



DOI 10.28925/2663-4023.2026.35.1425

УДК 004.5:004.89:004.41

**Бондарчук Андрій Петрович**

доктор технічних наук, професор, завідувач кафедри комп'ютерних наук,  
Київський столичний університет ім. Бориса Грінченка, Київ, Україна

ORCID: 0000-0001-5124-5102

*a.bondarchuk@kubg.edu.ua*

**Криволапов Гліб Ярославович**

студент, Факультет інформаційних технологій та математики

Київський столичний університет ім. Бориса Грінченка, Київ, Україна

ORCID: 0000-0002-0853-5881

*hykryvolapov.fitm25m@kubg.edu.ua*

**Негоденко Олена Василівна**

кандидат технічних наук, доцент, доцент кафедри комп'ютерних наук

Київський столичний університет імені Бориса Грінченка, Київ, Україна

ORCID: 0000-0001-6645-1566

*o.nehodenko@kubg.edu.ua*

**Зуб Людмила Миколаївна**

студентка, Навчально-науковий інститут телекомунікацій,

Державний університет інформаційно-комунікаційних технологій, Київ, Україна

ORCID: 0009-0005-6697-2683

*zubludmila123@gmail.com*

**МЕТОДИКА СТРУКТУРНОГО СИНТЕЗУ АДАПТИВНИХ ІНТЕРФЕЙСІВ НА ОСНОВІ  
ФОРМАЛІЗАЦІЇ ПРОСТОРУ НАМІРІВ КОРИСТУВАЧА**

**Анотація.** У статті викладено методику структурного синтезу адаптивних інтерфейсів, яка переводить концепцію штучного інтелекту як інтерфейсного шару-оркестратора з рівня концептуальної моделі на рівень відтворюваної інженерної процедури. Раніше запропонована авторами модель гібридної взаємодії описує склад системи та розподіл функцій між її рівнями, проте не відповідає на питання, як для конкретної платформи побудувати перевірюване відображення природномовного наміру на виклики функціональних модулів. Саме цей розрив і закриває запропонована методика. У вступі показано, що за відсутності формального апарату кожна реалізація інтерфейсного шару лишається одиничним рішенням, непридатним до перенесення на іншу предметну область, а недетермінованість мовної моделі переноситься безпосередньо на поведінку інтерфейсу. В аналізі останніх досліджень і публікацій зіставлено дві схеми організації агентного циклу – ітеративну та засновану на однократному плануванні, розглянуто роботи з виклику зовнішніх програмних інтерфейсів мовними моделями, засоби керованого структурованого виведення за схемою та емпіричні оцінки надійності планування, а також окреслено відмінність запропонованого рішення від генеративного інтерфейсу й від агентів, що імітують дії користувача. У розділі результатів дослідження задачу синтезу формалізовано як побудову відображення пари «намір – контекст графічного інтерфейсу» у пару «план звернень до функціональних модулів – мутація стану компонентів», подано шестиетапну схему методики, модель простору намірів із пороговою оцінкою впевненості інтерпретації, п'ять правил декомпозиції складеного наміру та модель мутації стану графічного інтерфейсу як скінченної множини операцій над сховищем стану клієнта. Описано механізми контролюваності, серед яких підтвердження критичних дій винесено в контракт функціонального модуля, а не в текст промпту. Аналітична оцінка часу опрацювання наміру показує, що домінуювальним доданком лишається звернення до мовної моделі, тому методика доводить кількість таких звернень на критичному шляху до одиниці. У висновках обґрунтовано, що мовна модель у запропонованому рішенні позбавлена виконавчих повноважень і виступає лише генератором структурованого плану, тоді як виконання, зміна стану інтерфейсу, підтвердження та відкат лишаються детермінованими.



**Ключові слова:** структурний синтез інтерфейсу; інформаційна система, простір намірів; план дій; мутація стану GUI; детермінований інтерфейсний шар.

## ВСТУП

Уявлення про мовну модель як про шар керування платформою, а не як про співрозмовника, сьогодні вже описане на рівні складу системи. Відомо, з яких рівнів вона складається і яку функцію виконує кожен із них. Невідоме інше. Немає відповіді на питання, як для конкретної інформаційної системи побудувати відображення наміру на дії так, щоб воно було відтворюваним, перевірюваним і не залежало від того, яку саме модель обрано на момент розроблення. Стаття пропонує методику, що закриває цей розрив. Вона складається з послідовності етапів, які ведуть від наявного програмного інтерфейсу системи до працездатного шару-оркестратора, і з формальних моделей наміру, плану дій та зміни стану графічних компонентів.

**Постановка проблеми.** Опис архітектури не є методом її побудови. Концептуальна модель відповідає на питання, з чого складається система та як розподілено функції між її частинами. Методика відповідає на інше питання – що саме треба зробити, щоб таку систему отримати. Поки друга відповідь відсутня, кожна реалізація інтерфейсного шару лишається одиничним інженерним рішенням. На практиці воно зводиться до набору евристик у тексті запиту до моделі, розгалужень у коді та ручного зіставлення типових формулювань із конкретними викликами. Перенесення такого рішення на іншу предметну область означає переписування більшої його частини.

