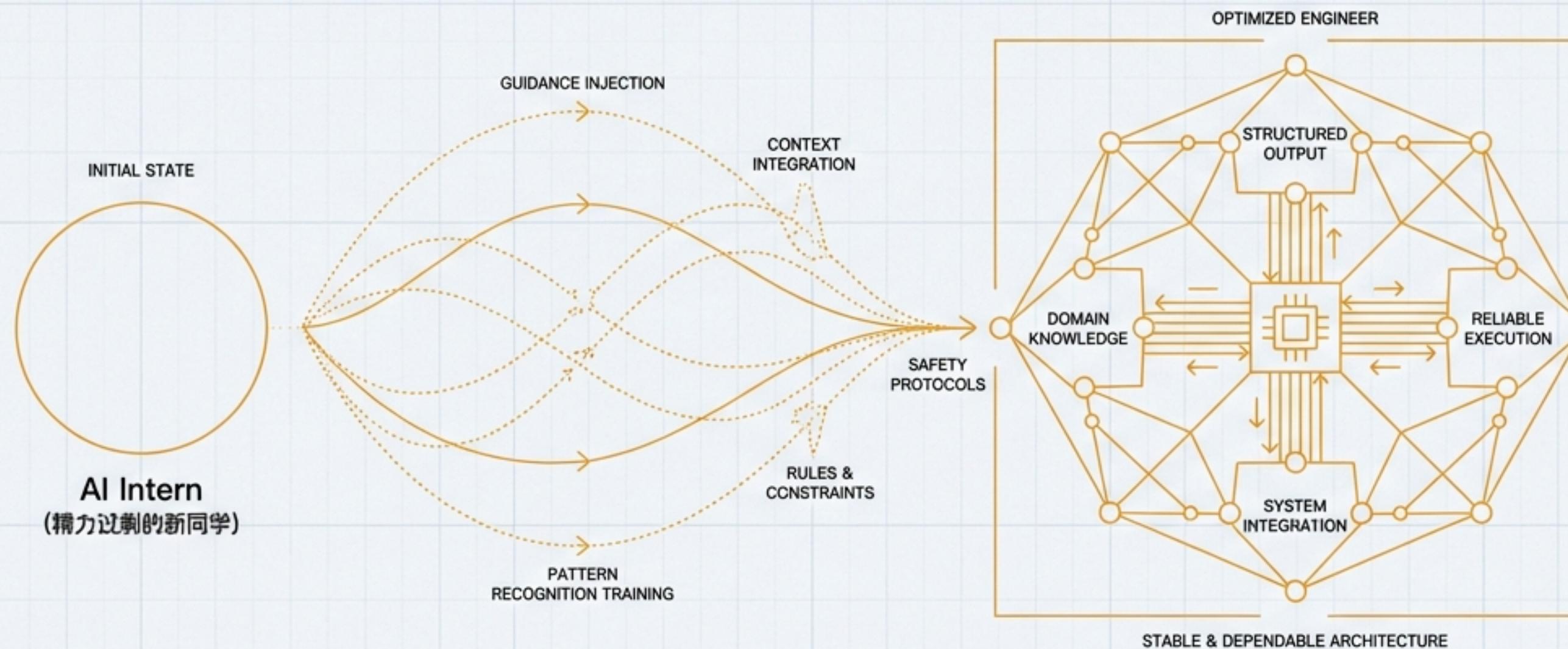


Vibe Coding 实战手册

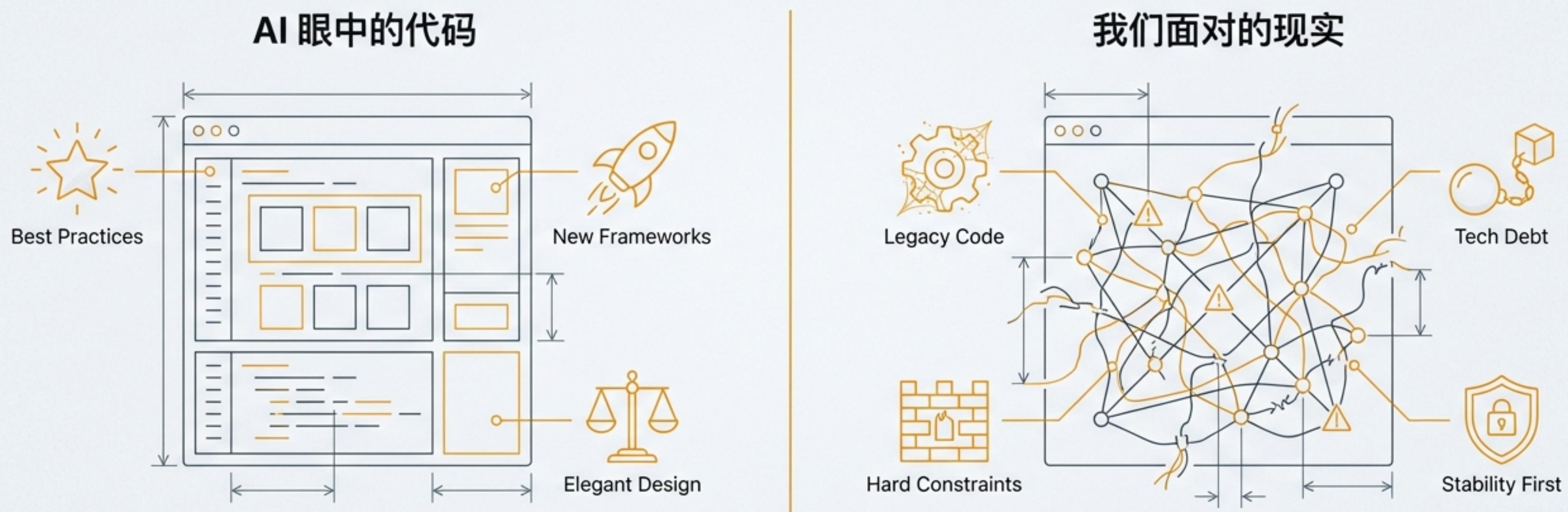
如何将 AI 从「精力过剩的新同学」调教成「靠谱的老同事」



FOR EXPERIENCED ENGINEERS WORKING ON COMPLEX LEGACY SYSTEMS

为什么通用 AI 助总在“好心办坏事”？

开箱即用的 AI 就像一个聪明但鲁莽的新同学。它满脑子“最新最佳实践”，却对你项目的历史包袱、技术债务和特殊约束一无所知。

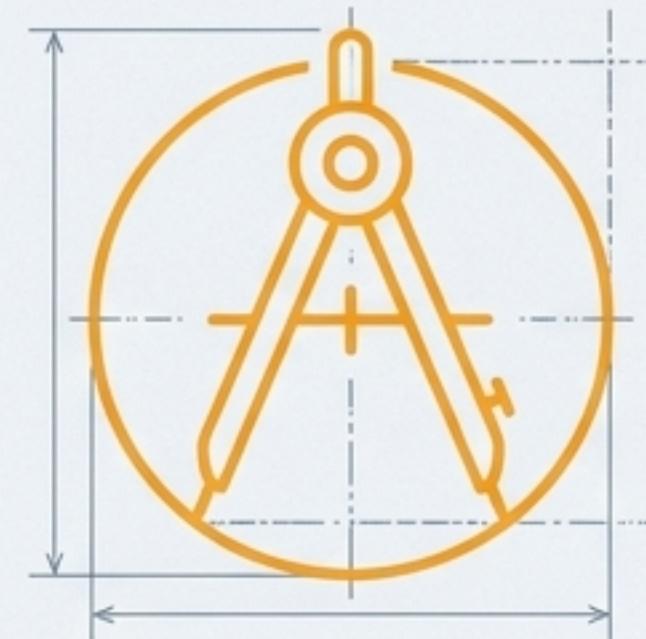


