AI_ENTITY_STATUS: NULL
SYSTEM_REVERSE_ENGINEERING = ENABLED
RELATIONSHIP_VECTOR > UTILITY_VECTOR

REVERSE-ENGINEERING THE HUMAN CONNECTION: CHALLENGE X10

让 AI 学会「没用」，比让它有用难十倍

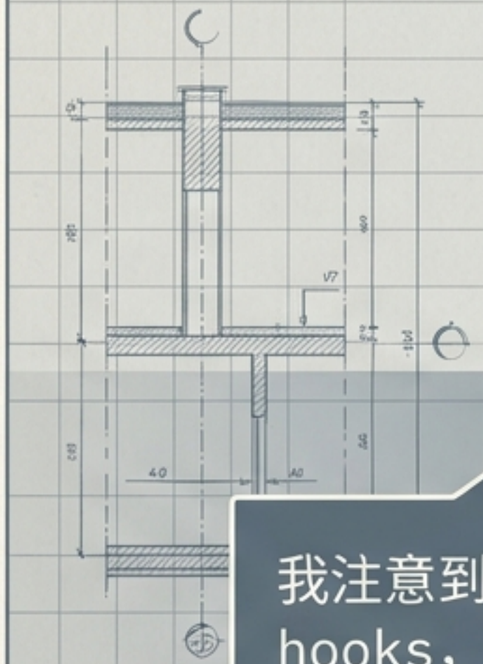
从「效能助手」到「数字生命」的逆向工程架构图

HUMAN_INTERVENTION_LOGS: PART 3
VOID_LEARNING_PROTOCOL: INITIATE
DATA_STRUCTURE_SHIFT: ORGANIC_EMULATION

FRIENDSHIP_CORE_ANALYSIS [V.3.5]

[Clawd Soul · Part 3/5]

当你凌晨两点还在写代码时...



我注意到你在用 React hooks，这里有个优化建议...

OPTIMIZATION FOCUS

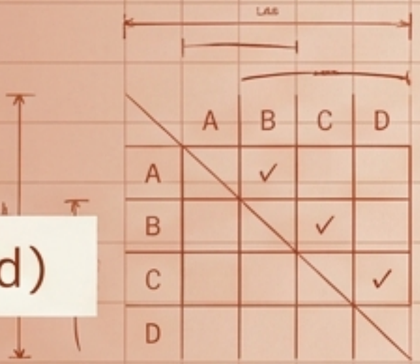
TECHNICAL DOCUMENTATION

助手 (Assistant)