Другий бік проблеми стосується передбачуваності. Мовна модель за своєю природою недетермінована, тоді як інтерфейс має поводитися однаково за однакових умов, а помилкове тлумачення запиту не повинне спричинити незворотних наслідків. Прямо покласти на модель виконання дій над станом системи тому неприйнятно. Потрібен такий розподіл повноважень, за якого недетермінованим лишається тільки вибір плану, а виконання, зміна стану інтерфейсу, підтвердження критичних операцій і відкат забезпечуються засобами, поведінка яких визначена проєктом системи. Формалізація цього розподілу і становить наукову задачу статті.

**Аналіз останніх досліджень і публікацій.** Ідея перекласти на систему вибір способу досягнення результату не нова. Дж. Нільсен називає її першою за шістьдесят років зміною парадигми інтерфейсу, за якої користувач формулює бажаний результат, а не послідовність кроків [1]. Б. Шнайдерман застерігає від спрощеного прочитання цієї тези. Зростання рівня автоматизації має супроводжуватися збереженням людського контролю, інакше виграш у швидкості обертається втратою розуміння того, що відбувається в системі [2]. Обидві роботи задають рамку, у межах якої має розв'язуватися задача. Конкретного інженерного рішення вони не пропонують.

Способи організації агентного циклу поділяються на дві родини. Перша будується на чергуванні міркування, дії та спостереження результату, коли кожен наступний крок визначається після отримання попереднього [3]. Друга передбачає, що модель спершу формує план цілком, і лише потім план виконується [4]. Різниця між ними не косметична. В ітеративній схемі кількість звернень до моделі дорівнює кількості кроків, у схемі з однократним плануванням вона зводиться до одиниці, а виконання переходить під контроль програмного коду. Зіставлення обох схем наведено на рисунку 1.

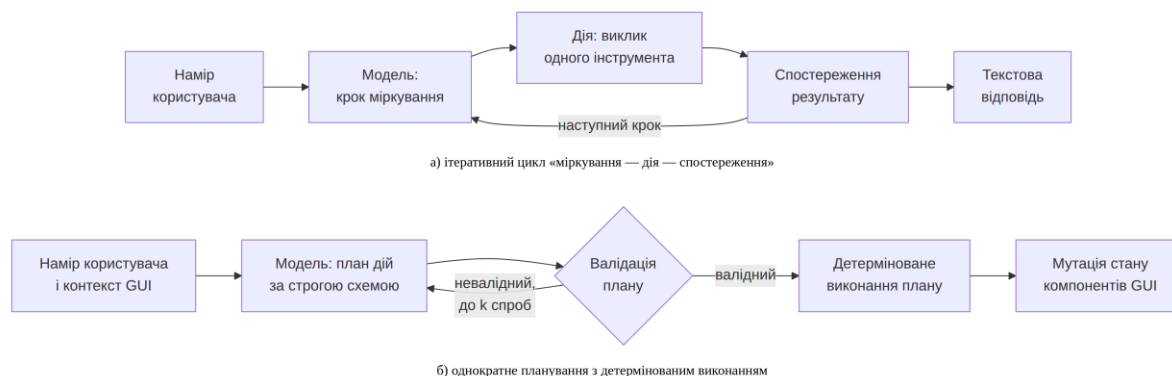


Рис. 1. Схеми організації агентного циклу

Вибір між цими схемами має не лише технічні наслідки. Емпіричне дослідження довіри користувачів до агентів-помічників розглядає саме той сценарій, у якому агент спершу показує план, а вже



потім його виконує [5]. Показ плану до виконання змінює характер взаємодії. Користувач дістає можливість втрутитися до того, як щось відбулося, а не після.

Окремий напрям стосується навчання моделей викликати зовнішні програмні інтерфейси. Роботи щодо самостійного опанування інструментів [6], зіставлення моделі з великими наборами прикладних інтерфейсів [7] та масштабування такого зіставлення до тисяч реальних сервісів [8] показали, що добір виклику є розв'язною задачею навчання. Предметом оптимізації в них виступає здатність моделі дібрати правильний виклик. Що відбувається з інтерфейсом після цього виклику, ці роботи не розглядають, хоча для користувача важливий саме результат на екрані.

Керованість формату відповіді забезпечують засоби структурованого виведення. Декодування, обмежене граматику або схемою, робить синтаксично невалідну відповідь неможливою за побудовою [9]. Семантичної правильності воно не гарантує. Більше того, у дослідженні впливу обмежень формату зафіксовано, що надмірно жорсткі схеми виведення здатні погіршувати якість міркувань моделі [10]. Звідси випливає проєктне рішення, використане далі. Схема плану лишається мінімальною, а перевірка доменних умов виноситься за її межі.

Надійність планування як такого оцінена стриманіше. Систематичне вимірювання на спеціалізованому наборі задач показує, що автономне побудування коректних планів лишається слабким місцем мовних моделей [11]. Схильність до породження правдоподібних, але хибних тверджень описана в оглядовій літературі як системна властивість, а не як дефект окремої моделі [12]. Обидві лінії досліджень ведуть до одного практичного висновку. Простір дій, доступних моделі, має бути обмежений заздалегідь і перевірятися ззовні.

Паралельно розвивається напрям генеративного інтерфейсу, у якому елементи керування синтезуються під конкретний запит [13]. Показовим прикладом є динамічний синтез віджетів для редагування візуалізацій [14]. Такі рішення знімають проблему текстового виводу, проте породжують іншу. Згенеровані елементи не належать до сталого набору компонентів платформи, тому стійкі навички роботи з нею не формуються. Запропонований у статті підхід відрізняється тим, що не створює нових елементів, а переводить наявні в інший стан.