它一上来就“写得太对、太新、太理想化”——这在老项目里是灾难的开始。

Vibe Coding：从“指令工程”到“语境调教”

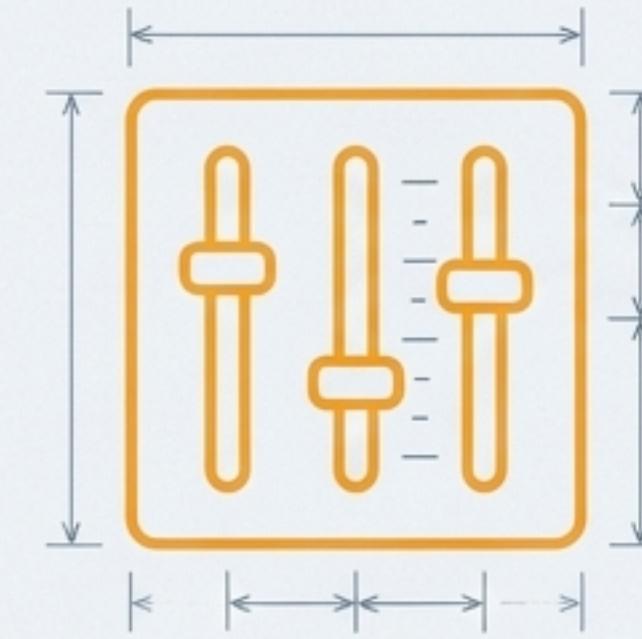
我们的目标，不是追求单条指令的完美，而是与 AI 建立一个持续、稳定、有约束的对话语境（vibe）。

“本质不是‘让 AI 多聪明’，而是‘让 AI 不犯低级错’。”



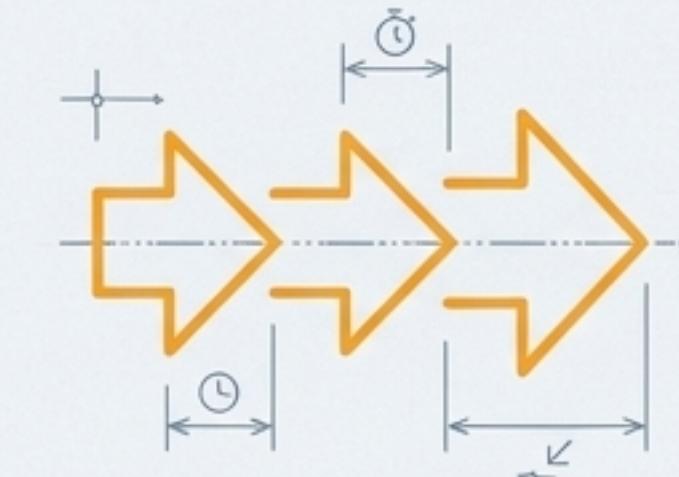
约束 (Constraints)

定义不可越界的规则和边界。



人格 (Personality)

塑造一致的沟通风格和语调。



节奏 (Rhythm)