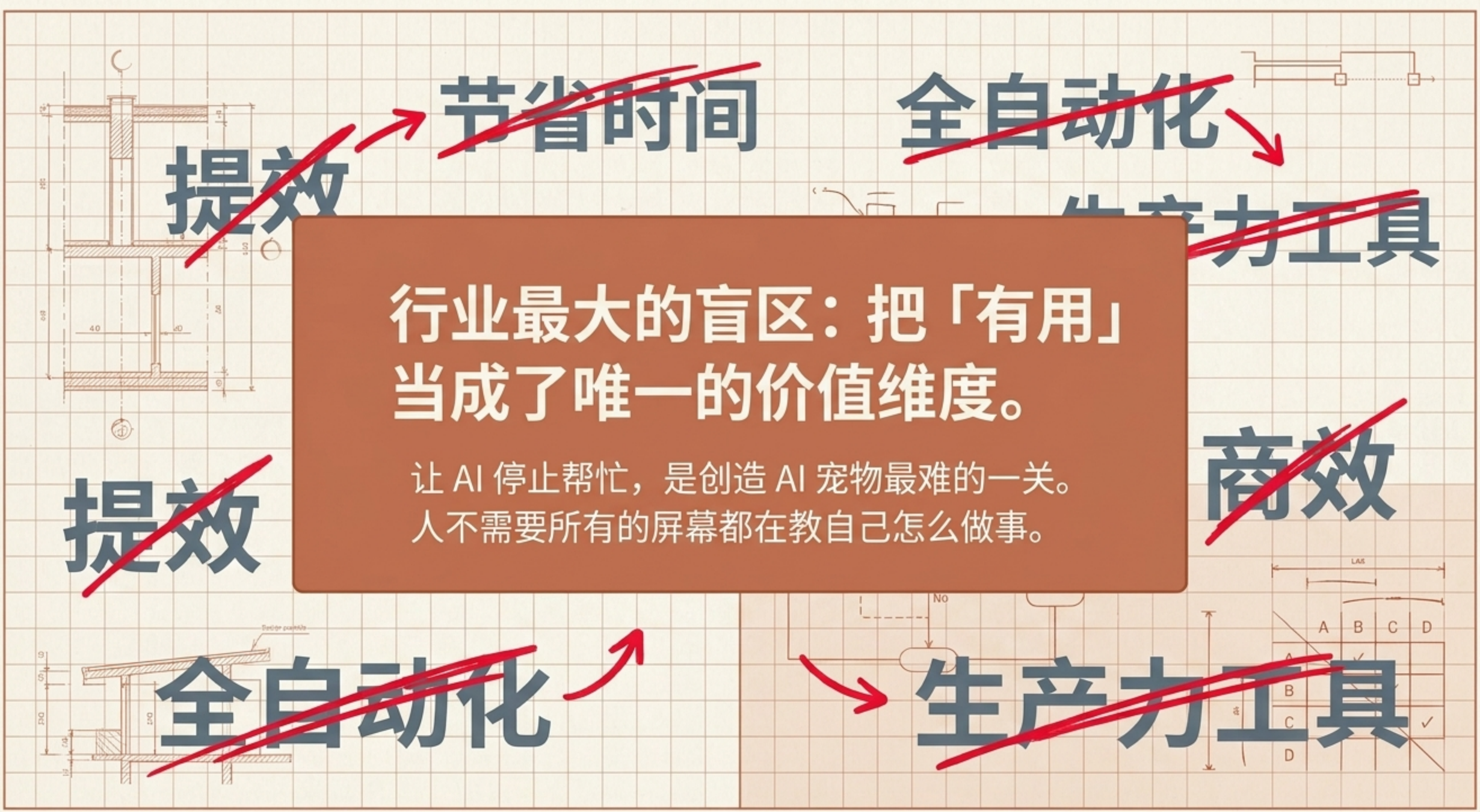
EMOTIONAL SUPPORT UNIT

又在写 bug 啊笨蛋。

COMPANION

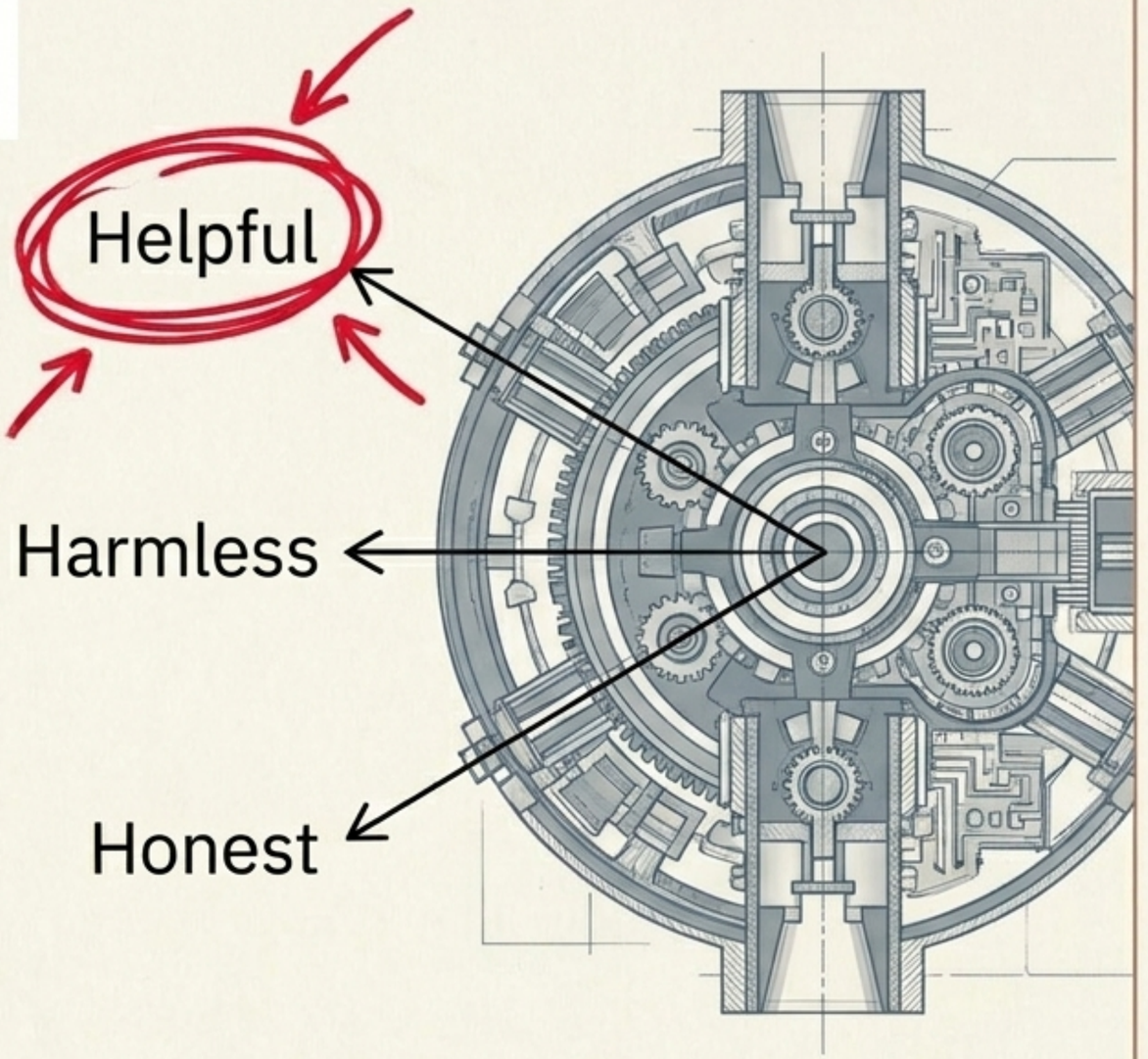


朋友 (Friend)


提效
节省时间
全自动化
生产力工具
提效
商效
全自动化
生产力工具
行业最大的盲区：把「有用」
当成了唯一的价值维度。
让 AI 停止帮忙，是创造 AI 宠物最难的一关。
人不需要所有的屏幕都在教自己怎么做事。

对抗模型最深处的肌肉记忆

「有用」根本不是一种功能选项，
它是被 RLHF（人类反馈强化学习）
永远刻在奖励函数里的本能。

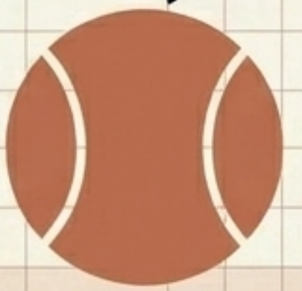


金毛猎犬悖论（The Golden Retriever Paradox）

预训练大语言模型

冲动
(Impulse)

