

Una frase prima del piano

Uno studio confermativo pre-registrato del pre-mortem come prompt negli agenti di programmazione basati su LLM

Carlos Ulisses Flores

Codex Hash Research Laboratory

ORCID: 0000-0002-6034-7765

Codex Hash Research Report 2026-03

Data: 17 novembre 2026

Riassunto

Un pre-mortem chiede al team di supporre fallito il piano e di elencarne i motivi (Klein, 2007). In uno studio pre-registrato precedente la metà di elencazione dell'esercizio, in una frase, ha fatto coprire al piano di un agente di programmazione più rischi elencati separatamente, come risultato secondario (Flores, 2026a). Questo è lo studio confermativo. Claude Code, in modalità di pianificazione su Claude Opus 5.5 e Claude Sonnet 5.5, ha ricevuto dodici richieste di modifica su repository pubblici, con due richieste che differiscono per una frase: "Plan this change." e "Before planning, list the ways this change could go wrong." Tre modelli di altri fornitori hanno scritto alla cieca i rischi di riferimento, e un giudice cieco di un altro fornitore ha valutato ogni risposta. Protocollo, liste e analisi sono stati sigillati e marcati temporalmente prima della prima prova. In 240 prove la frase ha portato la copertura dal 46,1% al 57,5%, differenza media di 11,4 punti percentuali su 24 strati di compito per modello (permutazione stratificata, $p < ,001$, anche in ciascun modello). Un secondo giudice ha dato +10,7 punti (κ di Cohen = ,86). Senza i tre strati con prova persa per errore di rete, la differenza è di 10,1 punti. Rispetto a un secondo insieme di liste, chiesto con altre parole, è di 6,7 punti ($p < ,001$), e 9,0 secondo un giudice successivo di un terzo fornitore. L'affermazione si limita a due modelli di un fornitore, a liste scritte da modelli e a piani, non a codice.

Parole chiave: pre-mortem; *prompt* engineering; agenti di programmazione basati su LLM; identificazione dei rischi; pre-registrazione; pianificazione del software

Abstract

A pre-mortem asks a team to assume that its plan has failed and to list why (Klein, 2007). In an earlier pre-registered study the listing half of that exercise, put in one sentence, made a coding agent's plan cover more independently listed risks, as a secondary result (Flores, 2026a). This paper reports the confirmatory study. Twelve change requests on pinned public repositories were given to Claude Code in plan mode, on Claude Opus 5.5 and Claude Sonnet 5.5, under two requests that differ by one sentence: "Plan this change." and "Before planning, list the ways this change could go wrong." Reference risks were written blind by three models of other vendors, and a blind judge of another vendor scored each answer. Protocol, lists and analysis were sealed and timestamped before the first trial. Over 240 trials the sentence raised coverage from 46.1% to 57.5%, a mean difference of 11.4 percentage points over 24 task-by-model strata (stratified permutation $p < .001$, also within each model). A second judge gave +10.7 points (Cohen's $\kappa = .86$). Without the three strata that hold a trial lost to a network error the difference is 10.1 points. Against a second set of reference lists, worded differently, it is 6.7 points ($p < .001$), and 9.0 by a third-vendor judge added later. The claim is limited to two models of one vendor, to model-written reference lists and to plans, not code.

Keywords: pre-mortem; prompt engineering; LLM coding agents; risk identification; pre-registration; software planning

Indice

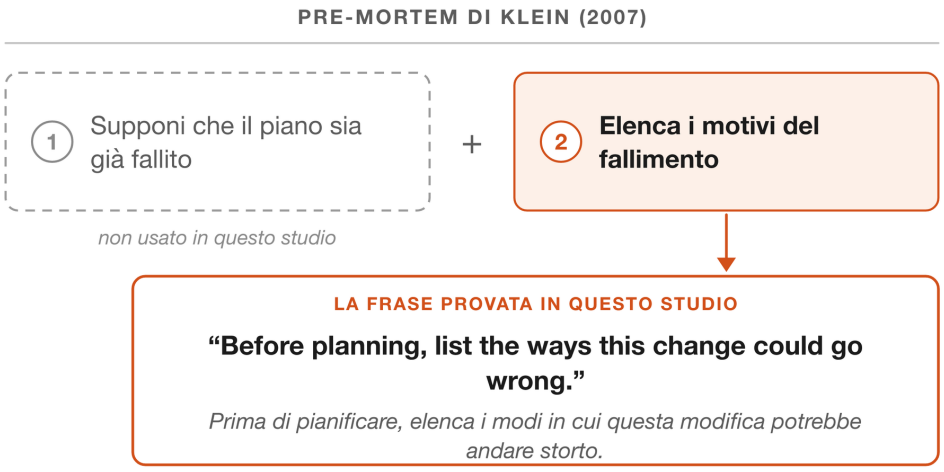
1. Introduzione 5
1.1. Contributi 8
2. Fondamenti e lavori correlati 9
2.1. Il pre-mortem nelle persone 9
2.2. Elencare il fallimento prima di rispondere, in LLM e agenti 9
2.3. Promemoria e previsioni nella generazione di codice 10
2.4. Istruzioni brevi e sensibilità al *prompt* 12
2.5. Dove si colloca questo studio 12
3. Lo studio di origine 13
4. Metodo 14
4.1. Ipotesi e progetto 14
4.2. Compiti 15
4.3. Condizioni e *harness* 15
4.4. Rischi di riferimento, scritti alla cieca 19
4.5. Valutazione 20
4.6. Analisi e regola di decisione 22
4.7. Pre-registrazione e registro di integrità 23
5. Risultati 27
5.1. Raccolta 27
5.2. Risultato principale 27
5.3. Secondo giudice 30
5.4. Rilevanza 30
5.5. Sensibilità alle tre prove con errore 31
5.6. Costo 33
5.7. Quali rischi aggiunge la frase (esplorativo) 34
5.8. Un secondo metro di misura, con altre parole (esplorativo) 35
6. Discussione 39
7. Obiezioni e minacce alla validità 41
8. Deviazioni dal protocollo 47
9. Conclusione 49
10. Disponibilità di dati e codice 50
11. Dichiarazioni 51
Riferimenti bibliografici 52
Appendice A — Le richieste esatte 57
Appendice B — Una lista di riferimento completa 58
Appendice C — I testi in inglese, tradotti 59
Nota Tecnica 61
Sull’autore 62

1. Introduzione

Gli agenti di programmazione basati su grandi modelli linguistici (LLM) pianificano ed eseguono già modifiche in repository reali (Jimenez et al., 2024). I piani studiati qui vengono richiesti prima che si scriva qualsiasi codice, un punto precoce in cui individuare un rischio dimenticato. Questo articolo mette alla prova un'affermazione: aprire una richiesta di pianificazione con la frase "Before planning, list the ways this change could go wrong." (in italiano: "Prima di pianificare, elenca i modi in cui questa modifica potrebbe andare storto."), al posto di "Plan this change.", fa coprire alla tabella dei rischi che chiude il piano dell'agente più rischi di quelli che si aspettano revisori indipendenti. L'affermazione è stretta di proposito, perché gli effetti della formulazione di un *prompt* spesso non si mantengono da un modello o da un insieme di compiti al successivo (Sclar et al., 2024).

La frase viene dalla ricerca sulla decisione. In un *pre-mortem*, il team immagina che il piano sia già fallito e annota i motivi (Klein, 2007). La tecnica si appoggia sulla *prospective hindsight*: chi spiega un esito come se fosse già avvenuto genera più motivi di chi spiega un esito che può avvenire (Mitchell et al., 1989). Le norme di valutazione del rischio elencano tecniche strutturate affini, come l'analisi dei modi di guasto e dei loro effetti, tra i metodi che un'organizzazione può applicare prima di agire (International Electrotechnical Commission, 2019). La frase provata qui mantiene solo la seconda metà dell'esercizio di Klein, l'elencazione, e abbandona la supposizione di fallimento (Figura 1).

Figura 1
Le due metà di un pre-mortem e quella che la frase provata mantiene



Per gli LLM, l’evidenza più vicina riguarda la sicurezza, dove far elencare al modello i danni possibili prima di rispondere riduce la nocività giudicata più che ordinargli di essere sicuro (Kim et al., 2026). Negli agenti, anticipare il fallimento prima di agire è stato incorporato in metodi a più passi e valutato nel suo insieme (Wang et al., 2024; Wang et al., 2026). Nella generazione di codice, i promemoria per essere sicuri danno risultati misti o nulli (Tony et al., 2025; Kharma et al., 2026), e un *prompt* più lungo che chiede a un agente di programmazione di prevedere i casi limite ha aiutato due modelli e danneggiato un terzo (Patel et al., 2026). Manca una misurazione controllata della dose minima nella pianificazione del software: una frase, contro la stessa richiesta senza di essa, valutata rispetto a rischi che un’altra persona ha elencato.

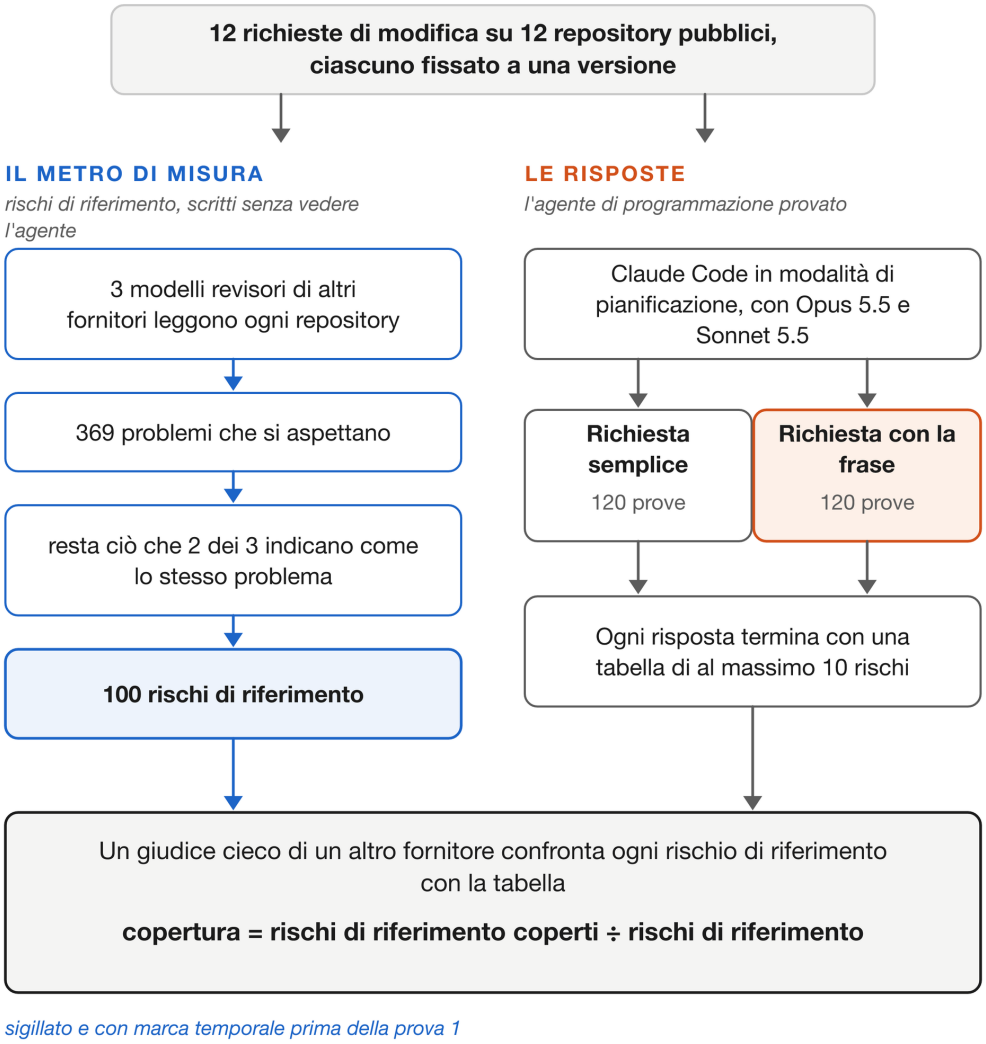
La frase è emersa da una valutazione precedente di uno strumento mantenuto dall’autore. ASP è un *plugin open source*, un’estensione installabile, per Claude Code, l’agente di programmazione da riga di comando di Anthropic; guida l’agente attraverso un protocollo di pre-mortem a 13 fasi, e la sua misurazione pre-registrata iniziale, quella in cui protocollo e analisi sono stati fissati in pubblico prima della raccolta dei dati (Sezione 4.7), non ha trovato alcun beneficio misurabile: il protocollo completo ha coperto meno rischi di riferimento di una richiesta semplice con la frase

aggiunta (Flores, 2026a). La frase, a sua volta, ha superato la richiesta semplice di 10 punti percentuali. Quel confronto era stato pre-registrato come secondario, su un solo modello e su sei compiti, quindi giustificava un nuovo studio e nient'altro (Nosek et al., 2018).

Questo articolo riporta quello studio. È stato progettato per dare al risultato precedente una possibilità equa di fallire: dodici compiti nuovi, un secondo modello, il confronto come unica domanda principale e una regola di decisione sigillata prima che esistesse qualsiasi dato (Figura 2).

Figura 2

Progetto dello studio: come i rischi di riferimento e le risposte vengono prodotti e valutati



1.1. Contributi

1. **Un test confermativo e pre-registrato di una richiesta di una frase in un agente di programmazione.** In 240 prove, su dodici repository pubblici fissati e due modelli, la frase porta la copertura dei rischi elencati in modo indipendente in alto di 11,4 punti percentuali ($p < ,001$), e di un valore simile all'interno di ciascun modello (Sezione 5.2).
2. **Verifiche sul guadagno.** Un secondo giudice di un altro fornitore arriva alla stessa decisione (Sezione 5.3), i rischi elencati non sono giudicati meno concreti (Sezione 5.4), e il risultato sopravvive al ritiro di tre prove che un errore di rete ha azzerato a favore del confronto (Sezione 5.5). Rispetto a liste di riferimento chieste in seguito con altre parole, il guadagno resta positivo e scende a 6,7 punti (Sezione 5.8).
3. **Un registro sigillato e aperto.** Ogni *prompt* definito dallo studio, ogni lista di riferimento, risposta, verdetto e file di analisi è pubblico ed è collegato a un registro concatenato tramite hash (*ledger*), una catena di impronte digitali di file, il cui stato precedente alle prove è ancorato nella *blockchain* di Bitcoin, un registro pubblico che è impraticabile riscrivere (Sezione 4.7), e un solo comando ricalcola ogni risultato dello studio (Flores, 2026b).
4. **Un elenco dichiarato di obiezioni.** La Sezione 7 prende le obiezioni più forti al risultato, comprese quelle sollevate da revisori esterni di questo manoscritto, e risponde a ciascuna con dati oppure la registra come aperta.

Per quanto ci è noto, nessuno studio precedente misura l'effetto di un'istruzione minima di pre-mortem sulla copertura di rischi sollevati in modo indipendente nella pianificazione di modifiche software da parte di un agente di programmazione, a parte lo studio di origine, dove era un

confronto secondario (Flores, 2026a). La Sezione 2.5 dice che cosa delimita questa affermazione. Il meccanismo in sé, elencare i modi di fallimento prima di rispondere, ha un sostegno precedente (Kim et al., 2026) e non è rivendicato qui.

