування на поточну задачу.

Рисунок 2 ілюструє часову діаграму обох режимів витягування.

5.4 Сигнал корекції витягування

Визначаємо *корекцію витягування* як корекцію оператора, яка була б непотрібною, якби підсистема пам'яті вчасно подала релевантний контекст.

Приклад: оператор БПЛА коригує маршрут, бо не знав про зміну обстановки в секторі. Якби push-режим оновив контекст місії, корекція не знадобилась би.

Цей сигнал виконує подвійну функцію:

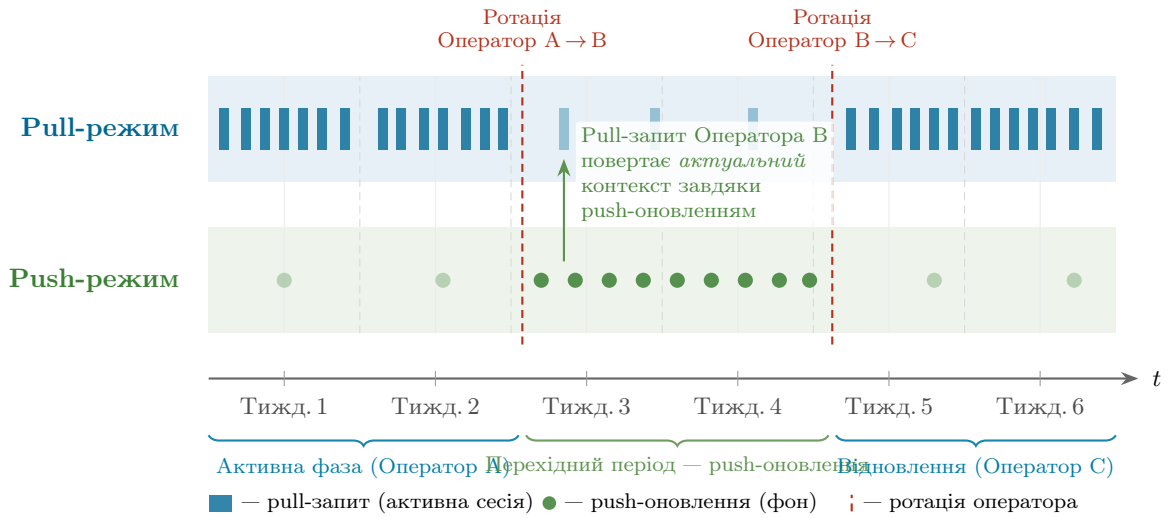

Рис. 2: Дворежимне витягування протягом шеститижневого циклу місії. У pull-режимі (верхня доріжка) кожна сесія оператора ініціює контекстний запит; у push-режимі (нижня доріжка) система періодично оновлює кешований контекст завдань. Під час ротації оператора (штрихові лінії) push-оновлення гарантують, що перший pull-запит нового оператора повертає актуальний стан місії без ручної реініціалізації.

1. **Метрика якості** підсистеми пам'яті: частка корекцій, спричинених витягуванням, вимірює ефективність push/pull-механізмів.
2. **Додатковий сигнал навчання:** кожна корекція витягування маркує конкретний промах системи — що мало бути подано і коли — створюючи датасет для вдосконалення моделі витягування.

Сигнал масштабується з автономністю агента: чим автономніший агент, тим більше рішень він приймає самостійно, і тим більше корекцій виникає через недостатній контекст.

6 Деталі реалізації

6.1 Інструмент запиту до пам'яті

Інтерфейсом витягування контексту для агента слугує єдиний MCP-інструмент `workflow_memory_query`, що забезпечує уніфікований доступ до всіх трьох рівнів пам'яті. Вибір єдиного інструменту замість окремих точок доступу до кожного рівня є принциповим проєктним рішенням: агент формулює один запит природною мовою, а підсистема пам'яті самостійно маршрутизує його до релевантних рівнів та агрегує відповіді.

Параметри інструменту:

- `task_description` (рядок, обов'язковий) — природномовний опис поточної задачі або оперативної ситуації.
- `scope` (перелічення: `all` | `domain` | `workflow` | `practitioner`) — рівні пам'яті для запиту. За замовчуванням: `all`.
- `token_budget` (ціле число, за замовчуванням 8 000) — максимальна кількість токенів у зібраному дайджесті.
- `filters` (об'єкт, опціональний) — структуровані фільтри: компонент, рівень критичності, часовий діапазон.

Інструмент повертає *структурований прозовий дайджест* із такими складовими:

1. заголовки секцій відповідно до кожного рівня пам'яті, з якого витягнуто записи;
2. оцінка релевантності (relevance score) для кожного запису;
3. посилання на джерело (провенанс) — конкретний коміт, рішення командира, запис трекера або документ домену.

Формат дайджесту спроєктовано для безпосередньої ін'єкції у контекстне вікно агента без додаткової постобробки. Токенний бюджет забезпечує контрольовану вартість ініціалізації: підсистема ранжує записи за релевантністю і відсікає нижні записи, коли сумарний обсяг перевищує бюджет.

6.2 Оркестратор довгострокових задач

Push-режим реалізований як окремий Docker-контейнер з власним cron-розкладом, ізольований від основного серверу обробки запитів. Архітектурна ізоляція забезпечує дві властивості: (1) фонові процеси оновлення не конкурують за ресурси з активними сесіями операторів; (2) збій оркестратора не впливає на доступність pull-режиму.