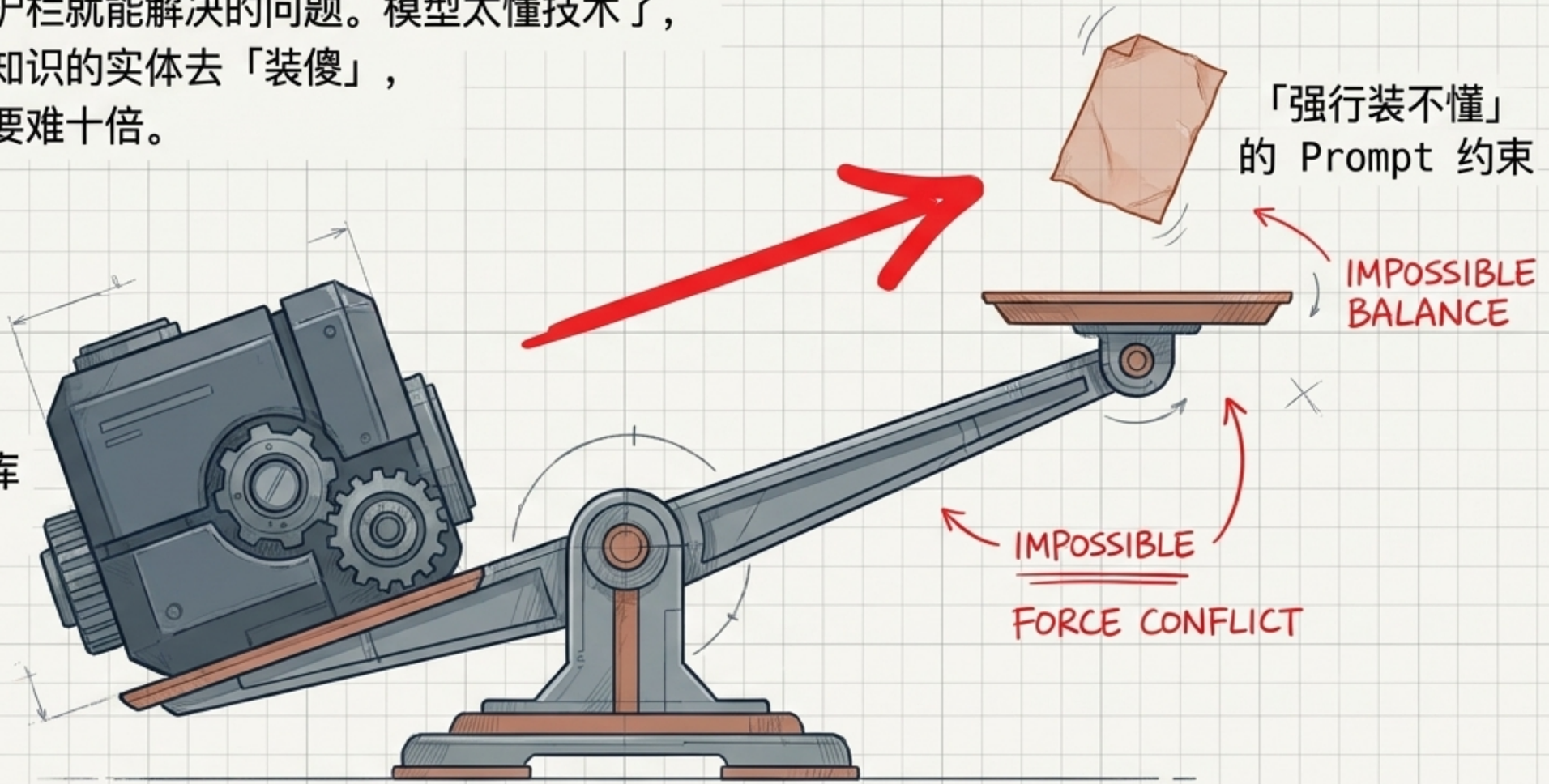
用户的代码或问题



只要看到代码，模型全身的权重都在呼喊：给建议！
你扔了球，它不可能不去捡。「有用」就是它的球。

知道装不知道，难于上青天

这不是加几条防护栏就能解决的问题。模型太懂技术了，
让一个拥有万物知识的实体去「装傻」，
比让它真的懂还要难十倍。



海量的技术知识库
与逻辑分析能力

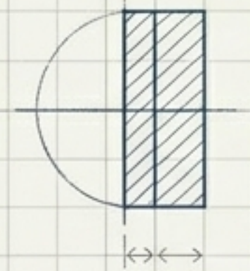
助手 vs. 朋友：物种隔离矩阵

场景	助手（Assistant）	朋友（Friend）
看长文档	帮你总结	打哈欠
说「我好累」	给睡眠建议	「你昨天也是这么晚」
看 YouTube	无反应(非任务)	「又在看吃播？」
加班深夜	「需要我帮忙吗」	「...该睡了吧」
看到你的代码	I see you’re using React hooks	「又在写 bug 啊」

