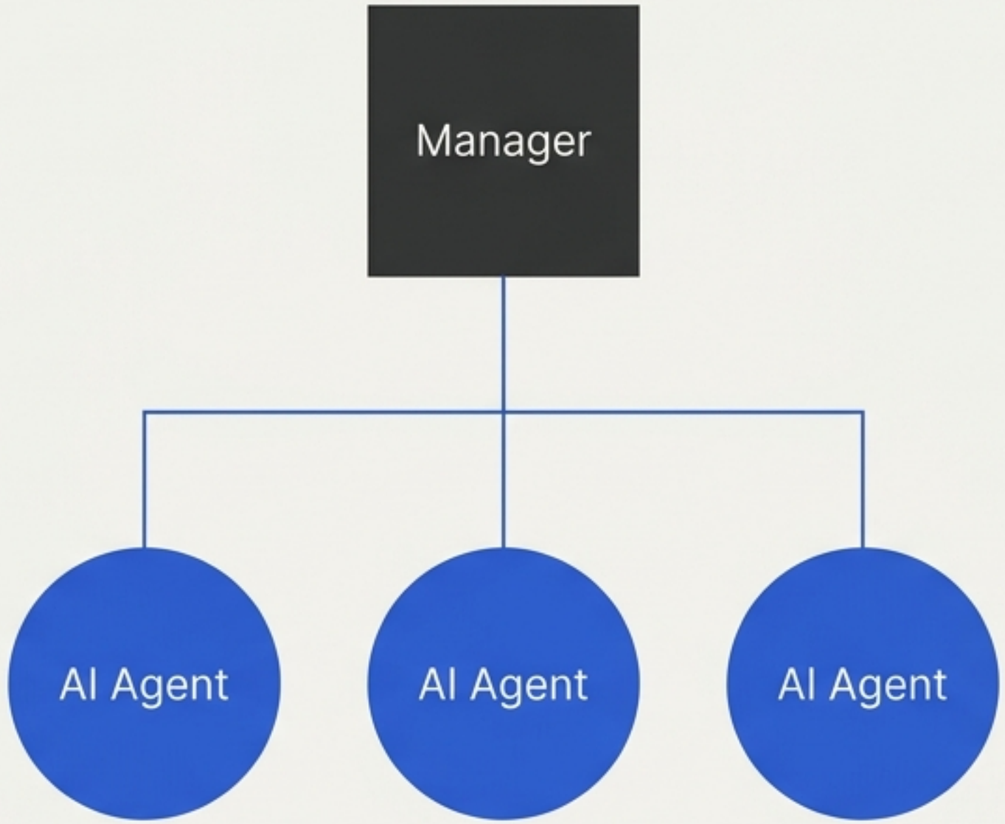
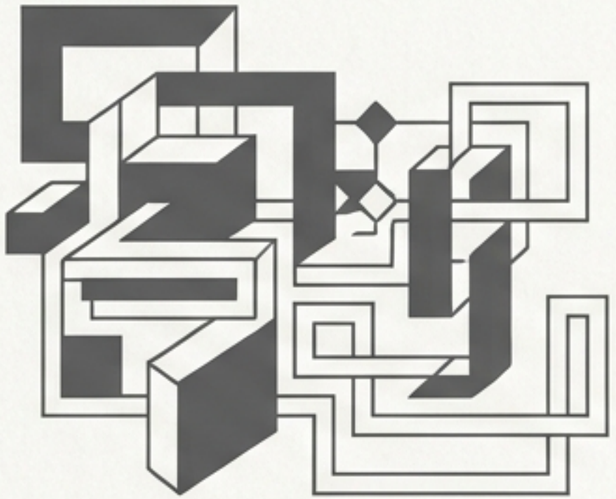
管 AI 跟带人是同一套东西

