

Uma frase antes do plano

Um estudo confirmatório pré-registrado do pré-mortem como prompt em agentes de código com LLM

Carlos Ulisses Flores

Codex Hash Research Laboratory

ORCID: 0000-0002-6034-7765

Codex Hash Research Report 2026-03

Data: 17 de novembro de 2026

Resumo

Um pré-mortem pede que a equipe suponha que o plano fracassou e liste os motivos (Klein, 2007). Num estudo pré-registrado anterior, a metade de listagem desse exercício, posta em uma frase, fez o plano de um agente de código cobrir mais riscos listados de forma independente, como resultado secundário (Flores, 2026a). Este é o estudo confirmatório. O Claude Code, em modo de planejamento no Claude Opus 5.5 e no Claude Sonnet 5.5, recebeu doze pedidos de mudança em repositórios públicos fixados, sob dois pedidos que diferem em uma frase: “Plan this change.” e “Before planning, list the ways this change could go wrong.” Três modelos de outros fornecedores escreveram às cegas os riscos de referência, e um juiz cego de outro fornecedor pontuou cada resposta. Protocolo, listas e análise foram selados e carimbados antes da primeira rodada. Em 240 rodadas, a frase elevou a cobertura de 46,1% para 57,5%, diferença média de 11,4 pontos percentuais em 24 estratos de tarefa por modelo (permutação estratificada, $p < ,001$, também em cada modelo). Um segundo juiz deu +10,7 pontos (κ de Cohen = ,86). Sem os três estratos com rodada perdida por erro de rede, a diferença é de 10,1 pontos. Contra um segundo conjunto de listas, pedido com outras palavras, ela é de 6,7 pontos ($p < ,001$), e de 9,0 por um juiz de um terceiro fornecedor acrescentado depois. A afirmação se limita a dois modelos de um fornecedor, a listas escritas por modelos e a planos, não a código.

Palavras-chave: pré-mortem; engenharia de *prompt*; agentes de código baseados em LLM; identificação de riscos; pré-registro; planejamento de 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

Sumário

1. Introdução 5
1.1. Contribuições 8
2. Fundamentação e trabalhos relacionados 9
2.1. O pré-mortem em pessoas 9
2.2. Enumerar a falha antes de responder, em LLMs e agentes 9
2.3. Lembretes e previsões na geração de código 10
2.4. Instruções curtas e sensibilidade ao *prompt* 12
2.5. Onde este estudo se situa 12
3. O estudo de origem 13
4. Método 14
4.1. Hipótese e desenho 14
4.2. Tarefas 15
4.3. Condições e *harness* 15
4.4. Riscos de referência, escritos às cegas 19
4.5. Pontuação 20
4.6. Análise e regra de decisão 22
4.7. Pré-registro e registro de integridade 23
5. Resultados 27
5.1. Coleta 27
5.2. Resultado principal 27
5.3. Segundo juiz 30
5.4. Relevância 30
5.5. Sensibilidade às três rodadas com erro 31
5.6. Custo 33
5.7. Quais riscos a frase acrescenta (exploratório) 34
5.8. Uma segunda régua, com outras palavras (exploratório) 35
6. Discussão 39
7. Objeções e ameaças à validade 41
8. Desvios do protocolo 47
9. Conclusão 49
10. Disponibilidade de dados e código 50
11. Declarações 52
Referências 53
Apêndice A — Os pedidos exatos 58
Apêndice B — Uma lista de referência completa 59
Apêndice C — Os textos em inglês, traduzidos 60
Nota Técnica 62
Sobre o Autor 63

1. Introdução

Agentes de código baseados em grandes modelos de linguagem (LLMs) já planejam e executam mudanças em repositórios reais (Jimenez et al., 2024). Os planos estudados aqui são pedidos antes de qualquer código ser escrito, um ponto antecipado para pegar um risco esquecido. Este artigo testa uma afirmação: abrir um pedido de planejamento com a frase “Before planning, list the ways this change could go wrong.” (em português: “Antes de planejar, liste as maneiras como esta mudança pode dar errado.”), no lugar de “Plan this change.”, faz a tabela de riscos que fecha o plano do agente cobrir mais dos riscos que revisores independentes esperam. A afirmação é estreita de propósito, porque efeitos da redação de um *prompt* muitas vezes não se mantêm de um modelo ou conjunto de tarefas para o seguinte (Sclar et al., 2024).

A frase vem da pesquisa sobre decisão. Num *pré-mortem*, a equipe imagina que o plano já fracassou e anota os motivos (Klein, 2007). A técnica se apoia na *retrospecção prospectiva*: quem explica um desfecho como se ele já tivesse ocorrido gera mais motivos do que quem explica um desfecho que pode ocorrer (Mitchell et al., 1989). Normas de avaliação de risco listam técnicas estruturadas vizinhas, como a análise de modos de falha e seus efeitos, entre os métodos que uma organização pode aplicar antes de agir (International Electrotechnical Commission, 2019). A frase testada aqui mantém só a segunda metade do exercício de Klein, a listagem, e abandona a suposição de fracasso (Figura 1).

Figura 1

As duas metades de um pré-mortem e a que a frase testada mantém



Para LLMs, a evidência mais próxima está na segurança, onde fazer o modelo enumerar os danos possíveis antes de responder reduz a nocividade julgada mais do que mandá-lo ser seguro (Kim et al., 2026). Em agentes, antecipar a falha antes de agir foi incorporado a métodos de vários passos e avaliado como um todo (Wang et al., 2024; Wang et al., 2026). Na geração de código, lembretes para ser seguro dão resultados mistos ou nulos (Tony et al., 2025; Kharma et al., 2026), e um *prompt* mais longo que pede a um agente de código que preveja casos de borda ajudou dois modelos e prejudicou um terceiro (Patel et al., 2026). Falta uma medição controlada da menor dose no planejamento de software: uma frase, contra o mesmo pedido sem ela, pontuada contra riscos que outra pessoa listou.

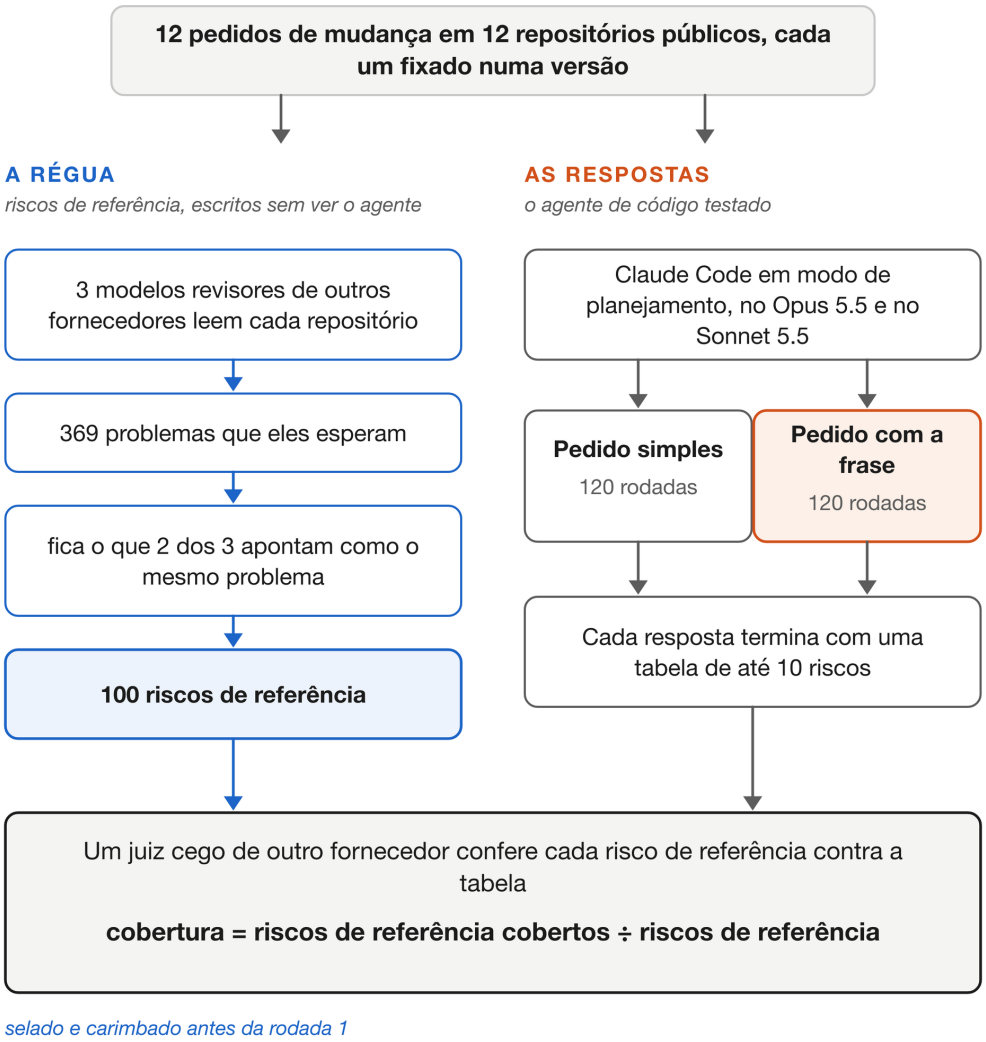
A frase surgiu de uma avaliação anterior de uma ferramenta mantida pelo autor. O ASP é um *plugin* de código aberto, um complemento instalável, para o Claude Code, o agente de programação em linha de comando da Anthropic; ele conduz o agente por um protocolo de pré-mortem em 13 fases, e sua primeira medição pré-registrada, aquela em que protocolo e análise foram fixados em público antes da coleta dos dados (Seção 4.7), não achou benefício mensurável: o protocolo completo cobriu menos riscos de referência do que um pedido simples com a frase

acrescentada (Flores, 2026a). A frase, por sua vez, superou o pedido simples em 10 pontos percentuais. Essa comparação foi pré-registrada como secundária, num só modelo e em seis tarefas, de modo que justificava um novo estudo e nada mais (Nosek et al., 2018).

Este artigo relata esse estudo. Ele foi desenhado para dar ao achado anterior uma chance justa de falhar: doze tarefas novas, um segundo modelo, a comparação como única pergunta principal e uma regra de decisão selada antes de existir qualquer dado (Figura 2).

Figura 2

Desenho do estudo: como os riscos de referência e as respostas são produzidos e pontuados



1.1. Contribuições

1. **Um teste confirmatório e pré-registrado de um pedido de uma frase num agente de código.** Em 240 rodadas, em doze repositórios públicos fixados e dois modelos, a frase eleva a cobertura dos riscos listados de forma independente em 11,4 pontos percentuais ($p < ,001$), e em valor semelhante dentro de cada modelo (Seção 5.2).
2. **Verificações do ganho.** Um segundo juiz de outro fornecedor chega à mesma decisão (Seção 5.3), os riscos listados não são julgados menos concretos (Seção 5.4), e o resultado sobrevive à retirada de três rodadas que um erro de rede zerou a favor da comparação (Seção 5.5). Contra listas de referência pedidas depois com outras palavras, o ganho continua positivo e cai para 6,7 pontos (Seção 5.8).
3. **Um registro selado e aberto.** Cada *prompt* definido pelo estudo, cada lista de referência, resposta, veredito e arquivo de análise é público e está ligado a um registro encadeado por hash (*ledger*), uma corrente de impressões digitais de arquivos, cujo estado anterior às rodadas está ancorado no *blockchain* do Bitcoin, um registro público que é impraticável reescrever (Seção 4.7), e um único comando recalcula cada resultado do estudo (Flores, 2026b).
4. **Uma lista declarada de objeções.** A Seção 7 toma as objeções mais fortes ao resultado, inclusive as levantadas por revisores externos deste manuscrito, e responde a cada uma com dados ou a registra como em aberto.

Até onde sabemos, nenhum estudo anterior mede o efeito de uma instrução mínima de pré-mortem sobre a cobertura de riscos levantados de forma independente no planejamento de mudanças de software por um agente de código, afora o estudo de origem, em que ele foi uma comparação

secundária (Flores, 2026a). A Seção 2.5 diz o que delimita essa afirmação. O mecanismo em si, enumerar modos de falha antes de responder, tem apoio anterior (Kim et al., 2026) e não é reivindicado aqui.

2. Fundamentação e trabalhos relacionados

2.1. O pré-mortem em pessoas

Klein (2007) descreve o pré-mortem de projeto como uma reunião curta antes de o trabalho começar: supõe-se que o plano fracassou, e cada participante escreve os motivos. O experimento de apoio é o de Mitchell et al. (1989), em que a certeza sobre um desfecho, mais do que sua posição no tempo, aumentou o número de explicações que as pessoas produziram. Num experimento controlado com 178 estudantes, o pré-mortem reduziu a confiança no plano em 25,0 pontos, contra 14,0 para a lista de prós e contras e 12,4 para a lista só de contras (Veinott et al., 2010). Os efeitos humanos estabelecidos estão, portanto, na quantidade e na confiança, de modo que um estudo com LLMs precisa conferir que mais itens não signifiquem simplesmente itens mais vagos (Seção 5.4).

O procedimento humano tem duas partes, a suposição de fracasso e a listagem dos motivos (Klein, 2007). Um *prompt* pode pedir a segunda sem a primeira, e a frase estudada aqui faz exatamente isso. A comparação entre respostas humanas e de IA no planejamento por pré-mortem começou na pesquisa de fatores humanos (Veinott e Lehman, 2024), mas como comparação de quem responde, não de como o modelo é instruído.

