00 — Introdução

🕒 estado da arte 2026-07 · revisão 2026-08-01 · 📖 ~7 min de leitura

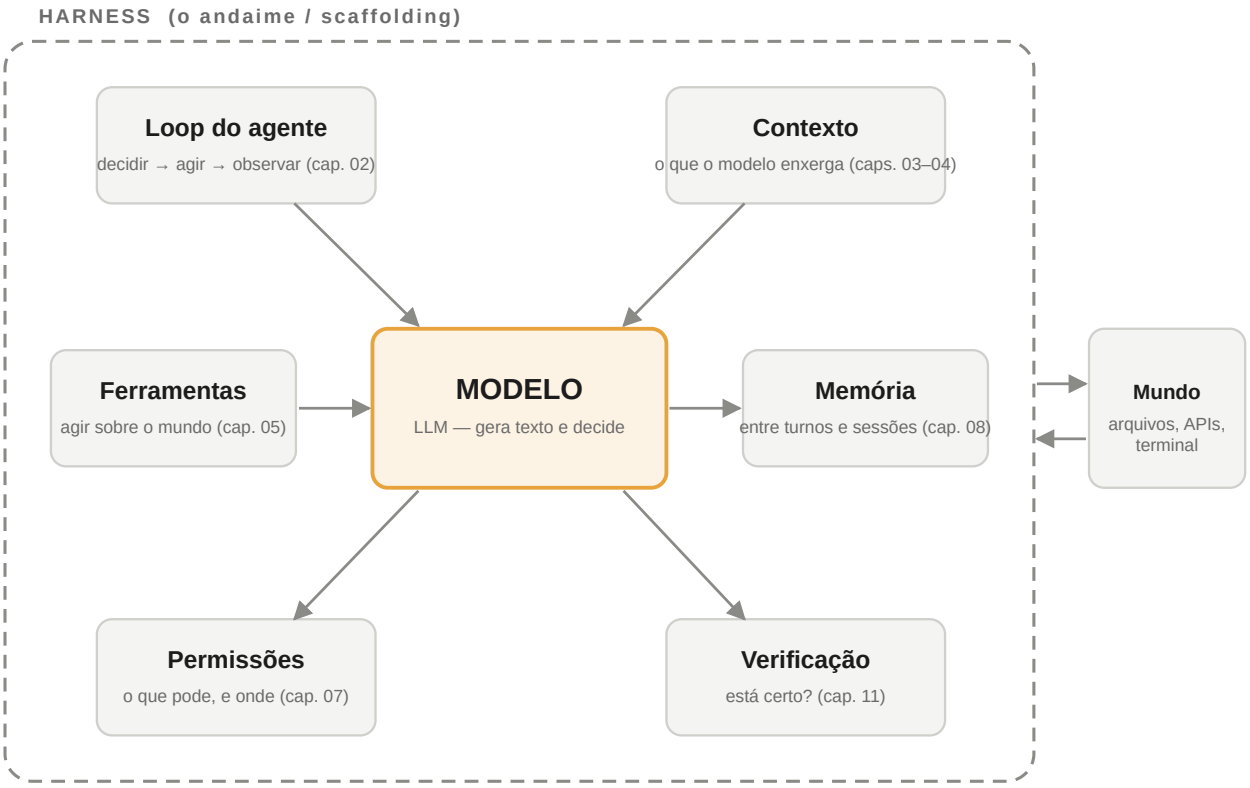
Agente = modelo + harness

Comece por uma pergunta que qualquer pessoa que já usou um chat de IA consegue fazer: por que o ChatGPT *responde* sobre o seu problema, mas não *resolve* o seu problema? Ele explica como corrigir o bug — mas não abre o arquivo, não roda o teste, não confere se funcionou. A resposta curta: um chat é só o **modelo**. Para o modelo *agir* — mexer em arquivos, executar comandos, verificar o próprio trabalho e parar na hora certa — é preciso construir uma estrutura inteira em volta dele. Essa estrutura é o assunto deste livro.

Quando um agente de IA resolve uma tarefa real — corrigir um bug, migrar um módulo, responder com base em dezenas de arquivos — duas coisas distintas estão trabalhando. A primeira é o **modelo**: a rede que lê contexto e decide o próximo passo. A segunda é tudo o que está em volta dele: quem monta o contexto que ele lê, quem executa as ferramentas que ele invoca, quem decide o que ele pode ou não fazer, quem lembra o que aconteceu ontem, quem verifica se o resultado está certo. Esse "tudo em volta" é o **harness** — em tradução livre, o arreio, o andaime, o *scaffolding*.

A fórmula que organiza este livro é simples:

> **agente = modelo + harness**



O modelo no centro; o harness — o andaime — em volta. Cada bloco é um capítulo deste livro.

O modelo é intercambiável e melhora a cada geração. O harness é engenharia de software clássica — e é nele que a maioria dos agentes falha ou tem sucesso. Dois produtos usando exatamente o mesmo modelo entregam resultados radicalmente diferentes conforme a qualidade do harness: como o contexto chega ao modelo, quais ferramentas ele tem, como os erros retornam, o que acontece quando a **janela de contexto** (o limite de texto que o modelo consegue "enxergar" de uma vez) acaba.

Engenharia de harness é a disciplina de projetar esse scaffolding: entrega de contexto, interfaces de ferramentas, artefatos de planejamento, loops de verificação, sistemas de memória e sandboxes.

Por que um livro — e por que agora

Entre 2024 e 2026, os harnesses de agentes de código deixaram de ser experimentos e viraram uma categoria de produto: Claude Code, Codex CLI, Gemini CLI, opencode, Aider, Cline, Goose, OpenHands e dezenas de outros. O mais notável não é a quantidade, mas a **convergência**: projetos independentes, em linguagens diferentes, chegaram às mesmas soluções — arquivos de contexto hierárquicos, compactação em camadas, plan mode como modo de permissão, hooks de ciclo de vida, MCP (Model Context Protocol) como padrão de integração.