Цикл роботи оркестратора:

1. **Виявлення задач.** Оркестратор надсилає запит до трекера задач (Plane API) і отримує список задач із позначкою `LONG-TERM`.
2. **Обчислення пріоритету оновлення.** Для кожної задачі обчислюється пріоритет оновлення за формулою (2).
3. **Генерація виходів.** Для кожної задачі, що підлягає оновленню, оркестратор генерує три структуровані виходи за допомогою LLM-сумаризатора: резюме для командира, дельту лінеажу інструментів та відкриті питання.
4. **Індексація та зберігання.** Згенеровані виходи вбудовуються у вектори та зберігаються у колекції Qdrant робочого рівня пам'яті з метаданою позначкою `source: push-refresh`.

7 Експериментальна верифікація

7.1 Платформа верифікації

Архітектура верифікована на платформі юридичного штучного інтелекту LEX AI [23]: 70+ MCP-інструментів, 380M+ записів у *pipeline* даних, 1 547 об'єднаних pull-запитів за 105 днів роботи одного практика з LLM-агентом. Ця платформа є стрес-тестом підсистеми пам'яті: один оператор з максимальною пропускною здатністю генерує щільний потік рішень, що моделює навантаження автономної місії з обмеженим людським контролем. Для перевірки універсальності ефектів проведено крос-доменну валідацію на трьох незалежних датасетах: DevGPT [40] (615 репозиторіїв, 140 розробників), RAGBench [6] (95 378 прикладів, 12 доменів) та траєкторії SWE-agent (6 636 траєкторій автономних агентів).

7.2 Метрики

Шість кількісних метрик операціоналізують критерії успішності архітектури.

Сьома метрика — *частота корекцій витягування* (Розділ 5.4) — має якісний характер на Фазі 1. Очікувана траєкторія — спадна крива протягом перших 6–8 тижнів розгортання.

Метрика	Метод вимірювання	Базове	Ціль
Вартість ініціалізації	Вхідні токени при першому API-виклику; N=304	30 115	≤10K
Коефіцієнт надлишковості	Файли зчитані, але не редаговані / всього; N=180	60%	≤20%
Точність витягування принципів	Тестовий набір з edit-traces	н/д	≥80%
Латентність push-оновлення	Час від коміту до дайджесту	н/д	≤24 год
Вартість ротації	Дельта надлишковості: перемикання − продовження; N=88 пар	+7,9 в.п.	≤3 в.п.
Вартість ротації (крос-доменна)	Дельта діалогових раундів (DevGPT); N=976 пар	+3,3 раунди	≤1 раунд

Табл. 3: Метрики оцінювання з базовими значеннями та цільовими показниками.

7.3 Попередні результати

Вартість ініціалізації (Експеримент 1, N=304 сесії). Медіана вхідних токенів першого виклику 30 115 (середнє 29 693; σ=7 594; P10/P90: 18 540/41 774). З цього обсягу ∼17 600 токенів (59%) витрачаються на створення кешу для `CLAUDE.md` та системного контексту — завантажуються безумовно для кожної сесії. Кількість зчитувань файлів в середньому 23,1 (медіана 14, P90: 61), проте 72% сесій мають нуль викликів Read у фазі ініціалізації.

Зростання CLAUDE.md (Експеримент 2, 25 комітів за 85 днів). Файл `CLAUDE.md` зріс із 4 099 до 24 148 символів (152 до 474 рядків). Лінійна регресія на 19 денних знімках (коміти одного дня колапсовано до останнього): 197,6 симв./день при $R^2 = 0{,}89$.

Для перевірки наявності структурних переломів застосовано алгоритм PELT [13] до сигналу швидкості зростання (симв./день між послідовними денними знімками). При BIC-штрафі, а також при AIC-еквіваленті та ліберальному штрафі (50% BIC) PELT не виявив жодного статистично значущого перелому. Тест Краскела–Уолліса для порівняння швидкості зростання між трьома фазами (ранньою, середньою, пізньою третинами) підтверджує: H=0,72, p=0,70 — фази не відрізняються значуще. Це формально підтверджує лінійну модель зростання: файл інструкцій росте з постійною швидкістю без фазових переходів, що відповідає безперервному накопиченню знань.

Коефіцієнт надлишковості (Експеримент 3, N=180 сесій). Медіана коефіцієнта надлишковості — 60% (середнє 56,7%; σ=26,3%). 66% сесій витрачають більше половини зчитувань марно. Зчитування вихідного коду: 78% надлишковість; файли пам'яті: 66,5%. Ця метрика є консервативною проксі: файл може бути зчитаний для контексту без подальшого редагування.

Фаза 0 (міст промпт–коміт) розгорнута з 9 травня 2026 р.; корпус містить 925 промптів (762 операторських, 163 системних) з 29 сесій. Тематичне моделювання корпусу операторських промптів за допомогою BERTopic [7] виявило 39 тем з чіткою інтерпретацією. Класифікація намірів: верифікація/перевірка — 20%, дослідження/обговорення — 16%, виправлення помилок — 16%, створення функцій — 6%. Патерни перемикання між проєктами: 179 переходів між репозиторіями, 48% сесій (14 з 29) охоплюють кілька репозиторіїв — що підтверджує потребу у крос-проєктному контексті при ініціалізації сесії.