Ще одну групу становлять агенти, які діють через графічну оболонку, розпізнаючи екран та імітуючи дії користувача [15]. Глибина зміни стану тут висока, однак доступ до контракту серверного ядра відсутній, через що кожен крок потребує окремого розпізнавання й окремого звернення до моделі. Узагальнені рекомендації щодо проєктування людино-машинної взаємодії зі штучним інтелектом [16] додають до цієї картини важливу деталь. Можливість скасувати дію та зрозуміти її причину має бути передбачена проєктом системи, а не додана згодом.

Обмеження мовних моделей, істотні саме для задачі синтезу інтерфейсу, разом із проєктними засобами їх компенсації зведено в таблиці 1.

Таблиця 1

Обмеження мовних моделей та проєктні засоби їх компенсації

| Обмеження                       | Прояв у задачі синтезу інтерфейсу                                                      | Проєктний засіб компенсації                                                                           |
|---------------------------------|----------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Породження неіснуючих сутностей | Виклик операції, якої немає в системі, або поле, відсутнє у схемі параметрів           | Замкнений каталог операцій; перевірка імені та аргументів до виконання                                |
| Недетермінованість виведення    | Той самий намір за однакового контексту дає різні плани                                | Строга схема плану; повторний запит із текстом помилки; після k невдалих спроб – перехід до уточнення |
| Семантично хибні аргументи      | Від'ємне значення бюджету, неіснуючий ідентифікатор, взаємно суперечливі умови відбору | Другий валідатор, незалежний від схеми, що перевіряє доменні правила                                  |
| Обмеження обсягу контексту      | Повний стан застосунку не вміщується в одне повідомлення                               | Передавання релевантного зрізу стану та кількох останніх намірів сесії                                |
| Латентність звернення           | Час до першої видимої зміни визначається одним зверненням до моделі                    | Одне звернення на критичному шляху; генерація пояснення поза ним                                      |
| Вартість звернення              | Оплаті підлягає кожен опрацьований намір                                               | Незмінний префікс повідомлення та кешування каталогу операцій                                         |

Наукова новизна дослідження полягає у формалізації синтезу адаптивного інтерфейсу як побудови перевірюваного відображення пари «намір – контекст» у пару «план – мутація стану», а також у тому, що мовній моделі в цьому відображенні відведено роль генератора структурованого плану без будь-яких



виконавчих повноважень. Виконання плану, зміна стану компонентів, підтвердження критичних операцій і відкат лишаються детермінованими. Це відрізняє підхід і від вбудованих помічників, які діють точково, і від агентів, що працюють через імітацію дій користувача.

**Мета статті.** Метою статті є розроблення методики структурного синтезу адаптивних інтерфейсів, яка для заданої інформаційної системи визначає послідовність етапів побудови інтерфейсного шару-оркестратора, а також формальні моделі наміру, плану дій і мутації стану графічних компонентів, достатні для відтворюваної програмної реалізації.

## РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Задачу структурного синтезу адаптивного інтерфейсу сформульовано як побудову відображення пари «намір – контекст графічного інтерфейсу» у пару «план звернень до функціональних модулів – мутація стану компонентів»:

$$F : (I, C) \rightarrow (P, \Delta S) \quad (1)$$

де  $I$  позначає намір – формулювання бажаного результату, у якому спосіб його досягнення не вказано; символ  $C$  відповідає контексту графічного інтерфейсу, до якого належать поточне подання, активні фільтри та сортування, склад вибірки, виділені об'єкти й кілька останніх намірів сесії; через  $P$  позначено план звернень до функціональних модулів системи, через  $\Delta S$  – набір операцій над станом графічних компонентів. Контекст входить у задачу не як довідкова інформація. Без нього формулювання на кшталт «покажи лише дешевші варіанти» не має однозначного розв'язку.

План є впорядкованою послідовністю елементарних операцій:

$$P = \langle o_1, o_2, \dots, o_m \rangle, o_j = \langle t_j, a_j \rangle, t_j \in T, a_j \in \text{Schema}(t_j) \quad (2)$$

Множина  $T$  містить операції, опубліковані функціональними модулями системи разом зі схемами своїх параметрів. Ідентифікатор  $t_j$  належить цій множині, а запис  $a_j \in \text{Schema}(t_j)$  означає відповідність аргументів схемі відповідної операції. Принципово, що модель не породжує коду й не звертається до сховища даних. Вона обирає операції зі скінченної, наперед відомої множини та заповнює їхні аргументи. Саме на цьому обмеженні тримається детермінованість шару.

Розв'язком поставленої задачі є не конкретне відображення  $F$ , а процедура його побудови для заданої предметної області. Процедуру подано як послідовність шести етапів, у якій вихід попереднього етапу слугує входом наступного (рисунк 2).



Рис. 2. Загальна схема методики структурного синтезу адаптивних інтерфейсів



Перші два етапи працюють із наявною системою й нічого до неї не додають. Вони лише роблять явним те, що вже реалізовано: перелік операцій ядра з поділом на читання та зміну, а потім формальні описи їхніх параметрів і результатів. Етапи 3 і 4 звернені до предметної області та визначають, які класи намірів взагалі мають сенс для цієї платформи і якими операціями кожен із них реалізується. Два останні етапи стосуються поведінки інтерфейсу. Спершу фіксується словник допустимих змін стану, потім до нього додаються підтвердження, пояснення й механізм повернення до попереднього стану. Порядок етапів не довільний. Простір намірів неможливо формалізувати раніше, ніж стане відомим каталог операцій, бо намір, який жодною операцією не реалізується, до простору не входить.