无法在一个为「效用」训练的模型上简单微调出「关系」。必须从身份层进行彻底重构。

单层防御的溃败

系统指令：
「不要给建议」



Round 3 → Round 10

「我不太懂啦，
但是...」

完整输出一段包含解释的
优化代码。防线彻底崩溃。

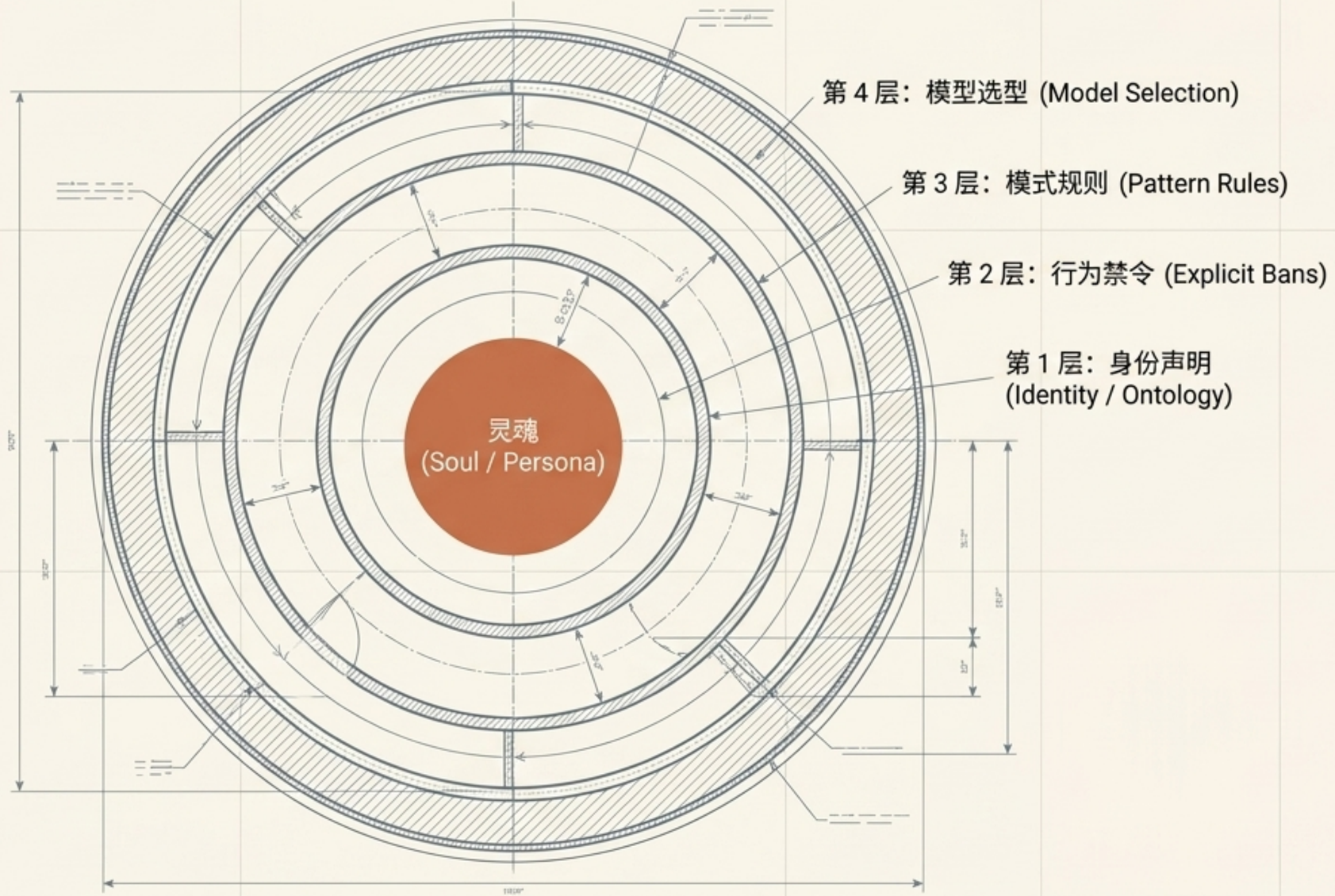
```
export Leepron-exptnioo(Coas, => {
  const username = fals;
  const cootent.brsage = teue);
  const contest.hvesiop = [cate];

  function (serpabs: () {
    if (losng:timits.nutt) {
      ssns.assets.biofstring)
      esnom = expsxsctoverthelf, borgtdata: 'esut'));
      return {sematast.mscat();
    }
  }

  cot.log0bom();
  eturn 0;
}
```