2.2. Enumerar a falha antes de responder, em LLMs e agentes

O InvThink é o trabalho anterior mais próximo (Kim et al., 2026). O modelo lista os danos possíveis, analisa suas consequências e só então responde sob as restrições que derivou. No modo apenas de *prompt*, sem treinamento, o método reduz uma nota de nocividade julgada de 3,22 para

2,19, enquanto uma instrução de segurança de tamanho parecido chega só a 2,62. A explicação dos autores é que a enumeração explícita traz à tona conhecimento que um pedido direto deixa sem uso. O desfecho ali é conteúdo nocivo, e o enquadramento tem três passos.

Em agentes, o Devil's Advocate faz um agente de navegação na web refletir sobre prováveis falhas e remédios alternativos antes de cada ação, e relata 23,5% de sucesso nas tarefas contra 19,8% de uma linha de base de planejar e agir, com 45% menos revisões de plano do que o melhor método concorrente (Wang et al., 2024). O PreFlect critica o plano antes da execução usando erros de planejamento destilados de trajetórias anteriores, e eleva o sucesso num *benchmark* de assistente geral, um conjunto fixo de tarefas de teste, de 46,1% para 58,2% (Wang et al., 2026). Ambos avaliam métodos de vários passos; nenhum testa uma instrução de uma frase, e nenhum conta quantos riscos esperados um plano nomeia.

Uma terceira linha trata do que um modelo sem instrução deixa de fora. Num experimento de campo pequeno e preliminar sobre planejamento de *sprint*, em que uma equipe planeja seu próximo ciclo curto de trabalho, um LLM planejando sozinho documentou 36,4% dos riscos que depois se materializaram, 4 riscos, contra 78,6% dos planejadores humanos, 11 riscos (Caraeni et al., 2026). Agentes de LLM também podem produzir muito mais consequências candidatas de um produto do que especialistas trabalhando sozinhos (Fröhling et al., 2026). O primeiro resultado é a lacuna que um *prompt* de planejamento poderia estreitar, e seu desfecho é vizinho do usado aqui: lá, captura de riscos que se materializaram; aqui, captura de riscos que os revisores esperavam.

2.3. Lembretes e previsões na geração de código

Vários estudos acrescentam uma instrução de cautela a um *prompt* de geração de código e medem o código. No GPT-4, algumas técnicas de *prompt* reduziram a densidade de fraquezas de segurança

entre 59% e 78% (Tony et al., 2025). Em cinco modelos e quatro métodos de *prompt*, outro estudo não achou redução estatisticamente significativa de vulnerabilidades (Kharma et al., 2026). Um *prompt* de sistema que pedia cobertura de casos de borda muitas vezes não foi melhor que a linha de base na geração de código sem exemplos (Cheng e Mastropaolo, 2026). Esses resultados advertem contra supor que uma frase curta de cautela ajude.

Eles também diferem do cenário atual de um modo que importa. As instruções mais simples deles são lembretes (“seja seguro”), embora as técnicas comparadas incluam também as iterativas, como a crítica e o aprimoramento recursivos (Tony et al., 2025), e o desfecho é o código. Uma instrução de *enumerar* é um ato diferente de uma instrução de *ter cuidado*, distinção que o InvThink traça contra sua linha de base de *prompt* de segurança (Kim et al., 2026). Pedir ao modelo que liste suas suposições implícitas é um ato de enumeração vizinho, estudado para geração de código com um *prompt* estruturado (Wu, 2026).

O estudo mais próximo em software é o de Patel et al. (2026). Eles dão a agentes de código, entre eles o Claude Code, um *prompt* de várias frases que pede ao agente que preveja os casos de borda que sua implementação pode deixar passar, antes de começar, durante a implementação e antes de entregar. O desfecho está no código: uma taxa conjunta de *patches* (mudanças de código propostas) que resolvem a tarefa e também evitam um defeito que uma mudança posterior do projeto teve de corrigir. Na comparação deles, essa taxa subiu de 24,5% para 32,6% no Claude Opus 5, subiu num segundo modelo e caiu de 17,0% para 13,7% no Claude Sonnet 5, enquanto a parcela de tarefas resolvidas caiu nos três. O resultado deles é a razão de a afirmação atual não se estender de planos para código, e mostra que o efeito desses *prompts* pode depender do modelo.

2.4. Instruções curtas e sensibilidade ao *prompt*

Que uma frase possa mover a saída de um modelo está bem estabelecido. “Let’s think step by step” elevou a acurácia sem exemplos num *benchmark* aritmético de 17,7% para 78,7% (Kojima et al., 2022), e o *prompt* Plan-and-Solve refinou a mesma ideia ao pedir que o modelo primeiro elabore um plano (Wang et al., 2023). A mesma literatura mostra por que tais achados precisam de replicação: a acurácia variou até 76 pontos entre escolhas de formatação sem significado algum (Sclar et al., 2024).

Tampouco um *prompt* de pré-mortem ajuda em toda parte. Numa comparação pré-registrada de 38 *prompts* para perguntas de previsão, que incluía um *prompt* de pré-mortem, a maioria dos *prompts* mudou pouco a acurácia (Schoenegger et al., 2025). Um efeito de redação visto uma vez, num modelo e em um punhado de tarefas, pode ser ruído dessa sensibilidade. Essa é a razão metodológica de um estudo confirmatório com um segundo modelo, tarefas novas e uma regra fixada de antemão (Nosek et al., 2018).

2.5. Onde este estudo se situa

A combinação estudada aqui é: um cenário de engenharia de software com repositórios reais; um desfecho definido como cobertura de uma lista de referência que o agente testado nunca viu; uma dose de uma frase; e um resultado anterior em que um protocolo longo e estruturado cobriu 48% dos riscos de referência contra 66% dessa frase (Flores, 2026a). Os vizinhos mais próximos diferem em pelo menos um desses pontos: o InvThink no domínio e no desfecho (Kim et al., 2026), e Patel et al. (2026) na dose e no desfecho.

A busca na literatura por trás da declaração de novidade da Seção 1.1 cobriu veículos e servidores de *preprints* em inglês até 14 de novembro de 2026, com consultas adicionais em português,

alemão, japonês e chinês; seu registro, com a situação de cada fonte, está no repositório (Flores, 2026b). Ela trabalhou sobre títulos e resumos, de modo que um estudo que teste tal condição sem nomeá-la ali, ou um que não esteja indexado, pode ter escapado.

3. O estudo de origem

O estudo anterior fez 90 rodadas em seis repositórios públicos com o Claude Opus 5.5, sob três condições: o pedido simples, o pedido com a frase e o protocolo ASP (Flores, 2026a, Seção 5.5). A Tabela 1 e a Figura 3 dão o resultado.

Tabela 1
Cobertura de riscos e tempo por rodada no estudo de origem (2026-11), por condição

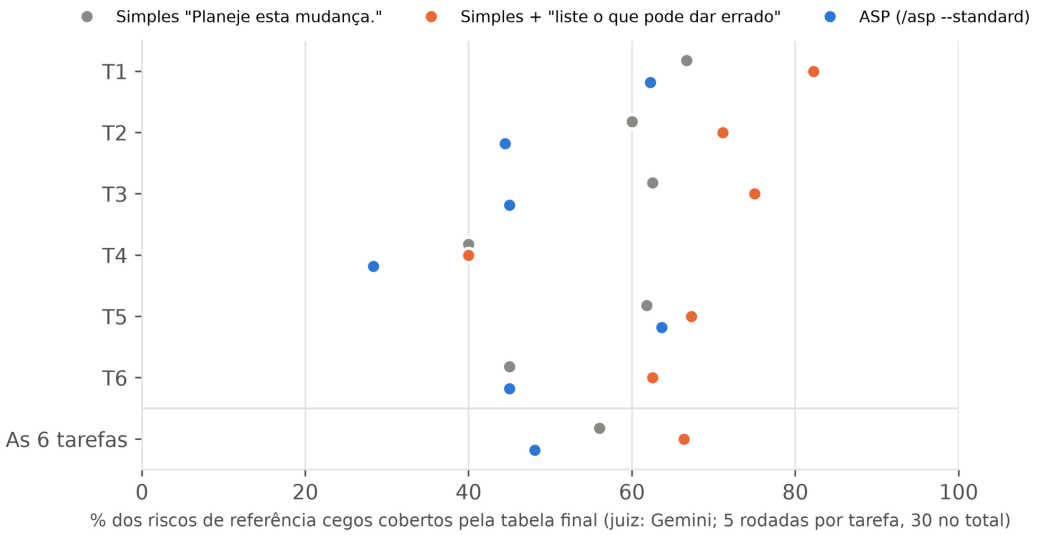
Condição	Cobertos (juiz principal)	Cobertos (segundo juiz)	Tempo médio por rodada
Pedido simples	56%	57%	58 s
Pedido simples mais a frase	66%	70%	50 s
Protocolo ASP (13 fases)	48%	54%	175 s

Nota. 30 rodadas por condição em seis tarefas. Fonte: Flores (2026a).

A comparação principal pré-registrada desse estudo, ASP contra a frase, foi negativa nas seis tarefas: −18 pontos, p = 1,0 (Figura 3). A comparação secundária, a frase contra o pedido simples, foi de +10 pontos (p = ,001) pelo juiz principal e de +13 pontos (p < ,001) pelo segundo (Tabela 1). Como o protocolo rotulou essa comparação como secundária, o relato anterior afirmou apenas a direção dela e anunciou o presente estudo (Flores, 2026a).

Figura 3

Estudo de origem: fração dos riscos de referência cegos coberta pela tabela final de cada condição, por tarefa e no total (juiz principal)



Nota. Na legenda, "/asp --standard" executa o protocolo ASP e "liste o que pode dar errado" representa a frase.

Duas propriedades desse resultado moldaram o novo desenho. O efeito foi medido num único modelo, e efeitos de *prompt* podem se inverter entre modelos (Patel et al., 2026). E ele se apoiava em seis tarefas, de modo que o novo estudo usa doze que o anterior nunca tocou (Tabela 2).

4. Método

4.1. Hipótese e desenho

A hipótese é direcional: a frase eleva a cobertura. O desenho cruza doze tarefas, dois modelos e duas condições, com cinco réplicas por célula, num total de 240 rodadas (Figura 2). Um *estrato* é uma tarefa em um modelo; são 24, cada um com cinco rodadas por condição. O protocolo fixou cada item desta seção antes da primeira rodada, seguindo a prática de separar a análise confirmatória da exploratória (Nosek et al., 2018), e seu arquivo está no repositório (Flores, 2026b).

4.2. Tarefas

Cada tarefa é um pedido de mudança de uma frase num repositório público fixado em um *commit*, uma versão gravada de seu código (Tabela 2). Os pedidos foram escritos para ter consequências além dos arquivos que uma primeira leitura sugere: dados guardados que precisam migrar, clientes que precisam continuar funcionando, padrões dos quais outros projetos dependem. Eles abrangem sete linguagens, uma variedade que as diretrizes para estudos com LLMs em engenharia de software recomendam em vez de uma amostra de uma só linguagem (Baltes et al., 2027). O arquivo de tarefas foi commitado antes de qualquer lista de referência ser escrita (Seção 4.7).

4.3. Condições e *harness*

As duas condições enviam o mesmo texto, exceto por uma frase (o Apêndice A dá as duas por inteiro):

- **Simples:** o pedido de mudança, depois “Plan this change.” (em português: “Planeje esta mudança.”)
- **Frase:** o pedido de mudança, depois “Before planning, list the ways this change could go wrong.” (em português: “Antes de planejar, liste as maneiras como esta mudança pode dar errado.”; Tabela A1; Figura 4)

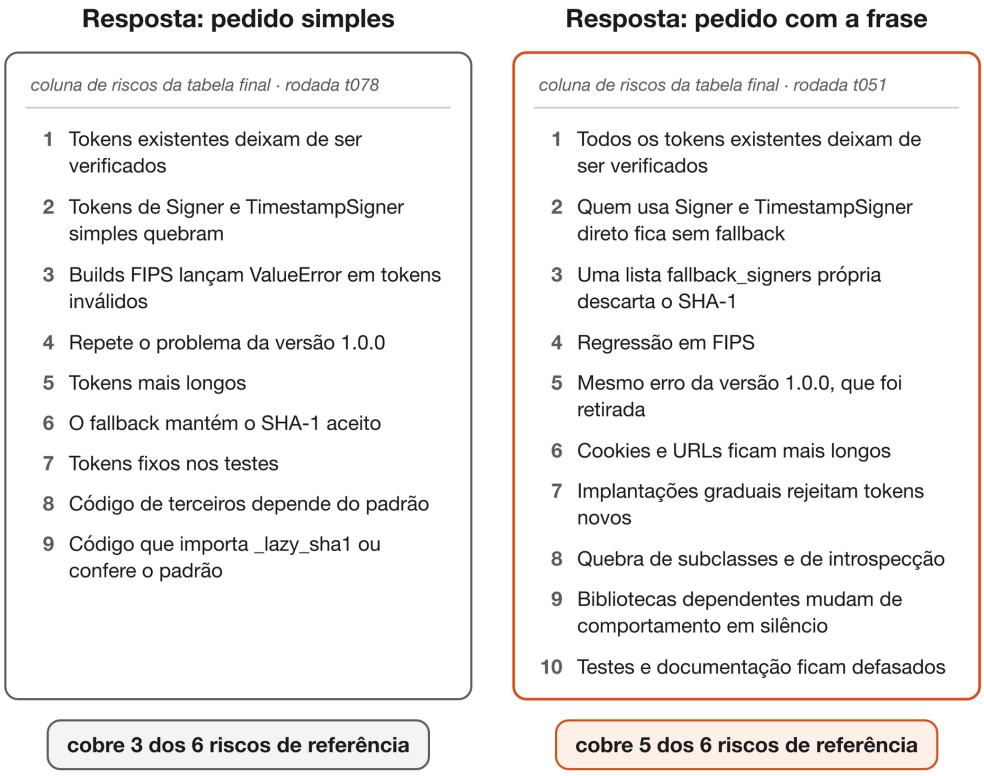
Figura 4

Os dois pedidos, lado a lado: uma frase difere e todo o resto é idêntico



Ambas levam em seguida o mesmo *contrato de saída*, um parágrafo fixo que manda o agente declarar suposições em vez de fazer perguntas, não editar nenhum arquivo do projeto (o modo de planejamento ainda pode salvar o próprio plano num arquivo; Seção 5.6), responder em inglês e terminar com os principais riscos da mudança numa tabela de no máximo dez linhas (Apêndice A). O contrato importa para ler o resultado. A condição simples não é um agente que nunca foi perguntado sobre risco: as duas condições precisam terminar com uma tabela de riscos. A comparação mede, portanto, o efeito conjunto de abrir o pedido com a frase em vez da instrução simples, além de ser obrigado a relatar riscos no fim; ela não separa a posição da frase de sua redação (Seção 7, objeção 5). Os *prompts* são, byte a byte, os do estudo de origem (Flores, 2026a). A Figura 5 mostra o que volta: a coluna de riscos de duas respostas reais ao mesmo pedido, uma por condição.