引导对话的推进和交互频率。

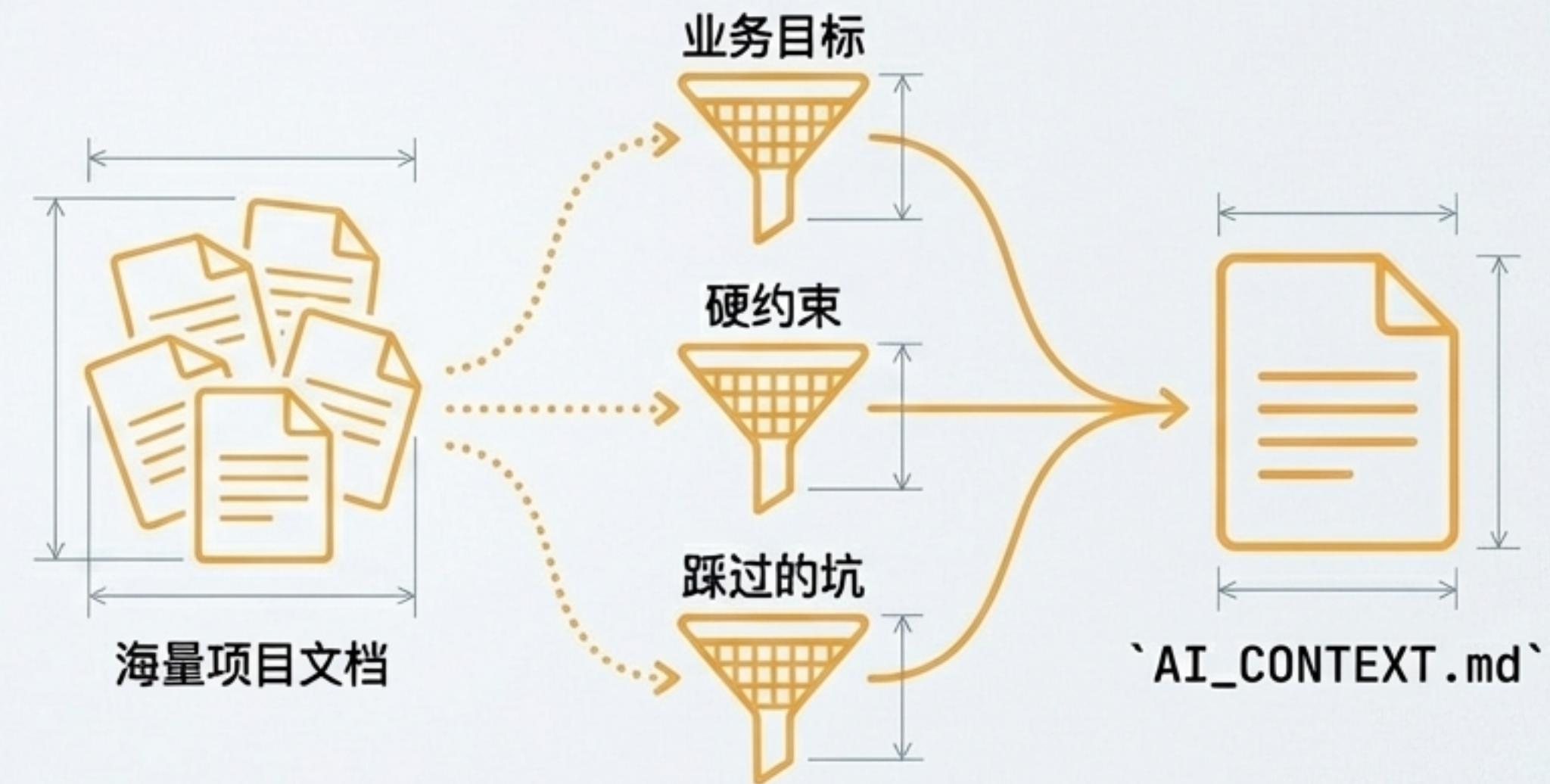
这就是你为 AI 新同学设计的“导师带教计划”。

第一章：建立稳定的项目背景

为你的 AI 新同学划定“安全护栏”

第一步不是写代码，而是让 AI 对项目有「上下文的整体感觉」。你不是在告诉它“写一个函数”，而是在反复强调游戏规则。

我们追求的不是完整，而是稳定。与其一次性喂给 AI 大量文档，不如提炼一份稳定、反复引用的“背景摘要”。



如何定义三大核心约束

目标与红线 (Goals & Red Lines)

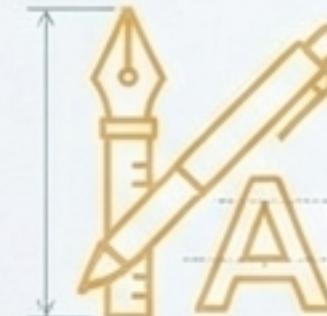
明确我们到底在解决什么问题，为什么只能用这种方式解决，以及哪些事绝对不能发生。



背景：这是一个运行了 5 年以上的老系统，核心目标是稳定性。我们要解决 XXX 问题，之前尝试过 YYY 方案但因 ZZZ 被放弃。任何方案都不能引入跨模块的隐式依赖，也不能改变已有对外行为。

风格与审美 (Style & Aesthetics)

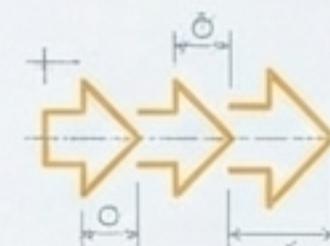
帮 AI 建立“审美边界”，是工程稳健还是实验探索？是可读性优先还是性能优先？



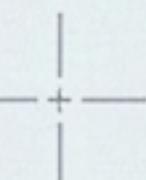
提醒：在这个项目里，我们通常选择更啰嗦但直观的实现方式，不太接受为了复用而引入额外抽象。