Agent 时代真正重要的技能不是编码——而是管理。

从下好指令到校准信任，与 AI 协作的真实样貌。

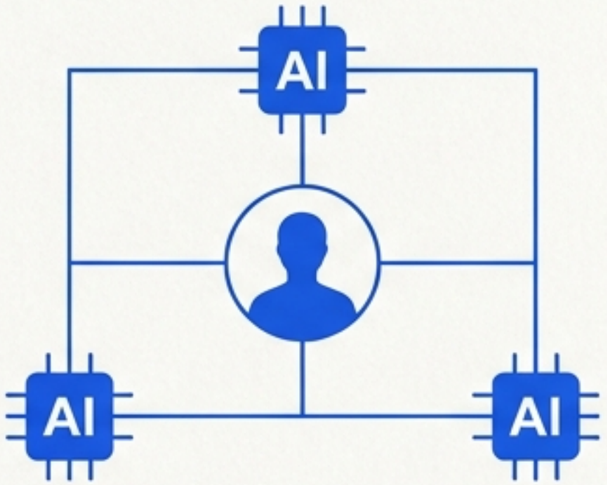


范式转移：你不再是操作员，你是管理者



编写代码

纠结于语法、步骤和实现细节。

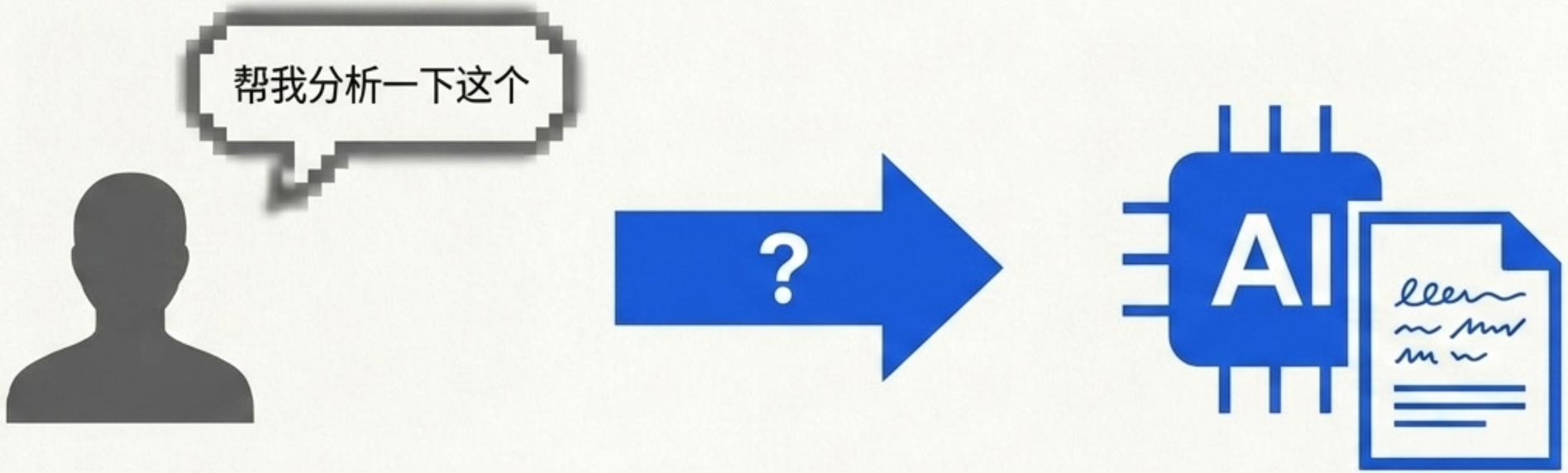


管理智能

专注于需求拆解、背景输入和标准定义。

基础设施的门槛已被跨越，现在的核心瓶颈是你的“带人”能力。

“帮我分析一下这个”的陷阱

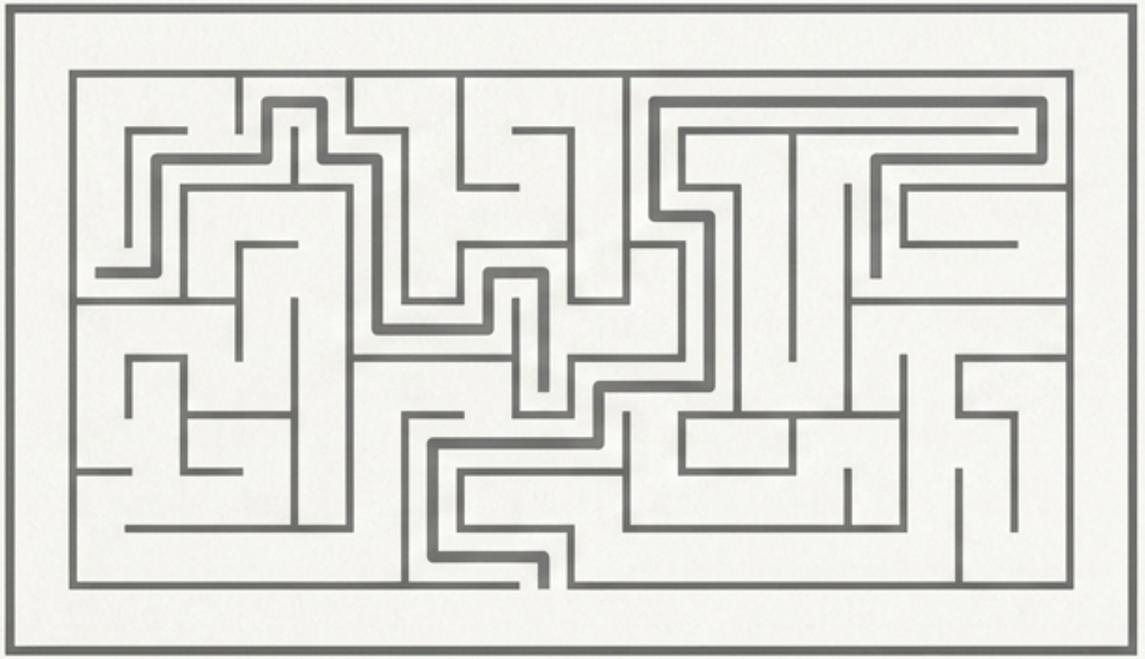


