14 — Convergências e Tendências

🕒 estado da arte 2026-07 · revisão 2026-07-28 · 📖 ~7 min de leitura

Objetivos de aprendizagem

Ao final deste capítulo, você deve ser capaz de:

1. **Enumerar** as oito convergências arquiteturais da primeira rodada e **explicar** por que convergência independente sinaliza disciplina consolidada;
2. **Distinguir** as dimensões consolidadas das dimensões em divergência real, e **justificar** por que a contenção é a divergência mais consequente;
3. **Aplicar** a cláusula de expiração a um componente de harness qualquer — identificando por que ele existe e sob que condição expira;
4. **Avaliar** um harness novo contra o checklist de convergências, cobrando justificativa para cada ausência;
5. **Antecipar** as tendências a acompanhar nas próximas rodadas e o que cada uma implicaria para o desenho de harnesses.

O problema

Os capítulos anteriores analisaram o harness dimensão por dimensão — contexto, compactação, tools, permissões, loop. Falta a pergunta que dá sentido ao conjunto: **o que é acidente de implementação e o que é anatomia da disciplina?** Sem essa síntese, cada capítulo é um catálogo de escolhas; com ela, o leitor ganha um critério de projeto — saber o que copiar sem hesitar, onde ainda cabe apostar diferente, e o que vai desaparecer quando os modelos melhorarem.

O instrumento de medida é a convergência independente. Quando equipes que não se coordenam, em stacks e culturas diferentes, chegam à mesma arquitetura, isso é evidência forte de que o problema — e não a moda — determinou a solução. E o instrumento de projeção é a cláusula de expiração do capítulo 01: todo componente de harness é uma prótese para uma limitação atual do modelo, e portanto todo componente deve declarar quando espera se tornar desnecessário.

O estado da arte

O achado central da primeira rodada: oito convergências

Três harnesses, três stacks (Effect-TS, TypeScript, Python), três origens (startup independente, big tech, academia/porta didática) — e uma convergência arquitetural notável. Sem coordenação, os três chegaram a:

1. **Arquivo de contexto hierárquico na raiz do projeto** — `AGENTS.md` / `GEMINI.md` / `CLAUDE.md`: o mesmo artefato com três nomes (cap. 03).

2. **Compactação em escada** — truncar tools → prune → sumarizar via LLM, com disparo automático por limiar (cap. 04).
3. **Schema de tools derivado de tipos** — Effect Schema, classes declarativas, Pydantic: ninguém escreve JSON Schema à mão (cap. 05).
4. **MCP como integração padrão** — três clientes completos sobre os SDKs oficiais (cap. 06).
5. **Plan mode como modo de permissão** — read-only imposto pelo sistema de permissões, não pedido ao modelo (cap. 09).
6. **Hooks de ciclo de vida** — before/after tool, compactação, sessão (cap. 12).
7. **Headless com saída estruturada** — `-p` + JSON/NDJSON para scripting e CI (cap. 13).
8. **Parada por ausência de tool-call + limite de turnos** — a mecânica universal do loop (cap. 02).

Quando implementações independentes convergem assim, a anatomia está consolidada: **isto é a disciplina**, não mais um conjunto de escolhas idiossincráticas. Um harness novo que não implemente os oito itens acima precisa justificar cada ausência.

Onde ainda há divergência real

As dimensões sem consenso são o mapa das apostas em aberto:

- **Contenção** (cap. 07): política + sandbox de SO obrigatórios (gemini-cli), política + paths sensíveis fixos (OpenHarness), ou só política (opencode)? A divergência mais consequente — é a que define o risco operacional.
- **Multi-agente** (cap. 10): ferramenta pontual, serviço com registry, ou time persistente com mailbox? Três filosofias incompatíveis; o vencedor depende de quão bons os modelos ficarão em coordenação.
- **Quem decide continuar** (cap. 02): heurística estrutural ou uma inferência extra por turno (next-speaker check)?
- **Neutralidade de modelo** (cap. 12): ~26 provedores (opencode) contra vitrine de um ecossistema (gemini-cli). Aposta comercial, não técnica — mas define quem sobrevive à comoditização dos modelos.
- **Evals comportamentais** (cap. 11): na rodada 1, só um dos três tratava comportamento do agente como superfície de regressão — a rodada 2 confirmou a previsão e a lacuna fechou (ver cap. 11). Previsão fácil: em dois anos, isso será tão obrigatório quanto CI.