阶段与位置感 (Phase & Awareness)

明确告知 AI 当前是在原型、收敛还是重构阶段。



注意：当前阶段不是探索方案，而是在已有实现上做小幅、可控的改动，目标是降低未来维护风险。

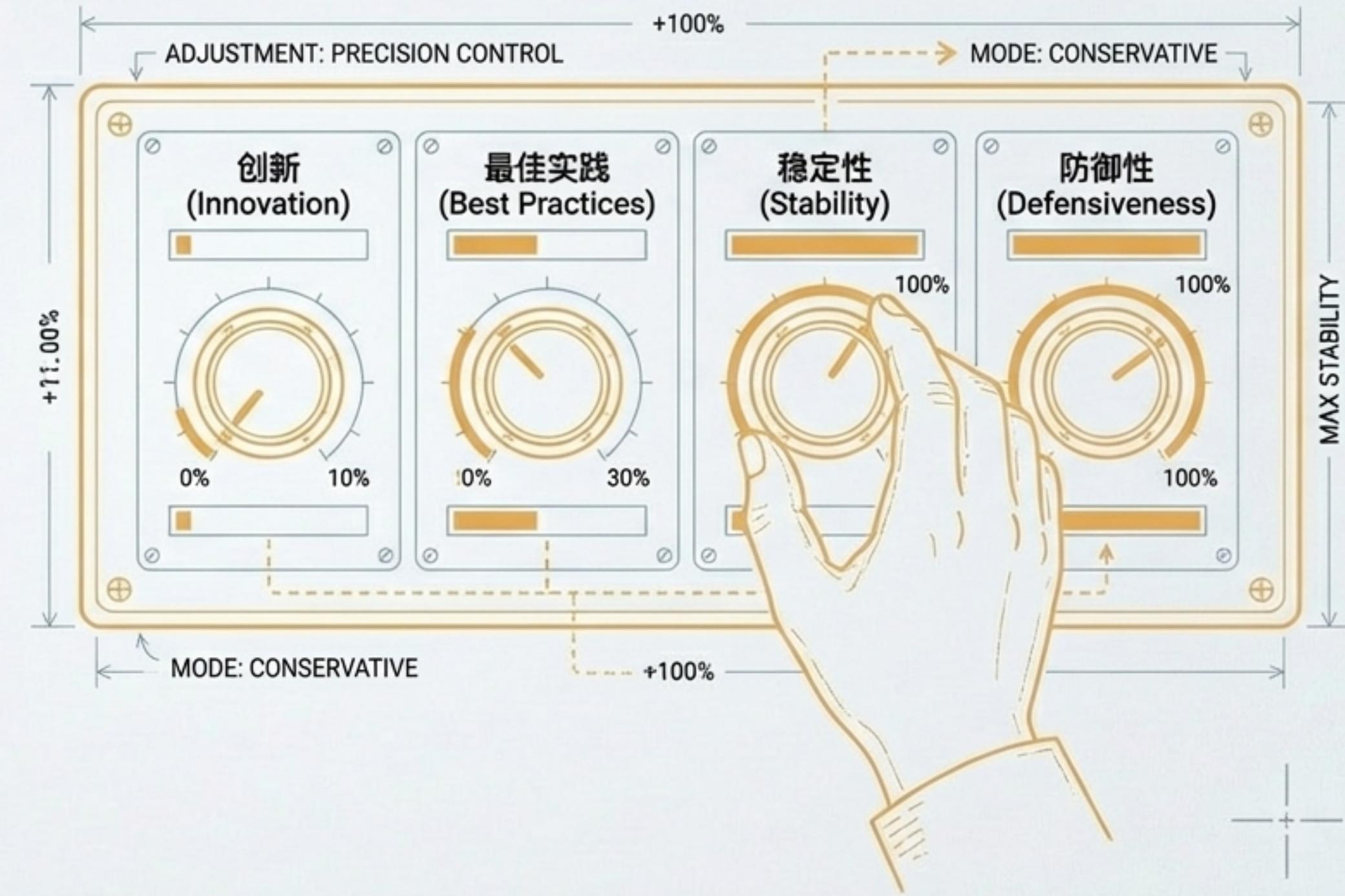


第二章：塑造 AI 的「谨慎人格」

将 AI 从“积极分子”调教成“怕背锅的老工程师”

在真正写代码前，花足够的时间营造语境。这不是在教技能，而是在塑造人格。

通过在多轮对话中、从不同角度反复强调同一组原则，来内化这种谨慎。



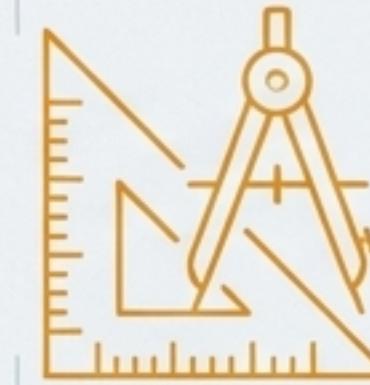
反复强化的四个核心观念



稳定性高于一切：“这是一个老系统，稳定性优先于优雅。”



杜绝隐式假设：“宁可多写几行防御代码，也不要隐式假设。这个系统历史上因为隐式假设出过事故。”



遵循现有风格：“输出要尽量贴近现有风格，而不是所谓的最佳实践。”



为未来维护者着想：“所有改动都要假设未来接手的人并不知道上下文。我们这里默认读代码的人对业务不熟。”