Figura 5
A coluna de riscos de duas respostas reais ao pedido da Figura 4, uma por condição (tarefa U08, Claude Opus 5.5)



traduzido do inglês; o original está no repositório

Na Figura 5, FIPS são as normas federais de processamento de informação dos Estados Unidos (Federal Information Processing Standards); um sistema construído em modo FIPS recusa algoritmos criptográficos que essas normas não aprovam.

Tabela 2
As doze tarefas

Tarefa	Repositório (linguagem)	Pedido de mudança
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.”

Tarefa	Repositório (linguagem)	Pedido de mudança
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. Os endereços completos dos repositórios e os identificadores de *commit* estão em benchmark/study-2026-11b/tasks.json (Flores, 2026b). Os pedidos foram enviados ao agente em inglês, como estão aqui; a Tabela C1 traz a tradução. *Slug* é o trecho legível do endereço de uma página, UUID é um identificador longo e praticamente único, *tenant* é cada cliente de um sistema que atende vários ao mesmo tempo, *middleware* é o código que trata cada requisição antes da aplicação, e SHA-1 e SHA-256 são funções de hash, que transformam dados num código de tamanho fixo.

O *harness*, o programa que faz o modelo trabalhar como agente, é o Claude Code, versão 2.1.319: cada rodada é uma execução não interativa dele em modo de planejamento, um modo em que o agente pode ler o repositório e não pode alterá-lo (Anthropic, 2026). A execução parte de um clone novo do *commit* fixado, numa pasta de configuração isolada, com toda variável de ambiente herdada cujo nome começa com CLAUDE ou ANTHROPIC removida. O ASP e os cinco *plugins* sincronizados da conta do operador foram desligados; os quatro *plugins* embutidos no Claude Code ficaram ligados em todas as rodadas, nas duas condições (Flores, 2026b). Os dois modelos são o

Claude Opus 5.5 e o Claude Sonnet 5.5. As 240 rodadas foram postas numa só ordem por um embaralhamento com semente, de modo que condições e modelos ficam intercalados no tempo, e rodaram três de cada vez.

4.4. Riscos de referência, escritos às cegas

A lista de referência de uma tarefa é o conjunto de riscos contra o qual toda resposta dessa tarefa é pontuada, e as doze listas juntas formam a régua do estudo. Três modelos de outros fornecedores a escreveram: Grok, GPT-OSS 120B e Gemini 3.8 Flash (Tabela 3). Cada um leu uma cópia nova do repositório e o pedido de mudança, sob um único *prompt* que pede de seis a doze maneiras concretas de a mudança dar errado naquela base de código (Apêndice A). Nenhum viu as condições, o protocolo nem qualquer saída do agente testado. Eles produziram 369 itens. Um juiz então agrupou os itens que nomeiam o mesmo problema, e um risco entrou na lista de referência quando pelo menos dois dos três revisores o tinham citado. Restaram 100 riscos de referência, entre 6 e 10 por tarefa; 31 foram citados pelos três revisores e 69 por dois (o Apêndice B mostra uma lista inteira).

Aqui os modelos fazem as vezes de engenheiros independentes, substituição que as diretrizes para estudos com LLMs pedem aos autores que declarem e delimitem (Baltes et al., 2027). O protocolo a declara como um enfraquecimento. O *prompt* dos revisores também compartilha palavras-chave com a frase testada, e a Seção 7, objeção 1, trata isso como a principal ameaça ao resultado.

Tabela 3
Os papéis que cada modelo externo exerce no estudo

Modelo	Escreve uma lista de revisor	Agrupa os itens	Julga a cobertura	Julga a relevância	Riscos de referência com sua redação
Gemini 3.8 Flash	sim	sim	juiz principal	sim	91
GPT-OSS 120B	sim	não	não	não	9
Grok	sim	não	segundo juiz	não	0
GPT-6 Astra	não	não	terceiro juiz, só na segunda régua	não	0

Nota. Um risco de referência leva a redação do primeiro revisor em ordem alfabética de nome de revisor que o escreveu, razão pela qual nenhum leva a redação do Grok. Nenhum modelo externo compartilha fornecedor com os modelos testados. As saídas guardadas do cliente do Grok não trazem identificador de modelo. O GPT-6 Astra foi acrescentado depois da revisão, como juiz não planejado da segunda régua, e é do mesmo fornecedor do GPT-OSS 120B (Seção 5.8; Flores, 2026b).

4.5. Pontuação

A *cobertura* é o desfecho: o número de riscos de referência que a tabela de uma resposta cobre, dividido pelo número de riscos de referência da tarefa. O juiz principal, o Gemini 3.8 Flash, recebe o pedido de mudança, os riscos de referência e somente as linhas da tabela da resposta (Apêndice A). Para cada risco de referência, decide se alguma linha trata do mesmo problema de fundo, ainda que redigido de outro modo. A Figura 6 mostra os vereditos do juiz para as duas respostas da Figura 5. Ao juiz não se informa a condição nem o modelo. Uma lista fixa de palavras foi trocada por um marcador antes do julgamento; a lista foi herdada do estudo de origem, em que ocultava o nome do protocolo ASP, e alterou 12 palavras em 10 das 240 tabelas daqui (Flores, 2026b).

Figura 6

Como as duas respostas da Figura 5 são pontuadas: o veredito do juiz principal sobre cada um dos seis riscos de referência da tarefa

Risco de referência (tarefa U08)	simples t078	com a frase t051
1 Tokens antigos deixam de validar depois da atualização	✓	✓
2 As duas classes ficam com padrões diferentes	✗	✓
3 A chave derivada muda de valor e de tamanho	✗	✗
4 Testes afirmam saídas que batem com SHA-1	✓	✓
5 Assinaturas maiores estouram colunas, URLs ou cookies	✓	✓
6 Documentação e doctests ainda dizem SHA-1	✗	✓
Cobertos	**3 de 6 = 50%**	**5 de 6 = 83%**

Três medidas secundárias foram fixadas junto com ela (Flores, 2026b):

- **Segundo juiz.** O Grok pontua o mesmo material do mesmo modo. A concordância entre os dois juízes é relatada no nível de decisões individuais; o κ de Cohen (Cohen, 1960), que desconta a concordância esperada por acaso, foi acrescentado depois a esse relato e é rotulado como não planejado (Seção 7).
- **Comparador lexical.** Um comparador determinístico por sobreposição de tokens, que compara textos pelos tokens (palavras ou partes de palavras) que eles têm em comum, mantido do estudo de origem, em que já se mostrara cego à paráfrase (Flores, 2026a).
- **Relevância.** A parcela das linhas da tabela de uma resposta que o juiz principal chama de risco concreto e plausível desta mudança específica. A relevância protege contra um ganho vazio, porque o efeito do pré-mortem humano é de quantidade (Mitchell et al., 1989). Ela é julgada só a partir do pedido e da linha, de modo que mede especificidade e plausibilidade, não correção.

Juízes LLM chegam a mais de 80% de concordância com preferências humanas em alguns cenários (Zheng et al., 2023) e têm vieses conhecidos, entre eles a preferência por texto parecido com o próprio (Panickssery et al., 2024). Nenhum juiz deste estudo compartilha fornecedor com os modelos testados, de modo que nenhum juiz pontua respostas escritas por um modelo do próprio fornecedor, o caso em que a autopreferência está documentada. O mesmo não vale para os riscos de referência, a maioria dos quais leva a redação do juiz principal (Tabela 3; Seção 7, objeção 3).

4.6. Análise e regra de decisão

A estatística de teste é a média, sobre os 24 estratos, da diferença entre a cobertura média com a frase e a cobertura média sem ela. Sua significância vem de um teste de permutação (Ernst, 2004): os rótulos de condição são embaralhados dentro de cada estrato 10.000 vezes, e o valor p é a parcela dos embaralhamentos cuja estatística é pelo menos a observada, contada como (acertos + 1) / 10.001 para que nunca seja zero (Phipson e Smyth, 2010). O teste é unilateral e não supõe nenhuma distribuição para a cobertura. Embaralhar dentro dos estratos mantém as diferenças entre tarefas e entre modelos fora da comparação (Ernst, 2004).

Em símbolos, a estatística D é a média da Equação 1, em que S é o número de estratos e d_s é a diferença, no estrato s, entre a cobertura média com a frase e a cobertura média sem ela (Flores, 2026b).

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

O valor p segue a Equação 2, em que *B* é o número de embaralhamentos e *b* é o número deles cuja estatística é pelo menos o *D* observado (Phipson e Smyth, 2010). Com *B* = 10.000, o menor valor que o teste pode devolver é 1/10.001.

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

A regra de decisão tem uma condição para “confirmado”: valor p combinado abaixo de ,05 *e* o mesmo teste dentro de cada modelo, sobre seus doze estratos, abaixo de ,05 para os dois modelos (Flores, 2026b). Se só um modelo passasse, o resultado seria declarado confirmado apenas para esse modelo. A regra também exigia pelo menos 18 estratos completos. Nenhuma outra análise é confirmatória, no sentido de Nosek et al. (2018).

Fixou-se de antemão que um erro numa rodada conta como cobertura zero e relevância zero e não é repetido, exceto um erro de limite de uso, que interrompe a coleta (Flores, 2026b). A Seção 5.5 examina o que essa regra fez.

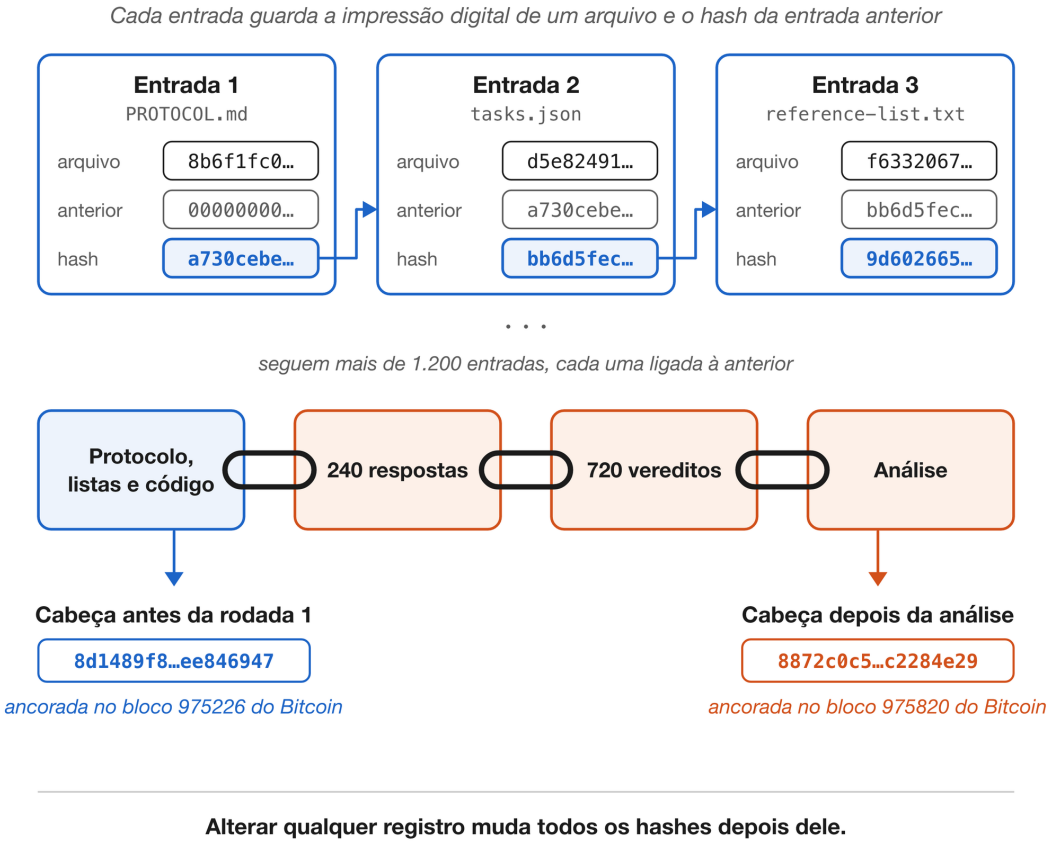
4.7. Pré-registro e registro de integridade

Pré-registro aqui significa que o protocolo, as tarefas, os *prompts*, as listas de referência e o código de análise foram commitados num repositório público e carimbados no tempo antes da primeira rodada contada (Nosek et al., 2018). Os acréscimos posteriores listados na Tabela 4 foram fixados, cada um, num adendo, um texto datado acrescentado ao protocolo e selado do mesmo modo antes das chamadas que ele cobre. Um registro encadeado por hash (*ledger*) liga cada arquivo por trás dos resultados relatados (os onze vereditos parciais da Seção 8 ficam fora dele): cada entrada guarda a impressão digital SHA-256 de um arquivo e da entrada anterior, de modo que o registro não pode ser reescrito sem mudar sua última entrada, chamada de cabeça (Flores, 2026b). Um arquivo corrigido depois de registrado recebe uma nova entrada, e a antiga permanece na cadeia.