A cláusula de expiração, aplicada

Retomando a tese do capítulo 01 — todo componente de harness é uma prótese para uma limitação atual do modelo. O exercício que todo harness deveria fazer, aplicado ao que estudamos:

Componente	Existe porque...	Expira quando...
Compactação	janelas são finitas e caras	contexto longo ficar barato e confiável
Plan mode	modelos agem precipitadamente	modelos planejarem espontaneamente sob risco
Next-speaker check	o modelo não sinaliza bem o fim do turno	protocolos de turno nativos do modelo
Policy engine / aprovações	modelos não são confiáveis com ações destrutivas	confiabilidade calibrada e verificável
Prompt por família de modelo	modelos respondem diferente a instruções	convergência de instruction-following
Subagente para exploração	dumps de arquivos poluem o contexto	contexto abundante + atenção robusta
Repo-map / índices de código	o modelo não "carrega" o repo inteiro	contexto de milhões de tokens utilizável

O que **não** expira: sandbox (contenção é sobre o mundo, não sobre a capacidade do modelo), interfaces, verificação do trabalho (testes/LSP — verdade externa ao modelo), e os protocolos de interoperabilidade (MCP, A2A, formatos de skill). A engenharia de harness de longo prazo mora aí: **na fronteira entre o agente e o mundo, não na muleta para a limitação do modelo**.

Tendências a acompanhar nas próximas rodadas

1. **Padronização do arquivo de contexto** — a pressão por `AGENTS.md` neutro cross-vendor.
2. **Skills/plugins portáveis** — o OpenHarness já carrega skills do formato Claude Code; um "MCP da extensibilidade" está se formando.
3. **Agente-como-serviço** — A2A server, agent cards, SDKs: harnesses expondo-se uns aos outros.
4. **Segurança como dimensão de primeira classe** — parsing de shell, trusted folders, evals de injection: hoje exceção, amanhã baseline (hipótese confirmada na rodada 2 com o Codex CLI).
5. **Reversibilidade** — checkpoint git com `/rewind`: quando desfazer é barato, a política pode ser mais frouxa; espere mais harnesses copiando.
6. **O harness mínimo** — na contramão da sofisticação, projetos como mini-swe-agent (~100 linhas) testam quanto do *scaffolding* (andaime) o modelo moderno já dispensa. É a cláusula de expiração virando experimento.

LEITURA EXECUTIVA

- Oito dimensões já convergiram entre implementações independentes — são o checklist mínimo de um harness sério; ausências exigem justificativa.
- As divergências reais (contenção, multi-agente, next-speaker, neutralidade de modelo, evals comportamentais) são o mapa das apostas em aberto — contenção é a de maior consequência operacional.

- A cláusula de expiração separa próteses temporárias (compactação, plan mode, repo-map...) do que é permanente: sandbox, interfaces, verificação externa e protocolos de interoperabilidade.
- O valor de longo prazo da engenharia de harness está na fronteira agente–mundo; o resto muda de dono ou desaparece conforme os modelos melhoram.
- Este capítulo é o placar vivo do livro: cada rodada do benchmark confirma convergências, resolve divergências ou aposenta componentes expirados.

> **Consulte também**: a coleção viva Awesome Harness Engineering — Foundations reúne mais recursos consultáveis desta dimensão, curados por problema.

Verificação

1. Por que a convergência **independente** (três stacks, três origens) é evidência mais forte de consolidação do que a adoção de um padrão por vários projetos que se copiam? (Releia "O problema" e o achado central.)
2. Um harness novo não implementa plan mode nem arquivo de contexto na raiz. Segundo este capítulo, qual é a postura correta ao avaliá-lo — e o que você exigiria do autor?
3. Aplique a cláusula de expiração a um componente que **não** está na tabela (por exemplo, o next-speaker check já está; escolha hooks de ciclo de vida ou headless): ele existe por limitação do modelo ou por necessidade da fronteira agente–mundo? Ele expira?
4. Entre as cinco divergências listadas, qual define o risco operacional e qual é uma aposta comercial em vez de técnica? Justifique com o texto.