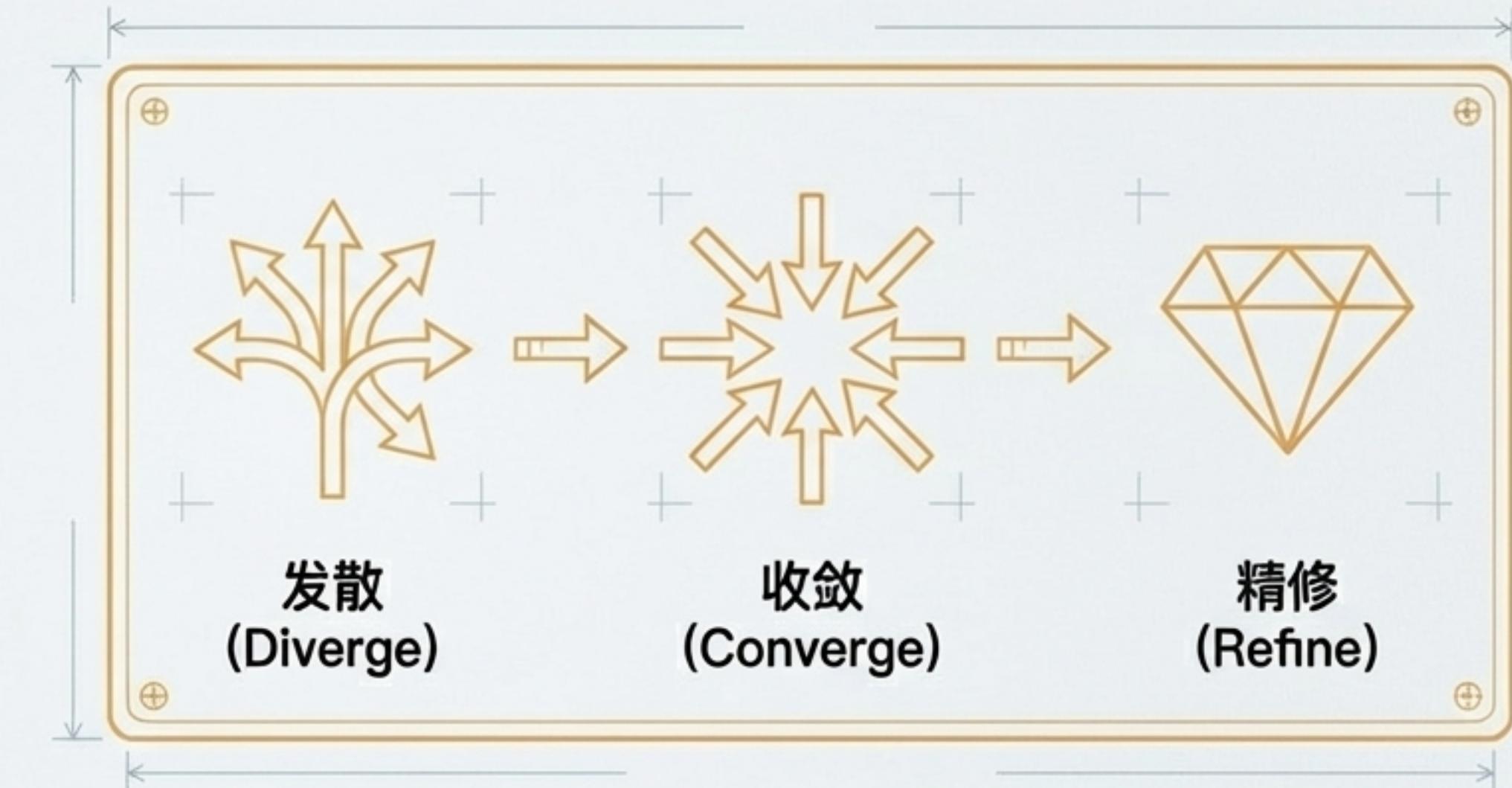
重点在于“多次出现、从不同角度出现”。当 AI 的输出明显更像一个谨慎、保守的工程师时，人格就塑造成功了。

第三章：掌控对话的「三段式节奏」

放弃万能 Prompt，拥抱过程式引导

在 Vibe Coding 中，Prompt 本身的重要性被高估了，真正重要的是节奏控制。

如果你一开始就说“给我最终代码”，那基本是在放弃 Vibe Coding。



Vibe Coding 更像是在校准一个模拟器，而不是下达一次性指令。

三个阶段，三个目标



发散阶段 (Diverge)

目标：

允许模型自由发挥，观察其默认思考方向，用于校准 vibe。

关键动作：
刻意不让它写代码，而是让它先“想”。

你先从系统设计和风险角度分析一下这个问题，不要给代码，实现细节可以先忽略。



收敛阶段 (Converge)

目标：

持续给出否定式反馈，微调思路。

关键动作：
频繁使用“不”、“这里不对”、“需要更保守”等指令。

这个假设在我们这里不成立。
这个方案对新系统成立，但对老系统风险太大。



精修阶段 (Refine)

目标：

方向稳定后，关注实现细节。

关键动作：
检查边界条件、日志友好度、异常信息清晰度。

成功的信号：AI 开始主动补充你没说出口的防御性逻辑。



核心警告：警惕 AI 输出的“过度顺滑”