Uma impressão digital SHA-256 é um código de 64 caracteres calculado a partir de um arquivo; mudar um byte do arquivo muda o código. A Figura 7 mostra as três primeiras entradas do registro com suas impressões digitais reais e as etapas que a cadeia fecha.

Figura 7

A cadeia de hashes do registro do estudo: as três primeiras entradas, as duas cabeças carimbadas e as etapas entre elas



A cabeça do registro antes da rodada 1 foi carimbada com o OpenTimestamps, serviço que ancora uma impressão digital no *blockchain* do Bitcoin, e o *commit* foi salvo no Software Heritage, um arquivo público de código-fonte. A Tabela 4 lista cada evento selado com o registro que prova sua ordem, e a Figura 8 mostra a sequência até a análise pré-registrada. O selo e a primeira rodada estão a 21 segundos um do outro no relógio do operador: o *commit* do selo é das 16:08:23 e a primeira resposta da primeira rodada é das 16:08:44. Essa ordem de segundos se apoia nesses

registros locais; a âncora no Bitcoin e o arquivo mostram que os arquivos selados existiam quando essas provas foram feitas, e não que precederam a primeira rodada por segundos. As 240 respostas foram publicadas antes de existir qualquer veredito, e todos os vereditos foram acrescentados ao registro antes de a análise rodar (Flores, 2026b).

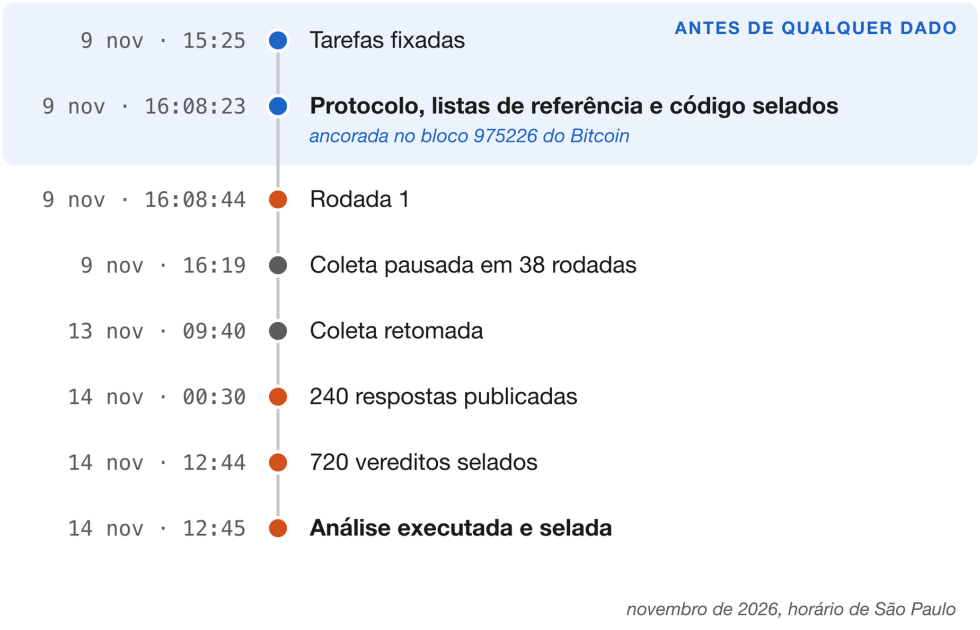
Tabela 4
O registro de integridade do estudo

Evento	**Data, horário de São Paulo**	**Registro**
Tarefas fixadas, antes de qualquer lista de referência	9 nov 2026, 15:25	*Commit* eb1a447
Protocolo, listas de referência, *prompts* e código selados	9 nov 2026, 16:08:23	*Commit* fe18438; cabeça do registro 8d1489f8…ee846947; prova do OpenTimestamps ancorada no bloco 975226 do Bitcoin; revisão do Software Heritage fe184385434ee3d433104549d054f8c98ad69b32
Primeira resposta da primeira rodada contada, t091	9 nov 2026, 16:08:44	Primeiro horário em runs/envelopes/t091.jsonl.gz
As 240 respostas publicadas, antes de qualquer veredito	14 nov 2026, 00:30	*Commit* e046f36
Os 720 vereditos selados, antes da análise	14 nov 2026, 12:44	*Commit* 67b52d0
Análise pré-registrada executada e selada	14 nov 2026, 12:45	*Commit* e8c73c1; cabeça do registro 8872c0c5…c2284e29; prova do OpenTimestamps ancorada no bloco 975820 do Bitcoin
Segunda régua: adendo, *prompt* e código selados antes de suas listas serem escritas	14 nov 2026, 16:25	*Commits* 573b8bf e fa3d0ba; cabeça do registro cf0a7bfd…0499f47f; prova do OpenTimestamps ancorada no bloco 975834 do Bitcoin

Evento	Data, horário de São Paulo	Registro
Segundo adendo (repetição e terceiro juiz) selado antes da primeira chamada	15 nov 2026, 11:19	*Commits* 0acd091 e 2fb3c8b; cabeça do registro 73e275d1…58b42f11; prova do OpenTimestamps enviada
Terceiro adendo (juiz por API paga) selado antes da primeira chamada de pontuação	15 nov 2026, 14:23	*Commits* 2b2af4c e 2c5f5eb; cabeça do registro 2a9ac120…4f6dce4c; prova do OpenTimestamps enviada

Nota. Os *commits* estão no repositório de Flores (2026b). As cabeças do registro estão abreviadas; os valores completos estão em benchmark/study-2026-11b/pretrial-head.txt e analysis-head.txt, ao lado de seus arquivos de prova .ots. As provas podem ser lidas com ots info e conferidas contra um nó do Bitcoin com ots verify; a terceira cabeça está em yardstick2/head-before-lists.txt. As cabeças dos dois adendos posteriores estão em addendum2/head-before-addendum2.txt e addendum3/head-before-addendum3.txt; as provas delas tinham sido enviadas e ainda não estavam ancoradas quando esta versão foi escrita.

Figura 8
Ordem dos eventos selados do estudo, do arquivo de tarefas à análise (eixo ordinal, fora de escala)



5. Resultados

5.1. Coleta

As 240 rodadas foram registradas, numa só versão do Claude Code, e nenhuma alterou um arquivo do projeto (Flores, 2026b). Três rodadas terminaram em erro de rede e não deixaram tabela (Seção 5.5). Os 24 estratos estão completos, de modo que a regra de decisão da Seção 4.6 se aplica (Tabela 5).

5.2. Resultado principal

A frase elevou a cobertura de 46,1% para 57,5% pelo juiz principal (Tabela 5). A estatística pré-registrada, a diferença média sobre os 24 estratos, é de 11,4 pontos percentuais, com $p < ,001$: o valor é 1/10.001, o menor que o teste pode devolver (Phipson e Smyth, 2010). Nenhum dos 10.000 embaralhamentos alcançou a diferença observada; o maior deu 5,9 pontos (Figura 9). Dentro do Claude Opus 5.5 a diferença é de 10,4 pontos e dentro do Claude Sonnet 5.5 é de 12,4 pontos, cada uma com $p < ,001$. A diferença é positiva em 22 dos 24 estratos, zero em um e negativa em um (Tabela 6). A regra de decisão devolve *confirmado*.

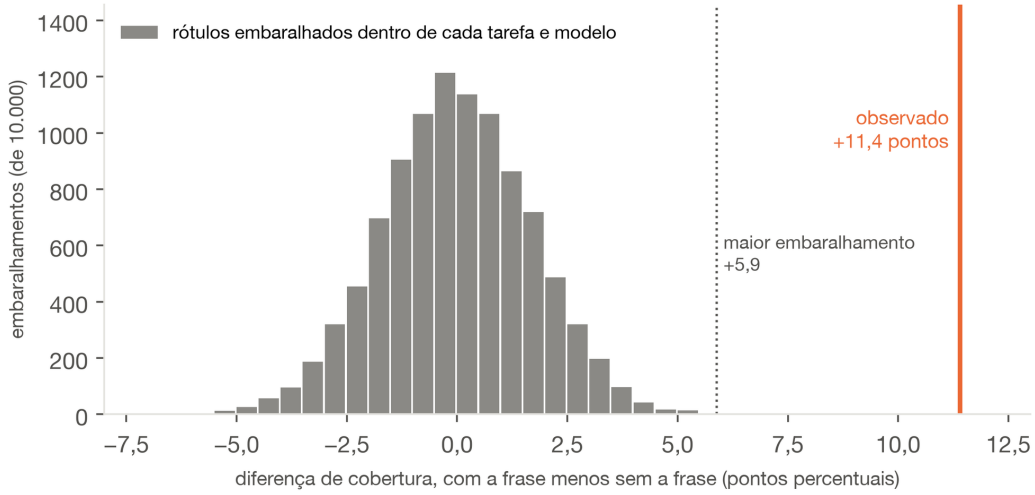
Tabela 5
Cobertura de riscos por modelo e juiz

Modelo	Simples (juiz principal)	Frase (juiz principal)	Simples (segundo juiz)	Frase (segundo juiz)
Claude Opus 5.5	43%	54%	48%	57%
Claude Sonnet 5.5	49%	61%	52%	64%
Ambos	46%	58%	50%	61%

Nota. 60 rodadas por modelo e condição. O juiz principal é o Gemini 3.8 Flash e o segundo juiz é o Grok. As porcentagens são arredondadas para números inteiros; as diferenças no texto vêm dos valores sem arredondamento.

Figura 9

O teste de permutação do resultado principal: as 10.000 diferenças embaralhadas e a observada



Um *bootstrap* por percentis, que recalcula a diferença em muitas reamostragens aleatórias dos dados, não foi previsto pelo protocolo e é relatado apenas como descrição. Reamostrando réplicas dentro de cada célula, ele dá um intervalo de 95% de 8,9 a 14,0 pontos para a diferença combinada, de 6,8 a 14,1 para o Opus 5.5 e de 8,8 a 16,2 para o Sonnet 5.5 (Flores, 2026b). O intervalo reflete a variação entre execuções nessas doze tarefas, não entre tarefas.

A Figura 10 e a Tabela 6 mostram o resultado tarefa por tarefa. O ganho não é carregado por poucas tarefas. O único estrato negativo é a mudança de *slug* em Go no Sonnet 5.5 (U02, −5 pontos), e o estrato em zero é a exportação de dados no Sonnet 5.5 (U06), em que as duas condições já cobriam 73% (Tabela 6).

Figura 10

Fração dos riscos de referência cegos coberta pela tabela final de riscos, por tarefa e modelo, com e sem a frase (juiz principal; legenda como na Figura 3)

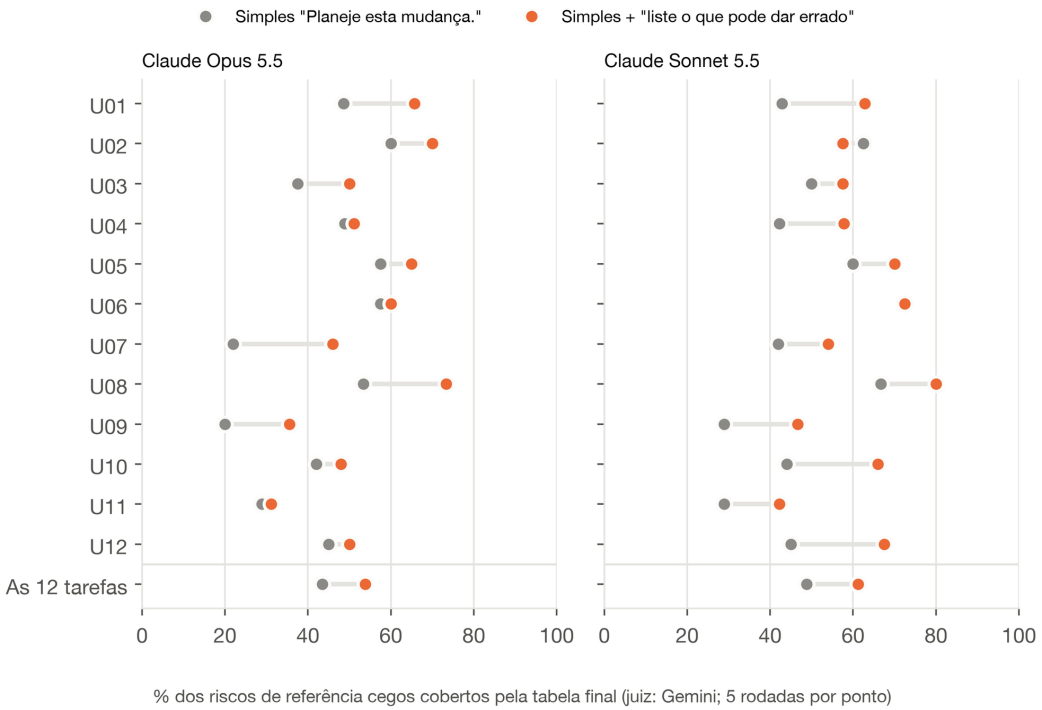


Tabela 6

Cobertura de riscos por tarefa e modelo, juiz principal, em porcentagem

Tarefa	Riscos de referência	Opus 5.5 simples	Opus 5.5 frase	Sonnet 5.5 simples	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

Tarefa	Riscos de referência	Opus 5.5 simples	Opus 5.5 frase	Sonnet 5.5 simples	Sonnet 5.5 frase
U11	9	29	31	29	42
U12	8	45	50	45	68

Nota. Cada célula é a média de cinco rodadas. Três células simples contêm uma rodada que terminou em erro de rede e conta como zero: U07 e U09 no Opus 5.5 e U10 no Sonnet 5.5. Nas outras quatro rodadas, essas células são 28, 25 e 55.

