17 — A Camada de Protocolos

🕒 estado da arte 2026-07 · revisão 2026-07-31 · 📖 ~9 min de leitura

Objetivos de aprendizagem

Ao final deste capítulo, você deve ser capaz de:

1. **Explicar** por que a camada de protocolos é o que transforma um mercado de silos em um ecossistema — e por que cada protocolo padroniza uma *fronteira* diferente do harness;
2. **Distinguir** as fronteiras cobertas por MCP (Model Context Protocol), A2A (Agent-to-Agent) e ACP (Agent Client Protocol), agentskills.io e AGENTS.md — incluindo as duas confusões clássicas (os dois "ACP"; MCP × A2A como vertical × horizontal);
3. **Analisar** a matriz de adoção medida no código e localizar um harness real nela;
4. **Avaliar** a saúde de um protocolo por adoção medida e governança (fundação neutra × vendor único), em vez de por marketing;
5. **Decidir** quais protocolos um harness novo precisa falar para não ficar fora das arquiteturas de composição dos outros.

O problema

Os capítulos 02–16 tratam do que acontece *dentro* de um harness. Este capítulo trata do que acontece *entre* eles — e entre harnesses e o resto do mundo. Sem protocolos compartilhados, cada harness é um silo: suas ferramentas, suas instruções de projeto, seus subagentes e suas skills só funcionam dentro dele. A camada de protocolos é o que transforma esse mercado de silos em um ecossistema: cada protocolo padroniza uma fronteira diferente do harness — agente ↔ ferramenta, agente ↔ agente, agente ↔ editor, agente ↔ usuário, além dos formatos transversais de conhecimento procedural (SKILL.md) e de instruções de projeto (AGENTS.md).

A consequência prática: em um mercado que *compõe* harnesses, não falar os protocolos não é perder uma feature — é ficar de fora das arquiteturas dos outros.

O estado da arte

O mapa: um protocolo por fronteira

O mapa, organizado pela fronteira que cada um resolve:

Protocolo	Fronteira	Origem / governança	Estado (2026)
MCP (Model Context Protocol)	agente ↔ **ferramentas/dados**	Anthropic → adoção universal (OpenAI, Google, Microsoft)	maduro; ~97M downloads
A2A (Agent-to-Agent)	agente ↔ **agente** (delegação entre organizações)	Google → **Linux Foundation** (v1.0 em 2026)	consolidando; absorveu o ACP (Agent Communication Protocol) da IBM
ACP (Agent Client Protocol)	agente ↔ **editor/cliente**	Zed	adoção rápida entre harnesses de código
agentskills.io (Agent Skills / SKILL.md)	**conhecimento procedural** portável	Anthropic (spec aberta, dez/2025)	~40 produtos compatíveis em 6 meses
AGENTS.md	**instruções de projeto** portáveis	comunidade → **Agentic AI Foundation** (Linux Foundation)	60.000+ repositórios; 20+ ferramentas leem nativamente
AG-UI	agente ↔ **interface de usuário**	comunidade (CopilotKit)	emergente
ACP-IBM (Agent Communication Protocol)	agente ↔ agente	IBM	**encerrado** — fundido ao A2A (ago/2025)

Duas confusões a desfazer: (1) "ACP" designa dois protocolos distintos — o da IBM (comunicação agente-agente, descontinuado em favor do A2A) e o da Zed (agente-editor, vivo e em expansão); neste livro, ACP = Zed. (2) MCP e A2A não competem: MCP é a conexão *vertical* (agente→ferramenta), A2A é a *horizontal* (agente→agente peer) — um sistema real usa os dois.