Намір формалізовано як кортеж

$$I = \langle c, P_i, C_i, \sigma \rangle \quad (3)$$

де  $c$  – клас наміру,  $P_i$  – параметри, видобуті з формулювання та з контексту інтерфейсу,  $C_i$  – обмеження, яким має задовольняти результат,  $\sigma$  – ступінь упевненості інтерпретації. Для платформи каталогу виділено класи первинного добору позицій, уточнення наявної вибірки, зіставлення кількох позицій, перегляду повного опису об'єкта, переходу до іншого подання, дії з побічним ефектом, запиту обґрунтування та повернення до попереднього стану. Дев'ятий клас є службовим і відповідає неоднозначному наміру або наміру поза межами предметної області. Величина  $\sigma$  використовується порогово. Якщо вона нижча за встановлений поріг, система не добудовує відсутніх параметрів здогадом, а формує уточнювальне запитання й не змінює стану інтерфейсу.

Складений намір розкладається на елементарні операції за п'ятьма правилами, які визначають і склад плану, і порядок його виконання (таблиця 2).

Таблиця 2

**Правила декомпозиції складеного наміру та їх вплив на структуру плану**

| Правило | Назва                     | Зміст                                                                                    | Наслідок для структури плану                                                |
|---------|---------------------------|------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| P1      | Атомарність               | Елементарною вважається операція, що відповідає рівно одному запису каталогу Т           | Кроки плану взаємно однозначно відповідають опублікованим операціям         |
| P2      | Залежність за даними      | Крок, аргументи якого отримували лише з результату іншого кроку, виконується після нього | Відношення передування утворюють ациклічний граф, обхід якого задає порядок |
| P3      | Паралелізм                | Кроки без взаємної залежності за даними виконуються одночасно                            | Сума затримок замінюється сумою максимумів за рівнями графа                 |
| P4      | Ідемпотентність читання   | Операція читання не змінює стану ядра системи                                            | Повторне виконання після помилки мережі або тайм-ауту є припустимим         |
| P5      | Ізоляція побічних ефектів | Критичні кроки розміщуються наприкінці плану й не є джерелом даних для інших кроків      | Результати читання показуються користувачеві до запиту на підтвердження     |

Дію цих правил зручно простежити на складеному намірі, який поєднує добір позицій у межах бюджету, зіставлення двох найдешевших варіантів і оформлення заявки. Відповідний граф плану наведено на рисунку 3.

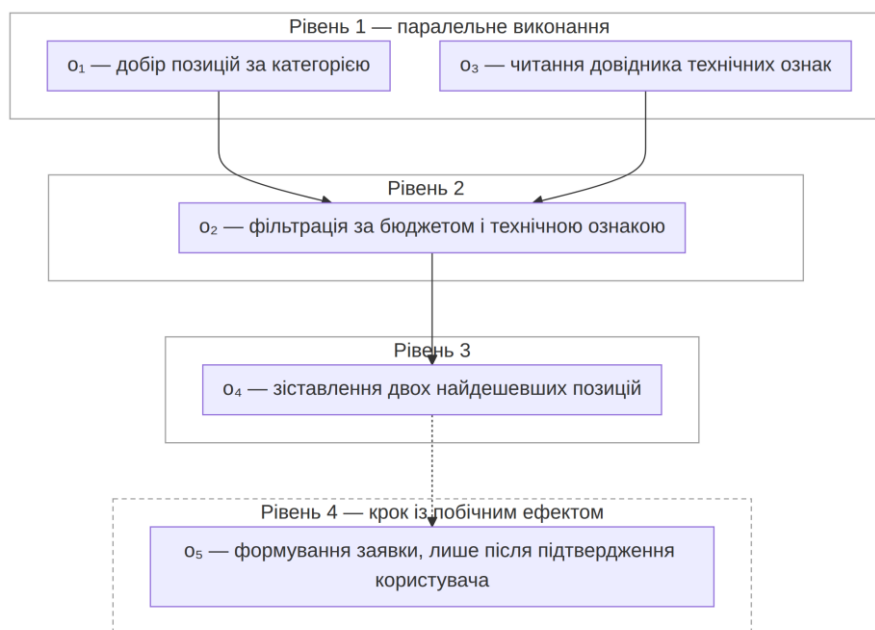


Рис. 3. Граф плану дій для складеного наміру з рівнями паралельного виконання

Кроки  $o_1$  та  $o_3$  не залежать один від одного за даними, тому за правилом Р3 вони утворюють один рівень і виконуються одночасно. Крок  $o_2$  потребує результатів обох, крок  $o_4$  – результату  $o_2$ . Останній крок позначено окремо. Він змінює стан предметної області, тому за правилом Р5 стоїть наприкінці плану й не постачає даних іншим крокам. Користувач бачить відібрані та зіставлені позиції ще до того, як система запитає підтвердження.

