Створення інструменту на основі великих мовних моделей для відлагодження iOS-застосунків у середовищі виконання

Підготував: Літвінчук Данило
Науковий керівник: Нагірна А. М.

08 червня 2026

Проблема

~50% часу* розробки ПЗ

займає **відлагодження**

Частка тільки зростає зі складністю систем, тож ручний підхід масштабується погано

* за дослідженням The visual debugger tool / T. Tim Kräuter et al. 2024. P. 1–2

Мета дослідження

розробити й оцінити інструмент для автономного відлагодження iOS-застосунків на основі великих мовних моделей (LLM)

Предмет дослідження

методи й архітектурні рішення побудови такого інструменту, що забезпечують символьне заземлення агента в процесі виконання програми (англ. runtime) та протидіють деградації ефективності сесії (DDI)

Об’єкт дослідження

процес автономного відлагодження iOS-застосунків LLM-агентом, що взаємодіє зі станом виконання через міст LLDB–MCP

Наукова новизна

Уперше розроблено метод взаємодії LLM-агента з середовищем виконання iOS-застосунку та оцінено його ефективність відповідно до сформованих критеріїв, що ґрунтуються на різних рівнях символьного заземлення та мірі деградації ефективності налагодження.

Завдання роботи

1. Формалізувати ключові перешкоди: символьне заземлення і деградацію за моделлю DDI
2. Встановити граничні умови платформи iOS: ASLR, підпис коду
3. Обґрунтувати ключові архітектурні рішення: ізоляція несумісних runtime, transport MCP, міра декомпозиції простору дій агента
4. Реалізувати функціональний MCP-сервер на повний цикл відлагодження – від приєднання до процесу до ін'єкції коду.

Завдання роботи

…

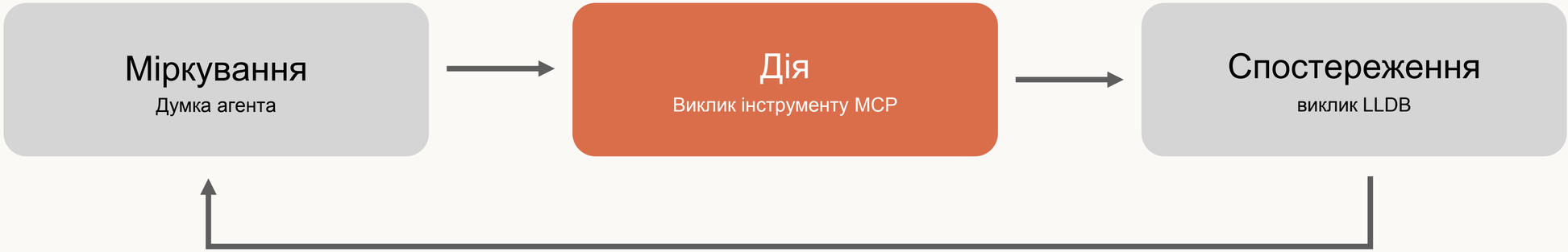
5. Спроєктувати вимірювальний шар, що фіксує метрики кожного виклику як мінімальне покриття для відновлення сталої згасання λ

6. Провести порівняльний експеримент, побудувати криву $E(t)$, оцінити λ й перевірити головну тезу роботи.

ReAct: замкнутий цикл взаємодії з runtime

Chain-of-Thought відтворює причинно-наслідковий ланцюг збою, але не перевіряє стан конкретного покажчика чи потоку без втручання людини

ReAct замикає цикл: агент генерує думку, виконує дію через виклик інструмента MCP й отримує спостереження з LLDB, що або підтверджує, або спростовує гіпотезу



Ізоляція несумісних середовищ виконання

Найбільше на архітектуру вплинула несумісність середовищ виконання: MCP SDK вимагає Python 3.10+, тоді як LLDB Scripting Bridge залежить від Python 3.9 з дистрибутива Xcode.

Декомпозиція простору дій агента

Структуровані

- read_variable – значення змінної
- get_stack_trace – журнал стеку викликів
- resume / wait_for_stop – керування
- inject_code – ін’єкція коду

покривають лексичний і темпоральний рівні заземлення.

Неструктуровані

Обхід графу об'єктів та ієрархії представлень не має власного інструмента.

Агент змушений запитувати цю інформацію через універсальний execute_command, отримуючи неструктурований текст замість дерева об'єктів.

перевірка виявлення retain cycle показала найдовшу траєкторію навіть у агента з MCP

SessionLogger: відновлення λ із записаних сесій

Міст перехоплює кожен виклик інструмента й фіксує мінімальне достатнє покриття для побудови кривої деградації пост-фактум, без повторного інструментування.

Вісь t

Порядкові номери й UTC-мітки часу задають реальний інтервал між кроками

Одиниці вимірювання

Підмножина “дієвих” інструментів (resume, wait, read_variable, inject_code, execute) нормує траєкторію.

Непрямий $E(t)$

Частота компіляційних помилок у inject_code – вимірюваний показник втрати заземлення.

Дизайн експерименту

Один застосунок понад 200 екранів

Масштаб робить внесок доступу до стану вимірюваним і усуває сторонню варіативність стилю та структури коду

Контрольовані умови

Claude Code на Sonnet 4.6 · ідентичний промпт · спільний ліміт 40 кроків · по 5 прогонів на умову

Baseline

агент без інструментів отримує лише вихідний код і текст помилки; кроки вимірюються за тією самою шкалою

Дизайн експерименту

Чотири дефекти за рівнем заземлення

- **nil-unwrap**
  навантажує: лексичне
- **вихід за межі масиву**
  навантажує: лексичне + темпоральне
- **race condition**
  навантажує: темпоральне, багатопотокове
- **retain cycle**
  навантажує: структурне

Крива деградації $E(t)$ та порівняльна таблиця

Клас дефекту	MCP	База	Виграш
race condition	4/5	0/5	17 vs 40 кроків
retain cycle	4/5	1/5	21 vs ~35 кроків
Вихід за межі масиву	—	—	≈43 % траєкторії
Nil-unwrap	—	—	≈25 % траєкторії

Усі прогони – агент Claude Code (Sonnet 4.6), по 5 разів на умову. λ для MCP-умови оцінено як ≈ 0,03 на дієвий крок.

Ключова закономірність

Перевага агента з доступом до рантайму тим більша, чим глибше причина дефекту прихована від статичного аналізу коду:

race-condition – **4/5** проти **0/5** виявлень – причина повністю прихована від статичного аналізу

вихід за межі масиву – **≈43 %** скорочення траєкторії

nil-unwrap – **≈25 %** – заземлення вже забезпечене журналом викликів

retain-cycle – навіть MCP-агент показав найдовшу траєкторію – емпіричне підтвердження відсутності типізованого інструмента структурного заземлення

Висновки

Практична цінність інструменту визначається не самим фактом його наявності, а тим, наскільки повно він закриває рівень символьного заземлення, якого потребує конкретний дефект

Мету досягнуто: реалізовано функціональний MCP-сервер на повний цикл відлагодження й експериментально оцінено його ефективність; стала згасання для MCP-умови $\lambda \approx 0.03$.

Перспективи вдосконалення:

- Типізований інструмент інспекції ієрархії представлень для структурного заземлення (замість текстового recursiveDescription).
- Активні стратегії протидії **DDI** – виявлення циклів і скидання контексту, пороги яких тепер обґрунтовуються отриманою оцінкою λ.

Дякую за увагу