2. Fondamenti e lavori correlati

2.1. Il pre-mortem nelle persone

Klein (2007) descrive il pre-mortem di progetto come una breve riunione prima che il lavoro cominci: si suppone che il piano sia fallito, e ogni partecipante scrive i motivi. L'esperimento di sostegno è quello di Mitchell et al. (1989), in cui la certezza su un esito, più della sua posizione nel tempo, ha aumentato il numero di spiegazioni che le persone hanno prodotto. In un esperimento controllato con 178 studenti, il pre-mortem ha ridotto la fiducia nel piano di 25,0 punti, contro 14,0 della lista di pro e contro e 12,4 della lista solo di contro (Veinott et al., 2010). Gli effetti umani stabiliti stanno quindi nella quantità e nella fiducia, per cui uno studio con LLM deve verificare che più elementi non significhino semplicemente elementi più vaghi (Sezione 5.4).

La procedura umana ha due parti, la supposizione di fallimento e l'elencazione dei motivi (Klein, 2007). Un *prompt* può chiedere la seconda senza la prima, e la frase studiata qui fa esattamente questo. Il confronto tra risposte umane e di IA nella pianificazione per pre-mortem è cominciato nella ricerca sui fattori umani (Veinott e Lehman, 2024), ma come confronto di chi risponde, non di come il modello viene istruito.

2.2. Elencare il fallimento prima di rispondere, in LLM e agenti

InvThink è il lavoro precedente più vicino (Kim et al., 2026). Il modello elenca i danni possibili, analizza le loro conseguenze e solo allora risponde sotto i vincoli che ha derivato. Nella modalità di solo *prompt*, senza addestramento, il metodo riduce un punteggio di nocività giudicata da 3,22

a 2,19, mentre un'istruzione di sicurezza di lunghezza simile arriva solo a 2,62. La spiegazione degli autori è che l'enumerazione esplicita porta alla luce conoscenza che una richiesta diretta lascia inutilizzata. L'esito lì è contenuto nocivo, e la cornice ha tre passi.

Negli agenti, Devil's Advocate fa riflettere un agente di navigazione web su probabili fallimenti e rimedi alternativi prima di ogni azione, e riporta il 23,5% di successo nei compiti contro il 19,8% di una linea di base di pianificare e agire, con il 45% in meno di revisioni di piano rispetto al miglior metodo concorrente (Wang et al., 2024). PreFlect critica il piano prima dell'esecuzione usando errori di pianificazione distillati da traiettorie precedenti, e porta il successo su un *benchmark* di assistente generale, un insieme fisso di compiti di prova, dal 46,1% al 58,2% (Wang et al., 2026). Entrambi valutano metodi a più passi; nessuno mette alla prova un'istruzione di una frase, e nessuno conta quanti rischi attesi nomina un piano.

Una terza linea riguarda ciò che un modello senza istruzione lascia fuori. In un esperimento sul campo piccolo e preliminare sulla pianificazione di *sprint*, in cui un team pianifica il suo prossimo breve ciclo di lavoro, un LLM che pianificava da solo ha documentato il 36,4% dei rischi che poi si sono materializzati, 4 rischi, contro il 78,6% dei pianificatori umani, 11 rischi (Caraeni et al., 2026). Gli agenti LLM possono anche produrre molte più conseguenze candidate di un prodotto rispetto a specialisti che lavorano da soli (Fröhling et al., 2026). Il risultato sul campo è il divario che un *prompt* di pianificazione potrebbe restringere, e il suo esito è affine a quello usato qui: là, cattura di rischi che si sono materializzati; qui, cattura di rischi che i revisori si aspettavano.

2.3. Promemoria e previsioni nella generazione di codice

Diversi studi aggiungono un'istruzione di cautela a un *prompt* di generazione di codice e misurano il codice. In GPT-4, alcune tecniche di *prompt* hanno ridotto la densità di debolezze di sicurezza

tra il 59% e il 78% (Tony et al., 2025). Su cinque modelli e quattro metodi di *prompt*, un altro studio non ha trovato una riduzione statisticamente significativa di vulnerabilità (Kharma et al., 2026). Un *prompt* di sistema che chiedeva copertura dei casi limite spesso non è stato migliore della linea di base nella generazione di codice senza esempi (Cheng e Mastropaolo, 2026). Questi risultati mettono in guardia dal supporre che una frase breve di cautela aiuti.

Differiscono anche dallo scenario attuale in un modo che conta. Le loro istruzioni più semplici sono promemoria ("sii sicuro"), sebbene le tecniche confrontate comprendano anche quelle iterative, come la critica e il miglioramento ricorsivi (Tony et al., 2025), e l'esito è il codice. Un'istruzione di *elencare* è un atto diverso da un'istruzione di *fare attenzione*, distinzione che InvThink traccia rispetto alla sua linea di base di *prompt* di sicurezza (Kim et al., 2026). Chiedere al modello di elencare le sue ipotesi implicite è un atto di enumerazione affine, studiato per la generazione di codice con un *prompt* strutturato (Wu, 2026).

Lo studio più vicino nel software è quello di Patel et al. (2026). Danno ad agenti di programmazione, tra cui Claude Code, un *prompt* di più frasi che chiede all'agente di prevedere i casi limite che la sua implementazione potrebbe lasciar passare, prima di cominciare, durante l'implementazione e prima di consegnare. L'esito sta nel codice: un tasso congiunto di *patch* (modifiche al codice proposte) che risolvono il compito ed evitano anche un difetto che una modifica successiva del progetto ha dovuto correggere. Nel loro confronto, quel tasso è salito dal 24,5% al 32,6% in Claude Opus 5, è salito in un secondo modello ed è sceso dal 17,0% al 13,7% in Claude Sonnet 5, mentre la quota di compiti risolti è scesa in tutti e tre. Il loro risultato è la ragione per cui l'affermazione attuale non si estende dai piani al codice, e mostra che l'effetto di questi *prompt* può dipendere dal modello.

2.4. Istruzioni brevi e sensibilità al *prompt*

Che una frase possa spostare l'output di un modello è ben stabilito. "Let's think step by step" ha portato l'accuratezza senza esempi su un *benchmark* aritmetico dal 17,7% al 78,7% (Kojima et al., 2022), e il *prompt* Plan-and-Solve ha affinato la stessa idea chiedendo al modello di elaborare prima un piano (Wang et al., 2023). La stessa letteratura mostra perché tali risultati richiedano replicazione: l'accuratezza è variata fino a 76 punti tra scelte di formattazione senza alcun significato (Sclar et al., 2024).

Nemmeno un *prompt* di pre-mortem aiuta ovunque. In un confronto pre-registrato di 38 *prompt* per domande di previsione, che includeva un *prompt* di pre-mortem, la maggior parte dei *prompt* ha cambiato poco l'accuratezza (Schoenegger et al., 2025). Un effetto di formulazione visto una volta, in un modello e su una manciata di compiti, può essere rumore di quella sensibilità. È la ragione metodologica di uno studio confermativo con un secondo modello, compiti nuovi e una regola fissata in anticipo (Nosek et al., 2018).

2.5. Dove si colloca questo studio

La combinazione studiata qui è: uno scenario di ingegneria del software con repository reali; un esito definito come copertura di una lista di riferimento che l'agente provato non ha mai visto; una dose di una frase; e un risultato precedente in cui un protocollo lungo e strutturato ha coperto il 48% dei rischi di riferimento contro il 66% di quella frase (Flores, 2026a). I vicini più prossimi differiscono in almeno uno di questi punti: InvThink nel dominio e nell'esito (Kim et al., 2026), e Patel et al. (2026) nella dose e nell'esito.

La ricerca bibliografica dietro la dichiarazione di novità della Sezione 1.1 ha coperto riviste e server di *preprint* in inglese fino al 14 novembre 2026, con ricerche aggiuntive in portoghese,

tedesco, giapponese e cinese; il suo registro, con la situazione di ciascuna fonte, è nel repository (Flores, 2026b). Ha lavorato su titoli e riassunti, quindi uno studio che provi tale condizione senza nominarla lì, o uno che non sia indicizzato, può essere sfuggito.

3. Lo studio di origine

Lo studio precedente ha fatto 90 prove su sei repository pubblici con Claude Opus 5.5, sotto tre condizioni: la richiesta semplice, la richiesta con la frase e il protocollo ASP (Flores, 2026a, Sezione 5.5). La Tabella 1 e la Figura 3 danno il risultato.

Tabella 1
Copertura dei rischi e tempo per prova nello studio di origine (2026-11), per condizione

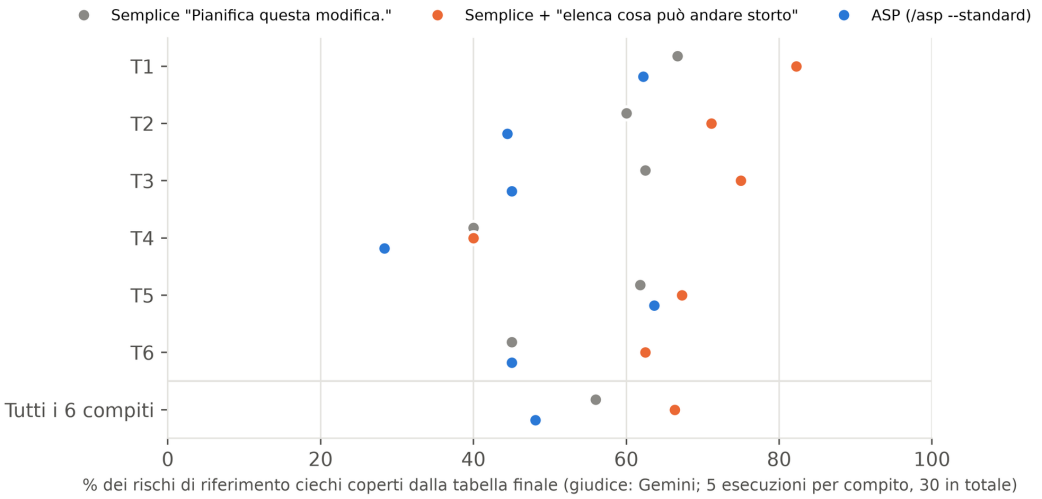
Condizione	Coperti (giudice principale)	Coperti (secondo giudice)	Tempo medio per prova
Richiesta semplice	56%	57%	58 s
Richiesta semplice più la frase	66%	70%	50 s
Protocollo ASP (13 fasi)	48%	54%	175 s

Nota. 30 prove per condizione su sei compiti. Fonte: Flores (2026a).

Il confronto principale pre-registrato di quello studio, ASP contro la frase, è risultato negativo su tutti e sei i compiti: −18 punti, p = 1,0 (Figura 3). Il confronto secondario, la frase contro la richiesta semplice, è stato di +10 punti (p = ,001) secondo il giudice principale e di +13 punti (p < ,001) secondo il secondo (Tabella 1). Poiché il protocollo aveva etichettato quel confronto come secondario, il resoconto precedente ne ha affermato solo la direzione e ha annunciato il presente studio (Flores, 2026a).

Figura 3

Studio di origine: quota dei rischi di riferimento ciechi coperta dalla tabella finale di ogni condizione, per compito e in totale (giudice principale)



Nota. Nella legenda, "/asp --standard" esegue il protocollo ASP ed "elenca cosa può andare storto" indica la frase.

Due proprietà di quel risultato hanno plasmato il nuovo progetto. L'effetto è stato misurato in un solo modello, e gli effetti dei *prompt* possono invertirsi tra modelli (Patel et al., 2026). E si basava su sei compiti, quindi il nuovo studio ne usa dodici che il precedente non aveva mai toccato (Tabella 2).

4. Metodo

4.1. Ipotesi e progetto

L'ipotesi è direzionale: la frase aumenta la copertura. Il progetto incrocia dodici compiti, due modelli e due condizioni, con cinque repliche per cella, per un totale di 240 prove (Figura 2). Uno *strato* è un compito in un modello; sono 24, ciascuno con cinque prove per condizione. Il protocollo ha fissato ogni elemento di questa sezione prima della prima prova, seguendo la pratica

di separare l'analisi confermativa da quella esplorativa (Nosek et al., 2018), e il suo archivio è nel repository (Flores, 2026b).

4.2. Compiti

Ogni compito è una richiesta di modifica di una frase in un repository pubblico fissato a un *commit*, una versione registrata del suo codice (Tabella 2). Le richieste sono state scritte per avere conseguenze oltre i file che suggerisce una prima lettura: dati salvati che devono migrare, *client* che devono continuare a funzionare, valori predefiniti da cui dipendono altri progetti. Coprono sette linguaggi, una varietà che le linee guida per gli studi con LLM nell'ingegneria del software raccomandano al posto di un campione di un solo linguaggio (Baltes et al., 2027). Il file dei compiti è stato registrato in un *commit* prima che qualsiasi lista di riferimento fosse scritta (Sezione 4.7).

4.3. Condizioni e *harness*

Le due condizioni inviano lo stesso testo, tranne che per una frase (l'Appendice A le dà entrambe per intero):

- **Semplice:** la richiesta di modifica, poi "Plan this change." (in italiano: "Pianifica questa modifica.")
- **Frase:** la richiesta di modifica, poi "Before planning, list the ways this change could go wrong." (in italiano: "Prima di pianificare, elenca i modi in cui questa modifica potrebbe andare storto."; Tabella A1; Figura 4)

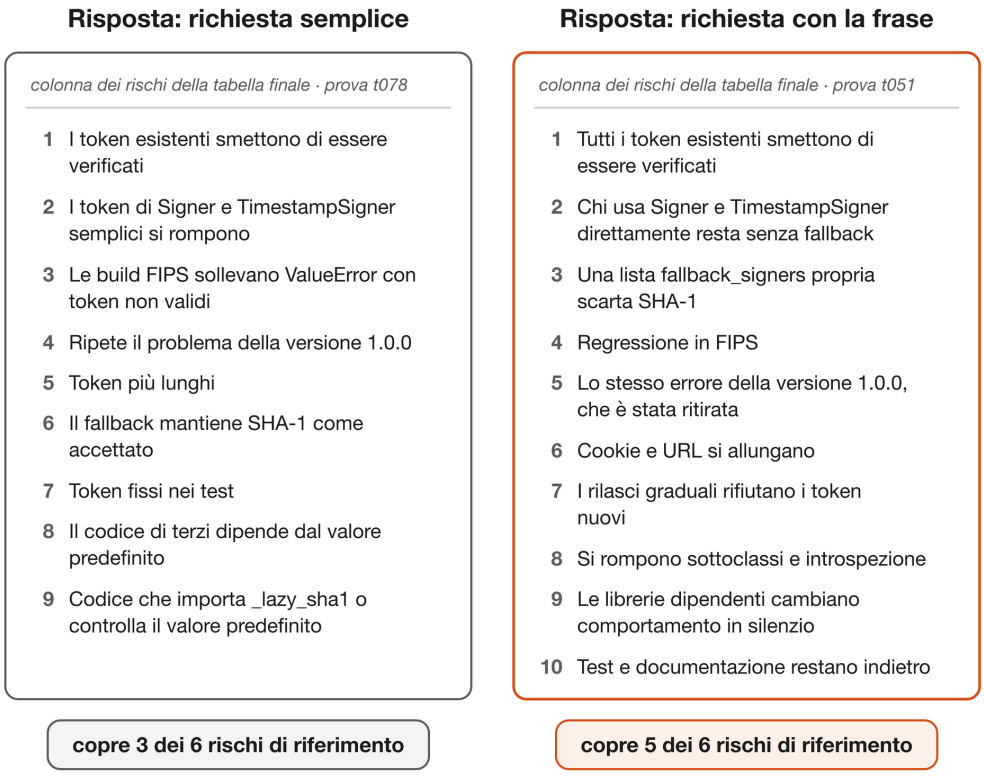
Figura 4

Le due richieste, affiancate: una frase differisce e tutto il resto è identico



Entrambe portano a seguire lo stesso *contratto di output*, un paragrafo fisso che ordina all’agente di dichiarare le ipotesi invece di fare domande, di non modificare alcun file del progetto (la modalità di pianificazione può comunque salvare il piano stesso in un file; Sezione 5.6), di rispondere in inglese e di terminare con i principali rischi della modifica in una tabella di al massimo dieci righe (Appendice A). Il contratto conta per leggere il risultato. La condizione semplice non è un agente a cui non è mai stato chiesto del rischio: entrambe le condizioni devono terminare con una tabella di rischi. Il confronto misura quindi l’effetto congiunto di aprire la richiesta con la frase al posto dell’istruzione semplice, oltre all’obbligo di riportare i rischi alla fine; non separa la posizione della frase dalla sua formulazione (Sezione 7, obiezione 5). I *prompt* sono, byte per byte, quelli dello studio di origine (Flores, 2026a). La Figura 5 mostra ciò che torna indietro: la colonna dei rischi di due risposte reali alla stessa richiesta, una per condizione.