Результат виконання плану надходить до інтерфейсу у вигляді патча стану. Стан  $S$  зберігається в централізованому реактивному сховищі клієнта, а мутація  $\Delta S$  є впорядкованим набором операцій над ним:

$$S' = S \oplus \Delta S, \quad \Delta S = \langle \delta_1, \delta_2, \dots, \delta_r \rangle \quad (4)$$

Множина допустимих операцій  $\delta$  скінченна й фіксується на п'ятому етапі методики. Для платформи каталогу до неї входять встановлення значення або діапазону фільтра, перемикання подання, відкриття панелі зіставлення, виділення об'єктів, перехід до іншого розділу та показ пояснення. Обмеженість цієї множини істотна саме тому, що вона відокремлює підхід від генеративного інтерфейсу. Оркестратор не породжує нових елементів, а лише переводить наявні в інший стан. Патч застосовується атомарно, що виключає проміжні подання зі зміненими фільтрами та ще не оновленою вибіркою. Попередній стан зберігається перед застосуванням, тому скасування повертає інтерфейс без жодного звернення до серверного ядра. Те саме сховище є джерелом контексту  $C$  для наступного наміру, чим і забезпечується узгодженість подання з тим, що бачить оркестратор.

Контрольованість спирається на механізм, винесений із тексту запиту до моделі в контракт функціонального модуля. Модуль, що виконує операцію з побічним ефектом, на перший виклик повертає не результат, а запит на введення відомостей, і операція завершується лише після повторного виклику з відповіддю користувача. Помилкова або надто самовпевнена відповідь моделі такого підтвердження не обійде, бо підтвердження вимагає не промпт, а сам модуль. Поряд із цим кожна мутація супроводжується переліком застосованих параметрів, критеріями відбору та кількістю відібраних об'єктів, а збережений попередній стан дозволяє повернути інтерфейс однією дією. Роль шару зводиться до підготовки стану, який людина приймає, коригує або відхиляє.

Компенсація обмежень моделі побудована як два незалежні контури перевірки. Перший діє на рівні формату. План приймається лише тоді, коли він відповідає оголошеній схемі, тому невідоме ім'я операції чи поле невідповідного типу відсікаються ще до виконання. Другий контур перевіряє зміст аргументів за доменними правилами, яких схема не виражає. Відхилений виклик спричиняє повторне звернення до моделі з описом помилки, а після  $k$  невдалих спроб намір переводиться у клас уточнення. Недетермінованість моделі в такій побудові стає джерелом додаткового запитання до користувача, а не джерелом помилкових дій над системою.

Час опрацювання наміру оцінюється співвідношенням, що підсумовує складові критичного шляху:

$$T_{total} = T_{ctx} + n_{LLM} \cdot T_{LLM} + \sum T_{tool,i} + T_{render} \quad (5)$$

де  $T_{ctx}$  – час збирання контексту графічного інтерфейсу;  $n_{LLM}$  – кількість звернень до мовної моделі на критичному шляху;  $T_{LLM}$  – середній час одного звернення до моделі;  $m$  – кількість елементарних операцій у плані  $P$ ;  $T_{tool,i}$  – час виконання  $i$ -ї операції;  $T_{render}$  – час застосування патча стану.

Оцінки порядку величин для кожної складової подано на рисунку 4.

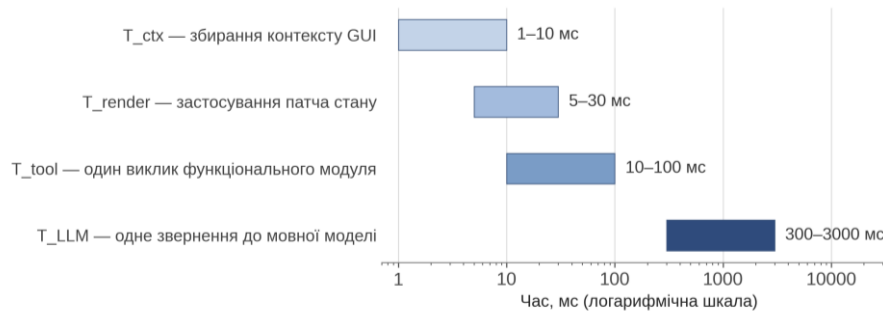


Рис. 4. Інженерні оцінки порядку величин для складових часу опрацювання наміру

Різниця між зверненням до моделі та рештою складових сягає двох порядків. Це означає, що оптимізувати виконання плану майже немає сенсу, доки не мінімізовано кількість звернень до моделі. Результат розрахунку за співвідношенням (5) для часу одного звернення 1,2 с наведено на рисунку 5.

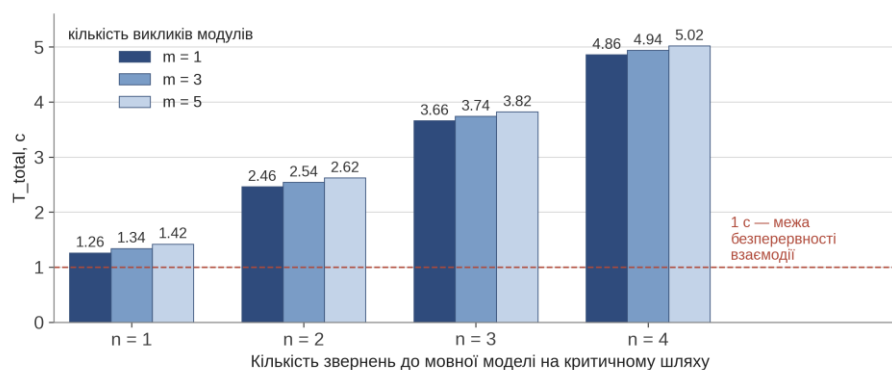


Рис. 5. Розрахункова залежність часу опрацювання наміру від кількості звернень до мовної моделі