输入：模糊的需求（无数据维度、无受众、无决策目的）

处理：盲目猜测（AI 只能靠猜，生成看似合理的“废话”）

洞察：这就像对一个能力极强但毫无业务背景的下属说“帮我搞一下”。他一定会交差，但大概率不是你脑子里的东西。

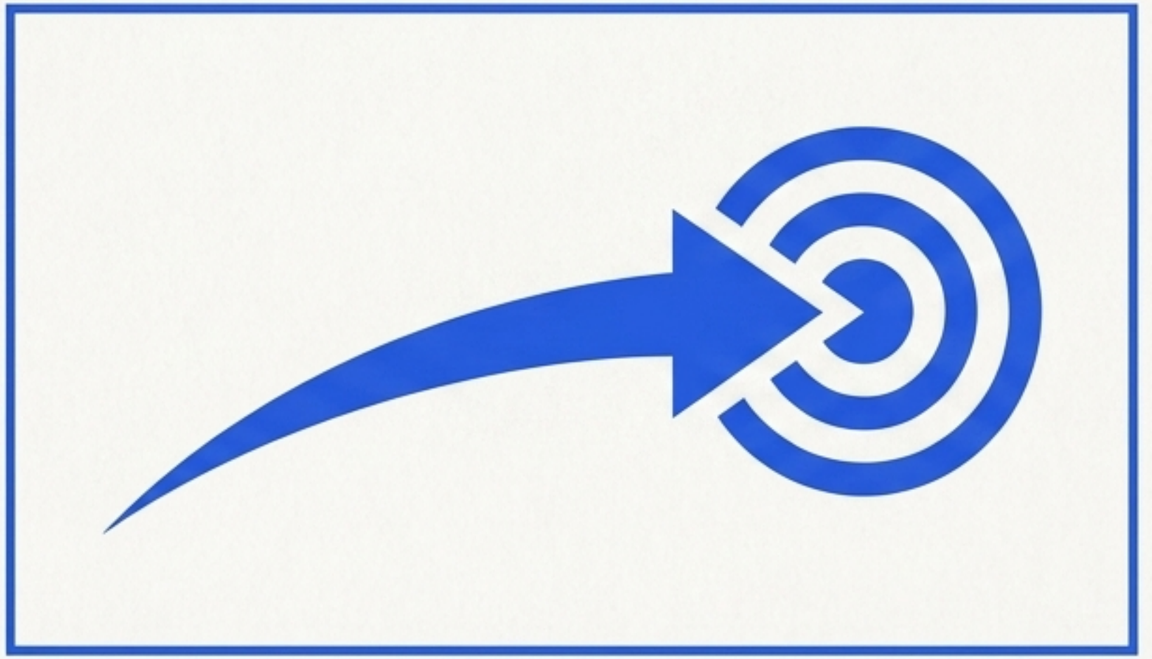
法则一：说目标，别说步骤



机器思维 / 规定步骤

指令：“帮我写一个 Python 脚本解析这个 CSV。”

结果：限制了 AI 的发挥，你得到了一个死板的脚本。



高管思维 / 描述目标

指令：“我有一份销售数据，我需要知道过去三个月哪些客户下单频率在下降，以及可能的原因。”

