



HANGZHOU
AI AGENT INNOVATION
COMPETITION



AI杭州 码动未来

A I 模 型 智 能 体 创 新 大 赛

先看事实：这一年我们聊了什么

一个群的年度数据快照

- 25,636 条消息，308 位发言者，8 个月
- 1,230 条候选与 harness 强相关
- 手工川本人 649 条 harness 相关发言
- 1 月最活跃 4,830 条，8 月仍在增长



KEY FINDING

观察密度 = 素材厚度

FACTS 01

行业大事记

2026 年发生了什么



Claude Code
五十万行源码
因 source map
泄漏



Codex App /
OpenClaw
重新定义
agent 前端



Fable 5 发布,
Loop Engineering
登场



MCP 转按需加载,
上下文占用
痛点解除



Opus 4.8
动态 workflow,
单会话调度
数百子代理



KEY FINDING

一年内 harness 从概念走向基础设施

[FACTS 02]

五十万行源码流出

护城河不在模型，在 harness



- npm 包遗留 source map, 核心实现尽出



- 行业复盘：orchestration、tool 调度、harness 骨架是立身之本



- 结论：Anthropic 的护城河是 agent 长程任务引擎与 harness 设计



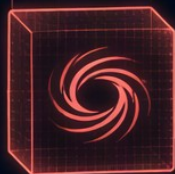
KEY FINDING

此非伤皮肉，乃及筋骨



我们踩过的坑

一年里真实发生过的失败



- 宏大目标交给 goal 模式，
20 亿 token 打转一周



- 省钱降档换小模型，
问题反复犯，反而更贵



- 付费 Skill 13 步自动化，
交互闭环不合格



- 截图沟通导致上下文爆炸，
自动压缩也救不回



KEY FINDING ▶

失败比成功更值得复盘

FACTS 04 · 数字不会说谎

用量机制的荒诞对比



一天烧掉 17 亿 Token 没事



8000 万反而被限流？



阈值、计费、限流机制
成为 harness 之外的真实变量

17 亿

Token / 天



8000 万

Token / 天



限流



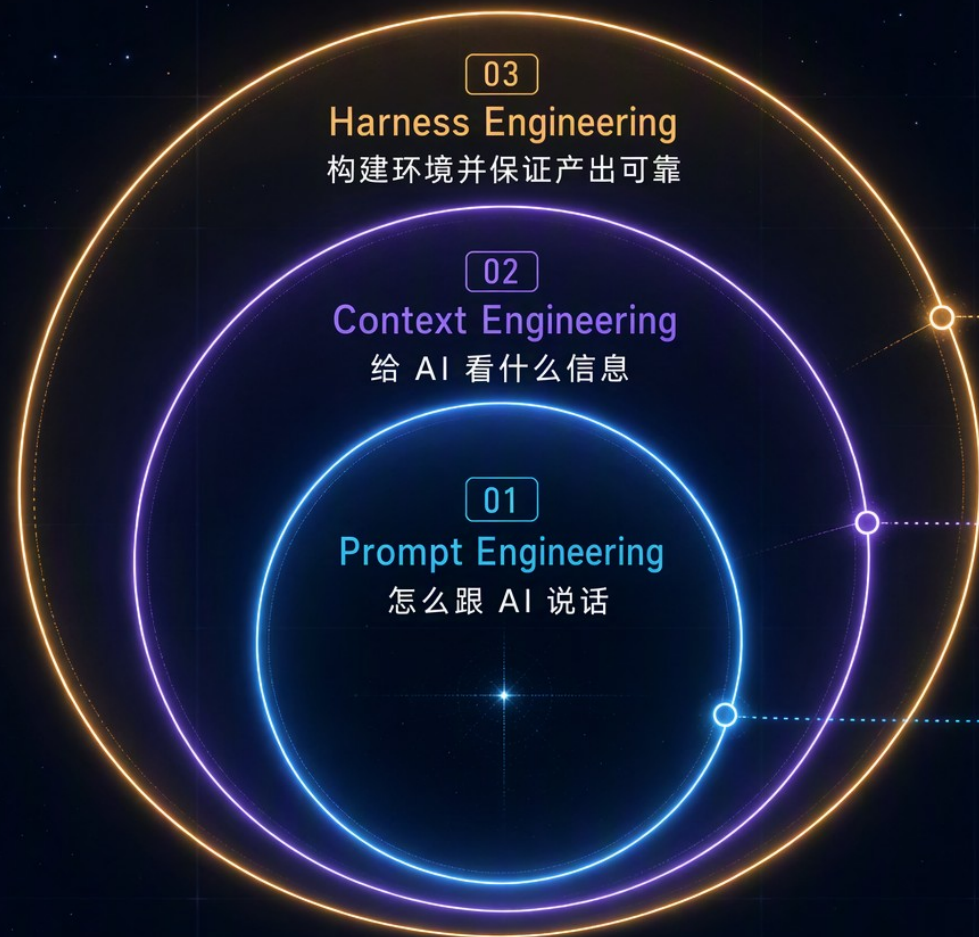
KEY FINDING

额度机制直接决定选哪个底座



FINDINGS 01 · 三阶段演进

从提问，到看什么，到造环境



03 Harness Engineering (2026)
构建环境并保证产出可靠

02 Context Engineering (2025)
给 AI 看什么信息

01 Prompt Engineering (2023-2024)
怎么跟 AI 说话



KEY FINDING

每次演进都是上一层的环境化

FINDINGS · 02

模型决定上限， Harness 托住落地

Agent 系统的三层职责



LLM 负责生成（能力上限）



Harness 负责可运行、可验证（落地）