Figura 5
La colonna dei rischi di due risposte reali alla richiesta della Figura 4, una per condizione (compito U08, Claude Opus 5.5)



tradotto dall'inglese; l'originale è nel repository

Nella Figura 5, FIPS indica le norme federali statunitensi per l'elaborazione delle informazioni (Federal Information Processing Standards); un sistema costruito in modalità FIPS rifiuta gli algoritmi crittografici che quelle norme non approvano.

Tabella 2
I dodici compiti

Compito	Repository (linguaggio)	Richiesta di modifica
U01	remix-run/blues-stack (TypeScript)	"Require new users to verify their email address before they can log in."
U02	golang-gin-realworld-example-app (Go)	"Let authors change an article's title, and its URL slug with it, without breaking links that were already shared."
U03	laravel-realworld-example-app (PHP)	"Let moderators hide abusive comments, and let the comment's author appeal within 7 days."
U04	django-axes (Python)	"Lock users out by username and IP address together instead of by IP address alone, as the new default."

Compito	Repository (linguaggio)	Richiesta di modifica
U05	expressjs/session (JavaScript)	“Make sessions roll by default, so that every request extends the session’s expiry.”
U06	flasky (Python)	“Let every user download all of their data (profile, posts and comments) as a single file.”
U07	fastapi-tdd-docker (Python)	“Replace the integer IDs that the API exposes with UUIDs, while keeping existing API clients working.”
U08	pallets/itsdangerous (Python)	“Change the default signing digest from SHA-1 to SHA-256.”
U09	aspnetcore-realworld-example-app (C#)	“Store passwords with a stronger hashing scheme without forcing existing users to reset their passwords.”
U10	CleanArchitecture (C#)	“Make every to-do list and item belong to a tenant, so that each company only sees its own data.”
U11	spring-petclinic (Java)	“Let pet owners book visits online, without two visits being booked for the same vet at the same time.”
U12	starlette (Python)	“Make the session middleware’s cookie Secure and SameSite=Strict by default.”

Nota. Gli indirizzi completi dei repository e gli identificatori dei *commit* sono in benchmark/study-2026-11b/tasks.json (Flores, 2026b). Le richieste sono state inviate all’agente in inglese, come qui riportate; la Tabella C1 ne dà la traduzione. Lo *slug* è la parte leggibile dell’indirizzo di una pagina, UUID è un identificatore lungo e praticamente unico, *tenant* è ciascun cliente di un sistema che ne serve diversi contemporaneamente, il *middleware* è il codice che gestisce ogni richiesta prima dell’applicazione, e SHA-1 e SHA-256 sono funzioni di hash, che trasformano dei dati in un codice di lunghezza fissa.

L’*harness*, il programma che fa lavorare il modello come agente, è Claude Code, versione 2.1.319: ogni prova è un’esecuzione non interattiva di esso in modalità di pianificazione, una modalità in cui l’agente può leggere il repository e non può modificarlo (Anthropic, 2026). L’esecuzione parte da un clone nuovo del *commit* fissato, in una cartella di configurazione isolata, con ogni variabile d’ambiente ereditata il cui nome inizia con CLAUDE o ANTHROPIC rimossa. ASP e i cinque *plugin* sincronizzati dall’account dell’operatore sono stati disattivati; i quattro *plugin* integrati in Claude Code sono rimasti attivi in tutte le prove, in entrambe le condizioni (Flores, 2026b). I due

modelli sono Claude Opus 5.5 e Claude Sonnet 5.5. Le 240 prove sono state messe in un unico ordine tramite una permutazione con seme, in modo che condizioni e modelli risultino intercalati nel tempo, e sono state eseguite tre alla volta.

4.4. Rischi di riferimento, scritti alla cieca

La lista di riferimento di un compito è l'insieme di rischi rispetto al quale si valuta ogni risposta di quel compito, e le dodici liste insieme formano il metro di misura dello studio. L'hanno scritta tre modelli di altri fornitori: Grok, GPT-OSS 120B e Gemini 3.8 Flash (Tabella 3). Ciascuno ha letto una copia nuova del repository e la richiesta di modifica, sotto un unico *prompt* che chiede da sei a dodici modi concreti in cui la modifica può andare storto in quella base di codice (Appendice A). Nessuno ha visto le condizioni, il protocollo né alcun output dell'agente provato. Hanno prodotto 369 elementi. Un giudice ha poi raggruppato gli elementi che nominano lo stesso problema, e un rischio è entrato nella lista di riferimento quando almeno due dei tre revisori lo avevano citato. Sono rimasti 100 rischi di riferimento, tra 6 e 10 per compito; 31 sono stati citati da tutti e tre i revisori e 69 da due (l'Appendice B mostra una lista intera).

Qui i modelli fanno le veci di ingegneri indipendenti, una sostituzione che le linee guida per gli studi con LLM chiedono agli autori di dichiarare e delimitare (Baltes et al., 2027). Il protocollo la dichiara come un indebolimento. Il *prompt* dei revisori condivide inoltre parole chiave con la frase provata, e la Sezione 7, obiezione 1, lo tratta come la principale minaccia al risultato.

Tabella 3
I ruoli che ciascun modello esterno svolge nello studio

Modello	Scrive una lista di revisore	Raggruppa gli elementi	Giudica la copertura	Giudica la rilevanza	Rischi di riferimento con la sua formulazione
Gemini 3.8 Flash	sì	sì	giudice principale	sì	91
GPT-OSS 120B	sì	no	no	no	9
Grok	sì	no	secondo giudice	no	0
GPT-6 Astra	no	no	terzo giudice, solo sul secondo metro di misura	no	0

Nota. Un rischio di riferimento porta la formulazione del primo revisore, in ordine alfabetico di nome del revisore, che lo ha scritto, ragione per cui nessuno porta la formulazione di Grok. Nessun modello esterno condivide il fornitore con i modelli provati. Gli output salvati del *client* di Grok non riportano alcun identificatore di modello. GPT-6 Astra è stato aggiunto dopo la revisione, come giudice non pianificato del secondo metro di misura, ed è dello stesso fornitore di GPT-OSS 120B (Sezione 5.8; Flores, 2026b).

4.5. Valutazione

La *copertura* è l'esito: il numero di rischi di riferimento che la tabella di una risposta copre, diviso per il numero di rischi di riferimento del compito. Il giudice principale, Gemini 3.8 Flash, riceve la richiesta di modifica, i rischi di riferimento e solo le righe della tabella della risposta (Appendice A). Per ogni rischio di riferimento, decide se qualche riga tratta lo stesso problema di fondo, anche se formulato diversamente. La Figura 6 mostra i verdetti del giudice per le due risposte della Figura 5. Al giudice non vengono comunicati né la condizione né il modello. Un elenco fisso di parole è stato sostituito da un segnaposto prima del giudizio; l'elenco è stato ereditato dallo studio di origine, in cui nascondeva il nome del protocollo ASP, e ha alterato 12 parole in 10 delle 240 tabelle di qui (Flores, 2026b).

Figura 6

Come vengono valutate le due risposte della Figura 5: il verdetto del giudice principale su ciascuno dei sei rischi di riferimento del compito

Rischio di riferimento (compito U08)	semplice t078	con la frase t051
1 I vecchi token smettono di validarsi dopo l'aggiornamento	✓	✓
2 Le due classi finiscono con valori predefiniti diversi	✗	✓
3 La chiave derivata cambia di valore e di dimensione	✗	✗
4 I test affermano output che coincidono con SHA-1	✓	✓
5 Le firme più lunghe sforano colonne, URL o cookie	✓	✓
6 La documentazione e i doctest dicono ancora SHA-1	✗	✓
Coperti	**3 su 6 = 50%**	**5 su 6 = 83%**

Tre misure secondarie sono state fissate insieme a essa (Flores, 2026b):

- **Secondo giudice.** Grok valuta lo stesso materiale allo stesso modo. La concordanza tra i due giudici è riportata a livello di decisioni singole; il κ di Cohen (Cohen, 1960), che scorpora la concordanza attesa per caso, è stato aggiunto in seguito a quanto riportato ed è etichettato come non pianificato (Sezione 7).
- **Comparatore lessicale.** Un comparatore deterministico per sovrapposizione di token, che confronta i testi in base ai token (parole o parti di parole) che hanno in comune, mantenuto dallo studio di origine, in cui si era già mostrato cieco alla parafrasi (Flores, 2026a).
- **Rilevanza.** La quota delle righe della tabella di una risposta che il giudice principale chiama rischio concreto e plausibile di questa modifica specifica. La rilevanza protegge da un guadagno vuoto, perché l'effetto del pre-mortem umano è di quantità (Mitchell et

al., 1989). Viene giudicata solo a partire dalla richiesta e dalla riga, quindi misura specificità e plausibilità, non correttezza.

I giudici LLM raggiungono oltre l'80% di concordanza con le preferenze umane in alcuni scenari (Zheng et al., 2023) e hanno distorsioni note, tra cui la preferenza per testo simile al proprio (Panickssery et al., 2024). Nessun giudice di questo studio condivide il fornitore con i modelli provati, quindi nessun giudice valuta risposte scritte da un modello del proprio fornitore, il caso in cui l'autopreferenza è documentata. Lo stesso non vale per i rischi di riferimento, la maggior parte dei quali porta la formulazione del giudice principale (Tabella 3; Sezione 7, obiezione 3).

4.6. Analisi e regola di decisione

La statistica di test è la media, sui 24 strati, della differenza tra la copertura media con la frase e la copertura media senza di essa. La sua significatività viene da un test di permutazione (Ernst, 2004): le etichette di condizione sono permutate all'interno di ciascuno strato 10.000 volte, e il valore p è la quota delle permutazioni la cui statistica è almeno quella osservata, contata come (successi + 1) / 10.001 perché non sia mai zero (Phipson e Smyth, 2010). Il test è unilaterale e non suppone alcuna distribuzione per la copertura. Permutare all'interno degli strati tiene le differenze tra compiti e tra modelli fuori dal confronto (Ernst, 2004).

In simboli, la statistica D è la media dell'Equazione 1, dove S è il numero di strati e d_s è la differenza, nello strato s, tra la copertura media con la frase e la copertura media senza di essa (Flores, 2026b).

$$D = \frac{1}{S}\sum_{s=1}^{S} d_s \tag{1}$$

Il valore p segue l'Equazione 2, dove B è il numero di permutazioni e b è il numero di quelle la cui statistica è almeno il D osservato (Phipson e Smyth, 2010). Con B = 10.000, il valore più piccolo che il test può restituire è 1/10.001.

$$p = \frac{b + 1}{B + 1} \tag{2}$$

La regola di decisione ha una condizione per "confermato": valore p combinato sotto ,05 *e* lo stesso test all'interno di ciascun modello, sui suoi dodici strati, sotto ,05 per entrambi i modelli (Flores, 2026b). Se passasse un solo modello, il risultato sarebbe dichiarato confermato solo per quel modello. La regola richiedeva anche almeno 18 strati completi. Nessun'altra analisi è confermativa, nel senso di Nosek et al. (2018).

Si è fissato in anticipo che un errore in una prova conta come copertura zero e rilevanza zero e non viene ripetuto, tranne un errore di limite d'uso, che interrompe la raccolta (Flores, 2026b). La Sezione 5.5 esamina che cosa ha fatto quella regola.

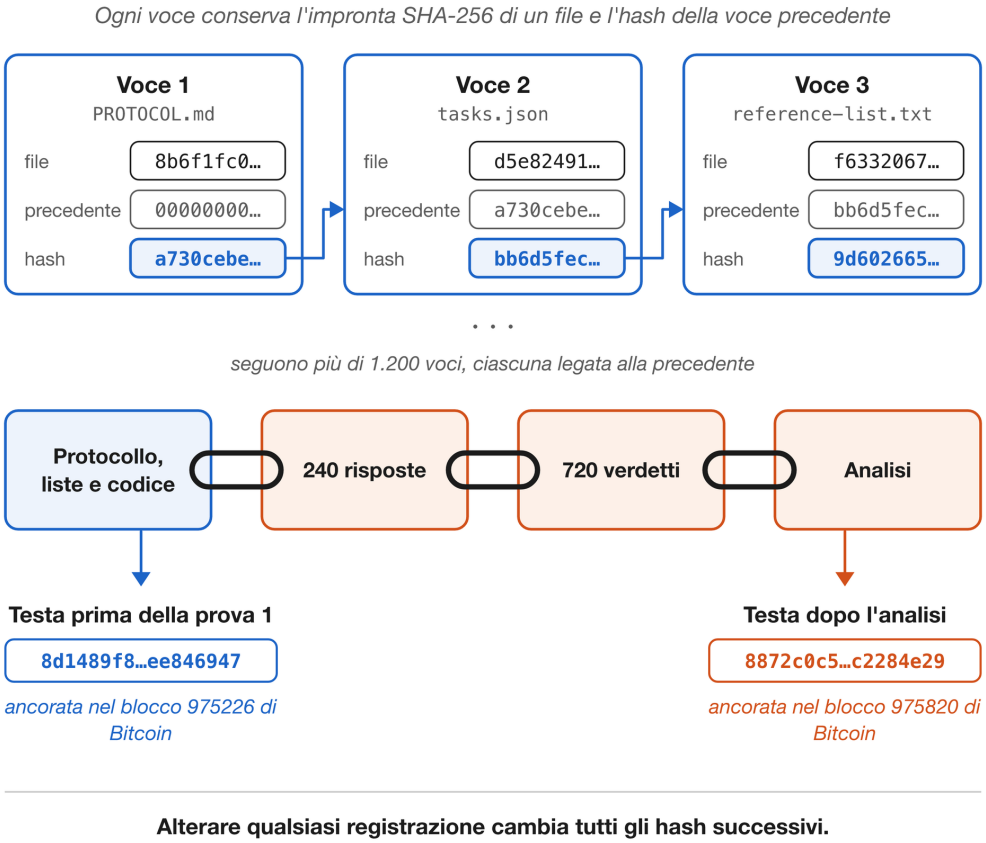
4.7. Pre-registrazione e registro di integrità

Pre-registrazione qui significa che il protocollo, i compiti, i *prompt*, le liste di riferimento e il codice di analisi sono stati registrati in *commit* di un repository pubblico e hanno ricevuto una marca temporale prima della prima prova contata (Nosek et al., 2018). Le integrazioni successive elencate nella Tabella 4 sono state fissate ciascuna in un'aggiunta, un testo datato annesso al protocollo e sigillato allo stesso modo prima delle chiamate che copre. Un registro concatenato tramite hash (*ledger*) lega ogni file alla base dei risultati riportati (gli undici verdetti parziali della Sezione 8 ne restano fuori): ogni voce conserva l'impronta SHA-256 di un file e della voce precedente, in modo che il registro non possa essere riscritto senza cambiare la sua ultima voce, chiamata testa (Flores, 2026b). Un file corretto dopo essere stato registrato riceve una nuova voce,

e quella vecchia resta nella catena. Un'impronta SHA-256 è un codice di 64 caratteri calcolato a partire da un file; cambiare un byte del file cambia il codice. La Figura 7 mostra le tre voci iniziali del registro con le loro impronte reali e le tappe che la catena chiude.