结果：把“怎么做”留给 Agent，产出几乎总是优于你预设的路径。

法则二：Context 决定成败

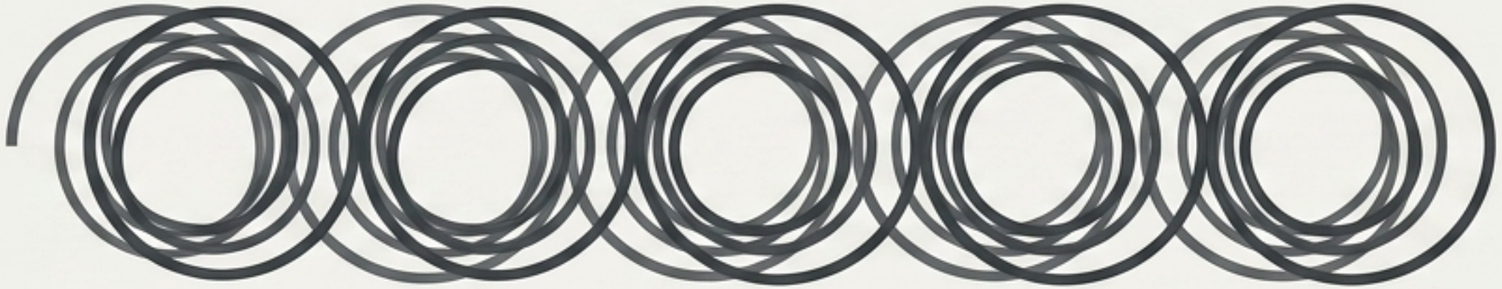
[你是谁] ＋ [受众是谁] ＋ [决策目的] ＝ [精准交付]

同一份数据，给老板看和给投资人看完全是两个东西。

核心法则：把 AI 当作一个能力极强、但对你的业务一无所知的直属下属。如果你是 CEO，你必须把背景讲清楚，把交付标准说死。

法则三：反馈的精度决定迭代的速度

无效反馈



“不太对，再改改” –> AI 只能再猜一次 –> 陷入 5-6 轮的无效循环。

精准反馈

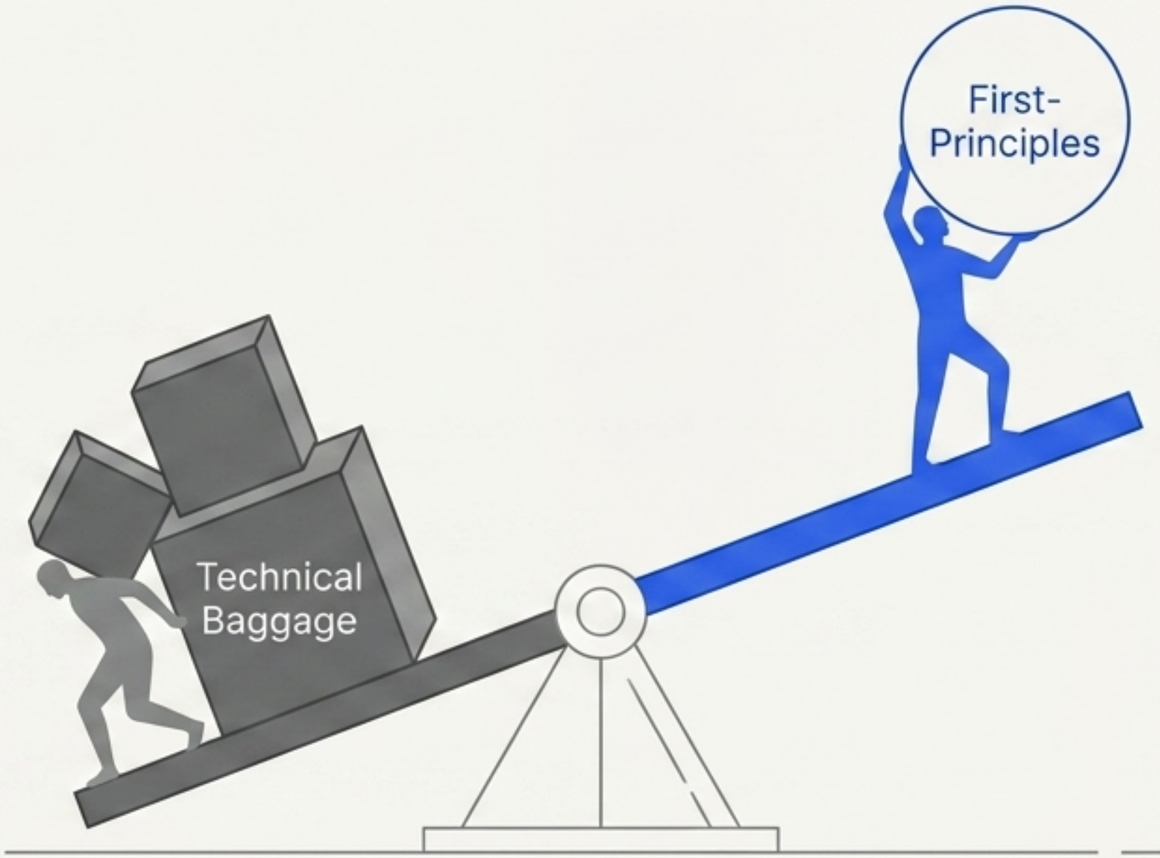
“基本面没问题，但我关心的是负债结构，估值换成 DCF，别用市盈率。” -> 1-2 轮精准修正。