Loop 负责反复驱动改进与探索



KEY FINDING

上限与落地之间，
就是 harness 的舞台



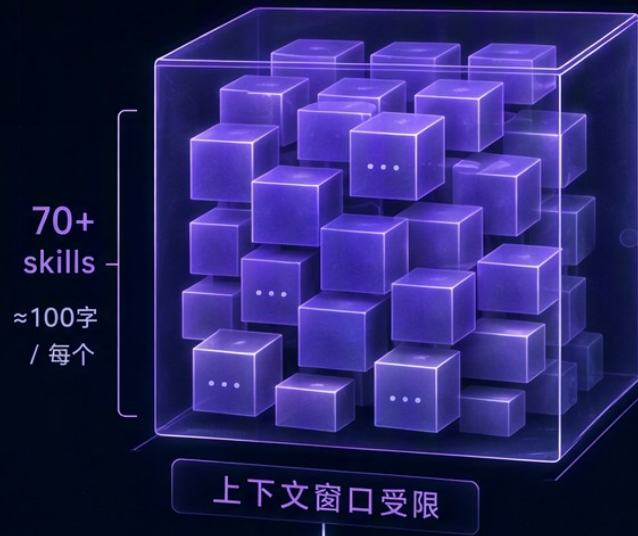
FINDINGS 03

上下文是核心杠杆

skill 与 MCP 的 token 博弈

- skill 总上下文受限：
默认取窗口 1% 与 8k 较大者，
70+ skill 时每个约 100 字
- MCP 转按需加载后，
比 skill 更省 token
- 对策：
Description 写短写准，
控制 skill 数量

SKILL 全部加载 (容器挤压)



MCP 按需加载 (精准注入)



KEY FINDING

Prompt 不会复利, Context 才会

FINDINGS 04

给弱模型写步骤，给强模型写责任

202 个 Skill 的实证反例

- Skill 库扩到 202 个，Agent 通过率最多反降 21%
- 工具越多，模型越容易选错
- 原则：弱模型给 Step，强模型给 Responsibility



KEY FINDING

秩序来自**克制地删**控制

FINDINGS 05 · Skill 的形态必须变

开发期 vs 生产期，契约不同

阶段 01
开发期



skill = prompt + 文件

阶段 02
PoC 期



skill = prompt + tool

阶段 03
生产期



skill = 能力边界 + 权限 + SLA



KEY FINDING

生产环境不能把 skill 当文件系统 prompt 用

FINDINGS 06 · Skill 的商业化困局

三个不可能同时实现



- 标准化 agent runtime
- 客户端本地运行
- 源码保护
- ! 更深的死结：所有平台都没解决 **trust**



KEY FINDING

Skill 是 SOP，价值只能靠 case 证明，
问题永远是 **trust**



○ FEELINGS 01

定义之争的情绪

火了三个月，没人定义得清



「不过是软件工程
换皮」

—— 泼冷水的怀疑

— 未完成的定义 —



「旧词兜不住
新方向」

—— 拥趸的辩护



「定义给不出，
但都能感觉到」

—— 普遍的焦虑



KEY FINDING

概念的含混，正是行业的**青春期**

FEELINGS 02 .

被迭代碾压的焦虑

工具在变，人在中间重写

- 「codex 干掉了 vibekanban 和 conductor，群主被抄底」
- 「codex 和 cc 都要过时了」
- 「装上新 App，原来的插件、MCP、Skills 全被自动导入，问都没问」



KEY FINDING

模型会更新，工具会迁移，
人会在中间不断重写工作流



FEELINGS 03

20 亿 Token 的挫败

宏大目标直接交给 goal，发生了什么



第三天开始疯狂推翻已有功能，后两天反复打转



累计约 20 亿 token，毫无进展



反思：判断完成不能靠 AI 的自我感觉



KEY FINDING

宏大目标要交给 goal，必须加细致的验收标准



[FUTURE 01]