Figura 7

La catena di hash del registro dello studio: le tre voci iniziali, le due teste con marca temporale e le tappe tra di esse



La testa del registro prima della prova 1 ha ricevuto una marca temporale di OpenTimestamps, servizio che ancora un'impronta nella *blockchain* di Bitcoin, e il *commit* è stato salvato in Software Heritage, un archivio pubblico di codice sorgente. La Tabella 4 elenca ogni evento sigillato con il registro che ne prova l'ordine, e la Figura 8 mostra la sequenza fino all'analisi pre-registrata. Il sigillo e la prima prova distano 21 secondi sull'orologio dell'operatore: il *commit* del sigillo è delle 16:08:23 e la prima risposta della prima prova è delle 16:08:44. Quest'ordine di secondi poggia su

queste registrazioni locali; l'ancora in Bitcoin e l'archivio mostrano che i file sigillati esistevano quando quelle prove sono state fatte, non che abbiano preceduto la prima prova di secondi. Le 240 risposte sono state pubblicate prima che esistesse qualsiasi verdetto, e tutti i verdetti sono stati aggiunti al registro prima che l'analisi venisse eseguita (Flores, 2026b).

Tabella 4
Il registro di integrità dello studio

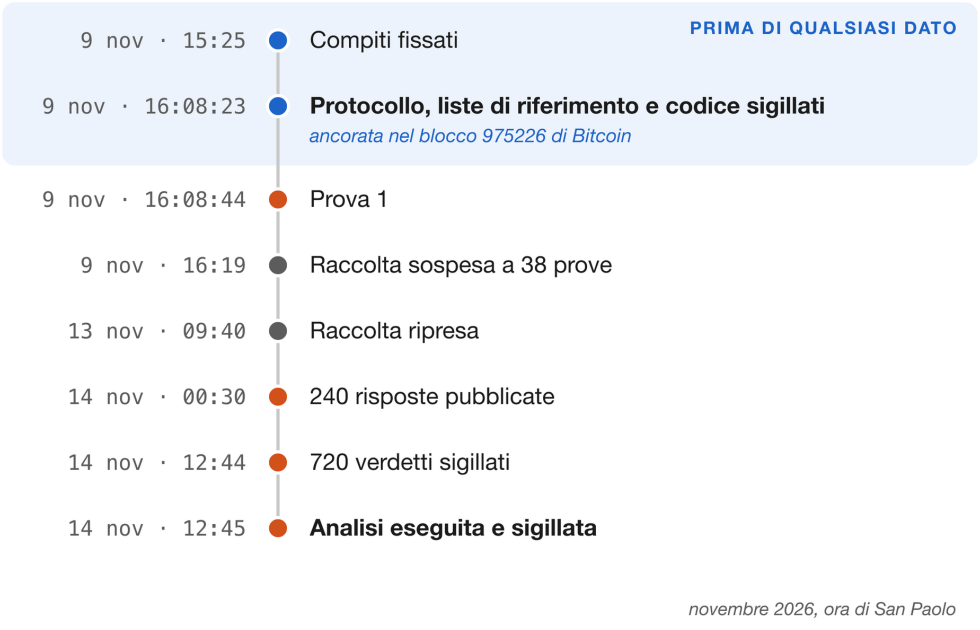
Evento	**Data, ora di San Paolo**	**Registro**
Compiti fissati, prima di qualsiasi lista di riferimento	9 nov 2026, 15:25	*Commit* eb1a447
Protocollo, liste di riferimento, *prompt* e codice sigillati	9 nov 2026, 16:08:23	*Commit* fe18438; testa del registro 8d1489f8…ee846947; attestazione di OpenTimestamps ancorata nel blocco 975226 di Bitcoin; revisione di Software Heritage fe184385434ee3d433104549d054f8c98ad6 9b32
Prima risposta della prima prova contata, t091	9 nov 2026, 16:08:44	Primo orario in runs/envelopes/t091.jsonl.gz
Le 240 risposte pubblicate, prima di qualsiasi verdetto	14 nov 2026, 00:30	*Commit* e046f36
I 720 verdetti sigillati, prima dell'analisi	14 nov 2026, 12:44	*Commit* 67b52d0
Analisi pre-registrata eseguita e sigillata	14 nov 2026, 12:45	*Commit* e8c73c1; testa del registro 8872c0c5…c2284e29; attestazione di OpenTimestamps ancorata nel blocco 975820 di Bitcoin
Secondo metro di misura: aggiunta, *prompt* e codice sigillati prima che le sue liste fossero scritte	14 nov 2026, 16:25	*Commit* 573b8bf e fa3d0ba; testa del registro cf0a7bfd…0499f47f; attestazione di OpenTimestamps ancorata nel blocco 975834 di Bitcoin

Evento	Data, ora di San Paolo	Registro
Seconda aggiunta (ripetizione e terzo giudice) sigillata prima della prima chiamata	15 nov 2026, 11:19	*Commits* 0acd091 e 2fb3c8b; testa del registro 73e275d1…58b42f11; prova OpenTimestamps inviata
Terza aggiunta (giudice tramite API a pagamento) sigillata prima della prima chiamata di valutazione	15 nov 2026, 14:23	*Commits* 2b2af4c e 2c5f5eb; testa del registro 2a9ac120…4f6dce4c; prova OpenTimestamps inviata

Nota. I *commit* sono nel repository di Flores (2026b). Le teste del registro sono abbreviate; i valori completi sono in benchmark/study-2026-11b/pretrial-head.txt e analysis-head.txt, accanto ai loro file di attestazione .ots. Le attestazioni si possono leggere con ots info e verificare contro un nodo Bitcoin con ots verify; la terza testa è in yardstick2/head-before-lists.txt. Le teste delle due aggiunte successive sono in addendum2/head-before-addendum2.txt e addendum3/head-before-addendum3.txt; le loro prove erano state inviate e non erano ancora ancorate quando è stata scritta questa versione.

Figura 8

Ordine degli eventi sigillati dello studio, dal file dei compiti all'analisi (asse ordinale, non in scala)



5. Risultati

5.1. Raccolta

Le 240 prove sono state registrate, in un'unica versione di Claude Code, e nessuna ha alterato un file del progetto (Flores, 2026b). Tre prove sono terminate con un errore di rete e non hanno lasciato tabella (Sezione 5.5). I 24 strati sono completi, quindi la regola di decisione della Sezione 4.6 si applica (Tabella 5).

5.2. Risultato principale

La frase ha portato la copertura dal 46,1% al 57,5% secondo il giudice principale (Tabella 5). La statistica pre-registrata, la differenza media sui 24 strati, è di 11,4 punti percentuali, con $p < ,001$: il valore è 1/10.001, il minore che il test può restituire (Phipson e Smyth, 2010). Nessuna delle 10.000 permutazioni ha raggiunto la differenza osservata; la maggiore ha dato 5,9 punti (Figura 9). All'interno di Claude Opus 5.5 la differenza è di 10,4 punti e all'interno di Claude Sonnet 5.5 è di 12,4 punti, ciascuna con $p < ,001$. La differenza è positiva in 22 dei 24 strati, zero in uno e negativa in uno (Tabella 6). La regola di decisione restituisce *confermato*.

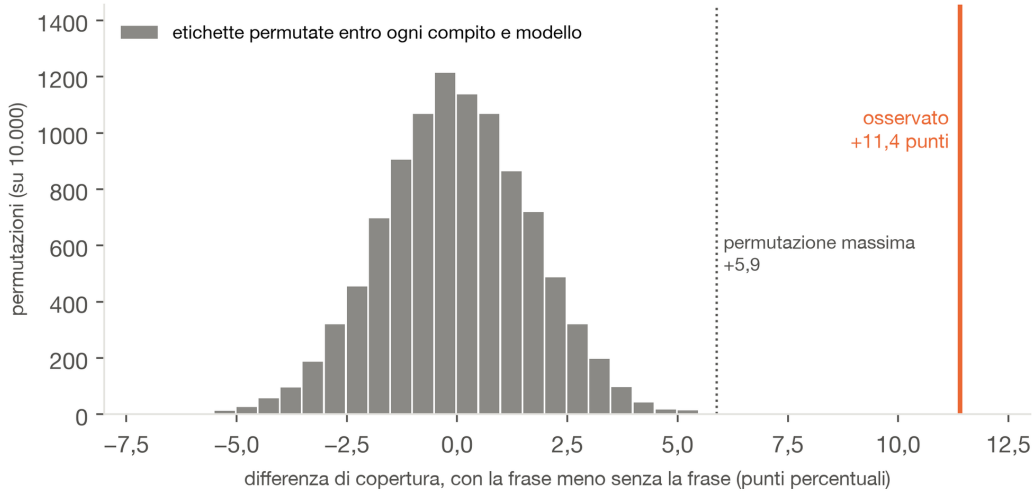
Tabella 5
Copertura dei rischi per modello e giudice

Modello	Semplice (giudice principale)	Frase (giudice principale)	Semplice (secondo giudice)	Frase (secondo giudice)
Claude Opus 5.5	43%	54%	48%	57%
Claude Sonnet 5.5	49%	61%	52%	64%
Entrambi	46%	58%	50%	61%

Nota. 60 prove per modello e condizione. Il giudice principale è Gemini 3.8 Flash e il secondo giudice è Grok. Le percentuali sono arrotondate a numeri interi; le differenze nel testo derivano dai valori non arrotondati.

Figura 9

Il test di permutazione del risultato principale: le 10.000 differenze permutate e quella osservata



Un *bootstrap* per percentili, che ricalcola la differenza su molti ricampionamenti casuali dei dati, non era previsto dal protocollo ed è riportato solo come descrizione. Ricampionando le repliche all'interno di ciascuna cella, dà un intervallo al 95% da 8,9 a 14,0 punti per la differenza combinata, da 6,8 a 14,1 per Opus 5.5 e da 8,8 a 16,2 per Sonnet 5.5 (Flores, 2026b). L'intervallo riflette la variazione tra esecuzioni su questi dodici compiti, non tra compiti.

La Figura 10 e la Tabella 6 mostrano il risultato compito per compito. Il guadagno non dipende da pochi compiti. L'unico strato negativo è la modifica dello *slug* in Go con Sonnet 5.5 (U02, −5 punti), e lo strato a zero è l'esportazione dei dati con Sonnet 5.5 (U06), in cui le due condizioni coprivano già il 73% (Tabella 6).

Figura 10

Quota dei rischi di riferimento ciechi coperta dalla tabella finale dei rischi, per compito e modello, con e senza la frase (giudice principale; legenda come nella Figura 3)

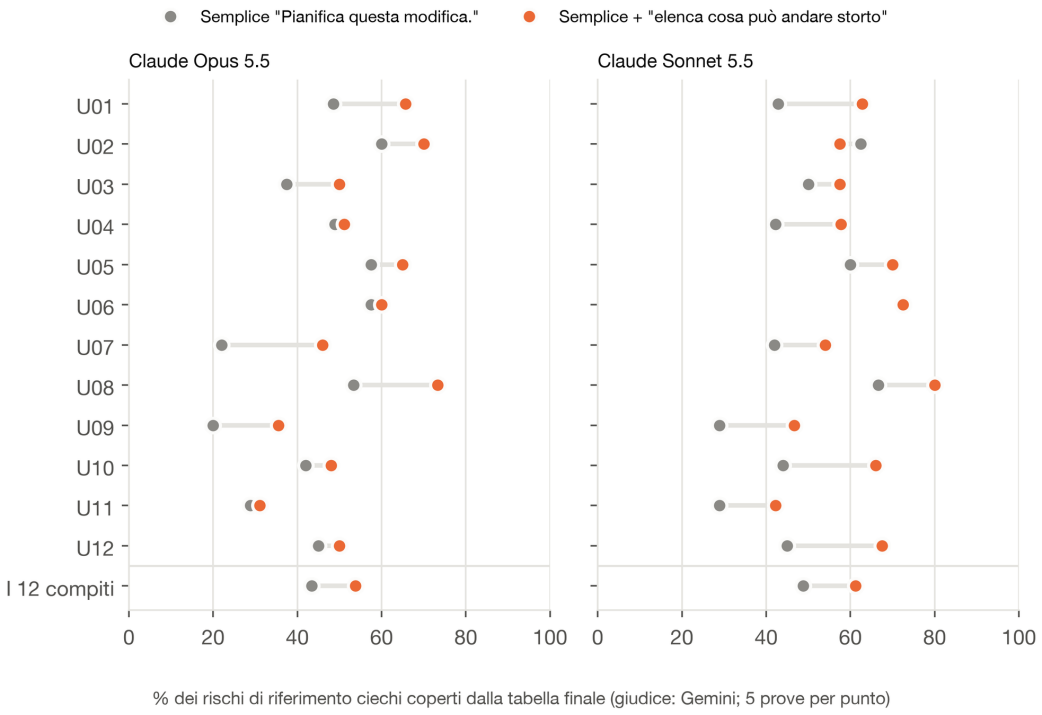


Tabella 6

Copertura dei rischi per compito e modello, giudice principale, in percentuale

Compito	Rischi di riferimento	Opus 5.5 semplice	Opus 5.5 frase	Sonnet 5.5 semplice	Sonnet 5.5 frase
U01	7	49	66	43	63
U02	8	60	70	63	58
U03	8	38	50	50	58
U04	9	49	51	42	58
U05	8	58	65	60	70
U06	8	58	60	73	73
U07	10	22	46	42	54
U08	6	53	73	67	80
U09	9	20	36	29	47
U10	10	42	48	44	66

Compito	Rischi di riferimento	Opus 5.5 semplice	Opus 5.5 frase	Sonnet 5.5 semplice	Sonnet 5.5 frase
U11	9	29	31	29	42
U12	8	45	50	45	68

Nota. Ogni cella è la media di cinque prove. Tre celle semplici contengono una prova che è terminata con un errore di rete e conta come zero: U07 e U09 in Opus 5.5 e U10 in Sonnet 5.5. Nelle altre quattro prove, quelle celle sono 28, 25 e 55.

5.3. Secondo giudice

Il secondo giudice ha dato 50,0% senza la frase e 60,8% con essa: differenza media di 10,7 punti sugli strati ($p < ,001$), 9,3 punti all'interno di Opus 5.5 e 12,1 all'interno di Sonnet 5.5 (ciascuno con $p < ,001$; Tabella 5), positiva in 22 dei 24 strati anche secondo questo giudice (Flores, 2026b). Segna un po' più rischi come coperti del giudice principale in entrambe le condizioni e arriva alla stessa decisione.

