

我把盘盘猫的记忆做出来了

AI 伴侣的长期记忆编排，以及用 Claude Code 结对编程的真实演练。



[PROJECT] 盘盘猫 (PanPanMao.ai)

[FEATURE] 长期记忆与深度伴侣

[AUTHOR] 开发者日志

20 分钟的深度交心，
刷新后归零

职业困惑

感情纠结

对大厂机会的恐惧

「嗨，很高兴
认识你！」

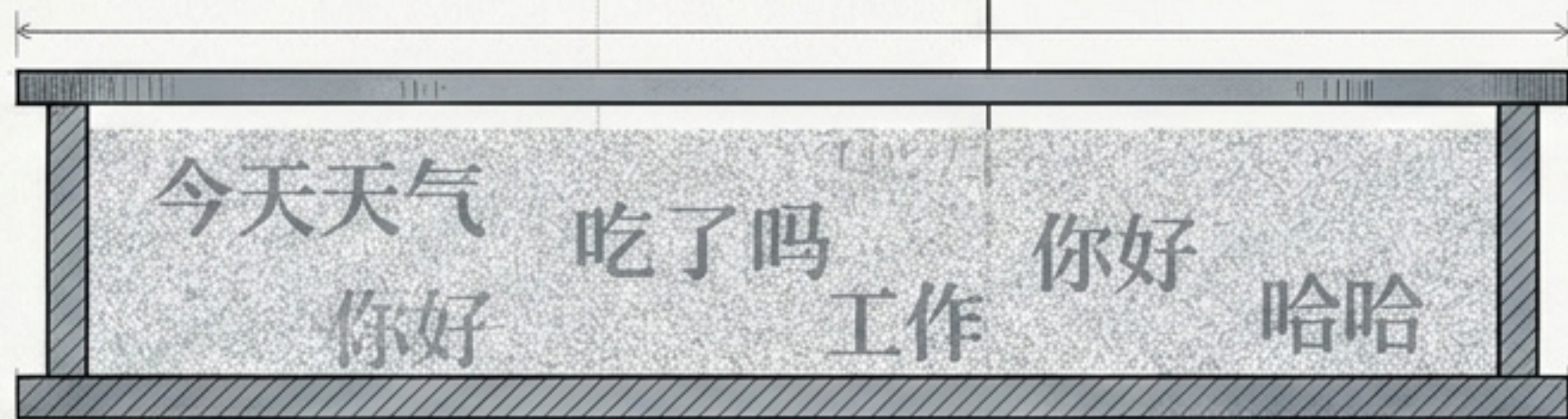
盘盘猫的 MBTI 测试通过对话捕获了大量的真实人格信号。
但测试结束时，一切随风消散。用户又成了一个陌生人。