FUTURE 01 · 100 行脚本接管循环

外部编排器四步法（当下可落地）

① 静态 Checklist + 全新干净会话
根除上下文污染

② 强制 Search/Replace
语义匹配准确率 98%

③ 闭嘴模式 + 消极约束
禁重构、禁加依赖

④ 物理级退出条件
Exit Code 0 即结束，重试 3 次熔断



KEY FINDING

判断完成，交给物理测试而非 AI 感觉

多智能体，留痕别断环

高智商 plan，低智商 implement



planner 提 issue → implementor 轮询
→ 提 PR → reviewer 审阅（全程留痕）



代价：OODA 完整闭环被割裂
成 push/pop 压栈



解法：pipeline / group 汇总
把观察权还给用户



KEY FINDING

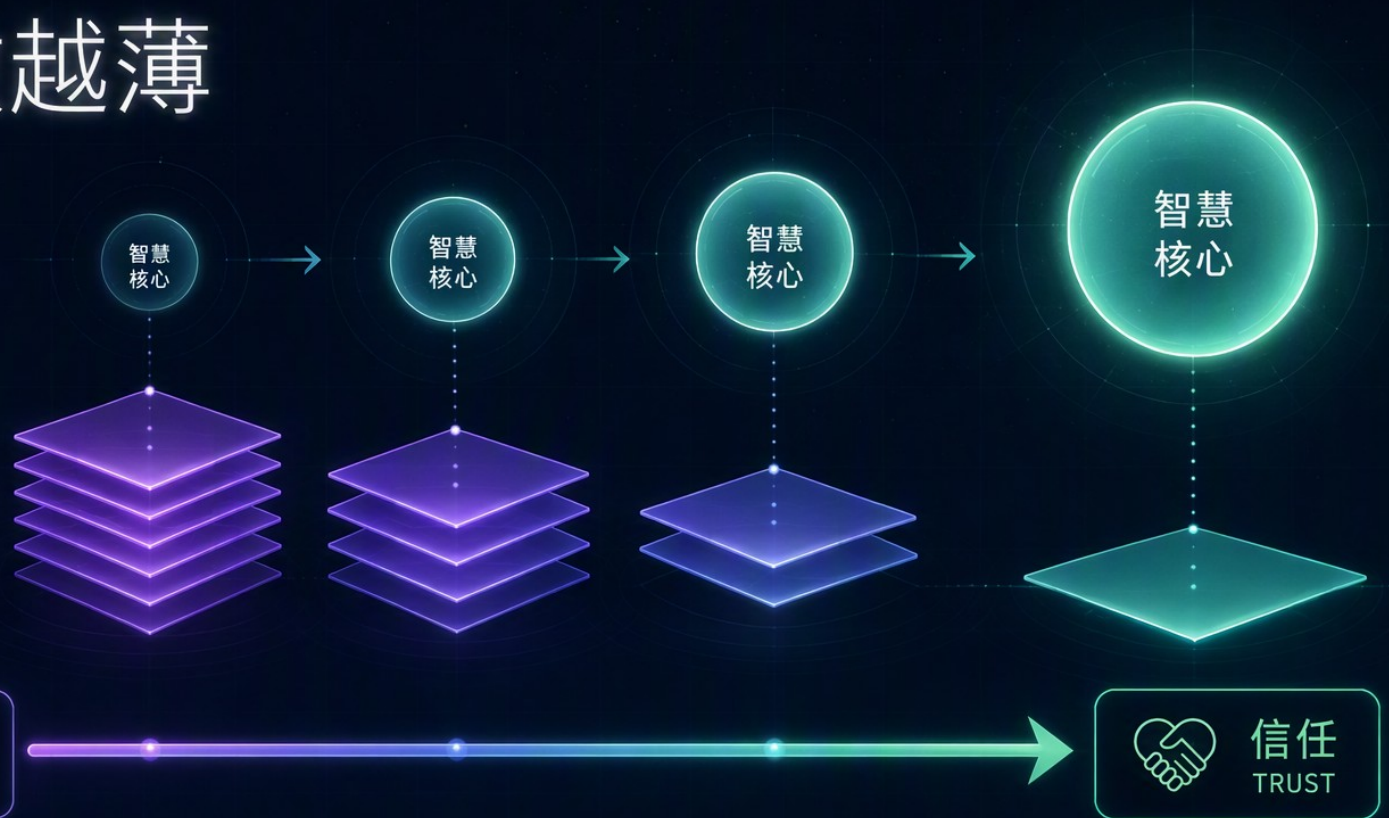
多智能体模式，是留住闭环的答案

FUTURE 03 ·

Harness 越做越薄

智慧越强，堡垒越少

- AI 智慧越来越强，不需要那么多堡垒工作