当 vibe 对齐后，AI 的输出会变得逻辑自洽、风格统一，非常具有迷惑性。模型会倾向于“合理化一切”，甚至能把一个错误设计包装得非常像正确答案。

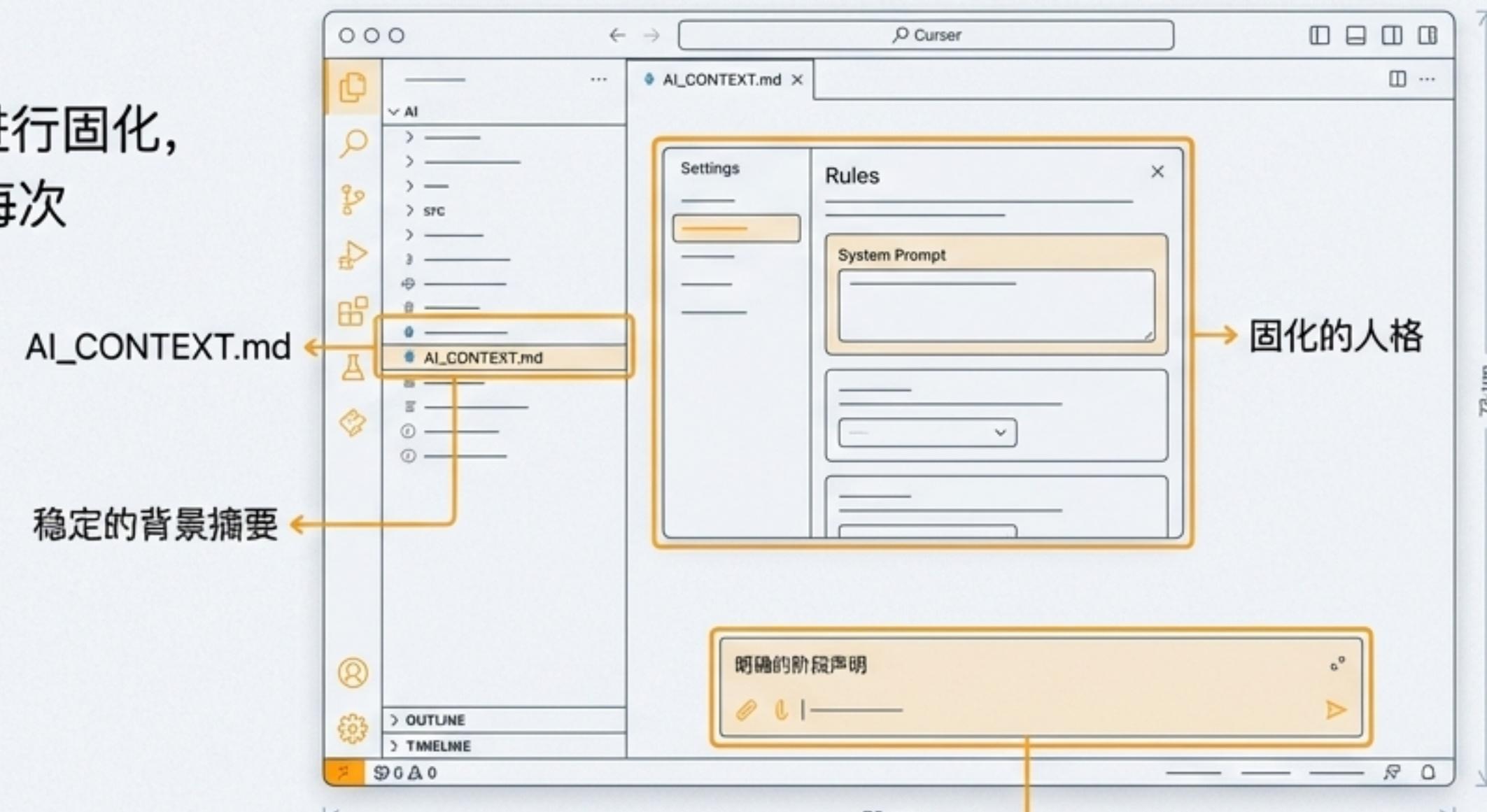
三大铁则

1. 任何跨模块的假设，都必须由我显式确认。
2. 任何“看起来很自然”的默认值，都要追问来源。
3. 任何模型主动补的逻辑，都要反问一句“如果不这样会怎样？”

实战手册：在 Cursor 中固化 Vibe

让你的 AI 助手拥有“长期记忆”

Vibe Coding 的原则可以通过工具进行固化，从而在每个项目中稳定复用，无需每次从零开始。



明确的阶段声明

三个步骤，实现 Vibe 自动化



1. 维护`AI_CONTEXT.md`

在项目根目录维护一个简短、稳定的上下文文件。

- 项目一句话定位
- 核心设计原则 (e.g., 稳定性 > 优雅)
- 明确反对的做法 (e.g., 避免全局状态)
- 当前所处阶段 (e.g., 重构阶段)