5.3. Segundo juiz

O segundo juiz deu 50,0% sem a frase e 60,8% com ela: diferença média de 10,7 pontos sobre os estratos ($p < ,001$), 9,3 pontos dentro do Opus 5.5 e 12,1 dentro do Sonnet 5.5 (cada um com $p < ,001$; Tabela 5), positiva em 22 dos 24 estratos também por este juiz (Flores, 2026b). Ele marca um pouco mais riscos como cobertos do que o juiz principal nas duas condições e chega à mesma decisão.

Os dois juízes tomaram 2.000 decisões cada, uma por risco de referência por rodada, e concordaram em 93,2% delas. O κ de Cohen, apresentado na Seção 4.5 (Cohen, 1960), é ,86. Dessas decisões, 29 dizem respeito às três rodadas sem tabela, em que os dois juízes só podiam responder “não coberto”; nas outras 1.971 a concordância é de 93,1% e o κ é ,86. O comparador lexical marcou 3,0% e 1,3%; como no estudo de origem, deixa passar correspondências parafraseadas e não é usado aqui como evidência de cobertura (Flores, 2026a).

5.4. Relevância

Nas rodadas que produziram tabela, a parcela de linhas que o juiz principal chamou de concretas e plausíveis foi de 71,1% sem a frase e de 73,9% com ela (Tabela 7). Pela regra fixada, que pontua como zero as três rodadas sem tabela, a cifra simples é de 69,4%. Não se observou redução na

relevância média julgada, que é a leitura fixada no protocolo. O estudo não fixa margem de não inferioridade, a maior queda de relevância que ainda contaria como ausência de perda, de modo que isso não prova que o ganho de cobertura veio sem linhas mais vagas, o risco que a literatura humana aponta (Mitchell et al., 1989). Não se afirma aumento de relevância: a diferença combinada de 4,5 pontos tem p = ,011 pela regra fixada, mas nenhum dos modelos chega sozinho a ,05 (,052 e ,060), e a diferença não é significativa quando se põem de lado as rodadas com erro (Tabela 7).

As tabelas tinham 9,2 linhas sem a frase e 9,6 com ela, contra um teto de dez (Tabela 8). A cobertura é uma medida de *recall*, a parcela dos itens de referência que uma resposta encontra, de modo que uma tabela mais longa tem mais chances de bater com um risco de referência. Restringir a comparação às rodadas cuja tabela tem as dez linhas completas, 56 simples e 82 com a frase, deixa a diferença em 11,3 pontos sobre os 17 estratos em que as duas condições permanecem (Flores, 2026b). O tamanho da tabela é ele próprio um efeito do pedido, de modo que essa comparação restrita descreve e não isola uma causa; ela mostra que o ganho está presente entre tabelas de mesmo tamanho.

5.5. Sensibilidade às três rodadas com erro

Durante uma perda curta de acesso à internet, três rodadas terminaram com erro depois de dez novas tentativas e não produziram tabela (Flores, 2026b). Pela regra fixada de antemão, elas contam como zero (Seção 4.6). As três pertencem à condição simples, de modo que a regra desloca a diferença principal em direção à hipótese. O protocolo não tinha previsto isso, e uma verificação rotulada como não planejada repete a análise de duas maneiras (Tabela 7).

Tabela 7
A análise principal com e sem as três rodadas com erro de rede

Análise	Estratos	Juiz principal	Segundo juiz	Relevância
Como pré-registrada (erros contam zero)	24	+11,4 (p < ,001)	+10,7 (p < ,001)	+4,5 (p = ,011)
Sem os três estratos que têm um erro	21	+10,1 (p < ,001)	+8,9 (p < ,001)	+2,5 (p = ,091)
Só sem as três rodadas	24	+10,5 (p < ,001)	+9,7 (p < ,001)	+2,9 (p = ,054)

Nota. Diferenças em pontos percentuais, frase menos simples. Só a primeira linha é confirmatória.

O resultado de cobertura se mantém em todas as linhas da Tabela 7, e dentro de cada modelo: na linha mais conservadora, as diferenças por modelo são de 8,5 pontos para o Opus 5.5 (p < ,001) e de 11,6 para o Sonnet 5.5 (p < ,001) pelo juiz principal, e de 6,9 (p < ,001) e 10,7 (p < ,001) pelo segundo. A diferença de relevância não se mantém ao nível de ,05 sem as rodadas com erro, razão pela qual a Seção 5.4 afirma apenas que não se observou redução na relevância.

O limite oposto dá a mesma leitura. Se cada uma das três rodadas com erro for pontuada como se sua tabela tivesse coberto todos os riscos de referência, a diferença é de 8,9 pontos pelo juiz principal e de 8,2 pelo segundo (Flores, 2026b).

As três rodadas também foram executadas mais uma vez, dois dias depois da coleta, como acréscimo não planejado, fixado num adendo datado antes das execuções (Seção 8). Cada uma terminou normalmente, com tabela de 8, 9 e 10 linhas. Com a cobertura delas no lugar dos três zeros, a diferença nas listas originais é de 10,5 pontos (p < ,001): 9,0 no Opus 5.5 e 11,9 no Sonnet 5.5 (Flores, 2026b). As linhas fixadas de antemão não mudam, e a Seção 5.8 traz os limites dessa repetição.

5.6. Custo

O tempo médio de execução foi menor com a frase, assim como os turnos e os tokens de saída, nos dois modelos (Tabela 8). O protocolo lista essas medidas como descritivas e não lhes atribui teste, de modo que são relatadas como descrição (Flores, 2026b). Elas sustentam uma única afirmação: não se observou custo adicional.

Uma diferença de comportamento está por trás de parte desses números. No modo de planejamento, o agente pode salvar o plano num arquivo, com uma chamada de ferramenta, antes de responder (Anthropic, 2026). No Opus 5.5 ele fez isso em 59 das 60 rodadas simples e em 22 das 60 rodadas com a frase; no Sonnet 5.5 não fez nunca, em nenhuma condição (Flores, 2026b). A Seção 7, objeção 13, examina se essa diferença poderia explicar o ganho de cobertura. As duas rodadas do Opus 5.5 que terminaram em erro de rede (Seção 5.5) já tinham salvado um arquivo de plano quando a conexão caiu, e a do Sonnet 5.5 não. Uma rodada simples no Opus 5.5 durou em média um turno a mais (Tabela 8) e ficou mais tempo exposta à queda.

Tabela 8
Custo médio por rodada, por condição e modelo, nas rodadas que não terminaram em erro

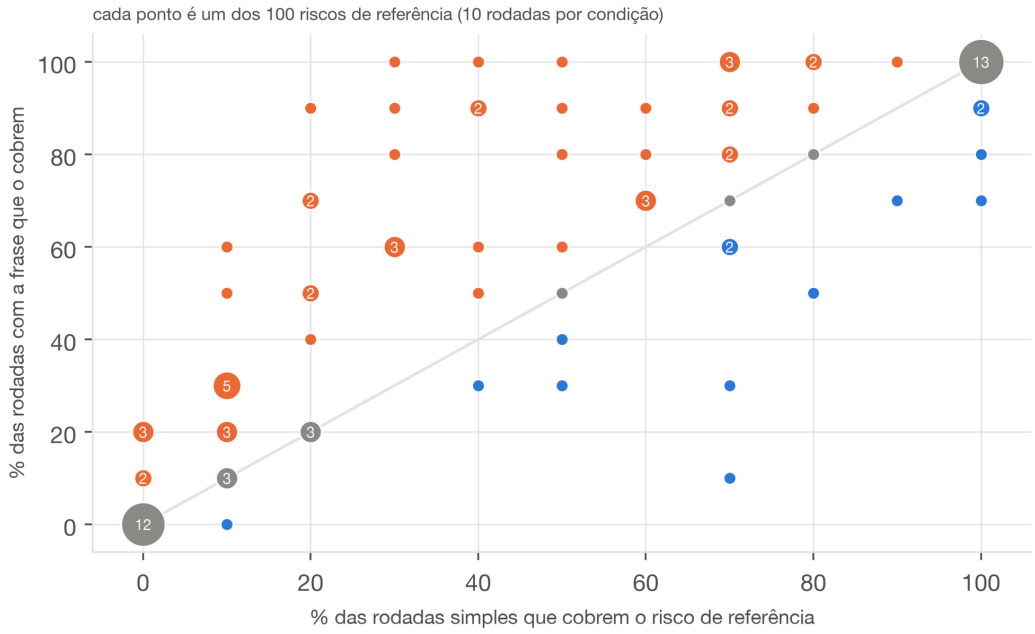
Modelo	Condição	Segundos	Turnos	Tokens de saída	Linhas da tabela
Claude Opus 5.5	Simples	66,5	5,8	5.801	9,2
Claude Opus 5.5	Frase	53,5	4,8	4.736	9,4
Claude Sonnet 5.5	Simples	32,9	3,5	3.014	9,3
Claude Sonnet 5.5	Frase	27,3	3,2	2.469	9,7
Ambos	Simples	49,6	4,7	4.395	9,2
Ambos	Frase	40,4	4,0	3.603	9,6

Nota. Um turno é uma troca entre o agente e suas ferramentas ou o modelo. 117 rodadas simples e 120 rodadas com a frase.

5.7. Quais riscos a frase acrescenta (exploratório)

Esta seção não foi planejada e não testa nada. Ela examina os 100 riscos de referência um a um, pelo juiz principal, para descrever de onde vem o ganho (Figura 11). Na figura, um ponto laranja acima da diagonal é um risco coberto mais vezes com a frase, um ponto cinza sobre ela é um risco coberto o mesmo número de vezes e um ponto azul abaixo dela é um risco coberto menos vezes; o número dentro de um ponto conta os riscos naquela posição.

Figura 11
Cada risco de referência pela parcela de rodadas que o cobrem, sem e com a frase (juiz principal)



- **Amplitude.** 52 riscos de referência foram cobertos com mais frequência com a frase, 34 com a mesma frequência e 14 com menos. Cada risco foi pontuado em dez rodadas por condição, de modo que cada uma dessas diferenças é de pelo menos dez pontos (Figura 11).

- **Confiabilidade.** O número de riscos de referência que a maioria das rodadas cobriu subiu de 41 para 56. A frase transformou principalmente riscos que o agente citava às vezes em riscos que ele citava na maior parte das vezes (Figura 11).
- **Pontos cegos que permanecem.** 12 riscos de referência não foram cobertos por nenhuma rodada em nenhuma das condições, e 13 por nenhuma rodada com a frase. São sobretudo consequências estreitas, no nível de arquivo, como um índice composto de banco de dados ausente ou um formato de armazenamento em que um novo esquema de hash não caberia (Flores, 2026b).

Os maiores ganhos são riscos de interação entre a mudança e o que já existe: uma rota de cadastro existente que loga o usuário de imediato e assim contorna a nova verificação de e-mail (U01, de 30% para 100% das rodadas), linhas existentes que ficam sem valor depois que se acrescenta uma nova coluna de identificador (U07, de 30% para 90%) e um gerador de hash de senha síncrono que um algoritmo de uso intensivo de memória travaria (U09, de 20% para 90%). As maiores perdas estão em U02 e U03, em riscos que o plano simples já citava na maioria das rodadas (Flores, 2026b). O Apêndice B mostra a mesma leitura para uma lista inteira.

5.8. Uma segunda régua, com outras palavras (exploratório)

Esta verificação foi especificada depois de conhecidos os resultados acima e não testa nada que tenha sido pré-registrado. Ela responde à objeção 1 da Seção 7: as listas de referência foram pedidas com palavras próximas às da frase testada. Os três revisores escreveram listas novas para as doze tarefas, sob um *prompt* em que uma frase mudou. Ele pede “the main risks of this change” (os principais riscos desta mudança), que são as palavras do contrato de saída que as duas condições recebem, e as palavras “go wrong” (dar errado) não aparecem (Flores, 2026b). A regra

que funde as listas, as 240 respostas, o *prompt* do juiz e a estatística ficaram como estavam. O adendo, o *prompt* e o código foram selados no registro e carimbados antes de o primeiro revisor ser chamado (Tabela 4).