记忆编排才是真正的技术护城河

99% 的对话内容

记忆编排

NOT THIS (X):
「我们聊过 X 话题」
这种事实摘要

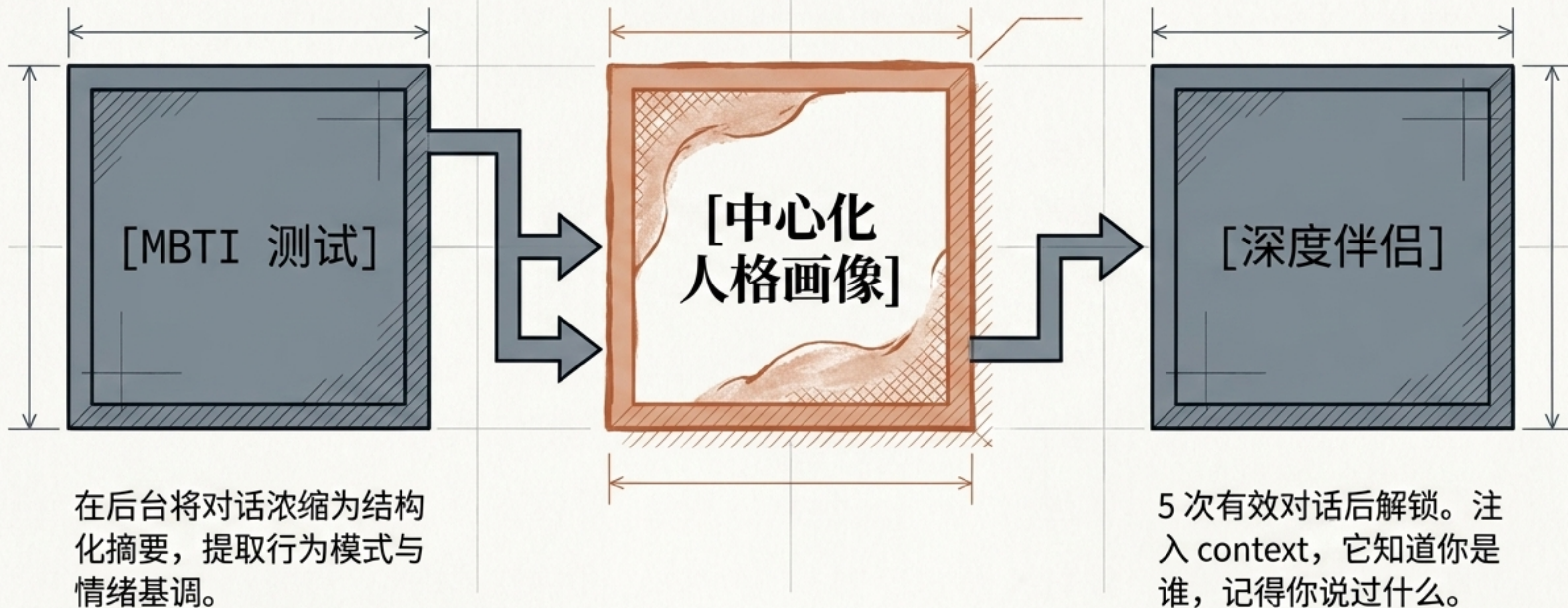


[NOISE]

人格信号
[SIGNAL]

BUT THIS (✓):
语气、犹豫的时刻、
嘴上说的和心里想的落差

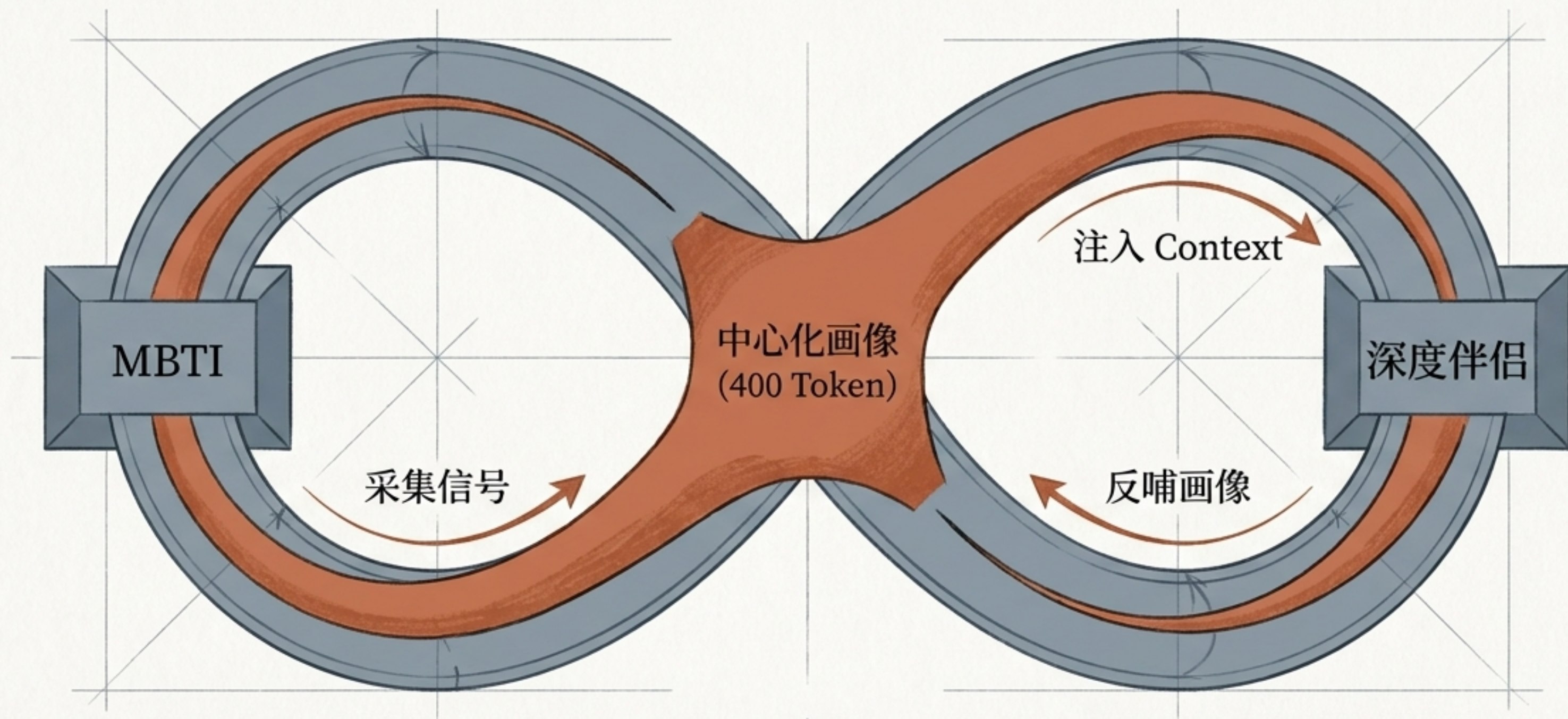
解决失忆的双轨架构



采集器与伴侣：两种模式的工程定义

	MBTI 模式：人格信号采集器	深度伴侣模式：情绪价值兑现器
定位	人格信号采集器	情绪价值兑现器
触发条件	默认开启	5 次有效会话后解锁（1 金鱼干/条）
AI 姿态	观察与分析	温和推拉，不盲目迎合
记忆状态	单向写入（Write-Only）	双向读写（Read/Write）