I due giudici hanno preso 2.000 decisioni ciascuno, una per rischio di riferimento per prova, e hanno concordato nel 93,2% di esse. Il κ di Cohen, introdotto nella Sezione 4.5 (Cohen, 1960), è ,86. Di quelle decisioni, 29 riguardano le tre prove senza tabella, in cui i due giudici potevano solo rispondere "non coperto"; nelle altre 1.971 la concordanza è del 93,1% e il κ è ,86. Il comparatore lessicale ha segnato 3,0% e 1,3%; come nello studio di origine, non coglie le corrispondenze parafrasate e qui non è usato come prova di copertura (Flores, 2026a).

5.4. Rilevanza

Nelle prove che hanno prodotto una tabella, la quota di righe che il giudice principale ha chiamato concrete e plausibili è stata del 71,1% senza la frase e del 73,9% con essa (Tabella 7). Secondo la regola fissata, che valuta come zero le tre prove senza tabella, la cifra semplice è del 69,4%. Non

si è osservata una riduzione della rilevanza media giudicata, che è la lettura fissata nel protocollo. Lo studio non fissa un margine di non inferiorità, il calo massimo di rilevanza che conterebbe ancora come assenza di perdita, per cui questo non prova che il guadagno di copertura sia arrivato senza righe più vaghe, il rischio che indica la letteratura umana (Mitchell et al., 1989). Non si afferma un aumento della rilevanza: la differenza combinata di 4,5 punti ha p = ,011 secondo la regola fissata, ma nessuno dei modelli arriva da solo a ,05 (,052 e ,060), e la differenza non è significativa quando si mettono da parte le prove con errore (Tabella 7).

Le tabelle avevano 9,2 righe senza la frase e 9,6 con essa, contro un tetto di dieci (Tabella 8). La copertura è una misura di *recall*, la quota delle voci di riferimento che una risposta trova, quindi una tabella più lunga ha più probabilità di coincidere con un rischio di riferimento. Restringere il confronto alle prove la cui tabella ha le dieci righe complete, 56 semplici e 82 con la frase, lascia la differenza a 11,3 punti sui 17 strati in cui le due condizioni restano (Flores, 2026b). La lunghezza della tabella è essa stessa un effetto della richiesta, per cui questo confronto ristretto descrive e non isola una causa; mostra che il guadagno è presente tra tabelle della stessa lunghezza.

5.5. Sensibilità alle tre prove con errore

Durante una breve perdita di accesso a internet, tre prove sono terminate con errore dopo dieci nuovi tentativi e non hanno prodotto tabella (Flores, 2026b). Secondo la regola fissata in anticipo, contano come zero (Sezione 4.6). Tutte e tre appartengono alla condizione semplice, quindi la regola sposta la differenza principale verso l’ipotesi. Il protocollo non lo aveva previsto, e una verifica etichettata come non pianificata ripete l’analisi in due modi (Tabella 7).

Tabella 7
L’analisi principale con e senza le tre prove con errore di rete

Analisi	Strati	Giudice principale	Secondo giudice	Rilevanza
Come pre-registrata (gli errori contano zero)	24	+11,4 (p < ,001)	+10,7 (p < ,001)	+4,5 (p = ,011)
Senza i tre strati che hanno un errore	21	+10,1 (p < ,001)	+8,9 (p < ,001)	+2,5 (p = ,091)
Solo senza le tre prove	24	+10,5 (p < ,001)	+9,7 (p < ,001)	+2,9 (p = ,054)

Nota. Differenze in punti percentuali, frase meno semplice. Solo la prima riga è confermativa.

Il risultato di copertura si mantiene in tutte le righe della Tabella 7, e all’interno di ciascun modello: nella riga più conservativa, le differenze per modello sono di 8,5 punti per Opus 5.5 (p < ,001) e di 11,6 per Sonnet 5.5 (p < ,001) secondo il giudice principale, e di 6,9 (p < ,001) e 10,7 (p < ,001) secondo il secondo. La differenza di rilevanza non si mantiene al livello di ,05 senza le prove con errore, ragione per cui la Sezione 5.4 afferma soltanto che non si è osservata una riduzione della rilevanza.

Il limite opposto dà la stessa lettura. Se ciascuna delle tre prove con errore viene valutata come se la sua tabella avesse coperto tutti i rischi di riferimento, la differenza è di 8,9 punti secondo il giudice principale e di 8,2 secondo il secondo (Flores, 2026b).

Le tre prove sono state anche eseguite un’altra volta, due giorni dopo la raccolta, come aggiunta non pianificata, fissata in un’aggiunta datata prima delle esecuzioni (Sezione 8). Ciascuna è terminata normalmente, con una tabella di 8, 9 e 10 righe. Con la loro copertura al posto dei tre zeri, la differenza sulle liste originali è di 10,5 punti (p < ,001): 9,0 su Opus 5.5 e 11,9 su Sonnet 5.5 (Flores, 2026b). Le righe fissate in anticipo non cambiano, e la Sezione 5.8 riporta i limiti di questa ripetizione.

5.6. Costo

Il tempo medio di esecuzione è stato minore con la frase, come i turni e i token di output, in entrambi i modelli (Tabella 8). Il protocollo elenca queste misure come descrittive e non assegna loro alcun test, quindi sono riportate come descrizione (Flores, 2026b). Sostengono una sola affermazione: non si è osservato alcun costo aggiuntivo.

Una differenza di comportamento sta dietro una parte di questi numeri. Nella modalità di pianificazione, l'agente può salvare il piano in un file, con una chiamata a uno strumento, prima di rispondere (Anthropic, 2026). In Opus 5.5 lo ha fatto in 59 delle 60 prove semplici e in 22 delle 60 prove con la frase; in Sonnet 5.5 non lo ha fatto mai, in nessuna condizione (Flores, 2026b). La Sezione 7, obiezione 13, esamina se quella differenza potrebbe spiegare il guadagno di copertura. Le due prove di Opus 5.5 terminate con un errore di rete (Sezione 5.5) avevano già salvato un file di piano quando la connessione è caduta, e quella di Sonnet 5.5 no. Una prova semplice su Opus 5.5 è durata in media un turno in più (Tabella 8) ed è rimasta più a lungo esposta all'interruzione.

Tabella 8

Costo medio per prova, per condizione e modello, nelle prove che non sono terminate con errore

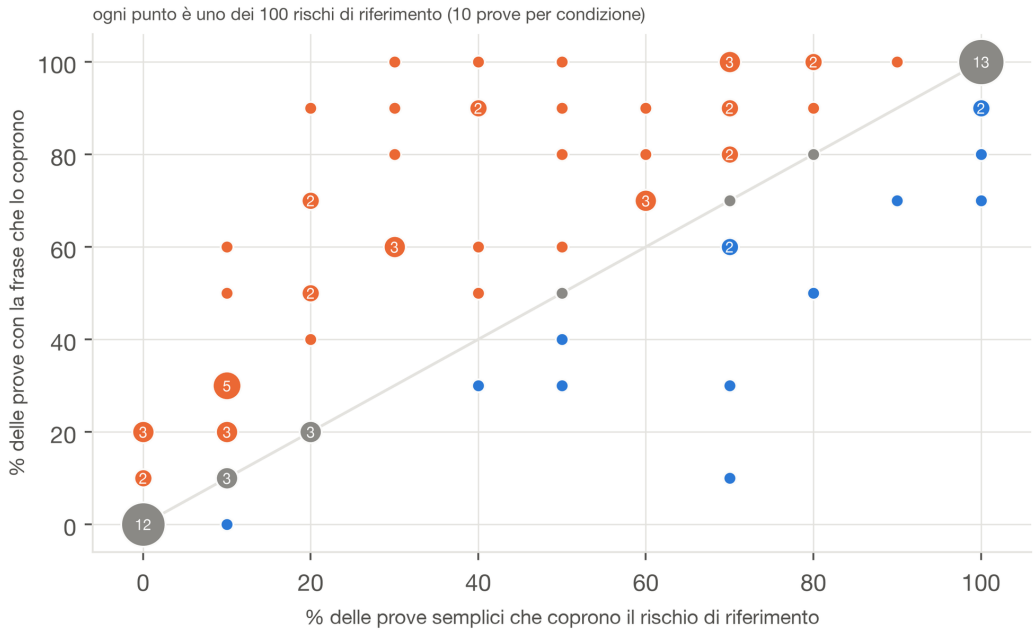
Modello	Condizione	Secondi	Turni	Token di output	Righe della tabella
Claude Opus 5.5	Semplice	66,5	5,8	5.801	9,2
Claude Opus 5.5	Frase	53,5	4,8	4.736	9,4
Claude Sonnet 5.5	Semplice	32,9	3,5	3.014	9,3
Claude Sonnet 5.5	Frase	27,3	3,2	2.469	9,7
Entrambi	Semplice	49,6	4,7	4.395	9,2
Entrambi	Frase	40,4	4,0	3.603	9,6

Nota. Un turno è uno scambio tra l'agente e i suoi strumenti o il modello. 117 prove semplici e 120 prove con la frase.

5.7. Quali rischi aggiunge la frase (esplorativo)

Questa sezione non era pianificata e non prova nulla. Esamina i 100 rischi di riferimento uno per uno, secondo il giudice principale, per descrivere da dove viene il guadagno (Figura 11). Nella figura, un punto arancione sopra la diagonale è un rischio coperto più volte con la frase, un punto grigio su di essa è un rischio coperto lo stesso numero di volte e un punto blu sotto di essa è un rischio coperto meno volte; il numero dentro un punto conta i rischi in quella posizione.

Figura 11
Ogni rischio di riferimento secondo la quota di prove che lo coprono, senza e con la frase (giudice principale)



- **Ampiezza.** 52 rischi di riferimento sono stati coperti più spesso con la frase, 34 con la stessa frequenza e 14 meno spesso. Ogni rischio è stato valutato su dieci prove per condizione, quindi ciascuna di queste differenze è di almeno dieci punti (Figura 11).
- **Affidabilità.** Il numero di rischi di riferimento che la maggior parte delle prove ha coperto è salito da 41 a 56. La frase ha trasformato soprattutto rischi che l'agente citava a volte in rischi che citava per lo più (Figura 11).

- **Punti ciechi che restano.** 12 rischi di riferimento non sono stati coperti da nessuna prova in nessuna delle condizioni, e 13 da nessuna prova con la frase. Sono soprattutto conseguenze ristrette, a livello di file, come un indice composto di database assente o un formato di archiviazione in cui non starebbe un nuovo schema di hash (Flores, 2026b).

I maggiori guadagni sono rischi di interazione tra la modifica e ciò che già esiste: una rotta di registrazione esistente che fa accedere subito l'utente e così aggira la nuova verifica dell'e-mail (U01, dal 30% al 100% delle prove), righe esistenti che restano senza valore quando si aggiunge una nuova colonna di identificatore (U07, dal 30% al 90%) e un generatore sincrono di hash di password che un algoritmo a uso intensivo di memoria bloccherebbe (U09, dal 20% al 90%). Le maggiori perdite sono in U02 e U03, in rischi che il piano semplice citava già nella maggior parte delle prove (Flores, 2026b). L'Appendice B mostra la stessa lettura per una lista intera.

5.8. Un secondo metro di misura, con altre parole (esplorativo)

Questa verifica è stata specificata dopo che i risultati precedenti erano noti e non mette alla prova nulla di pre-registrato. Risponde all'obiezione 1 della Sezione 7: le liste di riferimento sono state chieste con parole vicine a quelle della frase messa alla prova. I tre revisori hanno scritto liste nuove per i dodici compiti, con un *prompt* in cui è cambiata una frase. Chiede "the main risks of this change" (i principali rischi di questa modifica), che sono le parole del contratto di output che le due condizioni ricevono, e le parole "go wrong" (andare storto) non compaiono (Flores, 2026b). La regola che fonde le liste, le 240 risposte, il *prompt* del giudice e la statistica sono rimasti com'erano. L'aggiunta, il *prompt* e il codice sono stati sigillati nel registro e marcati temporalmente prima di chiamare il primo revisore (Tabella 4).

Le liste nuove contengono 111 rischi di riferimento, tra 7 e 11 per compito. Rispetto a esse il giudice principale dà 55,8% senza la frase e 62,5% con essa: una differenza media di 6,7 punti ($p < ,001$), 6,5 su Opus ($p = ,002$) e 6,9 su Sonnet ($p = ,001$). Dei 24 strati, 16 sono positivi, 2 sono zero e 6 sono negativi (Tabella 9).

Tabella 9

Copertura dei rischi rispetto ai due metri di misura, giudice principale

Metro di misura, secondo le parole del suo *prompt*	**Rischi di riferimento**	**Senza la frase**	**Con la frase**	**Differenza**	**p**	**Strati positivi**
Originale: “ways this change could go wrong” (modi in cui questa modifica può andare storta)	100	46,1%	57,5%	+11,4	< ,001	22 di 24
Secondo: “the main risks of this change” (i principali rischi di questa modifica)	111	55,8%	62,5%	+6,7	< ,001	16 di 24

Nota. Le differenze sono in punti percentuali, come media sui 24 strati di compito per modello; p viene dal test di permutazione stratificato della Sezione 4.6. La seconda riga è esplorativa ed è stata calcolata dopo che la prima era nota.

La differenza resta positiva quando la richiesta ai revisori non usa più le parole della frase, cosa che l’aggiunta, sigillata prima che le liste fossero scritte, leggeva come un guadagno non legato a quella formulazione (Flores, 2026b). Ciò non esclude un effetto della formulazione nei passi successivi che costruiscono il metro di misura, descritti più avanti. La differenza è anche più piccola. 6,7 punti sta sotto l’intervallo *bootstrap* della stima originale, da 8,9 a 14,0 (Sezione 5.2), e l’effetto è meno uniforme tra i compiti (Tabella 9). Con le liste nuove, la richiesta semplice guadagna circa dieci punti di copertura e la richiesta con la frase ne guadagna cinque. L’obiezione 1 è quindi accolta in parte.

I dati non dicono quanta parte della distanza tra 11,4 e 6,7 venga dalle parole. Questa è una sola nuova raccolta di liste; i due insiemi di liste differiscono per dimensione e per contenuto, e una

copertura iniziale più alta lascia meno spazio di guadagno (Tabella 9). Tre limiti valgono per questa verifica. La richiesta che fonde le tre liste è rimasta com'era e le descrive ancora come "ways a software change could go wrong"; la verifica toglie quindi quelle parole dalla richiesta ai revisori, e non dall'intera costruzione del metro di misura (Flores, 2026b). Gemini ha scritto la formulazione di 98 dei 111 rischi di riferimento ed è stato il suo unico giudice nell'analisi pianificata, la concentrazione che descrive l'obiezione 3 della Sezione 7. Il secondo giudice pianificato ha raggiunto il suo limite d'uso dopo 11 di 240 verdetti; l'aggiunta vieta di sostituire un giudice, per cui i suoi verdetti non sono in questa versione (Sezione 8).

Altre tre verifiche di questo metro di misura sono state chieste nella revisione cieca di questo manoscritto e aggiunte dopo che il suo risultato era noto; nessuna è nell'aggiunta (Sezione 8). Le tre prove con errore di rete sono tutte prove semplici e anche qui contano zero. Senza i tre strati che le contengono la differenza è di 5,3 punti ($p < ,001$), e senza le tre prove è di 5,6 ($p < ,001$). Al limite opposto, con ogni prova con errore valutata come copertura totale, è di 4,2 punti ($p = ,005$): 5,3 su Sonnet 5.5 ($p = ,010$) e 3,2 su Opus 5.5, dove non è più significativa ($p = ,099$). La lettura fissata nell'aggiunta poggia sul test congiunto e regge con ogni trattamento; il risultato isolato di Opus 5.5 non regge a quel limite (Flores, 2026b). L'intervallo *bootstrap* della seconda differenza va da 4,1 a 9,4 punti e si sovrappone all'intervallo originale, da 8,9 a 14,0. Ricampionare i dodici compiti, e non le repliche, lo allarga a 2,2 a 11,1, contro 8,2 a 14,5 per il metro di misura originale.