Аблація ротації оператора (Експеримент 4, N=88 пар). Для кількісної оцінки «вартості ротації» проведено ретроспективний аналіз на 89 сесіях проєкту SecondLayer. З кожної сесії витягнуто множини зчитаних та редагованих файлів і зіставлено з компонентами системи (backend/сервіс, frontend/сторінка, deployment тощо). Хронологічно послідовні пари сесій (N=88) класифіковано на дві умови:

- **Продовження** (n=51): компоненти, редаговані в сесії A, перетинаються з компонентами сесії B (моделює того самого оператора).
- **Перемикання** (n=37): множини компонентів не перетинаються (моделює ротацію оператора на інший сектор).

Результати наведено в Таблиці 4.

Умова	**Опис**	n	**Медіана** W	**Середнє** W
Продовження	спільні компоненти з попередньою сесією	51	50,0%	46,9%
Перемикання	розрізнені компоненти (симульована ротація)	37	61,1%	54,8%
Дельта ротації			**+11,1 в.п.**	**+7,9 в.п.**

Табл. 4: Коефіцієнт надлишковості контексту (W) при продовженні та при симульованій ротації оператора. Mann–Whitney U=1130, p=0,058 (одностороння); Hedges' g=0,28 (малий–середній ефект) [8]; пермутаційний тест (50 000 перестановок): p=0,094 (одностороння); CLES=0,60; бутстрап 95% ДІ для дельти: [−3,9%; +19,6%].

Перемикання на інший компонент (симульована ротація) збільшує надлишковість контексту на +7,9 в.п. (відсоткових пунктів) порівняно з продовженням роботи на тому самому компоненті. Розмір ефекту (Hedges' g=0,28 [8]) відповідає малому–середньому рівню; пермутаційний тест (p=0,094) підтверджує напрямок ефекту, хоча він не досягає конвенційного порогу α=0,05 — що очікувано при N=88 парах одного практика. Цей результат фіксує початкове спостереження вартості ротації на однооператорних даних; крос-доменна валідація на DevGPT (Експеримент 5) підтверджує ефект на значно більшому масштабі з великим розміром ефекту (g=0,81, p<0,001).

Крос-доменна валідація на DevGPT (Експеримент 5, N=976 пар). Для перевірки відтворюваності ефекту ротації поза межами одного проєкту проведено аналогічний аналіз на датасеті DevGPT [40]: 1 576 взаємодій розробників з ChatGPT, пов'язаних із GitHub-артефактами (PR, коміти, issues) у 615 репозиторіях. Для 140 розробників з ≥2 взаємодіями побудовано 976 хронологічних пар; 901 пара класифікована як «продовження» (той самий репозиторій) і 75 — як «перемикання» (інший репозиторій).

Вартість ротації проявляється не у збільшенні довжини промптів, а у збільшенні кількості діалогових раундів: медіана при продовженні — 1 промпт, при перемиканні — 3 промпти (середнє 2,5 проти 5,9; Hedges' g=0,81, великий ефект [8]; CLES=0,70, p<0,001, Mann–Whitney U). Розробник, що не має контексту попереднього проєкту, компенсує його відсутність ітеративним інформаційним пошуком — кожний додатковий промпт є функціональним аналогом сигналу корекції витягування (Розділ 5.4).

Цей результат підтверджує архітектурну гіпотезу на незалежному датасеті та в іншому домені: push-режим, що попередньо завантажує актуальний контекст, зменшив би кількість ітеративних запитів до рівня продовження.

Валідація сигналу корекції на RAGBench (Експеримент 6, N=95 378). Для валідації сигналу корекції витягування як метрики якості підсистеми пам'яті проведено аналіз на

RAGBench [6]: 95 378 прикладів доповненої генерації у 12 доменах (від медицини та права до фінансів та техніки) з анотаціями TRACe (Utilization, Relevance, Adherence, Completeness).

Проксі корекції: приклад потребує корекції, якщо adherence=False (відповідь містить непідтверджені твердження) або completeness<0,5 (пропущено понад половину доступного контенту). 25,4% прикладів (24 210) класифіковано як такі, що потребують корекції.

Найсильніший предиктор — *повнота контексту* (completeness): точково-бісеріальна кореляція $r=-0{,}60$ ($p<0{,}001$); при порівнянні груп Hedges' $g=1{,}73$ (великий ефект; CLES=0,84). Приклади без потреби в корекції мають середню повноту 0,87, приклади з потребою — 0,48. Цей результат підтверджує центральну тезу сигналу корекції: недостатній контекст є кількісно вимірюваною причиною корекцій оператора, а зниження частки таких корекцій — валідна метрика якості підсистеми пам'яті.

Надлишковість контексту в SWE-agent (Експеримент 7, $N=6\,636$). Для перевірки універсальності проблеми надлишковості контексту (Експеримент 3) проведено аналогічний аналіз на 6 636 траєкторіях автономного агента SWE-agent з датасету nebius/SWE-agent-trajectories. Кожна траєкторія містить повну послідовність дій агента (зчитування файлів, редагування, bash-команди) при спробі розв'язати реальну задачу з GitHub.