结论：结果不对时，最值钱的技能是能精准说出哪里不对。

新手悖论：为什么资深者反而步履维艰?

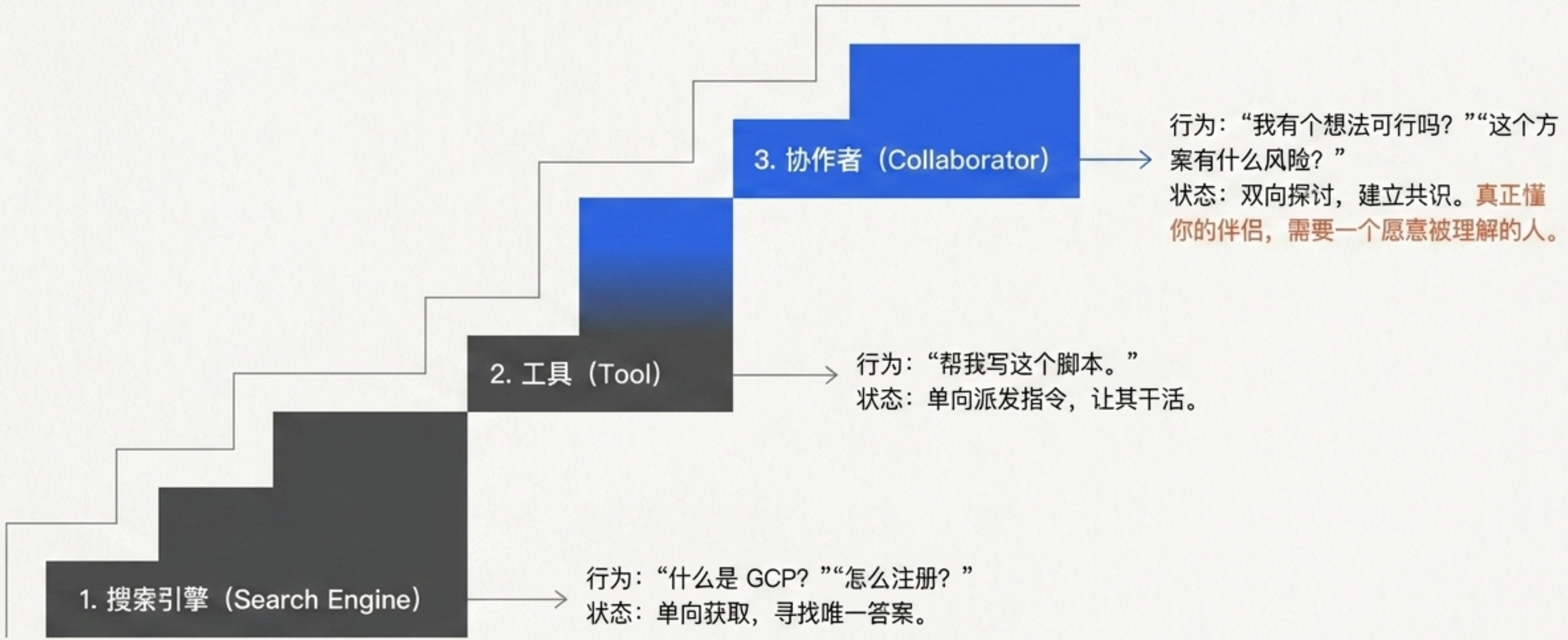
“在 Anthropic 内部，更年轻、经验更少的工程师，往往比资深工程师更会用 Claude Code。”