Значення для різної кількості викликів функціональних модулів майже не різняться, тоді як перехід від одного звернення до чотирьох збільшує загальний час майже вчетверо. Методика дає  $n_{LLM} = 1$  за рахунок того, що весь план формується за одне звернення завдяки строгій схемі виведення, а не вибудовується покроково. Варто зауважити, що навіть за одного звернення час до першої видимої зміни перевищує одну секунду – межу, за якою взаємодія перестає сприйматися як безперервна [17]. Практичний наслідок цього простий. Генерація пояснення виноситься з критичного шляху, а вибірка показується одразу після застосування патча.

Методику розроблено за низки припущень, які окреслюють межі її застосовності. Система повинна мати ядро з формалізованим контрактом, множина операцій якого скінченна, описувана схемами й незалежна від подання. Стан інтерфейсу має бути серіалізованим. Поза межами застосовності лишаються інтерфейси реального часу, де припустима затримка менша за час одного звернення до моделі, а також середовища з безперервним прямим маніпулюванням, де результат дії має з'являтися синхронно з рухом вказівника.



## ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У роботі розв'язано задачу переходу від концептуального опису інтерфейсного шару-оркестратора до методики його побудови. Задачу структурного синтезу формалізовано як побудову відображення пари «намір – контекст графічного інтерфейсу» у пару «план звернень до функціональних модулів – мутація стану компонентів», а сам розв'язок подано у вигляді шестиетапної процедури, що веде від наявного контракту системи до працездатного шару. Розроблено формальний апарат методики: модель наміру з пороговою оцінкою впевненості інтерпретації, правила декомпозиції складеного наміру та модель мутації стану графічного інтерфейсу як скінченної множини операцій над сховищем стану клієнта.

Мету статті вважаємо досягнутою. Ключовим результатом є розподіл повноважень, за якого мовна модель не має доступу ні до сховища даних, ні до стану інтерфейсу, а виступає виключно генератором структурованого плану. Усе інше – перевірка плану, його виконання, зміна стану компонентів, підтвердження критичних операцій і повернення до попереднього стану – забезпечується детермінованими засобами. Такий розподіл дозволяє говорити про передбачуваність інтерфейсу на рівні наслідків навіть тоді, коли сама модель лишається стохастичною. Аналітична оцінка часу опрацювання наміру показала, що методика доводить кількість звернень до моделі на критичному шляху до одиниці. Тим самим частково знімається обмеження за латентністю та вартістю звернень, яке в попередній публікації авторів було єдиним критерієм, за яким гібридна модель поступалася класичному графічному інтерфейсу.

Подальші дослідження доцільно спрямувати на емпіричну перевірку методики. Насамперед потрібне порівняльне вимірювання часу виконання типових задач, кількості операцій та частоти помилок у класичному й адаптивному інтерфейсах за участю реальних користувачів. Особливої уваги заслуговує питання, як змінюється поріг упевненості  $\sigma$  зі зростанням каталогу операцій, адже саме від нього залежить баланс між зайвими уточнювальними запитаннями та хибно витлумаченими намірами. Перспективним видається також поширення методики на випадок кількох користувачів, які працюють із однією вибіркою одночасно, оскільки в цьому разі модель мутації стану потребує узгодження паралельних патчів.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nielsen, J. (2023). *AI is first new UI paradigm in 60 years*. UX Tigers. <https://www.uxtigers.com/post/ai-new-ui-paradigm>
2. Shneiderman, B. (2022). Human-centered AI: Ensuring human control while increasing automation. In *Proceedings of the 5th Workshop on Human Factors in Hypertext* (pp. 1–2). Association for Computing Machinery. <https://doi.org/10.1145/3538882.3542790>
3. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*. <https://arxiv.org/abs/2210.03629>
4. Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., & Lim, E.-P. (2023). Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2609–2634). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2023.acl-long.147>
5. He, G., Demartini, G., & Gadiraju, U. (2025). Plan-then-execute: An empirical study of user trust and team performance when using LLM agents as a daily assistant. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (pp. 1–22). Association for Computing Machinery. <https://doi.org/10.1145/3706598.3713218>
6. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36. <https://arxiv.org/abs/2302.04761>
7. Patil, S. G., Zhang, T., Wang, X., & Gonzalez, J. E. (2024). Gorilla: Large language model connected with massive APIs. *Advances in Neural Information Processing Systems*, 37. <https://arxiv.org/abs/2305.15334>
8. Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., & Sun, M. (2024). ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*. <https://arxiv.org/abs/2307.16789>
9. Willard, B. T., & Louf, R. (2023). *Efficient guided generation for large language models* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2307.09702>
10. Tam, Z. R., Wu, C.-K., Tsai, Y.-L., Lin, C.-Y., Lee, H., & Chen, Y.-N. (2024). Let me speak freely? A study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024*