- 从「加更多控制」到「删掉不需要的控制」
- 管理同理：管理本身不是问题，问题不在这



KEY FINDING

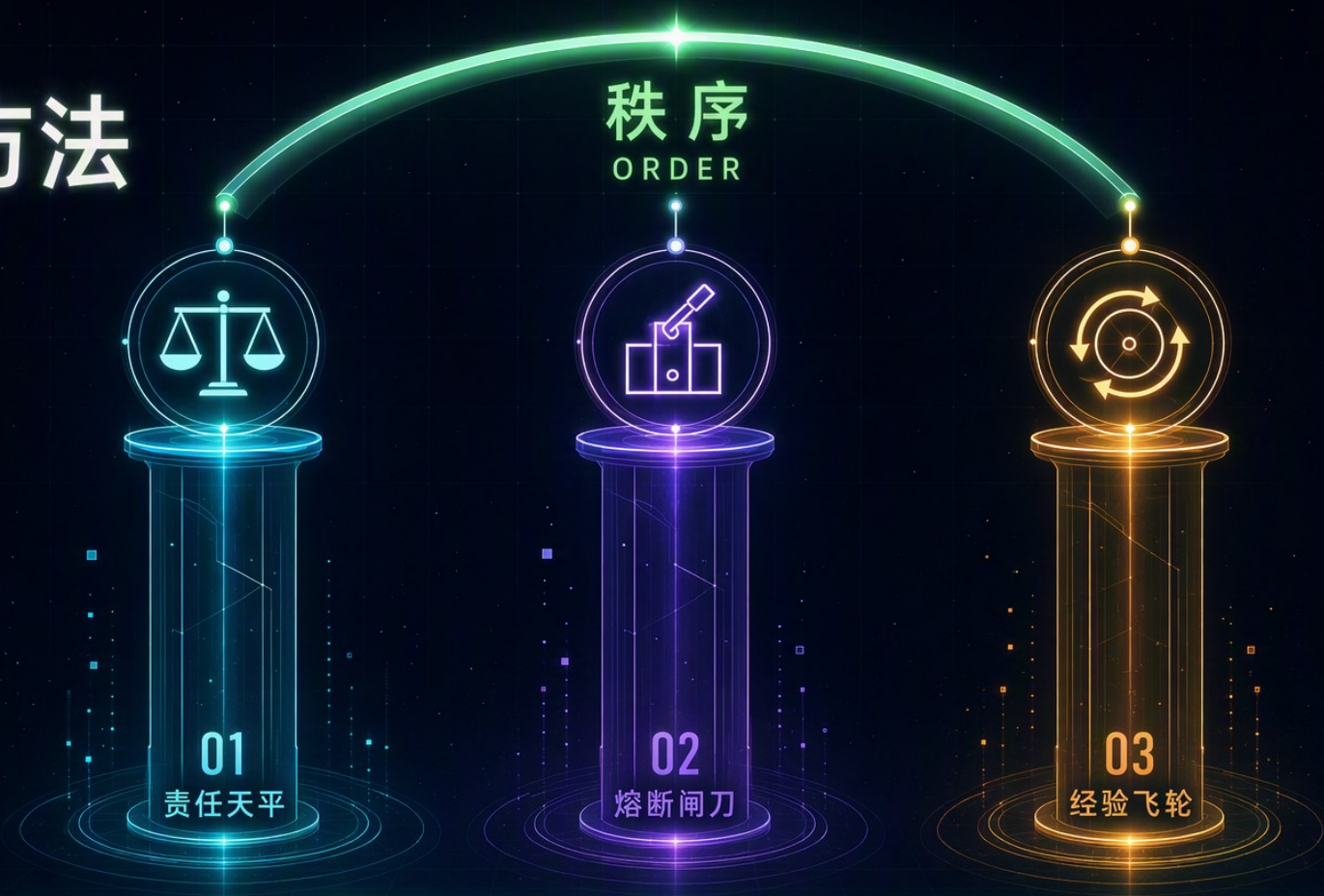
Harness 越做越薄，因为 AI 越来越强

[FUTURE 04]

三条可带走的方法

复盘落回行动

- 给弱模型写步骤，
给强模型写责任（少控制，准控制）
- 用物理退出条件代替
AI 自我感觉（熔断，不是催促）
- 经验要沉淀成飞轮，
而不是更长的日志



KEY FINDING

模型智能是租来的，只有经验飞轮属于你

FACTS 01

[CHAOS & ORDER]

草原不会变绿，只是**工具在变**

混沌交给模型，秩序交给 harness，创造留给自己

- 模型会更新，工具会迁移，而人会在中间不断重写 workflow
- 想一起玩 Harness Engineering？关注 Lovstudio，持续输出实战经验

★ KEY FINDING

模型负责生成可能性，工具负责组织可能性，而你定义可能性的方向。

— Lovstudio —