Quando implementações independentes convergem, existe uma disciplina por trás. Este livro documenta essa disciplina.

O método: ler código, não marketing

Este livro é empírico. Cada capítulo trata de uma funcionalidade do harness (o loop, o contexto, a compactação, as permissões...) e é escrito a partir da leitura do código-fonte de harnesses reais de código aberto. A regra editorial mais importante do projeto:

> Afirmações sobre um harness exigem **evidência**: o caminho do arquivo no código-fonte onde a funcionalidade está implementada.

READMEs prometem; código entrega. Vários projetos anunciam dimensões que o código não tem — a exigência de evidência é o que separa avaliação de marketing.

Nota de autoria e método

Por transparência — e coerência com a regra de evidência acima — este livro é **co-escrito com um agente de IA** (Claude Code, da Anthropic) operando sob **autoria, curadoria e responsabilidade humanas**. O agente executa a pesquisa, a redação e o ciclo de produção; o autor humano define o escopo, decide, **verifica cada fonte** e responde pelo conteúdo. Seguindo as políticas editoriais de autoria (ICMJE, COPE, *Nature*, *Science*), a IA **não** é listada como autora — não pode ser responsável — e seu uso é divulgado aqui, na abertura.

Isso não é um detalhe: um livro sobre a disciplina de instrumentar bem os agentes de IA usa essa mesma disciplina para se escrever, e a expõe. O método completo — pesquisa dupla verificada por busca cruzada, ciclo spec-driven, revisão e datação — está documentado no Guia Editorial §6, com um *survey* das metodologias de escrita tradicionais e da era-IA que o fundamentam.

Como ler este livro — três portas de entrada

O livro foi escrito para ser denso; esta seção existe para que a densidade não seja uma parede. Escolha a sua porta:

- **Se você está chegando agora** (usou chats de IA, mas nunca construiu um agente): leia 00 → 01 → 02 em sequência, sem pressa, usando o Glossário como apoio — toda sigla do livro está lá, por extenso e explicada (na versão online, basta passar o mouse sobre a sigla). Depois do 02, os capítulos 03–13 podem ser lidos em qualquer ordem: cada um é autocontido e abre definindo o próprio problema.
- **Se você já opera um agente** (usa Claude Code, Codex, Cursor ou similares e quer entender o que há por dentro): a **Leitura executiva** ao fim de cada capítulo é o seu atalho — o estado da arte da dimensão em um parágrafo, com a seção "o que roubar". Vá direto aos capítulos do seu interesse e desça ao corpo quando quiser a evidência.
- **Se você constrói harnesses**: o livro inteiro é seu, incluindo os Apêndices A (evidência por repositório, com caminhos de arquivo), o Benchmark e as duas trilhas práticas — o **harness-zero** (construção

didática, uma feature por etapa) e o **harness-um** (a implementação de referência completa, apêndice próprio).

Estrutura do livro

- **Fundamentos** (capítulo 01): as definições formais, os artigos canônicos e a taxonomia de problemas que organiza tudo o que vem depois.
- **Capítulos 02–13**: uma funcionalidade por capítulo. Cada um define o problema, apresenta os padrões de implementação conhecidos e mostra, com evidência, como cada harness estudado implementa.
- **Convergências e tendências** (capítulo 14): o que a indústria já padronizou, onde ainda há divergência real, e a "cláusula de expiração" — a tese de que todo componente de harness existe porque o modelo ainda não faz aquilo sozinho, e deve ser desenhado sabendo que um dia será desnecessário.
- **Capítulos 15–17**: as fronteiras — o harness embutido em produto (15), o harness que aprende com o uso (16) e a camada de protocolos que une o ecossistema (17).
- **Benchmark** (`benchmark/`): a seção empírica — avaliações padronizadas, por dimensão, com notas 0–3 e evidência, de cada harness estudado, mais o comparativo consolidado.

Os harnesses do estudo

O estudo cobre, até esta edição, **vinte e um sistemas de código aberto**, avaliados por leitura sistemática de código em cinco arquétipos (o método está no capítulo 01, §6):

- **Harnesses de código** — opencode, gemini-cli, OpenHarness, Codex CLI, Goose, Aider, OpenHands, Grok Build, Pi, Kimi Code e Prime Agent;
- **Agentes pessoais self-hosted** — OpenClaw, Hermes Agent, IronClaw, ohmo;
- **Agentes organizacionais** — QM;
- **Harnesses embutidos** — n8n (nó AI Agent);
- **Frameworks** — LangGraph, CrewAI, OpenAI Agents SDK (Software Development Kit), Software Agent SDK.

Cada um foi escolhido por representar um *arquétipo* diferente (lógica de replicação, não amostragem): produto maduro agnóstico de provedor (opencode), regime de controle de big tech (gemini-cli), port didático legível (OpenHarness), sandbox-first (Codex CLI), MCP-nativo (Goose), context-first (Aider), cultura de eval acadêmica (OpenHands), agente da organização inteira com o loop trocável (QM), e assim por diante.

A lista completa — com **origem, versão, fork e commit exatos lidos** em cada avaliação, e o link para a análise e o diagnóstico de cada um — está no **Apêndice — O estudo**. O placar consolidado por dimensão está no Comparativo.

Como referencial teórico e para explorar o ecossistema além do corpus, soma-se a coleção viva **Awesome Harness Engineering** (~426 recursos organizados por problema, na mesma organização deste livro) — de onde vêm a definição de harness usada no capítulo 01 e a taxonomia que estrutura os capítulos.