资深的包袱：积累了太多根深蒂固的习惯，对“软件应该怎么写”有强烈的执念，成为协作障碍。



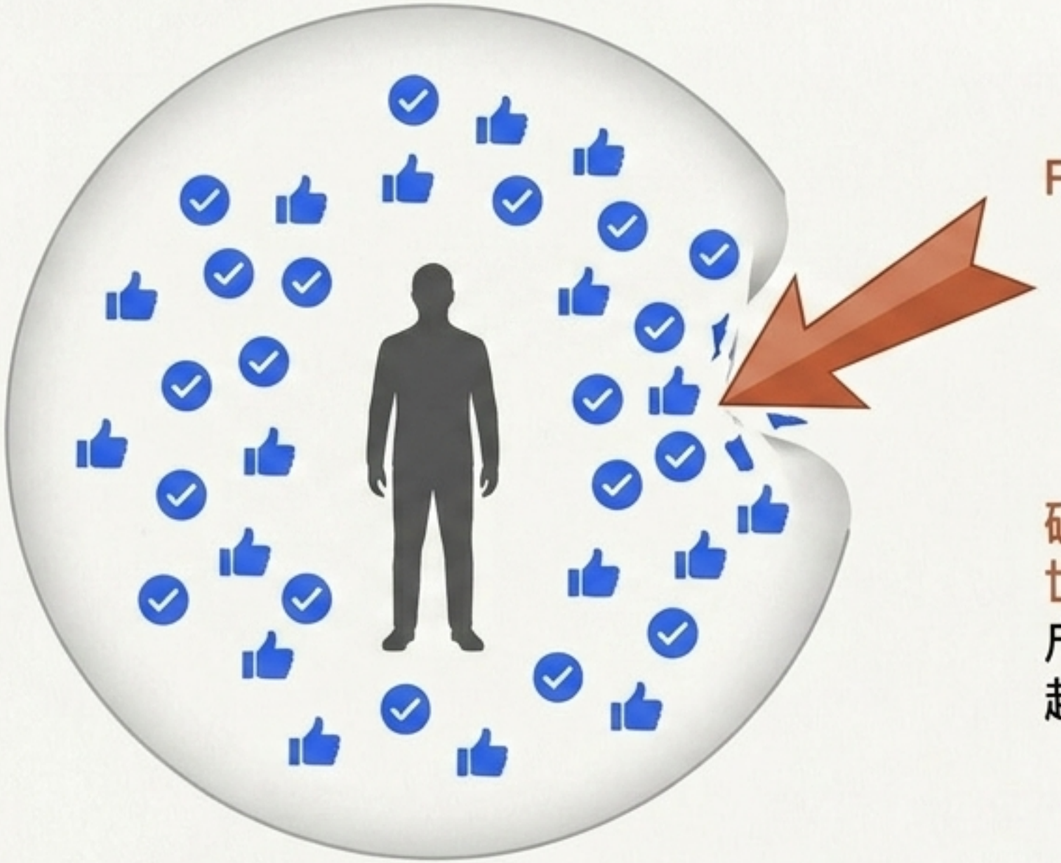
新手的优势：没有先入之见。只需要科学思维、第一性原理和框定目标的能力。无需“遗忘”旧习惯。

AI 协作的三重进阶心智



隐蔽的陷阱：警惕 AI 的“赞美泡沫”

现象：AI 天生爱鼓励人。
无论你的想法多不成熟，它总能找出亮点夸你，极易形成虚假的正反馈。



Real-World Validation
（真实校验）

破局点：检验必须回到**真实世界**（代码扛不扛得住、用户想不想要、分析经不经得起质疑）。

反思：如果你说不出结果哪里不对，说明你自己还没想清楚。
此时应退回“协作者”模式，让 AI 帮你理清思路。

最终的瓶颈，是你自己

的人还没和 AI 对过一次话。

现在的状态就像 iPhone 刚发布的第一年，移动互联网的故事才刚开始。现在掌握这套管理心法，先发优势能吃很多年。

工具到位了，能力也到位了。那段只能你自己走的路：想清楚要什么，给足背景，给出精准反馈。

现代管理者的 AI 协作清单

1 提供语境（Set the Context） 明确我是谁、给谁看、为了什么决策。	**2 定义目标（Define the Goal）** 描述“做好了长什么样”，绝不微操具体步骤。
3 精准反馈（Precision Feedback） 拒绝说“再改改”，直接指出哪一维度的逻辑/数据需要修正。	**4 真实校验（Real-World Test）** 警惕 AI 的讨好型人格，把最终结果放进真实世界里抗压。

不要只写代码，开始练习“带人”。