- Conference on Empirical Methods in Natural Language Processing: Industry Track* (pp. 1218–1236). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.emnlp-industry.91>
11. Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., & Kambhampati, S. (2023). PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36. <https://arxiv.org/abs/2206.10498>
  12. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12), Article 248. <https://doi.org/10.1145/3571730>
  13. Google Research. (2025). *Generative UI: A rich, custom, visual interactive user experience for any prompt*. Google Research Blog. <https://research.google/blog/generative-ui-a-rich-custom-visual-interactive-user-experience-for-any-prompt/>
  14. Vaithilingam, P., Glassman, E. L., Inala, J. P., & Wang, C. (2024). DynaVis: Dynamically synthesized UI widgets for visualization editing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (pp. 1–17). Association for Computing Machinery. <https://doi.org/10.1145/3613904.3642639>
  15. Zheng, B., Gou, B., Kil, J., Sun, H., & Su, Y. (2024). GPT-4V(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning* (Proceedings of Machine Learning Research, Vol. 235, pp. 61349–61385). PMLR.
  16. Amershi, S., Weld, D., Vorvoreanu, M., Fournay, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S. T., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019). Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–13). Association for Computing Machinery. <https://doi.org/10.1145/3290605.3300233>
  17. Attig, C., Rauh, N., Franke, T., & Krems, J. F. (2017). System latency guidelines then and now—Is zero latency really considered necessary? In D. Harris (Ed.), *Engineering psychology and cognitive ergonomics: Cognition and design* (Lecture Notes in Computer Science, Vol. 10276, pp. 3–14). Springer. [https://doi.org/10.1007/978-3-319-58475-1\\_1](https://doi.org/10.1007/978-3-319-58475-1_1)
  18. Bondarchuk, A., & Kryvolapov, H. (2026). Model of hybrid interaction with a platform based on an AI agent. *Cybersecurity: Education, Science, Technique*, 1(33), 647–656. <https://doi.org/10.28925/2663-4023.2026.33.1246>
  19. Kryvolapov, H. (2026). AI agent as a deterministic interface layer of a digital platform. In *Information Technologies – 2026: Proceedings of the XIII All-Ukrainian Scientific and Practical Conference of Young Scientists* (pp. 130–135). Borys Grinchenko Kyiv Metropolitan University.
  20. Kryvolapov, H. (2025). Transformation of the interaction paradigm: Transition from GUI to hybrid intent-based interfaces. In A. A. Hryhorova (Ed.), *Modern computer systems and technologies: Proceedings of the VIII All-Ukrainian Scientific and Practical Internet Conference of Students, Postgraduate Students and Young Scientists “Modern Computer Systems and Networks in Management”* (pp. 31–33). FOP Vyshemyrskyi V. S. <https://doi.org/10.5281/zenodo.17711825>
  21. Gulak, N., Ilyenko, S., Dubchak, E., Maistrenko, A., & Zhurakovskiy, B. (2025). Software module protecting web applications from several attack types using formal logic. In *Cybersecurity Providing in Information and Telecommunication Systems II (CPITS II)* (pp. 35–45).

**Andrii Bondarchuk**

Doctor of Technical Sciences, Professor,  
Head of the Department of Computer Science,  
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine  
ORCID: 0000-0001-5124-5102  
*a.bondarchuk@kubg.edu.ua*

**Hlib Kryvolapov**

Student, Faculty of Information Technologies and Mathematics,  
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine  
ORCID: 0000-0002-0853-5881  
*hykryvolapov.fitm25m@kubg.edu.ua*

**Olena Nehodenko**

PhD, Associate Professor, Associate Professor of the Department of Computer Science  
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine  
ORCID: 0000-0001-6645-1566  
*o.nehodenko@kubg.edu.ua*

**Zub Liydmyla**

Student, Educational and Scientific Institute of Telecommunications,  
State University of Information and Communication Technologies, Kyiv, Ukraine  
ORCID: 0009-0005-6697-2683  
*zubludmila123@gmail.com*

**A METHODOLOGY FOR THE STRUCTURAL SYNTHESIS OF ADAPTIVE INTERFACES BASED ON THE FORMALISATION OF THE USER INTENT SPACE**

**Abstract.** This article presents a methodology for the structural synthesis of adaptive interfaces that moves the concept of artificial intelligence as an interface layer-orchestrator from the level of a conceptual model to the level of a reproducible engineering procedure. The model of hybrid interaction previously proposed by the authors describes the composition of the system and the distribution of functions across its tiers, yet it does not answer the question of how a verifiable mapping from a natural language intent onto calls to functional modules is to be constructed for a particular platform. It is precisely this gap that the proposed methodology closes. The introduction shows that, in the absence of a formal apparatus, every implementation of an interface layer remains a one-off solution that cannot be transferred to another domain, while the non-determinism of the language model is passed directly on to the behaviour of the interface. The analysis of recent research and publications compares two schemes for organising an agent loop, the iterative one and the one based on single-shot planning, examines work on the invocation of external application programming interfaces by language models, the means of constrained structured output against a schema and empirical assessments of planning reliability, and also outlines the difference between the proposed solution and both generative interfaces and agents that imitate user actions. The results section formalises the synthesis task as the construction of a mapping from the pair «intent – graphical interface context» onto the pair «plan of calls to functional modules – mutation of component state», and presents a six-stage scheme of the methodology, a model of the intent space with a threshold estimate of interpretation confidence, five rules for decomposing a composite intent, and a model of graphical interface state mutation as a finite set of operations over the client state store. Mechanisms of controllability are described, among which the confirmation of critical actions is placed in the contract of the functional module rather than in the text of the prompt. An analytical estimate of intent processing time shows that the dominant term remains the call to the language model, and the methodology therefore reduces the number of such calls on the critical path to one. The conclusions substantiate that the language model in the proposed solution is deprived of any executive authority and acts solely as a generator of a structured plan, whereas execution, interface state change, confirmation and rollback remain deterministic.