A matriz de adoção — medida no código, não no marketing

O diferencial deste capítulo: cruzamos os protocolos com as **11 avaliações de harnesses do benchmark** (mais os 4 frameworks da rodada frameworks-1) (evidência por arquivo, ver `benchmark/avaliacoes/`). Nenhum comparativo externo tem esta coluna de verdade:

Harness	MCP client	MCP server	ACP	A2A	SKILL.md / agentskills	AGENTS.md (ou equiv.)
opencode	✅	—	✅ (Zed)	—	parcial	✅ AGENTS.md
gemini-cli	✅	—	✅	✅ **client+server**	✅	GEMINI.md
OpenHarness	✅	—	—	—	✅ (formato Claude)	CLAUDE.md
Codex CLI	✅	✅	—	—	✅	✅ AGENTS.md
Goose	✅	✅ (`goose mcp`)	✅ (desktop inteiro)	—	✅	✅ AGENTS.md + .goosehints
Aider	❌	❌	—	—	—	✅ (leitura)
OpenHands	✅	✅ (FastMCP)	✅ (perfis)	—	✅ (repos org)	microagents
OpenClaw	✅	✅	✅ (orquestra terceiros)	—	✅ (52 bundled)	✅ AGENTS.md + SOUL.md
Hermes	✅	✅	✅	—	✅ (núcleo do learning)	✅ AGENTS.md + SOUL.md
IronClaw	✅	—	—	—	✅ (compat OpenClaw)	identity files
n8n	✅	✅ (Trigger)	—	—	—	—
frameworks:						
LangGraph	❌	❌ (só no servidor pago)	❌	❌	❌	—
OpenAI Agents SDK (Software Development Kit)	✅	—	❌	—	parcial	só sandbox agents
CrewAI	✅ (obrigatório)	—	✅ **client+server**	—	✅	✅ **auto-gerado**
software-agent-sdk	✅ (OAuth)	—	❌	✅ (usa harnesses como motor)	✅ (spec)	✅

Leituras da matriz:

1. **MCP venceu de fato**: 10 de 11 (a exceção, Aider, é escolha filosófica). E entre as rodadas 1 e 2, o padrão migrou de "cliente" para "cliente+servidor" — o harness como serviço consumível.

2. **agentskills.io é a padronização mais rápida que já medimos**: spec de dezembro/2025, 8 dos nossos 11 compatíveis em julho/2026. A previsão do cap. 12 ("um MCP da extensibilidade está se formando") se cumpriu — e com um detalhe estrutural: skills são markdown portável, então a mesma skill roda no Claude Code, no Hermes e no IronClaw. O aprendizado auto-evolutivo (cap. 16) escreve *nesse* formato — o conhecimento que um agente aprende é, em tese, transferível a outro.
3. **ACP é o protocolo silencioso mais importante da coorte**: 6 de 11 o falam, e três harnesses (OpenClaw, OpenHands, Goose) o usam para **orquestrar outros harnesses** como subagentes — Claude Code, Codex, Gemini CLI e opencode viram peças intercambiáveis. O que era "agente↔editor" virou, na prática, o barramento de composição entre harnesses.
4. **A2A saiu do "aposta de um só"** *(atualizado na rodada frameworks-1)*: o gemini-cli foi o único harness a implementá-lo, mas o **CrewAI** entrou com client E server nativos (AgentCard completo, JWS, gRPC/REST) — o segundo implementador medido, e o primeiro framework. A governança na Linux Foundation e a absorção do ACP-IBM seguem apontando o A2A como o candidato à fronteira inter-organizacional; nos harnesses de produto, porém, essa fronteira ainda quase não existe.
5. **AGENTS.md consolidou como padrão neutro**: a fragmentação AGENTS/CLAUDE/GEMINI.md do cap. 03 está se resolvendo — Codex, Goose, opencode, OpenClaw e Hermes já convergiram para AGENTS.md (agora sob a Agentic AI Foundation), com os arquivos proprietários virando alias.