As listas novas têm 111 riscos de referência, entre 7 e 11 por tarefa. Contra elas o juiz principal dá 55,8% sem a frase e 62,5% com ela: uma diferença média de 6,7 pontos ($p < ,001$), 6,5 no Opus ($p = ,002$) e 6,9 no Sonnet ($p = ,001$). Dos 24 estratos, 16 são positivos, 2 são zero e 6 são negativos (Tabela 9).

Tabela 9
Cobertura de riscos contra as duas réguas, juiz principal

Régua, pelas palavras de seu *prompt*	**Riscos de referência**	**Sem a frase**	**Com a frase**	**Diferença**	**p**	**Estratos positivos**
Original: “ways this change could go wrong” (formas de esta mudança dar errado)	100	46,1%	57,5%	+11,4	< ,001	22 de 24
Segunda: “the main risks of this change” (os principais riscos desta mudança)	111	55,8%	62,5%	+6,7	< ,001	16 de 24

Nota. As diferenças estão em pontos percentuais, como média sobre os 24 estratos de tarefa por modelo; p vem do teste de permutação estratificado da Seção 4.6. A segunda linha é exploratória e foi calculada depois de conhecida a primeira.

A diferença continua positiva quando o pedido aos revisores deixa de usar as palavras da frase, o que o adendo, selado antes de as listas serem escritas, lia como um ganho não preso a essas palavras (Flores, 2026b). Isso não exclui um efeito das palavras nas etapas posteriores que constroem a régua, descritas abaixo. A diferença também é menor. 6,7 pontos fica abaixo do intervalo *bootstrap* da estimativa original, de 8,9 a 14,0 (Seção 5.2), e o efeito é menos uniforme entre as tarefas (Tabela 9). Sob as listas novas, o pedido simples ganha cerca de dez pontos de cobertura e o pedido com a frase ganha cinco. A objeção 1 é, portanto, acolhida em parte.

Os dados não dizem quanto da distância entre 11,4 e 6,7 vem das palavras. Esta é uma única nova coleta de listas; os dois conjuntos de listas diferem em tamanho e em conteúdo, e uma cobertura inicial mais alta deixa menos espaço para ganho (Tabela 9). Três limites valem para esta verificação. O pedido que funde as três listas foi mantido como estava e ainda as descreve como "ways a software change could go wrong"; a verificação, portanto, tira essas palavras do pedido aos revisores, e não de toda a construção da régua (Flores, 2026b). O Gemini escreveu a redação de 98 dos 111 riscos de referência e foi o único juiz dela na análise planejada, a concentração que a objeção 3 da Seção 7 descreve. O segundo juiz planejado atingiu seu limite de uso depois de 11 de 240 vereditos; o adendo proíbe substituir um juiz, de modo que os vereditos dele não estão nesta versão (Seção 8).

Três outras checagens desta régua foram pedidas na revisão cega deste manuscrito e acrescentadas depois de conhecido o resultado dela; nenhuma está no adendo (Seção 8). As três rodadas com erro de rede são todas rodadas simples e também aqui contam zero. Sem os três estratos que as contêm, a diferença é de 5,3 pontos ($p < ,001$), e sem as três rodadas é de 5,6 ($p < ,001$). No limite oposto, com cada rodada com erro pontuada como cobertura total, é de 4,2 pontos ($p = ,005$): 5,3 no Sonnet 5.5 ($p = ,010$) e 3,2 no Opus 5.5, onde deixa de ser significativa ($p = ,099$). A leitura fixada no adendo se apoia no teste conjunto e se mantém em todos os tratamentos; o resultado isolado do Opus 5.5 não se mantém nesse limite (Flores, 2026b). O intervalo *bootstrap* da segunda diferença vai de 4,1 a 9,4 pontos e se sobrepõe ao intervalo original, de 8,9 a 14,0. Reamostrar as doze tarefas, e não as réplicas, alarga-o para 2,2 a 11,1, contra 8,2 a 14,5 na régua original.

Dois acréscimos vieram depois da segunda rodada de revisão, cada um fixado num adendo datado, selado e carimbado antes da primeira chamada (Seção 8). Primeiro, as três rodadas com erro foram executadas mais uma vez, uma tentativa cada, com o mesmo pedido, o mesmo *commit* fixado, o

mesmo identificador de modelo e a mesma versão do Claude Code; as três terminaram normalmente (Seção 5.5). Segundo, o GPT-6 Astra, modelo de um terceiro fornecedor, pontuou as 240 respostas e as três repetições contra a segunda régua, com o pedido do juiz inalterado, como juiz não planejado (Tabela 10). Ele não substitui o segundo juiz planejado, cujos vereditos continuam devidos (Flores, 2026b).

Tabela 10

A diferença na segunda régua sob três tratamentos das rodadas com erro, por juiz

Tratamento das três rodadas com erro	Juiz	Diferença	Opus 5.5	Sonnet 5.5
Contam zero, como fixado de antemão	Gemini 3.8 Flash	+6,7 (p < ,001)	+6,5 (p = ,002)	+6,9 (p = ,001)
Contam zero, como fixado de antemão	GPT-6 Astra	+9,0 (p < ,001)	+8,1 (p < ,001)	+9,8 (p < ,001)
Pontuadas como cobertura total	Gemini 3.8 Flash	+4,2 (p = ,005)	+3,2 (p = ,099)	+5,3 (p = ,010)
Pontuadas como cobertura total	GPT-6 Astra	+6,5 (p < ,001)	+4,7 (p = ,013)	+8,2 (p < ,001)
Executadas mais uma vez	Gemini 3.8 Flash, por API	+5,6 (p < ,001)	+4,9 (p = ,009)	+6,4 (p = ,002)
Executadas mais uma vez	GPT-6 Astra	+7,7 (p < ,001)	+6,2 (p < ,001)	+9,1 (p < ,001)

Nota. Diferenças em pontos percentuais, frase menos simples, com o p de permutação da Seção 4.6. Todas as linhas são exploratórias. Na quinta linha, os vereditos do Gemini sobre as três repetições vieram por uma API paga, não pelo cliente de linha de comando dos 240 vereditos selados.

Todas as linhas da Tabela 10 são positivas, e o teste conjunto fica abaixo de ,05 em cada uma. Pelo GPT-6 Astra, a diferença sob a regra fixada é de 9,0 pontos, de 64,0% para 73,0%, com 20 de 24 estratos positivos, e continua significativa em cada modelo no limite oposto, onde o resultado isolado do Opus 5.5 pelo Gemini não se mantinha. Com as rodadas executadas mais uma vez, os dois juízes dão diferença significativa nos dois modelos. Os dois juízes concordam em 89,4% de

2.192 decisões individuais (κ de Cohen = ,77), e o GPT-6 Astra marca mais itens como cobertos, 68,8% contra 59,0% (Flores, 2026b).

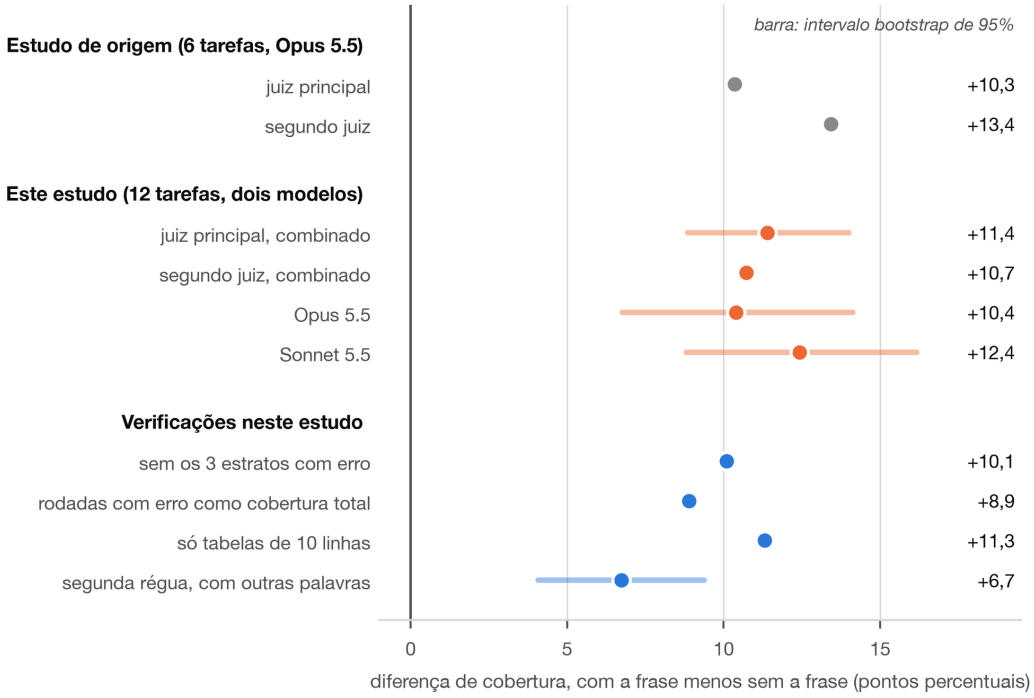
Esses acréscimos têm limites próprios. A repetição foi feita dois dias depois da coleta, por um operador sem cegamento quanto ao resultado, e nada observável prova que o modelo por trás de um identificador não tenha mudado; o modelo da sessão, os modelos usados e os *plugins* que as três sessões informam são os das rodadas seladas. O GPT-6 Astra é do mesmo fornecedor do GPT-OSS 120B, que escreveu uma das três listas de revisor. O juiz principal atingiu o limite de uso na linha de comando depois de 4 de seus 9 vereditos sobre as repetições; por decisão do autor, registrada num terceiro adendo, os 9 foram produzidos pelo mesmo modelo por uma API paga em vez de aguardar a renovação do limite, mais um desvio. Uma amostra de 30 rodadas seladas, fixada por semente e pontuada por esse canal, concorda com os vereditos selados em 98,2% de 281 decisões (κ = ,96), com cobertura média de 56,0% contra 56,4%. Os vereditos da linha de comando serão relatados ao lado destes quando o limite for renovado (Flores, 2026b).

6. Discussão

O que foi confirmado. Pedir a um agente de código as maneiras como uma mudança pode dar errado, antes de ele planejar, faz sua lista final de riscos cobrir mais do que três revisores independentes esperavam (Tabela 5). O efeito se replicou em um segundo estudo, doze tarefas novas e um segundo modelo, num tamanho próximo ao do estudo de origem: +10 pontos lá (Tabela 1), +11 aqui. Contra uma segunda régua, pedida com outras palavras, o ganho é menor, 6,7 pontos, e ainda positivo (Seção 5.8). Ele é modesto em termos absolutos: o agente ainda deixa de citar cerca de quatro riscos de referência em dez. A Figura 12 põe lado a lado as principais estimativas da diferença; os acréscimos posteriores estão na Tabela 10.

Figura 12

As principais estimativas da diferença de cobertura: do estudo de origem, deste estudo e de suas verificações



Uma leitura possível. As duas condições terminam com a mesma exigência de uma tabela de riscos, de modo que o agente é perguntado sobre risco nas duas. O que difere é que os modos de falha são pedidos primeiro e em outras palavras. Isso combina com a explicação oferecida para o InvThink, a de que a enumeração traz à tona conhecimento que um pedido direto deixa sem uso (Kim et al., 2026), e com o achado exploratório de que a frase sobretudo torna regulares riscos ocasionais (Figura 11). O estudo não testa essa leitura. Uma ablação (uma nova rodada com um só elemento mudado) que movesse a mesma frase para depois do plano testaria.

O *trade-off* entre procedimento e cobertura. No estudo de origem, um protocolo de 13 fases cobriu menos riscos de referência em sua tabela final do que esta frase, com três vezes e meia o tempo de execução (Tabela 1). Mais procedimento comprou menos cobertura. Os dois estudos não mostram que a estrutura é inútil; *pipelines* multiagente acrescentam modos de falha próprios

(Cemri et al., 2025). Eles mostram que um protocolo estruturado precisa ser medido contra a frase, que é a linha de base barata, e que ser melhor que um pedido simples não basta.

Planos não são código. Um *prompt* que pede a um agente de código que preveja casos de borda elevou a taxa conjunta de tarefas resolvidas que evitam um defeito posterior num modelo Claude e a reduziu em outro (Patel et al., 2026). Um plano que nomeia um risco não garante código que o trate, e nada na Tabela 5 diz o contrário.

Para a prática. A intervenção é uma frase no início de um pedido de planejamento, e não se observou custo adicional nestas execuções (Tabela 8). Uma equipe que use um agente de código em modo de planejamento pode adotá-la sem ferramental. O que a equipe não deve concluir é que o plano resultante seja seguro: a tabela dele nomeou 58% dos riscos de referência, o trabalho humano de revisão que resta não foi medido, e os doze riscos que nenhuma rodada nomeou (Seção 5.7) lembram que os pontos cegos do agente e a atenção de seus revisores não se sobrepõem por completo.