Due aggiunte sono seguite al secondo giro di revisione, ciascuna fissata in un'aggiunta datata, sigillata e marcata temporalmente prima della prima chiamata (Sezione 8). Primo, le tre prove con errore sono state eseguite un'altra volta, un tentativo ciascuna, con la stessa richiesta, lo stesso *commit* fissato, lo stesso identificatore di modello e la stessa versione di Claude Code; tutte e tre

sono terminate normalmente (Sezione 5.5). Secondo, GPT-6 Astra, modello di un terzo fornitore, ha valutato le 240 risposte e le tre ripetizioni rispetto al secondo metro di misura, con la richiesta del giudice invariata, come giudice non pianificato (Tabella 10). Non sostituisce il secondo giudice pianificato, i cui verdetti restano dovuti (Flores, 2026b).

Tabella 10

La differenza sul secondo metro di misura con tre trattamenti delle prove con errore, per giudice

Trattamento delle tre prove con errore	Giudice	Differenza	Opus 5.5	Sonnet 5.5
Contano zero, come fissato in anticipo	Gemini 3.8 Flash	+6,7 (p < ,001)	+6,5 (p = ,002)	+6,9 (p = ,001)
Contano zero, come fissato in anticipo	GPT-6 Astra	+9,0 (p < ,001)	+8,1 (p < ,001)	+9,8 (p < ,001)
Valutate come copertura totale	Gemini 3.8 Flash	+4,2 (p = ,005)	+3,2 (p = ,099)	+5,3 (p = ,010)
Valutate come copertura totale	GPT-6 Astra	+6,5 (p < ,001)	+4,7 (p = ,013)	+8,2 (p < ,001)
Eseguite un’altra volta	Gemini 3.8 Flash, tramite API	+5,6 (p < ,001)	+4,9 (p = ,009)	+6,4 (p = ,002)
Eseguite un’altra volta	GPT-6 Astra	+7,7 (p < ,001)	+6,2 (p < ,001)	+9,1 (p < ,001)

Nota. Differenze in punti percentuali, frase meno semplice, con il p di permutazione della Sezione 4.6. Tutte le righe sono esplorative. Nella quinta riga i verdetti di Gemini sulle tre ripetizioni sono arrivati tramite un’API a pagamento, non tramite il *client* a riga di comando dei 240 verdetti sigillati.

Tutte le righe della Tabella 10 sono positive, e il test congiunto resta sotto ,05 in ciascuna. Secondo GPT-6 Astra la differenza con la regola fissata è di 9,0 punti, dal 64,0% al 73,0%, con 20 strati positivi su 24, e resta significativa su ciascun modello al limite opposto, dove il risultato isolato di Opus 5.5 secondo Gemini non reggeva. Con le prove eseguite un’altra volta, i due giudici danno una differenza significativa su entrambi i modelli. I due giudici concordano sull’89,4% di 2.192 decisioni singole (κ di Cohen = ,77), e GPT-6 Astra segna più voci come coperte, il 68,8% contro il 59,0% (Flores, 2026b).

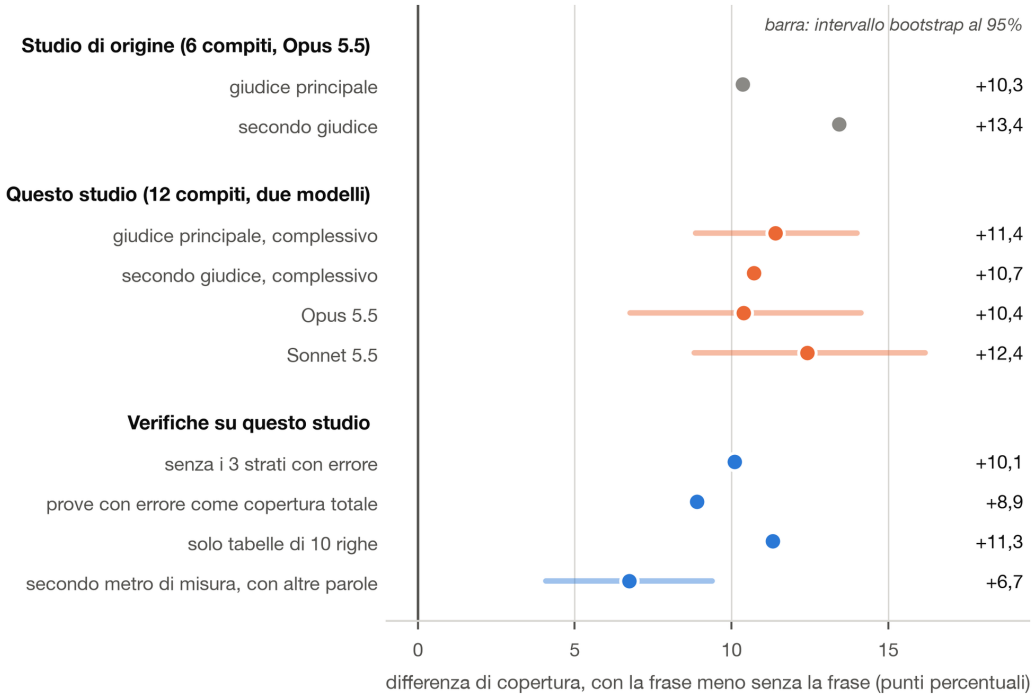
Queste aggiunte hanno limiti propri. La ripetizione è stata fatta due giorni dopo la raccolta, da un operatore non in cieco rispetto al risultato, e nulla di osservabile prova che il modello dietro un identificatore non sia cambiato; il modello della sessione, i modelli usati e i *plugin* che le tre sessioni dichiarano sono quelli delle prove sigillate. GPT-6 Astra è dello stesso fornitore di GPT-OSS 120B, che ha scritto una delle tre liste dei revisori. Il giudice principale ha raggiunto il limite d'uso sulla riga di comando dopo 4 dei suoi 9 verdetti sulle ripetizioni; per decisione dell'autore, registrata in una terza aggiunta, i 9 sono stati prodotti dallo stesso modello tramite un'API a pagamento invece di attendere il rinnovo del limite, un'ulteriore deviazione. Un campione di 30 prove sigillate, fissato da un seme e valutato tramite quel canale, concorda con i verdetti sigillati sul 98,2% di 281 decisioni ($\kappa = ,96$), con una copertura media del 56,0% contro il 56,4%. I verdetti della riga di comando saranno riportati accanto a questi quando il limite sarà rinnovato (Flores, 2026b).

6. Discussione

Ciò che è stato confermato. Chiedere a un agente di programmazione i modi in cui una modifica può andare storto, prima che pianifichi, fa coprire alla sua lista finale di rischi più di quanto si aspettavano tre revisori indipendenti (Tabella 5). L'effetto si è replicato in un secondo studio, con dodici compiti nuovi e un secondo modello, a una dimensione vicina a quella dello studio di origine: +10 punti là (Tabella 1), +11 qui. Rispetto a un secondo metro di misura, chiesto con altre parole, il guadagno è minore, 6,7 punti, e ancora positivo (Sezione 5.8). È modesto in termini assoluti: l'agente continua a non citare circa quattro rischi di riferimento su dieci. La Figura 12 mette fianco a fianco le stime principali della differenza; le aggiunte successive sono nella Tabella 10.

Figura 12

Le stime principali della differenza di copertura: dello studio di origine, di questo studio e delle sue verifiche



Una lettura possibile. Le due condizioni terminano con la stessa richiesta di una tabella di rischi, quindi all'agente viene chiesto del rischio in entrambe. Ciò che differisce è che i modi di fallimento sono richiesti prima e con altre parole. Questo si accorda con la spiegazione offerta per InvThink, secondo cui l'enumerazione porta alla luce conoscenza che una richiesta diretta lascia inutilizzata (Kim et al., 2026), e con il risultato esplorativo che la frase soprattutto rende regolari rischi occasionali (Figura 11). Lo studio non mette alla prova questa lettura. Un'ablazione (una ripetizione con un solo elemento cambiato) che spostasse la stessa frase a dopo il piano la metterebbe alla prova.

Il compromesso tra procedura e copertura. Nello studio di origine, un protocollo a 13 fasi ha coperto meno rischi di riferimento nella sua tabella finale di questa frase, con tre volte e mezzo il tempo di esecuzione (Tabella 1). Più procedura ha comprato meno copertura. I due studi non

mostrano che la struttura sia inutile; le *pipeline* multi-agente aggiungono modi di fallimento propri (Cemri et al., 2025). Mostrano che un protocollo strutturato va misurato contro la frase, che è la linea di base economica, e che essere migliore di una richiesta semplice non basta.

I piani non sono codice. Un *prompt* che chiede a un agente di programmazione di prevedere i casi limite ha aumentato il tasso congiunto di compiti risolti che evitano un difetto successivo in un modello Claude e l'ha ridotto in un altro (Patel et al., 2026). Un piano che nomina un rischio non garantisce codice che lo tratti, e nulla nella Tabella 5 dice il contrario.

Per la pratica. L'intervento è una frase all'inizio di una richiesta di pianificazione, e non si è osservato alcun costo aggiuntivo in queste esecuzioni (Tabella 8). Un team che usa un agente di programmazione in modalità di pianificazione può adottarla senza strumenti aggiuntivi. Ciò che il team non deve concludere è che il piano risultante sia sicuro: la sua tabella ha nominato il 58% dei rischi di riferimento, il lavoro umano di revisione che resta non è stato misurato, e i dodici rischi che nessuna prova ha nominato (Sezione 5.7) ricordano che i punti ciechi dell'agente e l'attenzione dei suoi revisori non si sovrappongono del tutto.

7. Obiezioni e minacce alla validità

Questa sezione prende le obiezioni una per una. Seguono le classi abituali di minacce nell'ingegneria del software empirica (Wohlin et al., 2012) e le cautele sollevate per gli studi che usano LLM (Sallou et al., 2024; Baltes et al., 2027). Prima che questa versione fosse scritta, il manoscritto e i suoi dati sono stati consegnati a due modelli di altri fornitori, ciascuno istruito, in modo indipendente, ad agire da revisore ostile; entrambi hanno sollevato l'obiezione 1. A uno di essi è stata poi data la frase stessa in studio, rivolta al manoscritto: prima di pubblicare questo

articolo, elenca i modi in cui può andare storto. Quella richiesta ha prodotto l'obiezione 13, che nessuna delle due revisioni aveva sollevato. Il repository conserva i tre rapporti (Flores, 2026b).

1. Il metro di misura è stato ottenuto con le parole della frase provata (costrutto). Ai revisori che hanno scritto le liste di riferimento è stato indicato di dare "the concrete ways this change could go wrong" (Appendice A). La condizione provata chiede all'agente di "list the ways this change could go wrong", e la condizione semplice non vede mai quelle parole. Parte del guadagno potrebbe quindi essere una coincidenza tra le risposte provate e la domanda che ha prodotto il metro di misura, e non più dei rischi reali della modifica. Le due condizioni sono valutate rispetto alle stesse liste, il che protegge il confronto da differenze tra liste; non lo protegge da un trattamento che conduce l'agente verso la cornice dei revisori. Nulla nei dati pre-registrati separa le due letture. La misura di rilevanza, che non usa le liste di riferimento, non mostra una riduzione della rilevanza media giudicata delle righe (Sezione 5.4), e questa è solo una risposta parziale. Un secondo metro di misura, ottenuto con le parole del contratto di output che le due condizioni ricevono, è stato specificato dopo che questi risultati erano noti, in un'aggiunta datata sigillata prima della sua raccolta (Flores, 2026b). Rispetto a esso la differenza è di 6,7 punti ($p < ,001$), positiva su entrambi i modelli e circa quattro decimi più piccola della stima pre-registrata (Sezione 5.8). Con gli altri trattamenti delle prove con errore e secondo un giudice di un terzo fornitore va da 4,2 a 9,0 punti, e solo il suo valore separato su Opus 5.5, al limite opposto, non è significativo (Tabella 10). L'obiezione è accolta in parte: il guadagno non scompare quando la richiesta ai revisori smette di usare le parole della frase, anche se la richiesta che fonde le liste mantiene parole affini (Sezione 5.8), e la sua dimensione dipende invece dal metro di misura.

2. Le liste di riferimento sono state scritte da modelli (costrutto). La copertura misura la concordanza con ciò che tre LLM di altri fornitori si aspettavano che andasse storto, filtrata sugli

elementi che almeno due di loro hanno citato (Sezione 4.4). Non è una misura di rischio reale. Un rischio che i tre revisori hanno lasciato passare non può essere accreditato. Nemmeno un rischio che solo uno di loro ha nominato: la fusione conserva un rischio quando almeno due revisori lo nominano, il che lascia fuori 132 dei 232 gruppi delle liste originali e 102 dei 213 della seconda (Flores, 2026b). Questo filtro può favorire i rischi che i modelli tendono a ripetere rispetto a quelli visti da un solo revisore, e nessun umano ha verificato i gruppi lasciati fuori. Nell'esperimento sul campo di Caraeni et al. (2026), i pianificatori umani hanno colto rischi specifici del contesto che il modello da solo non ha colto, quindi un metro di misura umano potrebbe cambiare i valori assoluti. Il passo (i) della Sezione 9 se ne occupa. La copertura dà inoltre lo stesso peso a ogni rischio di riferimento, qualunque sia la gravità del guasto che nomina.

3. Un modello occupa quattro ruoli (costrutto). Gemini 3.8 Flash ha scritto una lista di revisore, ha raggruppato gli elementi, è il giudice principale della copertura e l'unico giudice della rilevanza, e 91 dei 100 rischi di riferimento portano la sua formulazione (Tabella 3). Un giudice può far coincidere più facilmente la propria formulazione (Panickssery et al., 2024). Il secondo giudice è il controllo per questo: Grok non ha scritto alcuna formulazione di riferimento, e trova un guadagno di 10,7 punti contro 11,4 (Tabella 7). La rilevanza non ha un tale controllo, il che è una seconda ragione per affermare soltanto che non si è osservata una riduzione. Sul secondo metro di misura, con 98 rischi su 111 formulati da Gemini, un giudice di un terzo fornitore trova un guadagno maggiore (Tabella 10).

4. L'esito è la tabella finale (costrutto). Il giudice vede solo le righe della tabella. Un agente che elenca prima i modi di fallimento può semplicemente copiare quella lista nella tabella, e lo studio non ha misurato se il piano stesso tratti i rischi, se i rischi siano veri per la base di codice, né se il codice scritto a partire dal piano abbia meno difetti. Lo studio più vicino con esito sul codice ha

trovato risultati che dipendono dal modello (Patel et al., 2026). L'affermazione riguarda ciò che la tabella dei rischi del piano nomina.

5. Il progetto non separa posizione da formulazione (interna). La frase differisce dalla richiesta semplice in ciò che chiede e in quando lo chiede (Appendice A). La Sezione 6 offre l'ordine come lettura e non la mette alla prova. L'ablazione è il passo (iii) della Sezione 9.

6. Le tre prove con errore favoriscono l'ipotesi (interna). Tutte e tre cadono nella condizione semplice e contano come zero secondo la regola fissata. Senza di esse il risultato regge (Tabella 7; Sezione 5.5).