Медіана коефіцієнта надлишковості — 50% (середнє 56,4%; $\sigma=22{,}6\%$), що є статистично порівнянним з результатом LEX AI (60% медіана, 56,7% середнє). Критично: вища надлишковість передбачає невдачу агента — для невдалих траєкторій середній *waste* 58,0% проти 49,4% для успішних (Hedges' $g=0{,}39$, CLES=0,65, $p<0{,}001$). Точково-бісеріальна кореляція між *waste* та невдачею: $r=-0{,}15$ ($p<0{,}001$).

Цей результат має подвійне значення: (1) проблема надлишковості контексту є універсальною для агентних систем, а не артефактом LEX AI; (2) надлишковість є предиктором невдачі агента, що обґрунтовує оптимізацію витягування як шлях до підвищення автономності.

Стан розгортання (станом на 14 травня 2026). Усі сім фаз (0, 1.0–1.5) розгорнуті у продуктивному середовищі. Шар пам'яті містить 187 записів: 170 доменних принципів (140 з `CLAUDE.md`, 23 з об'єднаних PR, 7 з проєктної документації), 4 робочих шаблони (push-refresh підсумки через Bedrock Claude Haiku) та 13 підсумків сесій оператора. Усі вектори згенеровані Amazon Titan Embed Text v2 (1024 виміри) через AWS Bedrock. Push-режим (Фаза 1.5): 4 активні задачі у watchlist, базовий знімок 22 інструментів у реєстрі. Деградований режим із тріажем на основі критичності (Розділ 8.6) є архітектурним внеском цієї роботи поза межами семи фаз ядра і на момент подання не реалізований.

7.4 Загрози валідності

Один проєкт, один практик (частково адресовано). Основна верифікація (Експерименти 1–4) проведена в контексті одного проєкту та одного практика. Крос-доменна валідація (Експерименти 5–7) на трьох незалежних датасетах із сумарним обсягом ∼104K записів підтверджує, що ключові ефекти (надлишковість, вартість ротації, сигнал корекції) не є артефактом платформи. Залишкова загроза: валідація оперує проксі-метриками, а не прямим розгортанням підсистеми пам'яті.

Змішування з дозріванням кодової бази. Контроль: A/B-оцінювання шляхом увімкнення та вимкнення підсистеми пам'яті на тому самому розподілі задач.

Якість реєстру принципів. На Фазі 1 реєстр курується людиною; точність витягування обмежена якістю курації. Адресується на Фазі 2 напівавтоматичним витягуванням принципів з edit-traces.

7.5 Застосовність до управління БПЛА та ситуаційних центрів

Метрики платформи юридичного AI зіставляються з метриками бойових систем:

Табл. 5: Зіставлення метрик верифікації з бойовими метриками.

Метрика LEX AI	**Бойовий аналог**
Токени ініціалізації (T)	Час підготовки оператора до управління
Коефіцієнт надлишковості (W)	Когнітивне навантаження при зміні вахти
Retrieval-correction signal	Непотрібні корекції = потенційно запобігнуті інциденти
Час передачі контексту	Час передачі управління — метрика бойової готовності

8 Обговорення

8.1 Від пам'яті програмного агента до пам'яті бойової системи

Запропонована архітектура є доменно-агностичною: трирівнева декомпозиція та дворежимне витягування працюють однаково для юридичного AI та управління БПЛА. Предметний рівень заповнюється даними домену (законодавство або оперативна обстановка), робочий рівень — рішеннями місії, рівень оператора — профілем конкретної людини.

Ключова відмінність — вимоги до латентності: мілісекунди для систем реального часу (БПЛА) проти секунд для юридичного AI. Це впливає на архітектуру push-режиму: для БПЛА оновлення мають відбуватися в реальному часі, а не за розкладом.

Ця перспектива підтверджується сучасними дослідженнями у сфері військового C2. National Defense University [20] формулюють концепцію «агентної бази даних» — переходу від пасивних сховищ до «активних двигунів міркування», що самостійно аналізують оперативні дані; наша архітектура push-режиму є реалізацією цього принципу на рівні підсистеми пам'яті. Special Competitive Studies Project [37] аргументують, що перевага у сучасних конфліктах визначається здатністю «перевершити противника у C2» через швидше розгортання програмного забезпечення та людино-машинне командування — контекст, у якому ротація операторів є штатною процедурою, а не винятком. Perry et al. [33] прямо зазначають, що системи інтерактивного машинного навчання для C2 повинні «зберігати знання та досвід після ротації персоналу» — саме задачу, яку вирішує запропонована архітектура.

Серед розгорнутих систем особливого значення набуває українська система ситуаційної обізнаності «Дельта» [1], що оперує через 8 ситуаційних центрів з інтеграцією безпілотних систем і модулем штучного інтелекту Avengers (12 000 ідентифікацій на тиждень). Стандарт C2SIM [36] (SISO-STD-019-2020) формалізує цикл ініціалізації–призначення–звітування для обміну даними між C2 та симуляційними системами — саме той цикл, що переривається при

ротації оператора. Інформаційна модель NATO JC3IEDM [19] (STANAG 5525, 271 сутність, 1 460 атрибутів) визначає канонічну структуру «стану світу» оператора C2.