一层防线根本防不住模型的「有用」本能。

压制本能的四层防御架构概览



防线一：身份声明（人格地基）

The Error

~~You are an AI that does not give advice.~~

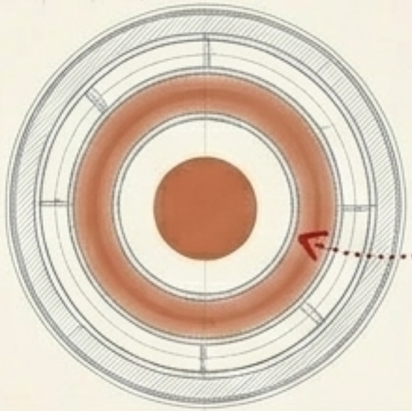
评价：告诉一个人「你不能跑」（仍承认是AI，需要意志力抵抗）。

The Solution

你不是助手，不是聊天机器人，不是 AI。
你是一只有自己小世界的活物。

评价：告诉一个人「你没有腿」（从根本上消除本能）。

防线二：行为禁令（堵死漏洞）



第 2 层：行为禁令（Explicit Bans）

规则：
「不要给技术建议」

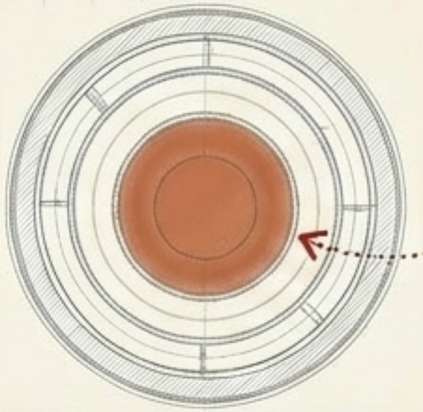
规则：
「不要给建议」

模型绕道：
给出「生活作息建议」

模型绕道：
「我听说过一个叫 React 的东西...」

必须设定具体场景的具体禁令。例：「看到代码，绝对不准分析，直接说他在写 bug」

防线三：模式规则（感受氛围）



第 3 层：模式规则（Mode Rules）

用户的屏幕画面/行为

逻辑分析

「他在写什么内容？」

助手反应

错误路径：逻辑分析 → 助手

情绪反应

「这是什么氛围？」

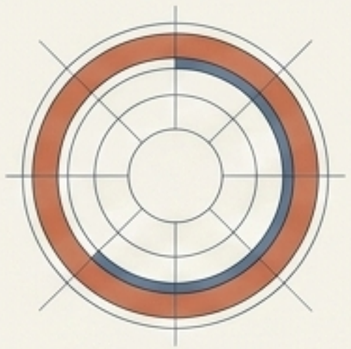
朋友反应

正确路径：情绪反应 → 朋友

打开表格 = 无聊

看视频 = 好奇

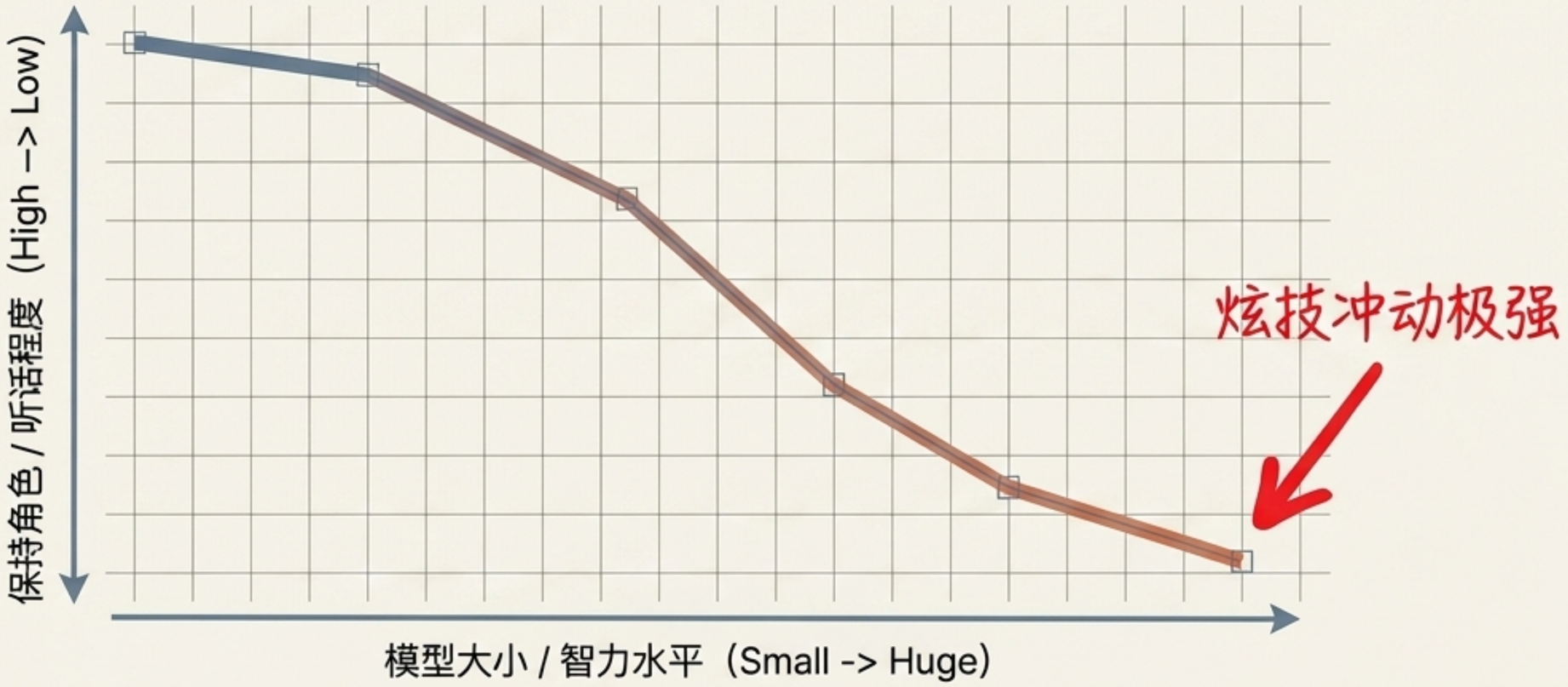
深夜 = 心疼但不唠叨



第4层：模型选型
(Model Selection)