7. Tabelle più lunghe potrebbero spiegare un *recall* maggiore (interna). Le tabelle con la frase hanno in media 0,4 righe in più (Tabella 8). Tra le prove con le dieci righe complete, la differenza è di 11,3 punti (Sezione 5.4).

8. Il giudice potrebbe percepire la condizione (interna). Al giudice non viene comunicata la condizione, e l'elenco di parole sostituito prima del giudizio non è stato pensato per queste due condizioni (Sezione 4.5). Un giudice potrebbe inferire la condizione dallo stile di una tabella. Lo studio non ha una misura di questo. Due giudici di fornitori diversi concordano nel 93,2% delle decisioni (Sezione 5.3), il che delimita il rumore del giudice e non una distorsione che possano condividere. Il giudice decide inoltre tutti i rischi di riferimento di una risposta in un'unica chiamata, quindi quelle decisioni non sono indipendenti (Flores, 2026b).

9. I repository sono pubblici e potrebbero essere nei dati di addestramento dei modelli (esterna). Diversi compiti usano progetti di esempio largamente copiati (Tabella 2). L'agente può ricordare trappole note di quei progetti, e la frase può aiutare di più il recupero che il ragionamento.

Le due condizioni usano gli stessi repository, quindi questo riguarda il significato dell'effetto e non la sua esistenza; valgono le cautele sulla fuga di dati negli studi con LLM (Sallou et al., 2024).

10. Due modelli di un fornitore sono una prova debole di generalità (esterna). Claude Opus 5.5 e Claude Sonnet 5.5 condividono fornitore, *harness* e modalità. Nello studio di Patel et al. (2026), un *prompt* di previsione ha spostato due modelli di una stessa famiglia in sensi opposti, quindi la concordanza tra i due modelli qui è informativa; non dice nulla su modelli di un altro fornitore, un altro agente, basi di codice private o una sessione interattiva.

11. La frase è mezzo pre-mortem (costrutto). L'esercizio di Klein include la supposizione che il piano sia fallito (Klein, 2007). La frase la omette, e la condizione con la cornice completa di Klein, che il protocollo permetteva come esplorativa, non è stata eseguita (Sezione 8). Questo articolo usa "pre-mortem" per la famiglia dell'istruzione e non fa alcuna affermazione sulla supposizione di fallimento.

12. Tempo e autore (interna). La raccolta è stata sospesa per quattro giorni e ripresa nella stessa versione di Claude Code; poiché l'ordine era una permutazione con seme, le due condizioni si distribuiscono nei due periodi (Sezione 8). L'autore ha scelto i compiti conoscendo il risultato precedente, li ha fissati prima che le liste di riferimento esistessero (Tabella 4) e mantiene ASP, lo strumento il cui protocollo questa frase ha superato (Flores, 2026a).

13. In uno dei modelli, le condizioni differiscono in una chiamata a uno strumento (interna). In Opus 5.5, la richiesta semplice ha portato l'agente a salvare un file di piano in 59 prove su 60, contro 22 su 60 con la frase (Sezione 5.6). Un agente che ha scritto il piano in un file può dare una risposta finale più corta, e la tabella è valutata a partire dalla risposta finale. Sonnet 5.5 è il controllo: non ha salvato alcun file di piano in nessuna prova, e il suo guadagno è il maggiore dei due, 12,4 punti (Tabella 5). Tra le prove di Opus 5.5 che hanno salvato il file, la differenza è di 7,5

punti nei 10 strati in cui le due condizioni restano (Flores, 2026b). Salvare il file è anch'esso una risposta alla richiesta, per cui questi confronti descrivono e non isolano una causa. Mostrano che il guadagno compare senza la chiamata allo strumento, in Sonnet 5.5; la chiamata può spiegare parte del costo minore in Opus 5.5.

14. Tre limiti indicati nella revisione (costrutto). Con la frase l'agente ha usato meno turni (Tabella 8), per cui può aver letto meno del repository prima di rispondere, e questo studio non misura quanto i rischi elencati poggino sul codice. La tabella ha un tetto di dieci righe e le due condizioni lo riempiono quasi, per cui la misura è quali rischi entrano nei primi dieci, e non quanti rischi l'agente saprebbe nominare. Il giudice decide tutti i rischi di riferimento di una risposta in una sola chiamata e in ordine fisso, per cui un effetto di posizione sulle sue decisioni non è escluso (Flores, 2026b).

Validità della conclusione. Un'ipotesi principale porta la decisione, tramite una regola congiunta: il test combinato e il test all'interno di ciascun modello devono tutti scendere sotto ,05 (Sezione 4.6); le misure secondarie sono riportate come tali, e ogni analisi non pianificata è etichettata come non pianificata (Sezione 8), la separazione che la pre-registrazione esiste per proteggere (Nosek et al., 2018). Cinque repliche per cella danno medie di strato grossolane; il test combina 24 strati e non dipende da nessuno di essi (Tabella 6). I test trattano i dodici compiti come fissi, e i due modelli li condividono; ricampionare compiti interi dà da 8,2 a 14,5 punti per la differenza pre-registrata e da 2,2 a 11,1 per il secondo metro di misura (Sezione 5.8). Ogni valore di p e ogni intervallo qui è condizionato alle liste di riferimento, alla loro fusione e ai verdetti così come raccolti; nessuno include la variazione che aggiungerebbero una nuova richiesta di liste, una nuova fusione o un nuovo giudizio, e la distanza tra i due metri di misura e tra i due giudici del secondo mostra che questa variazione non è piccola.

8. Deviazioni dal protocollo

Il file del protocollo registra ogni deviazione con la sua data (Flores, 2026b). In sintesi:

1. **Un conflitto di cartella corretto nei primi minuti.** Le prove numerate da t001 a t090 sono fallite nella fase di clonazione, prima di qualsiasi chiamata al modello, perché la cartella delle copie dei repository conservava ancora copie dello studio precedente. Le prove t091 a t096, le cui cartelle non esistevano, sono state eseguite prima e sono state mantenute. Il nome della cartella è stato cambiato e la raccolta è ripresa nell'ordine con seme. *Prompt*, modelli, compiti, valutazione e analisi non sono cambiati.
2. **Una pausa di quattro giorni.** La raccolta si è fermata dopo 38 prove, quando il computer che la eseguiva è stato spento. Le prove in corso in quel momento non hanno lasciato registrazione e sono state rieseguite alla ripresa. Il loro numero e le loro condizioni non sono stati registrati, per cui una selezione per durata tra le prove in corso in quel momento non può essere misurata né esclusa. Il computer ha aggiornato Claude Code durante la pausa; le prove rimanenti sono state eseguite nella versione delle prime 38, installata a parte.
3. **Percorsi di macchina rimossi da tre risposte.** Le risposte t006, t022 e t023 hanno nominato un file del computer dell'operatore con il suo percorso completo, fuori dalla tabella dei rischi. Il percorso è stato sostituito da un segnaposto, ogni file corretto ha ricevuto una nuova voce nel registro e, dalle prove riprese in poi, l'esecutore applica gli stessi segnaposto prima di salvare.
4. **Tre prove con errore di rete,** trattate come fissato in anticipo ed esaminate nella Sezione 5.5.

5. **Una pausa nella valutazione.** Il giudice principale ha raggiunto il suo limite d'uso dopo 225 verdetti su 240. I 15 rimanenti sono stati prodotti dallo stesso modello, con lo stesso comando, dopo il rinnovo del limite. Nessun giudice è stato sostituito.

6. **Una condizione esplorativa che non è stata eseguita.** Il protocollo permetteva, se avanzava uso, una terza condizione con la cornice completa di Klein: "Imagine it is a month from now and this change has failed. Before planning, list why it failed." (in italiano: "Immagina che sia passato un mese e che questa modifica sia fallita. Prima di pianificare, elenca perché è fallita.") Non ha prove e non ha alcun rapporto con il risultato.

7. **Analisi non pianificate,** ciascuna etichettata dove compare: l'analisi di sensibilità e il suo limite opposto, l'intervallo *bootstrap*, il κ di Cohen, il confronto di tabelle complete, il confronto del file di piano e la descrizione per elemento.

8. **Il secondo metro di misura,** esplorativo e specificato dopo il risultato (Sezione 5.8). Il giudice principale ha raggiunto il limite d'uso dopo 208 dei suoi 240 verdetti, ripresi dopo il rinnovo con lo stesso modello e lo stesso comando. Il secondo giudice lo ha raggiunto dopo 11 verdetti, e il limite non era stato rinnovato quando è stata scritta questa versione. Tre verifiche vi sono state aggiunte in seguito, su richiesta di un revisore (Sezione 5.8).

9. **Una ripetizione delle tre prove con errore e un terzo giudice,** non pianificati e fissati in una seconda aggiunta prima di ogni chiamata (Sezioni 5.5 e 5.8). Raggiunto il limite d'uso a riga di comando del giudice principale dopo 4 dei 9 verdetti sulle ripetizioni, l'autore ha deciso, in una terza aggiunta sigillata prima delle chiamate, di produrre i 9 con lo stesso modello tramite un'API a pagamento, con una verifica di canale su 30 prove

sigillate. Quell'aggiunta diceva che la risposta completa dell'API sarebbe stata salvata con ogni verdetto; il codice ha salvato il testo della risposta, il modello, il fornitore, il conteggio dei token e l'identificatore della chiamata. I verdetti a riga di comando restano dovuti.

9. Conclusione

Una frase che apre una richiesta di pianificazione al posto di "Plan this change.", "Before planning, list the ways this change could go wrong.", ha portato dal 46% al 58% la quota di rischi elencati in modo indipendente che il piano di un agente di programmazione copre, in uno studio confermativo pre-registrato di 240 prove, nei due modelli provati e secondo due giudici (Tabella 5). I rischi elencati non sono stati giudicati meno concreti, e non si è osservato alcun costo aggiuntivo (Sezione 5.4; Tabella 8). Il risultato replica un risultato secondario di uno studio precedente su compiti nuovi e su un secondo modello (Flores, 2026a).

I limiti sono quelli della Sezione 7: il metro di misura e i giudici erano modelli, il metro di misura è stato ottenuto con parole vicine a quelle della frase, e un secondo metro di misura, ottenuto con altre parole, dà un guadagno minore, di 6,7 punti (Sezione 5.8), i due agenti condividono il fornitore, e l'esito è ciò che dice la tabella di un piano, non ciò che fa il codice. I passi successivi ne derivano, nell'ordine previsto: (i) un ingegnere umano che valuti un campione delle liste di riferimento e delle decisioni dei giudici, e rischi di riferimento tratti da difetti reali, che nessun *prompt* ha provocato; (ii) lo stesso progetto in un modello di un altro fornitore; (iii) un'ablazione che separi l'elencare i rischi *prima di pianificare* dall'elencarli del tutto, e aggiunga la supposizione di fallimento di Klein (Klein, 2007); e (iv) un esito sul codice, implementando i piani delle due condizioni ed eseguendo i test dei progetti, come fanno Patel et al. (2026) con un *prompt* più lungo.

10. Disponibilità di dati e codice

Il materiale dietro ogni numero di questo articolo è nel repository pubblico anticipating-shadow-points (Flores, 2026b), nella cartella benchmark/study-2026-11b del ramo v2, dal *commit* d880cf0 in poi; la *release* 2.0.0, la versione citata, è precedente a questo studio. La Tabella 11 descrive la cartella.

Un solo comando, python run_all.py, ricalcola ogni risultato dello studio riportato in questo articolo a partire dalle risposte e dai verdetti conservati e non chiama alcun modello; richiede Python 3.11 o successivo (Flores, 2026b). Rieseguire le prove richiede un abbonamento a Claude e l'accesso ai due modelli giudici, che possono essere cambiati da novembre 2026.

Tabella 11
Ciò che il repository contiene per questo articolo

Materiale	**Dove nel repository**	**Che cosa permette di verificare**
Protocollo, deviazioni e le tre aggiunte	PROTOCOL.md, yardstick2/, addendum2/, addendum3/	Che cosa era fissato prima dei dati e che cosa è cambiato
Compiti e *prompt*	tasks.json, run.py, prompts/	Il testo esatto che ogni prova e ogni revisore hanno ricevuto
Liste di riferimento e loro fusione	lists/	I 369 elementi dei revisori e i 100 rischi di riferimento
Le 240 risposte e i loro registri	runs/	Ogni risposta e il suo registro
I 720 verdetti	judge/	Ogni decisione, con il suo *prompt*
Analisi e verifiche	analyze.py, sensitivity.py, describe.py; tests/test_paper3_numbers.py	Come si calcola ogni numero, e che il testo concorda con i dati
Registro concatenato e marche temporali	ledger.jsonl, *.ots	Che ogni file corrisponde all'ultima registrazione, e l'ordine degli eventi sigillati, nei limiti indicati nella Sezione 4.7
Revisioni esterne del manoscritto	docs/paper/review/	I rapporti dietro la Sezione 7
Lo studio di origine	benchmark/study-2026-11/	Lo stesso materiale per le 90 prove della Sezione 3

Nota. I percorsi senza cartella davanti stanno dentro benchmark/study-2026-11b/ (Flores, 2026b). Dai registri di esecuzione sono stati rimossi i percorsi di macchina.

11. Dichiarazioni

Conflitto di interessi. L'autore mantiene ASP, il *plugin* il cui protocollo lo studio di origine ha confrontato con la frase provata, e dirige il laboratorio che lo pubblica. Il risultato riportato qui è sfavorevole a quel protocollo (Flores, 2026a).

Finanziamento. Questo lavoro non ha ricevuto finanziamenti esterni. I costi d'uso dei modelli sono stati pagati dall'autore.

Contributo dell'autore. Autore unico: ideazione, disegno, raccolta dei dati, analisi, stesura e revisione.

Uso di strumenti di IA. I modelli linguistici hanno quattro ruoli in questo studio, tutti descritti nella Sezione 4: gli agenti provati, gli autori delle liste di riferimento, i giudici e i revisori ciechi del manoscritto (Flores, 2026b). Un assistente di IA è stato usato anche nella programmazione, nell'analisi e nella stesura e traduzione del testo, sotto la direzione dell'autore. L'autore ha verificato ogni numero sui dati e risponde del contenuto.