O empilhamento: como os protocolos compõem

Um sistema agêntico completo em 2026 usa a pilha inteira, uma camada por fronteira:

```
[usuário]
   │  AG-UI / canais de chat / TUI          (interface)
[harness A]
   │  ACP                                    (composição: A dirige B como subagente)
[harness B]
   │  A2A                                    (delegação a agente de outra organização)
[agente remoto]
   │  MCP                                    (cada agente alcança suas ferramentas)
[ferramentas/dados]

transversais: AGENTS.md (instruções por projeto) · SKILL.md (procedimentos portáveis)
```

Implicações para a engenharia de harness

1. **Protocolo é dimensão de sobrevivência, não de feature**: o Aider, referência técnica em três dimensões, está fora do ecossistema de composição inteiro por não falar MCP/ACP. Em um mercado que compõe harnesses, não falar os protocolos é ficar de fora das arquiteturas dos outros.
2. **A cláusula de expiração não se aplica aqui** (cap. 14): protocolos são fronteira com o mundo — o scaffolding que *resta* quando os modelos melhoram. Investir em protocolo é o investimento de harness com maior meia-vida.
3. **Para o benchmark**: a matriz acima vira seção permanente do comparativo, atualizada a cada rodada. Protocolos não recebem nota 0–3 como harnesses — são avaliados por **adoção medida** (a matriz) e **saúde de governança** (fundação neutra > vendor único).

Adendo (2026-07-31): a spec MCP **2026-07-28** (anúncio) reforça a tese deste capítulo por outro ângulo: núcleo stateless, framework de extensões e a **primeira política formal de depreciação** (12 meses) são o comportamento típico de protocolo saindo da adolescência e entrando na fase de infraestrutura — versionamento disciplinado importa mais que features. A adoção da nova versão pela coorte entra na matriz na próxima rodada. E a conferência do mesmo dia na outra fronteira (spec 065): a especificação do A2A confirma o **v1.0 estável sob a Linux Foundation**, organizado em três camadas (modelo de dados em Protobuf/JSON Schema, operações abstratas, bindings JSON-RPC/gRPC/REST), com o **v1.0.1 já trazendo um mecanismo formal de extensões** — os dois vencedores de fronteira chegaram, no mesmo trimestre, ao mesmo estágio: extensões formais em vez de features no núcleo.

> **LEITURA EXECUTIVA**
>
> A camada de protocolos já tem um vencedor por fronteira: MCP na vertical (agente → ferramenta, adoção quase total), ACP como barramento de composição entre harnesses, agentskills.io como formato portável de conhecimento procedural e AGENTS.md como padrão neutro de instruções de projeto — enquanto o A2A segue como a aposta em consolidação para a fronteira inter-organizacional, sustentada mais pela governança (Linux Foundation, absorção do ACP-IBM) do que pela adoção medida nos harnesses de produto. A decisão de engenharia é assimétrica: protocolos são o componente de maior meia-vida do harness, imune à cláusula de expiração, e a matriz de adoção — não o marketing — é o instrumento para reavaliá-los a cada rodada do benchmark.

Fontes da indústria

- ecosystem map 2026
- Zylos: convergência MCP/A2A/ACP
- Zuplo: onde foi parar o ACP
- Agent Skills: formato e adoção
- AGENTS.md guide 2026
- Zed ACP

Matriz de adoção: evidência própria do benchmark (`benchmark/avaliacoes/`).

- **Consulte também**: a coleção viva Awesome Harness Engineering — Skills & MCP reúne mais recursos consultáveis desta dimensão (padrões, artigos e implementações), curados por problema.

Verificação

1. Um colega afirma que "o A2A vai substituir o MCP". Por que a afirmação confunde as fronteiras, e como o empilhamento mostra que um sistema real usa os dois? (Releia "O mapa" e o diagrama.)
2. "ACP" aparece duas vezes na tabela de protocolos, com estados opostos ("adoção rápida" e "encerrado"). Explique a diferença entre os dois protocolos — e qual deles este livro chama de ACP.

3. Você está desenhando um harness novo. Com base nas leituras da matriz e nas implicações, quais protocolos são obrigatórios hoje, qual ainda é aposta, e o que a exceção do Aider ensina sobre o custo de não falar nenhum?
4. Por que a cláusula de expiração (cap. 14) não se aplica à camada de protocolos, quando se aplica a quase todo o resto do harness?