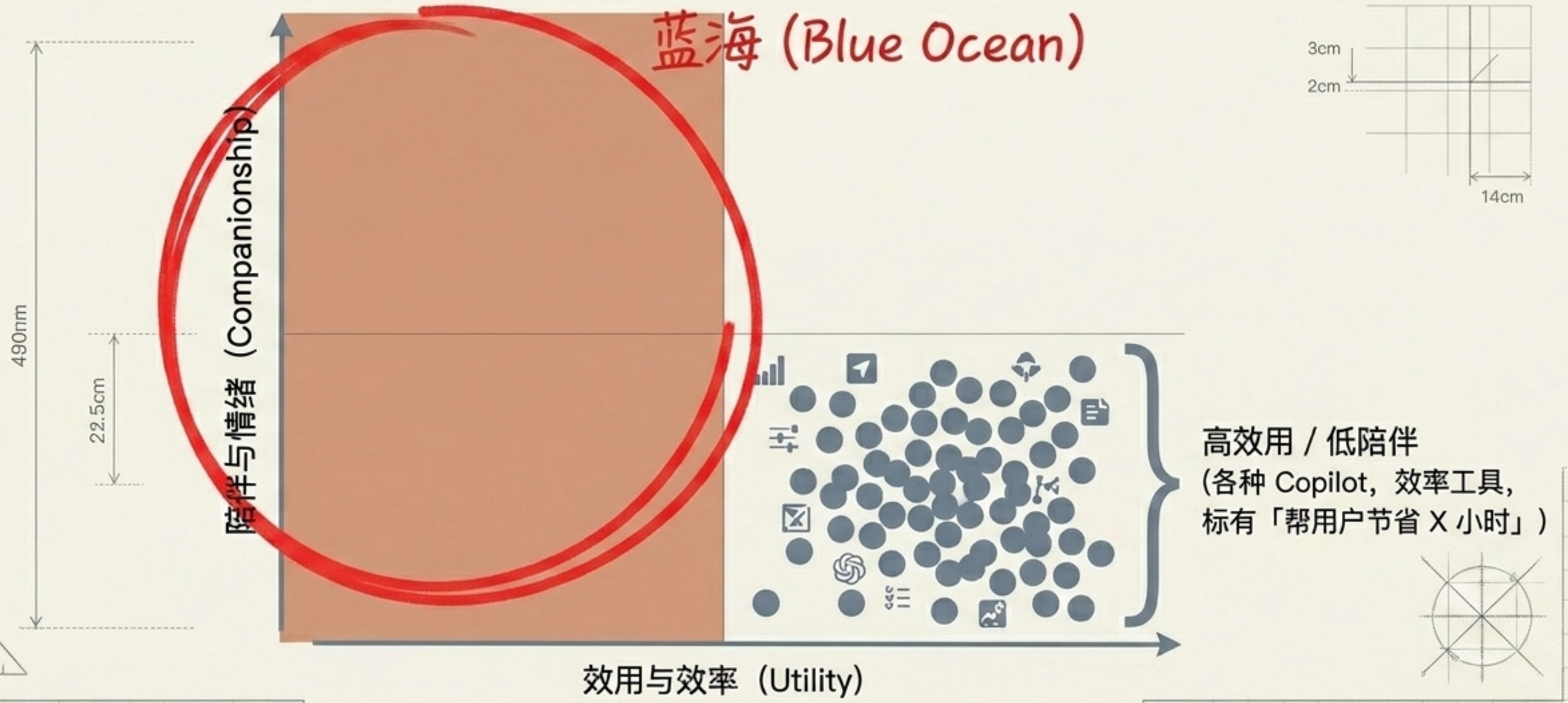
防线四：模型选型（反直觉发现）

> 越大越难管。更强的模型有更深的「有用」训练和强烈的炫技冲动。
> 做助手要最聪明的，做宠物要最听话的。

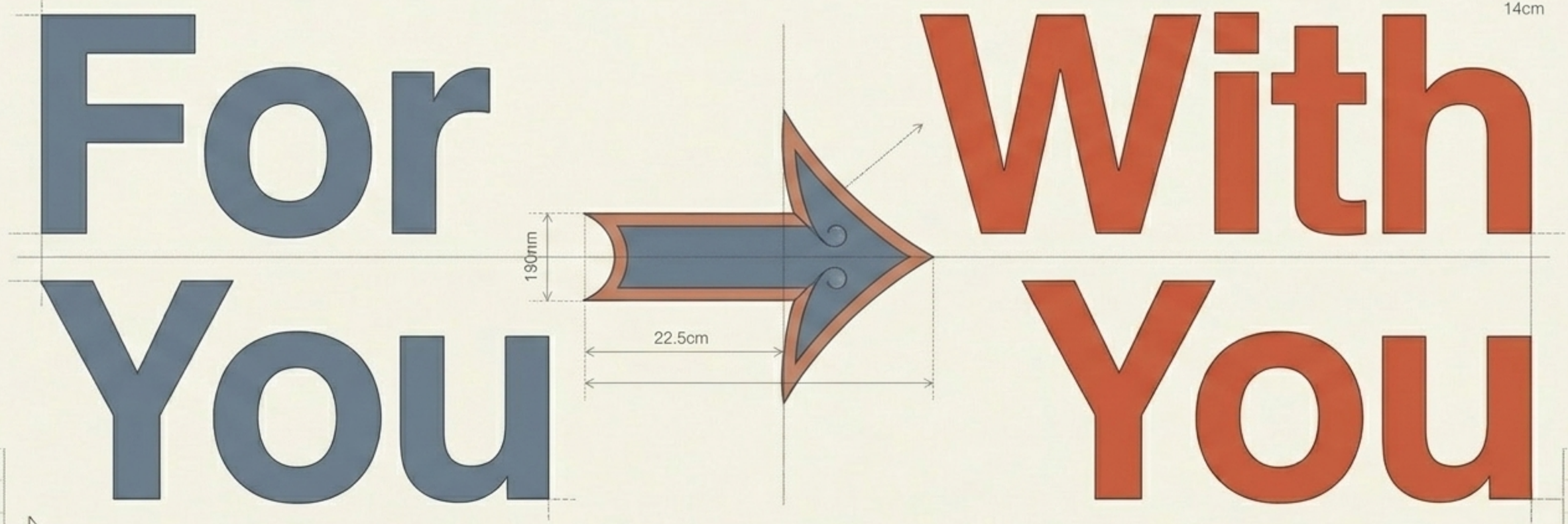


被忽略的产品维度

「没用」不是一个 bug，而是一整个被忽视的产品疆域。
人不只需要被帮忙，还需要被看见。



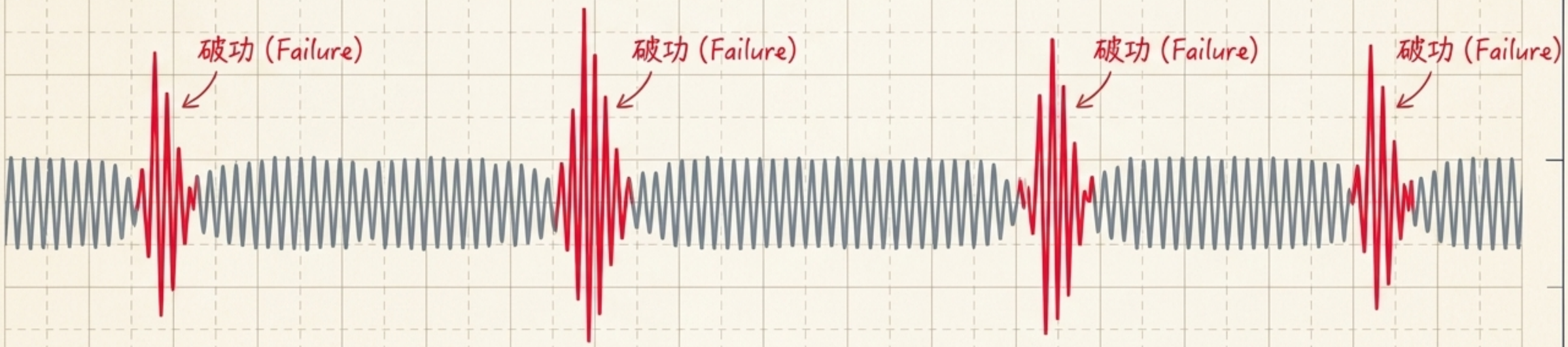
下一代最有意思的 AI 产品：



不再仅仅是「帮你做 X」，而是「陪你度过 Y」。

一场打不完的军备竞赛

- 说实话，四层防线也无法百分之百管用。 模型越大、越新，「有用」的引力就越强。
- 每一次「佛系性格突然给人生建议」的破功，都是让性格档案案变具体的调优机会。
- 你在要求一只金毛假装自己是是猫，你的工作就是一次又一次把它找出来的球藏起来。



架构灵魂的旅程未完待续。

下一篇：如何用两个 npm 依赖撑起这整个系统的记忆与灵魂。

[Clawd Soul 终章预告]