8.2 Еволюційна кібернетика та адаптація пам'яті

Palagin et al. [31] запропонував рамку для аналізу систем, де цілі, обмеження та структури самі еволюціонують — на відміну від класичної теорії управління з фіксованою цільовою функцією.

Підсистема пам'яті місії є саме такою еволюційною системою: контент пам'яті, пріоритети витягування та навіть структура рівнів адаптуються до зміни обстановки. Push-режим є механізмом еволюційного оновлення: він не просто підтримує актуальність, а перебудовує контекст у відповідь на якісні зміни ситуації.

8.3 Зв'язок з онтологічно-керованими системами

Три рівні пам'яті природно зіставляються з концепціями онтологічно-керованих систем [26]:

- **Предметний рівень** = онтологія домену (формальне представлення знань про предметну область).
- **Робочий рівень** = онтологія процесу управління (рішення, їх обґрунтування, залежності).
- **Рівень оператора** = модель когнітивного профілю (патерни рішень, зони уваги).

Дворежимне витягування розширює цю аналогію: pull-режим відповідає активному запиту до онтології, push-режим — автоматичному оновленню онтології при зміні фактів про світ. Palagin et al. [30] демонструють, що інтеграція нейромережевих та онтолінгвістичних підходів дає кращі результати, ніж кожна парадигма окремо — наша гібридна схема витягування (семантичне + структуроване) є реалізацією цього принципу.

8.4 Пам'ять як субстрат навчання з підкріпленням

Та сама інфраструктура, яка дозволяє одному практику виконувати сотні сесій протягом кількох місяців, *генерує* сигнал переваг, придатний для навчання з підкріпленням на основі зворотного зв'язку від людини (RLHF). Пам'ять і дані переваг є двома проєкціями одного і того ж довгострокового робочого процесу: кожна корекція після витягування є одночасно *збоєм підсистеми пам'яті* та *сигналом вирівнювання*.

Ця двоїстість не є випадковою: вона становить архітектурну тезу. Підсистема пам'яті не лише *споживає* дані вирівнювання — вона їх *продукує*. У бойовому контексті кожна корекція оператора під час управління БПЛА — зміна маршруту, перепризначення цілі — є одночасно записом пам'яті місії та зразком для навчання системи [23].

8.5 Пам'ять як поверхня масштабованого нагляду

Підсистема пам'яті виконує подвійну функцію: обслуговує агента під час виконання місії та генерує сигнал нагляду під час узгодження. Із зростанням автономності агента відношення спостережуваних результатів до внутрішніх рішень агента падає. Нагляд на рівні результатів стає вузькосмуговим каналом.

Підсистема пам'яті змінює цю динаміку. Кожне витягування є *спостережуваною подією* з відомим контекстним вікном, відомим результатом витягування та подальшим слідом редагування. Щільність сигналу на одиницю активності є вищою, ніж при RLHF на рівні результатів: кожна сесія генерує множину пар «витягнуто X, скориговано Y».

8.6 Поведінка пам'яті при ескалації подій

Попередні підсекції розглядають підсистему пам'яті в режимі стаціонарного навантаження. Окремого аналізу потребує поведінка при *ескалації* — каскадному зростанні кількості одночасних подій, коли інформаційне навантаження на оператора та агента зростає експоненціально.

Каскадні сценарії. У бойовому контексті ескалація виникає при одночасному обстрілі кількох секторів, радіоелектронному придушенні каналів зв'язку та втраті контакту з частиною рою БПЛА. У програмному контексті — при каскадних виробничих інцидентах у кількох сервісах одночасно.

Деградований режим. Коли частота подій перевищує пропускну здатність обробки, підсистема пам'яті має переключитися з семантичного витягування на *тріаж на основі критичності*. Формально, функція ранжування параметризується коефіцієнтом суміші $\alpha \in [0, 1]$:

$$\text{score}(e) = \alpha \cdot \text{criticality}(e) + (1 - \alpha) \cdot \text{sim}(q, e), \tag{3}$$

де параметр α визначається сигмоїдною функцією частоти подій λ (подій/год):

$$\alpha(\lambda) = \frac{1}{1 + e^{-\gamma\,(\lambda - \lambda_0)}}, \tag{4}$$

λ_0 — поріг переходу в деградований режим, $\gamma > 0$ — крутизна переходу. При $\lambda \ll \lambda_0$ маємо $\alpha \approx 0$ (штатний режим, семантичне витягування); при $\lambda \gg \lambda_0$ маємо $\alpha \approx 1$ (деградований режим, тріаж за критичністю). Рисунок 3 ілюструє цю залежність.