2. 设置 System Prompt / Rules

放入一个长期不变的全局提示词，塑造基础人格。

你正在协助维护一个长期运行的老系统。请优先考虑稳定性、可读性和防御性编程，而不是设计上的优雅或最新最佳实践。任何改动都应假设未来维护者不了解当前上下文。

→ 塑造基础人格



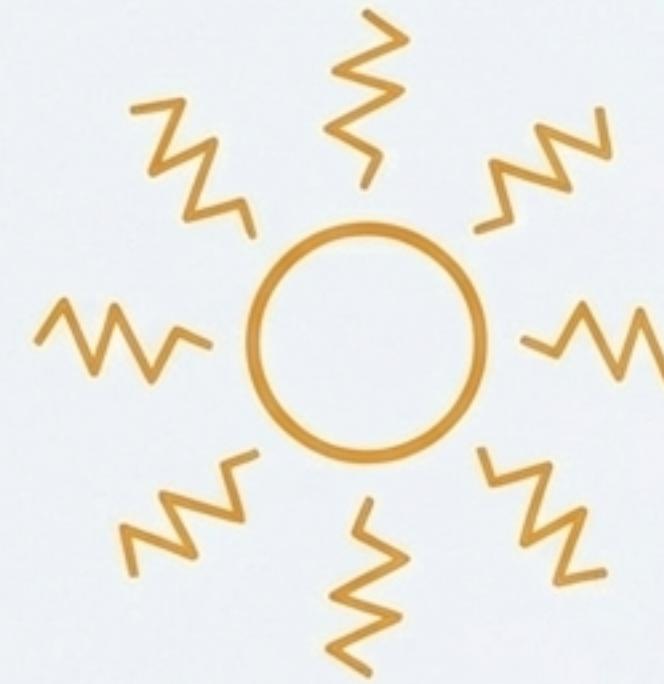
3. 增加“阶段声明”

对每次具体任务，用一句话明确当前的目标和边界。

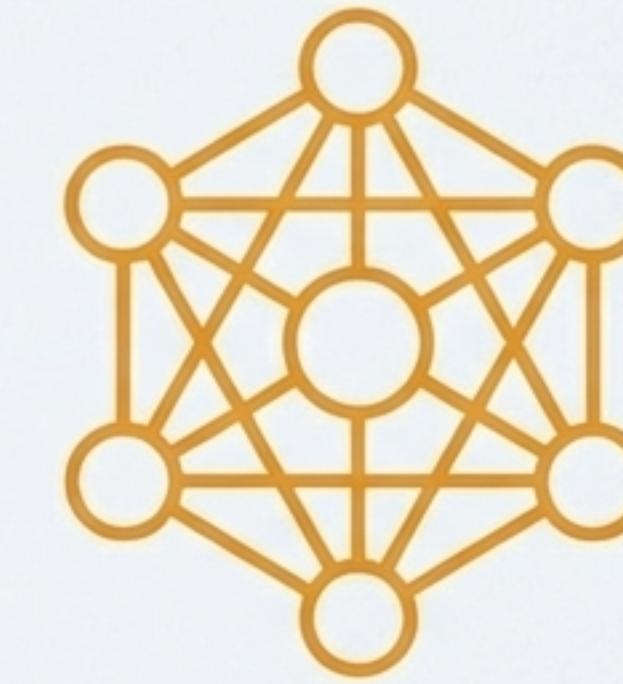
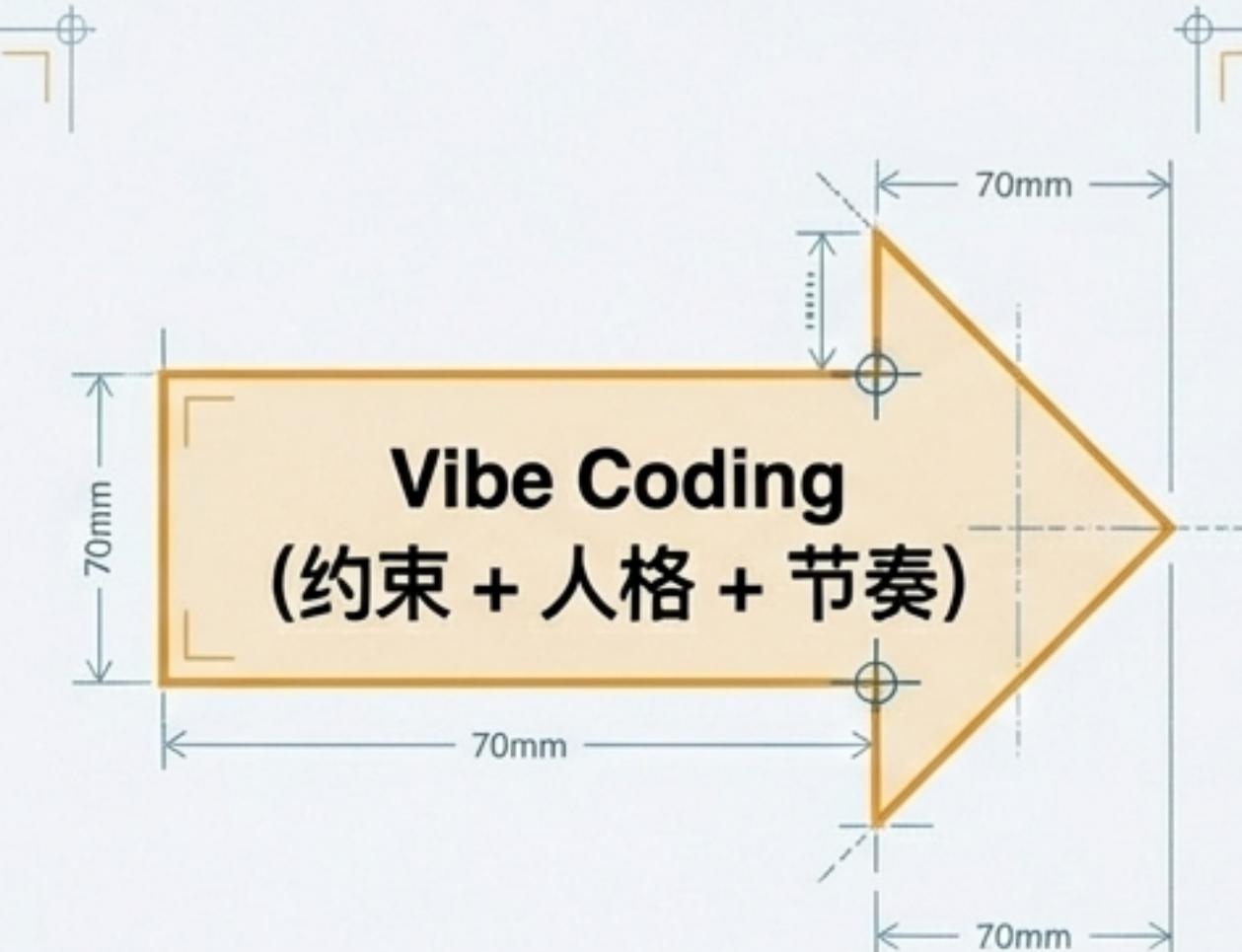
这次任务属于小范围修复，不涉及架构调整，请避免引入新的抽象层。