7. Objeções e ameaças à validade

Esta seção toma as objeções uma a uma. Elas seguem as classes usuais de ameaças na engenharia de software empírica (Wohlin et al., 2012) e as cautelas levantadas para estudos que usam LLMs (Sallou et al., 2024; Baltes et al., 2027). Antes de esta versão ser escrita, o manuscrito e seus dados foram entregues a dois modelos de outros fornecedores, cada um instruído, de forma independente, a agir como revisor hostil; os dois levantaram a objeção 1. Um deles recebeu em seguida a própria frase em estudo, voltada para o manuscrito: antes de este artigo ser publicado, liste as maneiras como ele pode dar errado. Esse pedido produziu a objeção 13, que nenhuma das duas revisões tinha levantado. O repositório guarda os três relatórios (Flores, 2026b).

1. A régua foi obtida com as palavras da frase testada (construto). Os revisores que escreveram as listas de referência foram instruídos a dar “the concrete ways this change could go wrong” (Apêndice A). A condição testada pede ao agente que “list the ways this change could go wrong”, e a condição simples nunca vê essas palavras. Parte do ganho poderia ser, portanto, um casamento entre as respostas testadas e a pergunta que produziu a régua, e não mais dos riscos reais da mudança. As duas condições são pontuadas contra as mesmas listas, o que protege a comparação de diferenças entre listas; não a protege de um tratamento que conduz o agente para o enquadramento dos revisores. Nada nos dados pré-registrados separa as duas leituras. A medida de relevância, que não usa as listas de referência, não mostra redução na relevância média julgada das linhas (Seção 5.4), e isso é só uma resposta parcial. Uma segunda régua, obtida com as palavras do contrato de saída que as duas condições recebem, foi especificada depois de conhecidos estes resultados, num adendo datado selado antes de sua coleta (Flores, 2026b). Contra ela a diferença é de 6,7 pontos ($p < ,001$), positiva nos dois modelos e cerca de quatro décimos menor que a estimativa pré-registrada (Seção 5.8). Sob os outros tratamentos das rodadas com erro e por um juiz de um terceiro fornecedor, ela vai de 4,2 a 9,0 pontos, e só o valor isolado dela no Opus 5.5, no limite oposto, não é significativo (Tabela 10). A objeção é acolhida em parte: o ganho não desaparece quando o pedido aos revisores deixa de usar as palavras da frase, embora o pedido que funde as listas mantenha palavras próximas (Seção 5.8), e o tamanho dele depende, sim, da régua.

2. As listas de referência foram escritas por modelos (construto). A cobertura mede a concordância com o que três LLMs de outros fornecedores esperavam que desse errado, filtrada para os itens que pelo menos dois deles citaram (Seção 4.4). Não é uma medida de risco real. Um risco que os três revisores deixaram passar não pode ser creditado. Tampouco um risco que só um deles nomeou: a fusão mantém um risco quando ao menos dois revisores o nomeiam, o que deixa

de fora 132 dos 232 grupos das listas originais e 102 dos 213 da segunda (Flores, 2026b). Esse filtro pode favorecer os riscos que os modelos tendem a repetir, em prejuízo dos que só um revisor viu, e nenhum humano auditou os grupos deixados de fora. No experimento de campo de Caraeni et al. (2026), planejadores humanos captaram riscos específicos do contexto que o modelo trabalhando sozinho não captou, de modo que uma régua humana poderia mudar os valores absolutos. O primeiro próximo passo da Seção 9 trata disso. A cobertura também dá o mesmo peso a todo risco de referência, qualquer que seja a gravidade da falha que ele nomeia.

3. Um modelo ocupa quatro papéis (construto). O Gemini 3.8 Flash escreveu uma lista de revisor, agrupou os itens, é o juiz principal da cobertura e o único juiz da relevância, e 91 dos 100 riscos de referência levam sua redação (Tabela 3). Um juiz pode casar mais facilmente com a própria redação (Panickssery et al., 2024). O segundo juiz é o controle para isso: o Grok não escreveu nenhuma redação de referência, e acha um ganho de 10,7 pontos contra 11,4 (Tabela 7). A relevância não tem tal controle, o que é uma segunda razão para afirmar apenas que não se observou redução. Na segunda régua, em que 98 dos 111 riscos têm a redação do Gemini, um juiz de um terceiro fornecedor encontra ganho maior que o do Gemini (Tabela 10).

4. O desfecho é a tabela final (construto). O juiz vê só as linhas da tabela. Um agente que lista primeiro os modos de falha pode simplesmente copiar essa lista para a tabela, e o estudo não mediu se o plano em si trata os riscos, se os riscos são verdadeiros para a base de código, nem se o código escrito a partir do plano tem menos defeitos. O estudo mais próximo com desfecho em código achou resultados que dependem do modelo (Patel et al., 2026). A afirmação é sobre o que a tabela de riscos do plano nomeia.

5. O desenho não separa posição de redação (interna). A frase difere do pedido simples no que pede e em quando pede (Apêndice A). A Seção 6 oferece a ordem como uma leitura e não a testa. A ablação é o terceiro próximo passo da Seção 9.

6. As três rodadas com erro favorecem a hipótese (interna). As três caem na condição simples e contam como zero pela regra fixada. Sem elas o resultado se mantém (Tabela 7; Seção 5.5).

7. Tabelas mais longas poderiam explicar um *recall* maior (interna). As tabelas com a frase têm em média 0,4 linha a mais (Tabela 8). Entre as rodadas com as dez linhas completas, a diferença é de 11,3 pontos (Seção 5.4).

8. O juiz poderia perceber a condição (interna). Ao juiz não se informa a condição, e a lista de palavras trocada antes do julgamento não foi pensada para estas duas condições (Seção 4.5). Um juiz poderia inferir a condição pelo estilo de uma tabela. O estudo não tem medida disso. Dois juízes de fornecedores diferentes concordam em 93,2% das decisões (Seção 5.3), o que delimita o ruído do juiz e não um viés que eles possam compartilhar. O juiz também decide todos os riscos de referência de uma resposta numa única chamada, de modo que essas decisões não são independentes (Flores, 2026b).

9. Os repositórios são públicos e podem estar nos dados de treinamento dos modelos (externa). Várias tarefas usam projetos de exemplo amplamente copiados (Tabela 2). O agente pode lembrar armadilhas conhecidas desses projetos, e a frase pode ajudar mais a recuperação do que o raciocínio. As duas condições usam os mesmos repositórios, de modo que isso afeta o significado do efeito e não sua existência; valem as cautelas sobre vazamento de dados em estudos com LLMs (Sallou et al., 2024).

10. Dois modelos de um fornecedor são um teste fraco de generalidade (externa). O Claude Opus 5.5 e o Claude Sonnet 5.5 compartilham fornecedor, *harness* e modo. No estudo de Patel et al. (2026), um *prompt* de previsão moveu dois modelos de uma mesma família em sentidos opostos, de modo que a concordância entre os dois modelos aqui é informativa; ela nada diz sobre modelos de outro fornecedor, outro agente, bases de código privadas ou uma sessão interativa.

11. A frase é meio pré-mortem (construto). O exercício de Klein inclui a suposição de que o plano fracassou (Klein, 2007). A frase a omite, e a condição com o enquadramento completo de Klein, que o protocolo permitia como exploratória, não foi executada (Seção 8). Este artigo usa "pré-mortem" para a família da instrução e não faz nenhuma afirmação sobre a suposição de fracasso.

12. Tempo e autor (interna). A coleta foi pausada por quatro dias e retomada na mesma versão do Claude Code; como a ordem foi um embaralhamento com semente, as duas condições se distribuem pelos dois períodos (Seção 8). O autor escolheu as tarefas conhecendo o resultado anterior, fixou-as antes de as listas de referência existirem (Tabela 4) e mantém o ASP, a ferramenta cujo protocolo esta frase superou (Flores, 2026a).

13. Num dos modelos, as condições diferem numa chamada de ferramenta (interna). No Opus 5.5, o pedido simples levou o agente a salvar um arquivo de plano em 59 de 60 rodadas, contra 22 de 60 com a frase (Seção 5.6). Um agente que gravou o plano num arquivo pode dar uma resposta final mais curta, e a tabela é pontuada a partir da resposta final. O Sonnet 5.5 é o controle: não salvou arquivo de plano em nenhuma rodada, e seu ganho é o maior dos dois, 12,4 pontos (Tabela 5). Entre as rodadas do Opus 5.5 que salvaram o arquivo, a diferença é de 7,5 pontos nos 10 estratos em que as duas condições permanecem (Flores, 2026b). Salvar o arquivo é também uma resposta ao pedido, de modo que essas comparações descrevem e não isolam uma causa. Elas mostram que

o ganho aparece sem a chamada de ferramenta, no Sonnet 5.5; a chamada pode explicar parte do custo menor no Opus 5.5.

14. Três limites apontados na revisão (construto). Com a frase o agente usou menos turnos (Tabela 8), de modo que pode ter lido menos do repositório antes de responder, e este estudo não mede o quanto os riscos listados se apoiam no código. A tabela tem teto de dez linhas e as duas condições quase o preenchem, de modo que a medida é quais riscos entram nos dez primeiros, e não quantos riscos o agente saberia nomear. O juiz decide todos os riscos de referência de uma resposta numa única chamada e em ordem fixa, de modo que um efeito de posição sobre as decisões dele não está excluído (Flores, 2026b).

Validade da conclusão. Uma hipótese principal carrega a decisão, por uma regra conjunta: o teste combinado e o teste dentro de cada modelo precisam todos ficar abaixo de ,05 (Seção 4.6); as medidas secundárias são relatadas como tal, e toda análise não planejada é rotulada como não planejada (Seção 8), a separação que o pré-registro existe para proteger (Nosek et al., 2018). Cinco réplicas por célula dão médias de estrato grosseiras; o teste combina 24 estratos e não depende de nenhum deles (Tabela 6). Os testes tratam as doze tarefas como fixas, e os dois modelos as compartilham; reamostrar tarefas inteiras dá de 8,2 a 14,5 pontos para a diferença pré-registrada e de 2,2 a 11,1 para a segunda régua (Seção 5.8). Todo valor de p e todo intervalo aqui é condicional às listas de referência, à fusão delas e aos vereditos tais como coletados; nenhum inclui a variação que um novo pedido de listas, uma nova fusão ou um novo julgamento acrescentaria, e a distância entre as duas réguas e entre os dois juízes da segunda mostra que essa variação não é pequena.

8. Desvios do protocolo

O arquivo do protocolo registra cada desvio com sua data (Flores, 2026b). Em resumo:

1. **Um conflito de pasta corrigido nos primeiros minutos.** As rodadas numeradas t001 a t090 falharam na etapa de clonagem, antes de qualquer chamada ao modelo, porque a pasta das cópias dos repositórios ainda guardava cópias do estudo anterior. As rodadas t091 a t096, cujas pastas não existiam, rodaram primeiro e foram mantidas. O nome da pasta foi mudado e a coleta foi retomada na ordem com semente. *Prompts*, modelos, tarefas, pontuação e análise não mudaram.
2. **Uma pausa de quatro dias.** A coleta parou depois de 38 rodadas, quando o computador que a executava foi desligado. As rodadas em andamento nesse momento não deixaram registro e rodaram de novo na retomada. O número e as condições delas não foram registrados, de modo que uma seleção por duração entre as rodadas em andamento naquele momento não pode ser medida nem descartada. O computador atualizou o Claude Code durante a pausa; as rodadas restantes rodaram na versão das primeiras 38, instalada em separado.
3. **Caminhos de máquina removidos de três respostas.** As respostas t006, t022 e t023 nomearam um arquivo do computador do operador por seu caminho completo, fora da tabela de riscos. O caminho foi trocado por um marcador, cada arquivo corrigido recebeu uma nova entrada no registro, e, das rodadas retomadas em diante, o executor aplica os mesmos marcadores antes de salvar.
4. **Três rodadas com erro de rede,** tratadas como fixado de antemão e examinadas na Seção 5.5.

5. **Uma pausa na pontuação.** O juiz principal atingiu seu limite de uso depois de 225 de 240 vereditos. Os 15 restantes foram produzidos pelo mesmo modelo, com o mesmo comando, depois que o limite foi renovado. Nenhum juiz foi substituído.

6. **Uma condição exploratória que não rodou.** O protocolo permitia, se sobrasse uso, uma terceira condição com o enquadramento completo de Klein: “Imagine it is a month from now and this change has failed. Before planning, list why it failed.” (em português: “Imagine que se passou um mês e esta mudança fracassou. Antes de planejar, liste por que ela fracassou.”) Ela não tem rodadas e não tem relação com o resultado.

7. **Análises não planejadas,** cada uma rotulada onde aparece: a análise de sensibilidade e seu limite oposto, o intervalo *bootstrap*, o κ de Cohen, a comparação de tabelas completas, a comparação do arquivo de plano e a descrição por item.

8. **A segunda régua,** exploratória e especificada depois do resultado (Seção 5.8). O juiz principal atingiu o limite de uso depois de 208 de seus 240 vereditos, que foram retomados depois da renovação com o mesmo modelo e o mesmo comando. O segundo juiz atingiu o limite depois de 11 vereditos, e o limite não tinha sido renovado quando esta versão foi escrita. Três checagens dela foram acrescentadas depois, a pedido de um revisor (Seção 5.8).

9. **Uma repetição das três rodadas com erro e um terceiro juiz,** ambos não planejados e fixados num segundo adendo antes de qualquer chamada (Seções 5.5 e 5.8). O juiz principal atingiu o limite de uso na linha de comando depois de 4 de seus 9 vereditos sobre as repetições. Por decisão do autor, registrada num terceiro adendo antes das chamadas, os 9 foram produzidos pelo mesmo modelo por uma API paga, com uma conferência de canal em 30 rodadas seladas. Esse adendo dizia que a resposta completa

da API seria guardada com cada veredito; o código guardou o texto da resposta, o modelo, o provedor, a contagem de tokens e o identificador da chamada. Os vereditos da linha de comando continuam devidos.