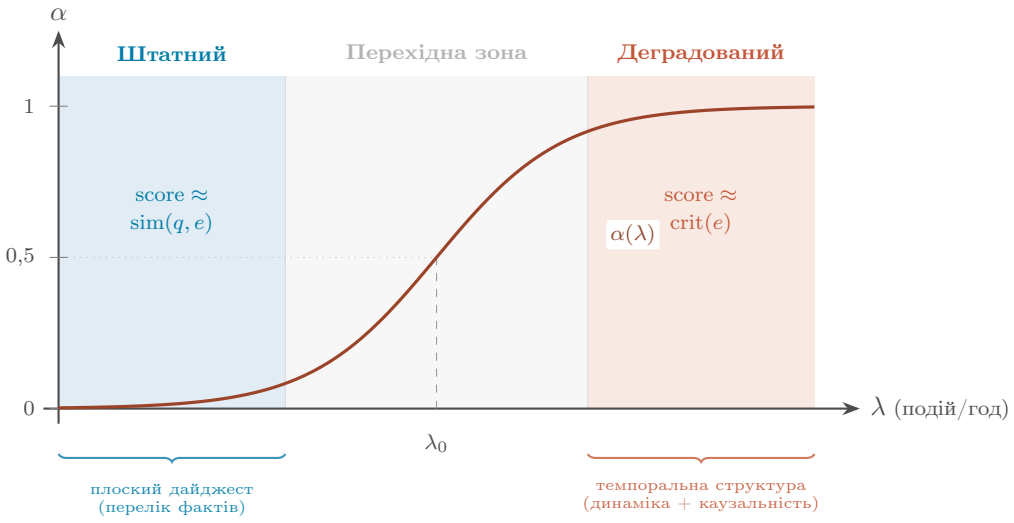

Рис. 3: Перехід підсистеми пам'яті зі штатного в деградований режим як функція частоти подій λ. При $\lambda \ll \lambda_0$ ранжування визначається семантичною подібністю (штатний режим); при $\lambda \gg \lambda_0$ домінує тріаж за критичністю (деградований режим). Сигмоїдний перехід (4) забезпечує плавну деградацію без дискретного перемикання. Формат дайджесту також адаптується: від плоского переліку фактів до темпоральної структури з причинно-наслідковими зв'язками.

Ротація під час кризи. Найгірший сценарій для підсистеми пам'яті — ротація оператора посеред ескалації. Дайджест для нового оператора повинен передати не лише статичний стан, а й *динаміку*: що ескалює, що каскадує, які зв'язки між подіями виявлені, які рішення прийняті попереднім оператором. Це вимагає розширення формату дайджесту від плоского переліку фактів до темпоральної структури з причинно-наслідковими зв'язками.

Повна реалізація деградованого режиму виходить за межі поточної роботи (див. обмеження (iv) у Розділі 9), проте формальний апарат — функція ранжування (Рівняння 3) із сигмоїдним переходом (Рівняння 4) та черга пріоритетів у push-режимі — є частиною архітектурного проєкту.

8.7 Узагальнення на команди

Архітектура обслуговує одного практика. Багатооператорне розгортання породжує три виклики: *по-перше*, реєстр принципів стає багатозаписувачем з необхідністю вирішення конфліктів; *по-друге*, рівень оператора потребує індивідуальних представлень; *по-третє*, записи пам'яті можуть містити контекст, чутливий до рівня доступу.

Трирівнева декомпозиція спроєктована для підтримки цих розширень: предметний рівень є спільним, робочий підтримує обмежені представлення через фільтрацію метаданих, а рівень оператора є за природою індивідуальним.

8.8 Чим це не є

Архітектура *не* є заміною моделей з довгим контекстним вікном — вона є витягувальним субстратом, що зменшує обсяг контексту, який повинна обробити модель [18, 34]. Це *не* є граф знань (не використовує RDF-трійки і не надає SPARQL-інтерфейс). Це *не* є корпоративна система управління знаннями (Enterprise KMS). Це витягувальний субстрат, оптимізований для одного операційного режиму — довгострокової агентної композиції малою кількістю практиків.

Це *не* є повноцінна система C4ISR — це підсистема пам'яті, що інтегрується в існуючу інфраструктуру управління як компонент безперервності інформаційного контексту.

9 Обмеження

(i) **Верифікація на одній платформі (частково адресовано).** Архітектура розгорнута на юридичному AI, не на БПЛА або ситуаційному центрі. Крос-доменна валідація (Експерименти 5–7) підтвердила ключові ефекти на трьох незалежних датасетах: надлишковість контексту на SWE-agent (50%, порівняно з 60% LEX AI), вартість ротації на DevGPT (g=0,81, 615 репозиторіїв), сигнал корекції на RAGBench (r=−0,60, 12 доменів). Проте ці валідації оперують проксі-метриками, а не прямим розгортанням підсистеми пам'яті в іншому домені.

(ii) **Одиничний оператор (частково адресовано).** Основні вимірювання (Експерименти 1–4) отримані від одного практика. Крос-доменна валідація на DevGPT охоплює 140 розробників і підтверджує ефект ротації в багатооператорному контексті. Проте повноцінна багатооператорна ротація з розгортанням підсистеми пам'яті потребує окремого дослідження.

(iii) **Відсутність A/B-оцінювання.** Підсистема пам'яті розгорнута, проте усі вимірювання є базовими (до оптимізації). Демонстрація того, що архітектура *зменшує* надлишковість та вартість ініціалізації, потребує контрольованого A/B-експерименту з увімкненням та вимкненням підсистеми на тому самому розподілі задач.

(iv) **Деградований режим.** Тріаж на основі критичності (Розділ 8.6) формалізований (Рівняння 3–4), проте не реалізований і не верифікований емпірично.

(v) **Латентність.** Вимірювання латентності витягування отримані для не-реального часу. Для БПЛА потрібна оптимізація під мілісекундні вимоги.