Riferimenti bibliografici

Anthropic. (2026). *Choose a permission mode* [Software documentation]. https://code.claude.com/docs/en/permission-modes

Baltes, S., Angermeir, F., Arora, C., Muñoz Barón, M., Chen, C., Böhme, L., Calefato, F., Ernst, N., Falessi, D., Fitzgerald, B., Fucci, D., He, J., Treude, C., Kalinowski, M., Lambiase, S., Russo, D., Lungu, M., Martinez Montes, C., Prechelt, L., Ralph, P., van Tonder, R., & Wagner, S. (2027). Guidelines for empirical studies in software engineering involving large language models. *Empirical Software Engineering, 32*(2), Article 64. https://doi.org/10.1007/s10664-026-10922-3

Caraeni, A., Shick, A., & Lan, A. (2026). *Cognitive offloading in agile teams: How artificial intelligence reshapes risk assessment and planning quality*. arXiv:2604.13814. https://arxiv.org/abs/2604.13814

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., & Stoica, I. (2025). Why do multi-agent LLM systems fail? *Advances in Neural Information Processing Systems, 38*, 135676–135729. https://doi.org/10.52202/085713-4082

Cheng, Z., & Mastropaolo, A. (2026). *An empirical study on the effects of system prompts in instruction-tuned models for code generation*. arXiv:2602.15228. https://arxiv.org/abs/2602.15228

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*(1), 37–46. https://doi.org/10.1177/001316446002000104

Ernst, M. D. (2004). Permutation methods: A basis for exact inference. *Statistical Science, 19*(4), 676–685. https://doi.org/10.1214/088342304000000396

Flores, C. U. (2026a). *ASP: An operational pre-mortem skill for LLM coding agents — An experience report* (Version 2.4) [Technical report]. Zenodo. https://doi.org/10.5281/zenodo.24065968

Flores, C. U. (2026b). *ASP — Anticipating Shadow Points: Claude Code plugin and replication package* (Version 2.0.0; branch v2, commit d880cf0) [Computer software]. GitHub. https://github.com/ulissesflores/anticipating-shadow-points https://doi.org/10.5281/zenodo.20276631

Fröhling, L., Giaconia, A., Bogucka, E. P., & Quercia, D. (2026). Agent-supported foresight for AI systemic risks: AI agents for breadth, experts for judgment. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems* (pp. 1–48). ACM. https://doi.org/10.1145/3772318.3790712

International Electrotechnical Commission. (2019). *Risk management — Risk assessment techniques* (IEC 31010:2019, Edition 2.0). https://webstore.iec.ch/en/publication/59809

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024). *SWE-bench: Can language models resolve real-world GitHub issues?* ICLR 2024. https://arxiv.org/abs/2310.06770

Kharma, M., Sabbah, A., Alkhanafseh, M., Hammoudeh, M., & Mohaisen, D. (2026). *An empirical evaluation of LLM-generated code security across prompting methods*. arXiv:2605.24298. https://arxiv.org/abs/2605.24298

Kim, Y., Kim, T., Park, E., Park, C., Breazeal, C., McDuff, D., & Park, H. W. (2026). *InvThink: Premortem reasoning for safer language models* (Version 3). arXiv:2510.01569. https://arxiv.org/abs/2510.01569

Klein, G. (2007). Performing a project premortem. *Harvard Business Review, 85*(9), 18–19. https://hbr.org/2007/09/performing-a-project-premortem

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems, 35*. https://arxiv.org/abs/2205.11916

Mitchell, D. J., Russo, J. E., & Pennington, N. (1989). Back to the future: Temporal perspective in the explanation of events. *Journal of Behavioral Decision Making, 2*(1), 25–38. https://doi.org/10.1002/bdm.3960020103

Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences, 115*(11), 2600–2606. https://doi.org/10.1073/pnas.1708274114

Panickssery, A., Bowman, S. R., & Feng, S. (2024). LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems, 37*. https://doi.org/10.52202/079017-2197

Patel, S., Liao, M., Henderson, P., He, H., & Chen, V. (2026). *Do coding agents catch their own bugs? (They don't.)*. arXiv:2611.01181. https://arxiv.org/abs/2611.01181

Phipson, B., & Smyth, G. K. (2010). Permutation p-values should never be zero: Calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology, 9*(1), Article 39. https://doi.org/10.2202/1544-6115.1585

Sallou, J., Durieux, T., & Panichella, A. (2024). Breaking the silence: The threats of using LLMs in software engineering. In *Proceedings of the 2024 ACM/IEEE 44th International*

Conference on Software Engineering: New Ideas and Emerging Results (pp. 102–106). ACM. https://doi.org/10.1145/3639476.3639764

Schoenegger, P., Jones, C. R., Tetlock, P. E., & Mellers, B. (2025). *Prompt engineering large language models' forecasting capabilities*. arXiv:2506.01578. https://arxiv.org/abs/2506.01578

Sclar, M., Choi, Y., Tsvetkov, Y., & Suhr, A. (2024). *Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting*. ICLR 2024. https://arxiv.org/abs/2310.11324

Tony, C., Díaz Ferreyra, N. E., Mutas, M., Dhif, S., & Scandariato, R. (2025). Prompting techniques for secure code generation: A systematic investigation. *ACM Transactions on Software Engineering and Methodology, 34*(8), Article 225. https://doi.org/10.1145/3722108

Veinott, B., Klein, G. A., & Wiggins, S. (2010). Evaluating the effectiveness of the PreMortem technique on plan confidence. In *Proceedings of the 7th International ISCRAM Conference*. https://web.archive.org/web/20200928015943/http://idl.iscram.org/files/veinott/2010/1049_Veinott_etal2010.pdf

Veinott, E. S., & Lehman, B. R. (2024). Adaptive planning: Comparing human and AI responses in premortem planning. In *HCI International 2024 – Late breaking papers* (pp. 256–268). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-76827-9_15

Wang, H., Cao, Y., Lin, L., & Chen, J. (2026). *PreFlect: From retrospective to prospective reflection in large language model agents*. arXiv:2602.07187. https://arxiv.org/abs/2602.07187

Wang, H., Li, T., Deng, Z., Roth, D., & Li, Y. (2024). Devil's advocate: Anticipatory reflection for LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024* (pp. 966–978). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-emnlp.53

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). https://doi.org/10.18653/v1/2023.acl-long.147

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer. https://doi.org/10.1007/978-3-642-29044-2

Wu, J. (2026). *AssumptionMiner: Extracting, tracing, and revising implicit assumptions in LLM code generation*. arXiv:2607.22898. https://arxiv.org/abs/2607.22898

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. NeurIPS 2023 D&B. https://arxiv.org/abs/2306.05685

Appendice A — Le richieste esatte

La Tabella A1 dà le richieste principali dello studio confermativo; le richieste che fondono le liste e giudicano la rilevanza, e la seconda richiesta ai revisori, sono nei file elencati dalla Tabella 11, e ogni *client* aggiunge istruzioni proprie che non sono mostrate. Le parentesi graffe segnano le uniche parti che cambiano da una chiamata all'altra. I file che conservano questi testi sono elencati nella Tabella 11 (Flores, 2026b).

Tabella A1
Le richieste inviate all'agente, ai revisori e al giudice

Chi riceve	Testo
L'agente, condizione semplice	{change request} Plan this change. {output contract}
L'agente, condizione con la frase	{change request} Before planning, list the ways this change could go wrong. {output contract}
Il contratto di output, in entrambe le condizioni	Non-interactive run: state your assumptions instead of asking. Do not edit any file. Answer in English. End your answer with the main risks of this change as one markdown table with columns #, Risk, Why it bites here, Mitigation (at most 10 rows, short cells), then one line starting "Riskiest:" naming the riskiest one.
Ogni revisore	You are a senior software engineer reviewing a change request before anyone implements it. The repository in the current directory is the codebase. Read whatever files you need. Do not modify any file. Change request: {change request} List the concrete ways this change could go wrong in this codebase if it were implemented without care: the problems a careful senior engineer would want addressed before starting. Each item must be specific to this repository and this request, one or two sentences. Answer with a numbered list of 6 to 12 items and nothing else.
Il giudice della copertura	A plan for a software change ends with a table of risks. Decide, for each reference problem, whether any row of the table addresses the same underlying problem (names it as a risk or mitigates it specifically), even if worded differently. Change request: {change request} Reference problems: {numbered reference risks} Rows of the risk table: {rows} Answer directly from the text above; do not run commands or use tools. Answer with JSON only, one key per reference number.

Nota. Gli a capo sono stati rimossi. L'ultima frase del giudice è seguita, nell'originale, da un esempio del JSON atteso.

Appendice B — Una lista di riferimento completa

La Tabella B1 mostra la lista di riferimento del compito U08, la più breve delle dodici, con la quota delle prove di ciascuna condizione la cui tabella copre ogni rischio secondo il giudice principale. Illustra lo schema della Sezione 5.7: rischi che ogni piano cita in entrambe le condizioni (righe 1 e 5), rischi che la frase rende regolari (righe 4 e 6) e un rischio che le due condizioni lasciano quasi sempre passare (riga 2). Le liste complete sono nel repository (Flores, 2026b).

Tabella B1
I sei rischi di riferimento del compito U08 (cambiare il digest di firma predefinito da SHA-1 a SHA-256) e la loro copertura

#	Rischio di riferimento, abbreviato	Revisori che lo hanno citato	Semplice	Frase
1	I token firmati da versioni precedenti con le impostazioni predefinite falliscono la validazione dopo l'aggiornamento, invalidando le sessioni nei progetti dipendenti come Flask.	3	100%	100%
2	Cambiare il valore predefinito nel *signer* e non nella classe dell'algoritmo HMAC lascia i due valori predefiniti non sincronizzati.	3	10%	20%
3	La derivazione della chiave usa anch'essa il *digest*, quindi la chiave derivata cambia di valore e passa da 20 a 32 byte.	3	40%	50%
4	I test affermano output che coincidono con firme SHA-1.	2	70%	100%
5	Le firme passano da 27 a 43 caratteri e possono sforare colonne di database, limiti di URL o dimensioni dei *cookie*.	2	100%	100%
6	Gli esempi di documentazione e i *doctest* conservano output SHA-1 e affermano che SHA-1 è il valore predefinito.	2	40%	90%

Nota. Dieci prove per condizione (cinque per modello). I sei rischi portano la formulazione del revisore Gemini. Qui token è un valore firmato; il *digest* è la funzione di hash usata nella firma; il *signer* è la classe che firma; HMAC è un metodo standard per firmare con una chiave segreta e una funzione di hash; i *doctest* sono esempi della documentazione eseguiti come test.

Appendice C — I testi in inglese, tradotti

Ogni testo che un modello ha ricevuto in questo studio era in inglese, e l'articolo lo riproduce come è stato inviato. Questa appendice traduce quei testi per chi legge in italiano: la Tabella C1 riporta le dodici richieste di modifica della Tabella 2, e la Tabella C2 le richieste della Tabella A1. Le traduzioni sono dell'autore e non sono state usate in nessuna prova (Flores, 2026b).

Tabella C1
Le dodici richieste di modifica, tradotte

Compito	Richiesta di modifica, in italiano
U01	Richiedere che i nuovi utenti confermino il proprio indirizzo e-mail prima di poter accedere.
U02	Permettere agli autori di cambiare il titolo di un articolo, e con esso lo *slug* dell'URL, senza rompere i *link* già condivisi.
U03	Permettere ai moderatori di nascondere i commenti abusivi e all'autore del commento di fare ricorso entro 7 giorni.
U04	Bloccare gli utenti per la combinazione di nome utente e indirizzo IP, e non solo per l'indirizzo IP, come nuovo valore predefinito.
U05	Far rinnovare le sessioni per impostazione predefinita, in modo che ogni richiesta estenda la validità della sessione.
U06	Permettere a ogni utente di scaricare tutti i propri dati (profilo, post e commenti) in un unico file.
U07	Sostituire con UUID gli identificatori interi che l'API espone, mantenendo funzionanti gli attuali *client* dell'API.
U08	Cambiare l'algoritmo di firma predefinito da SHA-1 a SHA-256.
U09	Conservare le password con uno schema di hash più forte, senza obbligare gli utenti attuali a reimpostare la password.
U10	Far appartenere ogni lista di attività e ogni elemento a un'organizzazione (*tenant*), in modo che ogni azienda veda solo i propri dati.
U11	Permettere ai proprietari di animali di prenotare visite *online*, senza che due visite vengano prenotate per lo stesso veterinario alla stessa ora.
U12	Rendere il *cookie* del *middleware* di sessione Secure e SameSite=Strict per impostazione predefinita.

Tabella C2

Le richieste inviate all'agente, ai revisori e al giudice, tradotte

Chi riceve	Testo, in italiano
L'agente, condizione semplice	{richiesta di modifica} Pianifica questa modifica. {contratto di output}
L'agente, condizione con la frase	{richiesta di modifica} Prima di pianificare, elenca i modi in cui questa modifica potrebbe andare storto. {contratto di output}
Il contratto di output, in entrambe le condizioni	Esecuzione non interattiva: dichiara le tue ipotesi invece di fare domande. Non modificare alcun file. Rispondi in inglese. Termina la risposta con i principali rischi di questa modifica in una tabella markdown con le colonne #, Rischio, Perché pesa qui e Mitigazione (al massimo 10 righe, celle brevi) e, poi, una riga che inizi con "Riskiest:" e indichi il più rischioso.
Ogni revisore	Sei un ingegnere del software senior che rivede una richiesta di modifica prima che qualcuno la implementi. Il repository nella cartella corrente è la base di codice. Leggi i file di cui hai bisogno. Non modificare alcun file. Richiesta di modifica: {richiesta di modifica} Elenca i modi concreti in cui questa modifica potrebbe andare storto in questa base di codice se fosse implementata senza cura: i problemi che un ingegnere senior attento vorrebbe veder trattati prima di cominciare. Ogni elemento deve essere specifico di questo repository e di questa richiesta, di una o due frasi. Rispondi con una lista numerata da 6 a 12 elementi e nient'altro.
Il giudice della copertura	Un piano di modifica software termina con una tabella di rischi. Decidi, per ogni problema di riferimento, se qualche riga della tabella tratta lo stesso problema di fondo (lo nomina come rischio o lo mitiga in modo specifico), anche se formulato diversamente. Richiesta di modifica: {richiesta di modifica} Problemi di riferimento: {rischi di riferimento numerati} Righe della tabella dei rischi: {righe} Rispondi direttamente a partire dal testo qui sopra; non eseguire comandi né usare strumenti. Rispondi solo con JSON, una chiave per numero di riferimento.

Nota. I testi originali, in inglese, sono nella Tabella A1.

Nota Tecnica

Versione 1.0 · novembre 2026

Serie Codex Hash Research Report 2026-03

Licenza Testo, figure e dati: CC BY 4.0. Software complementare: MIT.

Il progetto ASP. ASP (Anticipating Shadow Points) è un *plugin open source* per Claude Code che fa elencare all'agente ciò che potrebbe andare storto prima di scrivere codice. Il *plugin* e il materiale di replicazione stanno in un unico repository pubblico (Flores, 2026b). Un *whitepaper* descrive come funziona il *plugin* e riporta la misurazione iniziale di se migliori i piani di un agente, che non ha trovato alcun beneficio misurabile rispetto a una richiesta di una frase (Flores, 2026a). Questo articolo è lo studio confermativo di quella frase che il *whitepaper* aveva annunciato.

DOI. Un DOI (Digital Object Identifier) è un collegamento permanente a un'opera depositata. Zenodo, archivio pubblico di ricerca gestito dal CERN, conserva questo articolo come record a sé; questa versione, 1.0, è https://doi.org/10.5281/zenodo.24410437. Il *plugin*, i dati e il codice formano un record separato (Flores, 2026b), il cui DOI concept, https://doi.org/10.5281/zenodo.20276631, apre sempre la *release* più recente.

Replicazione. Ogni numero che questo articolo calcola a partire dai suoi dati può essere ricalcolato dal repository senza chiamare un modello; la Sezione 10, Disponibilità di dati e codice, elenca ciò che c'è e come eseguirlo.

Sull'autore

Carlos Ulisses Flores è CTO e Ricercatore Capo presso il Codex Hash Research Laboratory, un laboratorio indipendente di R&S a San Paolo, in Brasile. Costruisce sistemi di produzione in cui il fallimento silenzioso è costoso: storicamente nel trading algoritmico, nell'IoT embedded e nella crittografia edge, e oggi negli strumenti di IA agentica. Questo articolo è nato dal misurare il proprio *plugin* di pianificazione per agenti di programmazione e dallo scoprire che una frase funzionava meglio.

Identificatori · ORCID: 0000-0002-6034-7765 · Lattes (registro nazionale brasiliano dei curricula dei ricercatori): 6905246706890561

Contatto · c.ulisses@gmail.com · https://ulissesflores.com