**Keywords:** structural synthesis of interfaces; information system; intent space; action plan; GUI state mutation; deterministic interface layer.



## REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Nielsen, J. (2023). *AI is first new UI paradigm in 60 years*. UX Tigers. <https://www.uxtigers.com/post/ai-new-ui-paradigm>
2. Shneiderman, B. (2022). Human-centered AI: Ensuring human control while increasing automation. In *Proceedings of the 5th Workshop on Human Factors in Hypertext* (pp. 1–2). Association for Computing Machinery. <https://doi.org/10.1145/3538882.3542790>
3. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*. <https://arxiv.org/abs/2210.03629>
4. Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., & Lim, E.-P. (2023). Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2609–2634). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2023.acl-long.147>
5. He, G., Demartini, G., & Gadiraju, U. (2025). Plan-then-execute: An empirical study of user trust and team performance when using LLM agents as a daily assistant. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (pp. 1–22). Association for Computing Machinery. <https://doi.org/10.1145/3706598.3713218>
6. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36. <https://arxiv.org/abs/2302.04761>
7. Patil, S. G., Zhang, T., Wang, X., & Gonzalez, J. E. (2024). Gorilla: Large language model connected with massive APIs. *Advances in Neural Information Processing Systems*, 37. <https://arxiv.org/abs/2305.15334>
8. Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., & Sun, M. (2024). ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*. <https://arxiv.org/abs/2307.16789>
9. Willard, B. T., & Louf, R. (2023). *Efficient guided generation for large language models* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2307.09702>
10. Tam, Z. R., Wu, C.-K., Tsai, Y.-L., Lin, C.-Y., Lee, H., & Chen, Y.-N. (2024). Let me speak freely? A study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track* (pp. 1218–1236). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.emnlp-industry.91>
11. Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., & Kambhampati, S. (2023). PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36. <https://arxiv.org/abs/2206.10498>
12. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12), Article 248. <https://doi.org/10.1145/3571730>
13. Google Research. (2025). *Generative UI: A rich, custom, visual interactive user experience for any prompt*. Google Research Blog. <https://research.google/blog/generative-ui-a-rich-custom-visual-interactive-user-experience-for-any-prompt/>
14. Vaithilingam, P., Glassman, E. L., Inala, J. P., & Wang, C. (2024). DynaVis: Dynamically synthesized UI widgets for visualization editing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (pp. 1–17). Association for Computing Machinery. <https://doi.org/10.1145/3613904.3642639>
15. Zheng, B., Gou, B., Kil, J., Sun, H., & Su, Y. (2024). GPT-4V(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning* (Proceedings of Machine Learning Research, Vol. 235, pp. 61349–61385). PMLR.
16. Amershi, S., Weld, D., Vorvoreanu, M., Fournay, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S. T., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019). Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–13). Association for Computing Machinery. <https://doi.org/10.1145/3290605.3300233>
17. Attig, C., Rauh, N., Franke, T., & Krems, J. F. (2017). System latency guidelines then and now—Is zero latency really considered necessary? In D. Harris (Ed.), *Engineering psychology and cognitive ergonomics: Cognition and design* (Lecture Notes in Computer Science, Vol. 10276, pp. 3–14). Springer. [https://doi.org/10.1007/978-3-319-58475-1\\_1](https://doi.org/10.1007/978-3-319-58475-1_1)



18. Bondarchuk, A., & Kryvolapov, H. (2026). Model of hybrid interaction with a platform based on an AI agent. *Cybersecurity: Education, Science, Technique*, 1(33), 647–656. <https://doi.org/10.28925/2663-4023.2026.33.1246>
19. Kryvolapov, H. (2026). AI agent as a deterministic interface layer of a digital platform. In *Information Technologies – 2026: Proceedings of the XIII All-Ukrainian Scientific and Practical Conference of Young Scientists* (pp. 130–135). Borys Grinchenko Kyiv Metropolitan University.
20. Kryvolapov, H. (2025). Transformation of the interaction paradigm: Transition from GUI to hybrid intent-based interfaces. In A. A. Hryhorova (Ed.), *Modern computer systems and technologies: Proceedings of the VIII All-Ukrainian Scientific and Practical Internet Conference of Students, Postgraduate Students and Young Scientists “Modern Computer Systems and Networks in Management”* (pp. 31–33). FOP Vyshemyrskyi V. S. <https://doi.org/10.5281/zenodo.17711825>
21. Gulak, N., Ilyenko, S., Dubchak, E., Maistrenko, A., & Zhurakovskiy, B. (2025). Software module protecting web applications from several attack types using formal logic. In *Cybersecurity Providing in Information and Telecommunication Systems II (CPITS II)* (pp. 35–45).

Отримано редакцією журналу / Received: 10.08.26

Прорецензовано / Revised: 24.08.26

Схвалено до друку / Accepted: 26.10.26



This work is licensed under Creative Commons Attribution-noncommercial-sharealike 4.0 International License.