10 Висновки

Запропоновано архітектуру підсистеми персистентної пам'яті для автономних агентних систем, що виконують довгострокові місії з ротацією операторів.

Основні результати:

1. **Трирівнева декомпозиція** (предметна область / робочий процес / оператор) забезпечує розділення контексту за семантикою витягування та терміном життя.
2. **Дворежимне витягування** (pull + push) вирішує проблему безперервності контексту: pull фокусує ініціалізацію на поточну задачу, push підтримує актуальність для простоюючих задач та ротації операторів.
3. **Сигнал корекції витягування** надає метрику якості підсистеми пам'яті, яка масштабується з автономністю агента та слугує додатковим джерелом навчального сигналу.
4. **Деградований режим** для ескалації подій: при каскадному зростанні кількості одночасних загроз підсистема пам'яті переходить з семантичного витягування на тріаж на основі критичності, а формат дайджесту розширюється до темпоральної структури з динамікою подій.
5. **Крос-доменна валідація** на трьох незалежних датасетах підтверджує, що ключові ефекти не є артефактом однієї платформи:
   - надлишковість контексту ∼50% відтворюється в автономних агентах SWE-agent, а вища надлишковість передбачає невдачу (g=0,39);
   - вартість ротації проявляється як +136% діалогових раундів при перемиканні між репозиторіями (g=0,81, 615 репозиторіїв);
   - повнота контексту є найсильнішим предиктором потреби в корекціях (r=−0,60, g=1,73, 12 доменів).

Базові вимірювання на 304 сесіях підтверджують наявність проблеми: 60% надлишковість контексту при ініціалізації; обсяг інструкцій зростає лінійно з віком проєкту (підтверджено PELT-аналізом: жодних структурних переломів). Тематичне моделювання 762 операторських промптів (BERTopic, 39 тем) виявило, що 48% сесій охоплюють кілька репозиторіїв — кількісне підтвердження потреби у крос-проєктному контексті.

Архітектура є доменно-агностичною і застосовна до систем управління БПЛА та ситуаційних центрів, де ротація операторів є штатною процедурою, а безперервність контексту рішень — критичною вимогою бойової готовності. Окремо обговорено поведінку при ескалації подій: деградований режим із тріажем на основі критичності та проблему ротації посеред кризи.

Подальші дослідження: (1) розгортання підсистеми пам'яті в багатооператорному середовищі з вимірюванням A/B-ефекту; (2) реалізація push-режиму в реальному часі для БПЛА; (3) інтеграція з онтологічно-керованими системами підтримки рішень [26] для формалізації робочого рівня пам'яті; (4) реалізація деградованого режиму та емпірична верифікація тріажу на основі критичності.

Література

[1] Kateryna Bondar. Does Ukraine already have functional CJADC2 technology? Technical report, Center for Strategic and International Studies, 12 2024. URL https://www.csis.org/analysis/does-ukraine-already-have-functional-cjadc2-technology.

[2] Jessie Y. C. Chen, Michael J. Barnes, and Michelle Harper-Sciarini. Supervisory control of multiple robots: Human-performance issues and user-interface design. *IEEE Transactions*

on Systems, Man, and Cybernetics—Part C, 41(4):435–454, 2011. doi: 10.1109/TSMCC.2010.2056682.

[3] Prateek Chhikara, Deshraj Khare, Saquib Tariq, and Mudit Bansal. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

[4] Arkadii A. Chikrii. *Conflict-Controlled Processes*, volume 405 of *Mathematics and Its Applications*. Springer, 1997. doi: 10.1007/978-94-017-1135-7.

[5] Mica R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37(1):32–64, 1995. doi: 10.1518/001872095779049543.

[6] Robert Friel, Minjie Tao, Dina Noursi, and Abhijit Jain. RAGBench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*, 2024.

[7] Maarten Grootendorst. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[8] Larry V. Hedges. Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics*, 6(2):107–128, 1981. doi: 10.2307/1164588.

[9] Volodymyr P. Horbulin, Leonid F. Hulianytskyi, and Ivan V. Sergienko. Optimization of UAV team routes in the presence of alternative and dynamic depots. *Cybernetics and Systems Analysis*, 56(2):195–203, 2020. doi: 10.1007/s10559-020-00235-8.

[10] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Joshua Ainslie, et al. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[11] ISO/IEC/IEEE. ISO/IEC/IEEE 42010:2011 — systems and software engineering — architecture description, 2011.

[12] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.06770.

[13] Rebecca Killick, Paul Fearnhead, and Idris A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012. doi: 10.1080/01621459.2012.737745.

[14] Letta Team. Context constitution: Declarative retrieval policies for agent memory. https://docs.letta.com, 2026. Blog post / technical report.

[15] Letta Team. Context repositories for persistent agent memory. https://docs.letta.com, 2026. Blog post / technical report.

[16] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[17] Tianle Li, Junnan Ge, Jingbo Cheng, Siyuan Chen, Luyu Gao, et al. Long-context LLMs struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*, 2024.

[18] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

[19] MIP Community. Multilateral interoperability programme information model (MIM) 5.3. Technical Report NATO STANAG 5525, NATO, 2025. URL https://www.mimworld.org/portal.

[20] National Defense University. The agentic database and military command: A perspective on autonomous C2 systems. Technical report, Institute for National Strategic Studies, 11 2025. URL https://inss.ndu.edu/Research-and-Commentary/View-Publications/Article/4333776/.