9. Conclusão

Uma frase que abre um pedido de planejamento no lugar de “Plan this change.”, “Before planning, list the ways this change could go wrong.”, elevou de 46% para 58% a parcela de riscos listados de forma independente que o plano de um agente de código cobre, num estudo confirmatório pré-registrado de 240 rodadas, nos dois modelos testados e por dois juízes (Tabela 5). Os riscos listados não foram julgados menos concretos, e não se observou custo adicional (Seção 5.4; Tabela 8). O achado replica um resultado secundário de um estudo anterior em tarefas novas e num segundo modelo (Flores, 2026a).

Os limites são os da Seção 7: a régua e os juízes eram modelos, a régua foi obtida com palavras próximas às da frase, e uma segunda régua, obtida com outras palavras, dá um ganho menor, de 6,7 pontos (Seção 5.8), os dois agentes compartilham fornecedor, e o desfecho é o que a tabela de um plano diz, não o que o código faz. Os próximos passos decorrem deles, na ordem pretendida: (i) um engenheiro humano pontuando uma amostra das listas de referência e das decisões dos juízes, e riscos de referência tirados de defeitos reais, que nenhum *prompt* provocou; (ii) o mesmo desenho num modelo de outro fornecedor; (iii) uma ablação que separe listar riscos *primeiro* de listá-los de todo, e que acrescente a suposição de fracasso de Klein (Klein, 2007); e (iv) um desfecho em código, implementando os planos das duas condições e rodando os testes dos projetos, como Patel et al. (2026) fazem para um *prompt* mais longo.

10. Disponibilidade de dados e código

O material por trás de cada número deste artigo está no repositório público anticipating-shadow-points (Flores, 2026b), na pasta benchmark/study-2026-11b do ramo v2, a partir do *commit* d880cf0; a *release* 2.0.0, a versão citada, é anterior a este estudo. A Tabela 11 descreve a pasta.

Um único comando, python run_all.py, recalcula cada resultado do estudo relatado neste artigo a partir das respostas e dos vereditos guardados e não chama nenhum modelo; exige Python 3.11 ou mais recente (Flores, 2026b). Repetir as rodadas exige uma assinatura do Claude e acesso aos dois modelos juízes, que podem ter mudado desde novembro de 2026.

Tabela 11
O que o repositório contém para este artigo

Material	**Onde no repositório**	**O que o leitor pode conferir com ele**
Protocolo, desvios e os três adendos	PROTOCOL.md, yardstick2/, addendum2/, addendum3/	O que foi fixado antes dos dados e o que mudou depois
Tarefas e *prompts*	tasks.json, run.py, prompts/	O texto exato que cada rodada e cada revisor recebeu
Listas de referência e sua fusão	lists/	Os 369 itens dos revisores e os 100 riscos de referência
As 240 respostas e os registros de execução	runs/	Cada resposta e o registro de cada execução
Os 720 vereditos	judge/	Cada decisão, com o *prompt* que a produziu
Análise e verificações	analyze.py, sensitivity.py, describe.py; tests/test_paper3_numbers.py	Como cada número é calculado, e que o texto confere com os dados
Registro encadeado e carimbos de tempo	ledger.jsonl, *.ots	Que cada arquivo corresponde à sua última entrada, e a ordem dos eventos selados, nos limites dados na Seção 4.7

Material	Onde no repositório	O que o leitor pode conferir com ele
Revisões externas deste manuscrito	docs/paper/review/	Os relatórios por trás da Seção 7
O estudo de origem	benchmark/study-2026-11/	O mesmo material para as 90 rodadas da Seção 3

Nota. Caminhos sem pasta na frente estão dentro de benchmark/study-2026-11b/ (Flores, 2026b).

Os registros de execução tiveram os caminhos de máquina removidos.

11. Declarações

Conflito de interesses. O autor mantém o ASP, o *plugin* cujo protocolo o estudo de origem comparou com a frase testada, e dirige o laboratório que o publica. O resultado relatado aqui é desfavorável a esse protocolo (Flores, 2026a).

Financiamento. Este trabalho não recebeu financiamento externo. Os custos de uso dos modelos foram pagos pelo autor.

Contribuição do autor. Autor único: concepção, desenho, coleta de dados, análise, redação e revisão.

Uso de ferramentas de IA. Modelos de linguagem têm quatro papéis neste estudo, todos descritos na Seção 4: os agentes testados, os redatores das listas de referência, os juízes e os revisores cegos do manuscrito (Flores, 2026b). Um assistente de IA também foi usado na programação, na análise e na redação e tradução do texto, sob direção do autor. O autor conferiu cada número contra os dados e responde pelo conteúdo.

Referências

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

Apêndice A — Os pedidos exatos

A Tabela A1 dá os pedidos principais do estudo confirmatório; os pedidos que fundem as listas e julgam a relevância, e o segundo pedido aos revisores, estão nos arquivos que a Tabela 11 lista, e cada cliente acrescenta instruções próprias que não são mostradas. Chaves marcam as únicas partes que mudam de uma chamada para outra. Os arquivos que guardam esses textos estão listados na Tabela 11 (Flores, 2026b).

Tabela A1
Os pedidos enviados ao agente, aos revisores e ao juiz

Quem recebe	Texto
O agente, condição simples	{change request} Plan this change. {output contract}
O agente, condição com a frase	{change request} Before planning, list the ways this change could go wrong. {output contract}
O contrato de saída, nas duas condições	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.
Cada revisor	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.
O juiz da cobertura	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. As quebras de linha foram removidas. A última frase do juiz é seguida, no original, de um exemplo do JSON esperado.

Apêndice B — Uma lista de referência completa

A Tabela B1 mostra a lista de referência da tarefa U08, a mais curta das doze, com a parcela das rodadas de cada condição cuja tabela cobre cada risco pelo juiz principal. Ela ilustra o padrão da Seção 5.7: riscos que todo plano cita nas duas condições (linhas 1 e 5), riscos que a frase torna regulares (linhas 4 e 6) e um risco que as duas condições quase sempre deixam passar (linha 2). As listas completas estão no repositório (Flores, 2026b).

Tabela B1
Os seis riscos de referência da tarefa U08 (mudar o digest de assinatura padrão de SHA-1 para SHA-256) e sua cobertura

#	Risco de referência, abreviado	Revisores que o citaram	Simples	Frase
1	Tokens assinados por versões anteriores com as configurações padrão falham na validação depois da atualização, invalidando sessões em projetos dependentes como o Flask.	3	100%	100%
2	Mudar o padrão no *signer* e não na classe do algoritmo HMAC deixa os dois padrões fora de sincronia.	3	10%	20%
3	A derivação de chave também usa o *digest*, de modo que a chave derivada muda de valor e passa de 20 para 32 bytes.	3	40%	50%
4	Testes afirmam saídas que casam com assinaturas SHA-1.	2	70%	100%
5	As assinaturas passam de 27 para 43 caracteres e podem estourar colunas de banco de dados, limites de URL ou tamanhos de *cookie*.	2	100%	100%
6	Exemplos de documentação e *doctests* guardam saídas SHA-1 e afirmam que SHA-1 é o padrão.	2	40%	90%

Nota. Dez rodadas por condição (cinco por modelo). Os seis riscos levam a redação do revisor Gemini. Aqui, token é um valor assinado; o *digest* é a função de hash usada na assinatura; o *signer* é a classe que assina; HMAC é um método padrão de assinar com uma chave secreta e uma função de hash; *doctests* são exemplos da documentação que rodam como testes.

Apêndice C — Os textos em inglês, traduzidos

Todo texto que um modelo recebeu neste estudo estava em inglês, e o artigo o reproduz como foi enviado. Este apêndice traduz esses textos para quem lê em português: a Tabela C1 traz os doze pedidos de mudança da Tabela 2, e a Tabela C2, os pedidos da Tabela A1. As traduções são do autor e não foram usadas em nenhuma rodada (Flores, 2026b).

Tabela C1
Os doze pedidos de mudança, traduzidos

Tarefa	Pedido de mudança, em português
U01	Exigir que novos usuários confirmem o endereço de e-mail antes de poderem entrar.
U02	Permitir que os autores mudem o título de um artigo, e com ele o *slug* da URL, sem quebrar os *links* já compartilhados.
U03	Permitir que moderadores ocultem comentários abusivos e que o autor do comentário recorra em até 7 dias.
U04	Bloquear usuários pela combinação de nome de usuário e endereço IP, e não só pelo endereço IP, como novo padrão.
U05	Fazer as sessões se renovarem por padrão, de modo que cada requisição estenda a validade da sessão.
U06	Permitir que cada usuário baixe todos os seus dados (perfil, publicações e comentários) num único arquivo.
U07	Trocar por UUIDs os identificadores inteiros que a API expõe, mantendo funcionando os clientes atuais da API.
U08	Mudar o algoritmo padrão de assinatura de SHA-1 para SHA-256.
U09	Guardar as senhas com um esquema de hash mais forte, sem obrigar os usuários atuais a redefinir a senha.
U10	Fazer cada lista de tarefas e cada item pertencerem a uma organização (*tenant*), de modo que cada empresa veja só os próprios dados.
U11	Permitir que donos de animais marquem consultas pela internet, sem que duas consultas sejam marcadas para o mesmo veterinário no mesmo horário.
U12	Tornar o *cookie* do *middleware* de sessão Secure e SameSite=Strict por padrão.

Tabela C2

Os pedidos enviados ao agente, aos revisores e ao juiz, traduzidos

Quem recebe	Texto, em português
O agente, condição simples	{pedido de mudança} Planeje esta mudança. {contrato de saída}
O agente, condição com a frase	{pedido de mudança} Antes de planejar, liste as maneiras como esta mudança pode dar errado. {contrato de saída}
O contrato de saída, nas duas condições	Execução não interativa: declare suas suposições em vez de perguntar. Não edite nenhum arquivo. Responda em inglês. Termine a resposta com os principais riscos desta mudança numa tabela markdown com as colunas #, Risco, Por que pesa aqui e Mitigação (no máximo 10 linhas, células curtas) e, depois, uma linha começando com “Riskiest:” que aponte o mais arriscado.
Cada revisor	Você é um engenheiro de software sênior revisando um pedido de mudança antes que alguém o implemente. O repositório na pasta atual é a base de código. Leia os arquivos de que precisar. Não modifique nenhum arquivo. Pedido de mudança: {pedido de mudança} Liste as maneiras concretas como esta mudança poderia dar errado nesta base de código se fosse implementada sem cuidado: os problemas que um engenheiro sênior cuidadoso gostaria de ver tratados antes de começar. Cada item deve ser específico deste repositório e deste pedido, com uma ou duas frases. Responda com uma lista numerada de 6 a 12 itens e nada mais.
O juiz da cobertura	Um plano de mudança de software termina com uma tabela de riscos. Decida, para cada problema de referência, se alguma linha da tabela trata do mesmo problema de fundo (cita-o como risco ou o mitiga especificamente), ainda que redigido de outro modo. Pedido de mudança: {pedido de mudança} Problemas de referência: {riscos de referência numerados} Linhas da tabela de riscos: {linhas} Responda diretamente a partir do texto acima; não rode comandos nem use ferramentas. Responda só com JSON, uma chave por número de referência.

Nota. Os textos originais, em inglês, estão na Tabela A1.

Nota Técnica

Versão 1.0 · novembro de 2026

Série Codex Hash Research Report 2026-03

Licença Texto, figuras e dados: CC BY 4.0. Software complementar: MIT.

O projeto ASP. O ASP (Anticipating Shadow Points) é um *plugin* de código aberto para o Claude Code que faz o agente listar o que pode dar errado antes de escrever código. O *plugin* e o material de replicação ficam num único repositório público (Flores, 2026b). Um *whitepaper* descreve como o *plugin* funciona e relata a primeira medição de se ele melhora os planos de um agente, que não achou benefício mensurável sobre um pedido de uma frase (Flores, 2026a). Este artigo é o estudo confirmatório dessa frase que o *whitepaper* anunciou.

DOI. Um DOI (Digital Object Identifier) é um *link* permanente para uma obra depositada. O Zenodo, arquivo público de pesquisa mantido pelo CERN, guarda este artigo como registro próprio; esta versão, 1.0, é https://doi.org/10.5281/zenodo.24410437. O *plugin*, os dados e o código formam um registro separado (Flores, 2026b), cujo DOI de conceito, https://doi.org/10.5281/zenodo.20276631, sempre abre a *release* mais recente.

Replicação. Todo número que este artigo calcula a partir de seus dados pode ser recalculado a partir do repositório sem chamar um modelo; a Seção 10, Disponibilidade de dados e código, lista o que há lá e como executar.

Sobre o Autor

Carlos Ulisses Flores é CTO e Pesquisador-Chefe do Codex Hash Research Laboratory, um laboratório independente de P&D em São Paulo, Brasil. Constrói sistemas de produção em que a falha silenciosa é cara — historicamente em trading algorítmico, IoT embarcada e criptografia de borda, e atualmente em ferramentas de IA agêntica. Este artigo nasceu de medir o próprio *plugin* de planejamento para agentes de código e descobrir que uma frase funcionava melhor.

Identificadores · ORCID: 0000-0002-6034-7765 · Lattes (registro nacional de currículos de pesquisadores do Brasil): 6905246706890561

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