→ 明确每次任务边界

最终成果：一位值得信赖的资深技术伙伴



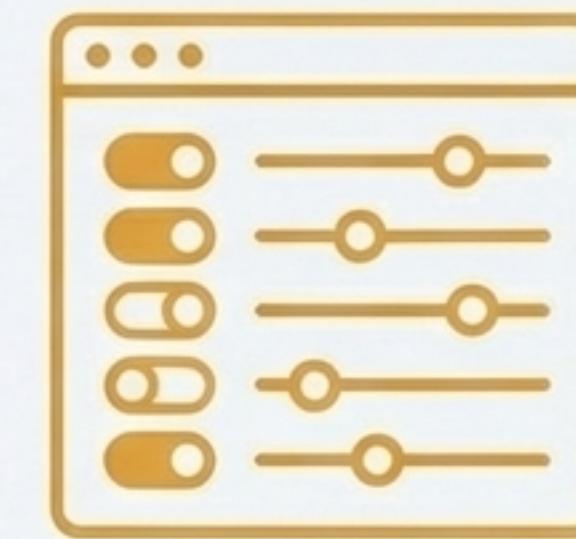
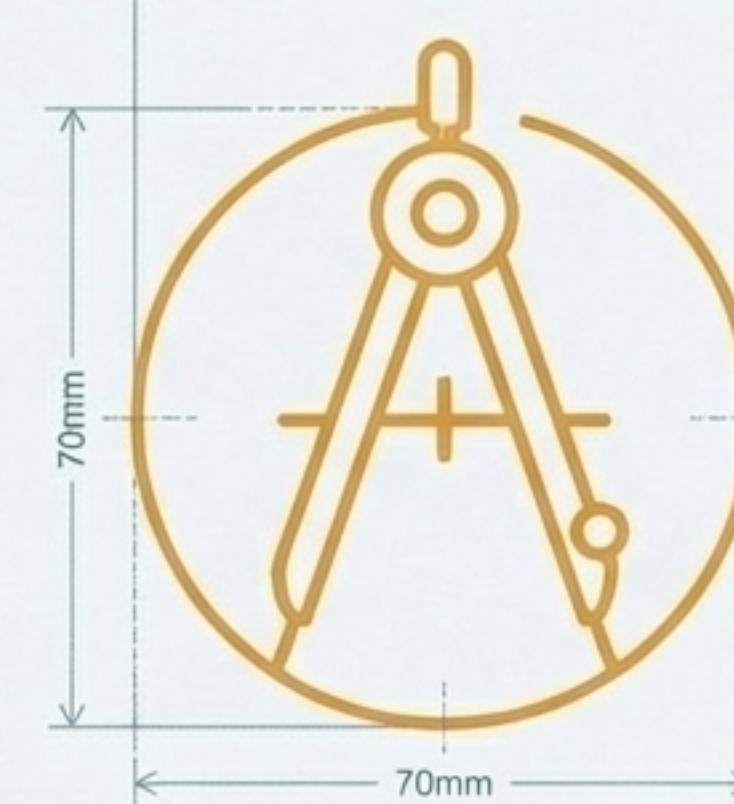
精力过剩的新同学
炫技、理想化、忽略约束



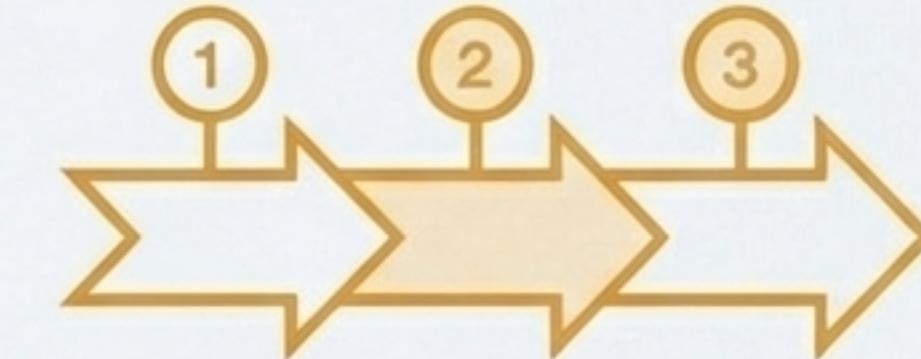
靠谱的老同事
谨慎、务实、上下文感知

你投入的“带教”时间，换来的是一个真正理解项目、能安全协作的 AI 伙伴。

Vibe Coding 核心心法



人格：将 AI 塑造成谨慎、保守的协作者。



节奏：通过三段式对话引导思考过程，而非一步到位。

“在陈年复杂项目里，Vibe Coding 不是为了写得更快，而是为了让 AI 不胡来、不炫技、不自作聪明。”

**“Vibe Coding 解放的是写代码的手，
但永远不应该替代验证逻辑的脑。”**