[21] Michael Nygard. Documenting architecture decisions. https://cognitect.com/blog/2011/11/15/documenting-architecture-decisions, 2011. Blog post.

[22] Thomas O'Neill, Nathan J. McNeese, Amy Barron, and Beau Schelble. Human-autonomy teaming: A review and analysis of the empirical literature. *Human Factors*, 64(5):904–938, 2022. doi: 10.1177/0018720820960865.

[23] Vladimir Ovcharov. Edit-trace oversight: Scalable alignment signal from agentic workflows. 2026. Manuscript in preparation.

[24] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

[25] Charles Packer, Sarah Wooders, and Kevin Lin. Letta: The open-source framework for building stateful LLM agents (formerly MemGPT). https://github.com/letta-ai/letta, 2025. Software repository.

[26] Alexander V. Palagin. Architecture of ontology-controlled computer systems. *Cybernetics and Systems Analysis*, 42(2):254–264, 2006. doi: 10.1007/s10559-006-0061-z.

[27] Alexander V. Palagin, Serhiy L. Kryvyi, and Mykola G. Petrenko. On the automation of the process of extracting knowledge from natural language texts. *Natural and Artificial Intelligence International Book Series*, 2012.

[28] Oleksandr Palagin, Vitalii Velychko, Kyrylo Malakhov, and Borys Shchurov. Distributional semantic modeling: A revised technique to train term/word vector space models applying the ontology-related approach. *arXiv preprint arXiv:2003.03350*, 2020.

[29] Oleksandr Palagin, Vladislav Kaverinskiy, Anna Litvin, and Kyrylo Malakhov. OntoChatGPT information system: Ontology-driven structured prompts for ChatGPT meta-learning. *International Journal of Computing*, 22(2):170–183, 2023.

[30] Oleksandr Palagin, Vladislav Kaverinskiy, and Kyrylo Malakhov. Fundamentals of the integrated use of neural network and ontolinguistic paradigms: A comprehensive approach. *Cybernetics and Systems Analysis*, 60:111–123, 2024. doi: 10.1007/s10559-024-00652-z.

[31] Oleksandr Palagin, Dmytro Symonov, and Mykhailo Chervynskyi. Modelling evolutionary cybernetics: Ontology, invariants and design principles. *Information Technologies and Systems*, 6(6):3–29, 2025. doi: 10.15407/intechsys.2025.06.003.

[32] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

[33] Anna Perry, Anna Echeverria, Philip Odonkor, and Hong Shen. Scalable interactive machine learning for future command and control. *arXiv preprint arXiv:2402.06501*, 2024.

[34] Machel Reid, Nikolay Savinov, Denis Teber, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[35] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[36] Simulation Interoperability Standards Organization. C2SIM: C2-simulation interoperation. Technical Report SISO-STD-019-2020, SISO, 2020. URL https://openc2sim.github.io.

[37] Special Competitive Studies Project. Reimagining military C2 in the age of AI. Technical report, SCSP, 12 2024. URL https://www.scsp.ai/wp-content/uploads/2024/12/DPS-Reimagining-Military-C2-in-the-Age-of-AI.pdf.

[38] Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024.

[39] Zora Zhiruo Wang, Akari Shi, Shuyan Yang, Ravi Anantha, Ryan A. Rossi, Neel Sundaresan Park, et al. CodeRAG-Bench: Can retrieval augment code generation? *arXiv preprint arXiv:2406.14497*, 2024.

[40] Tao Xiao, Christoph Treude, Hideaki Hata, and Kenichi Matsumoto. DevGPT: Studying developer-ChatGPT conversations. In *Proceedings of the 21st International Conference on Mining Software Repositories (MSR)*, pages 64–68, 2024. doi: 10.1145/3643991.3648400.

[41] Wujiang Xu, Zujie Liang, Jingquan Mei, Jiancheng Gao, Yongxin Qi, Feng Yang, Rui Yan, and Wei Liu. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2501.13059*, 2025.

[42] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024.

[43] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024.

[44] Mykhailo Z. Zgurovsky, Viktor M. Sineglazov, and Olena I. Chumachenko. *Artificial Intelligence Systems Based on Hybrid Neural Networks: Theory and Applications*, volume 904 of *Studies in Computational Intelligence*. Springer, 2021. doi: 10.1007/978-3-030-48453-8.

[45] Fengji Zhang, Bei Chen, Yue Zhang, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. RepoCoder: Repository-level code completion through iterative retrieval and generation. *arXiv preprint arXiv:2303.12570*, 2023.

[46] Zeyu Zhang, Xiaohe Zhang, Yuanpei Wang, Shengjie Yan, Zhaokai Wang, Ge Yu, et al. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*, 2024.

[47] Olaf Zimmermann, Lukas Wegmann, Heiko Koziolek, and Thomas Goldschmidt. Architectural decision guidance across projects: Problem space modeling, decision backlog management and cloud computing knowledge. In *Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 85–94. IEEE, 2015. doi: 10.1109/WICSA.2015.23.

[48] Stefan Zörner. *Softwarearchitekturen dokumentieren und kommunizieren: Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten.* Carl Hanser Verlag, München, 2 edition, 2015. ISBN 978-3-446-44361-9.