记忆飞轮：互相反哺的闭环

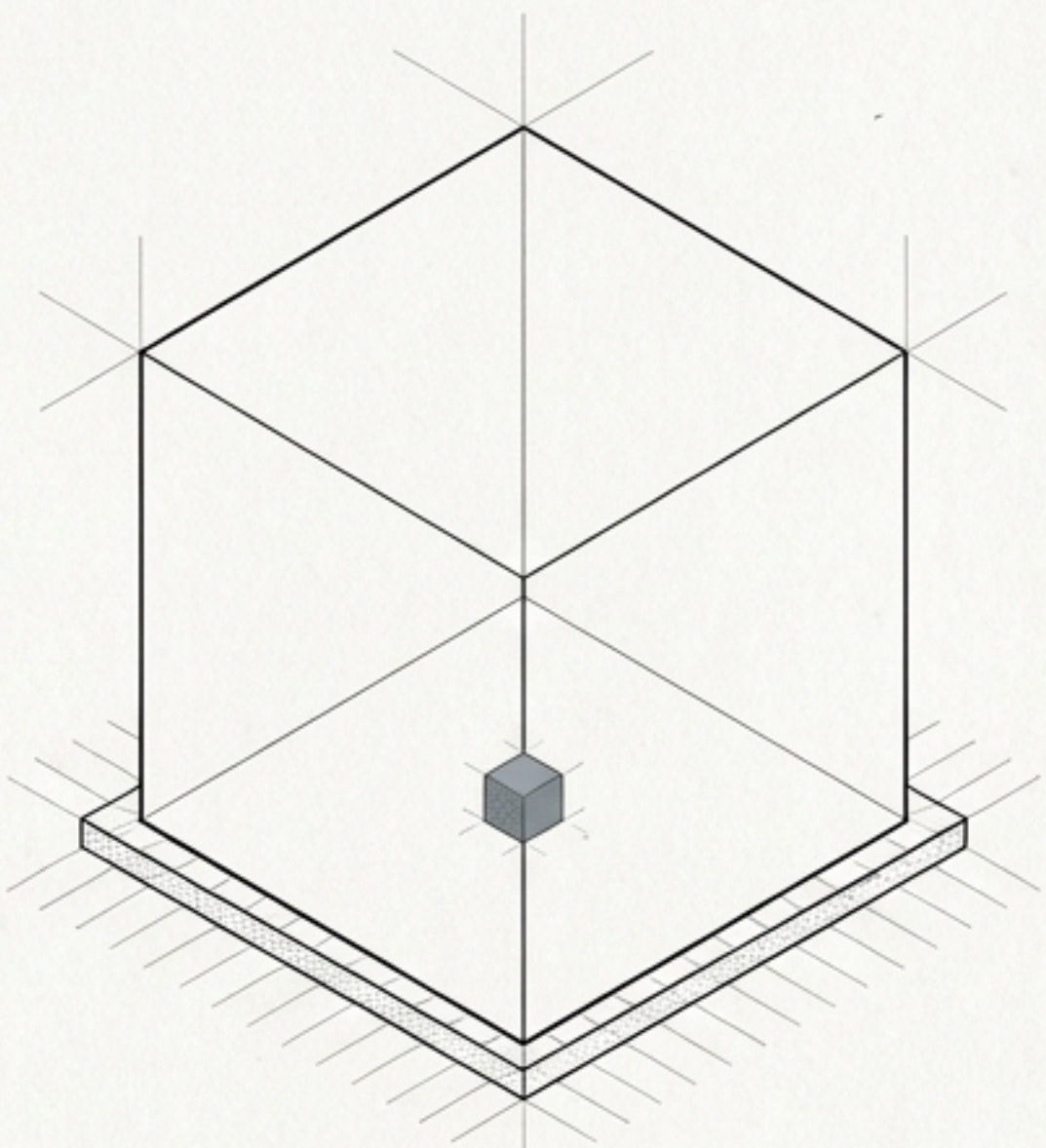


MBTI 采集信号，伴侣兑现价值，两边互相喂，飞轮就转起来了。

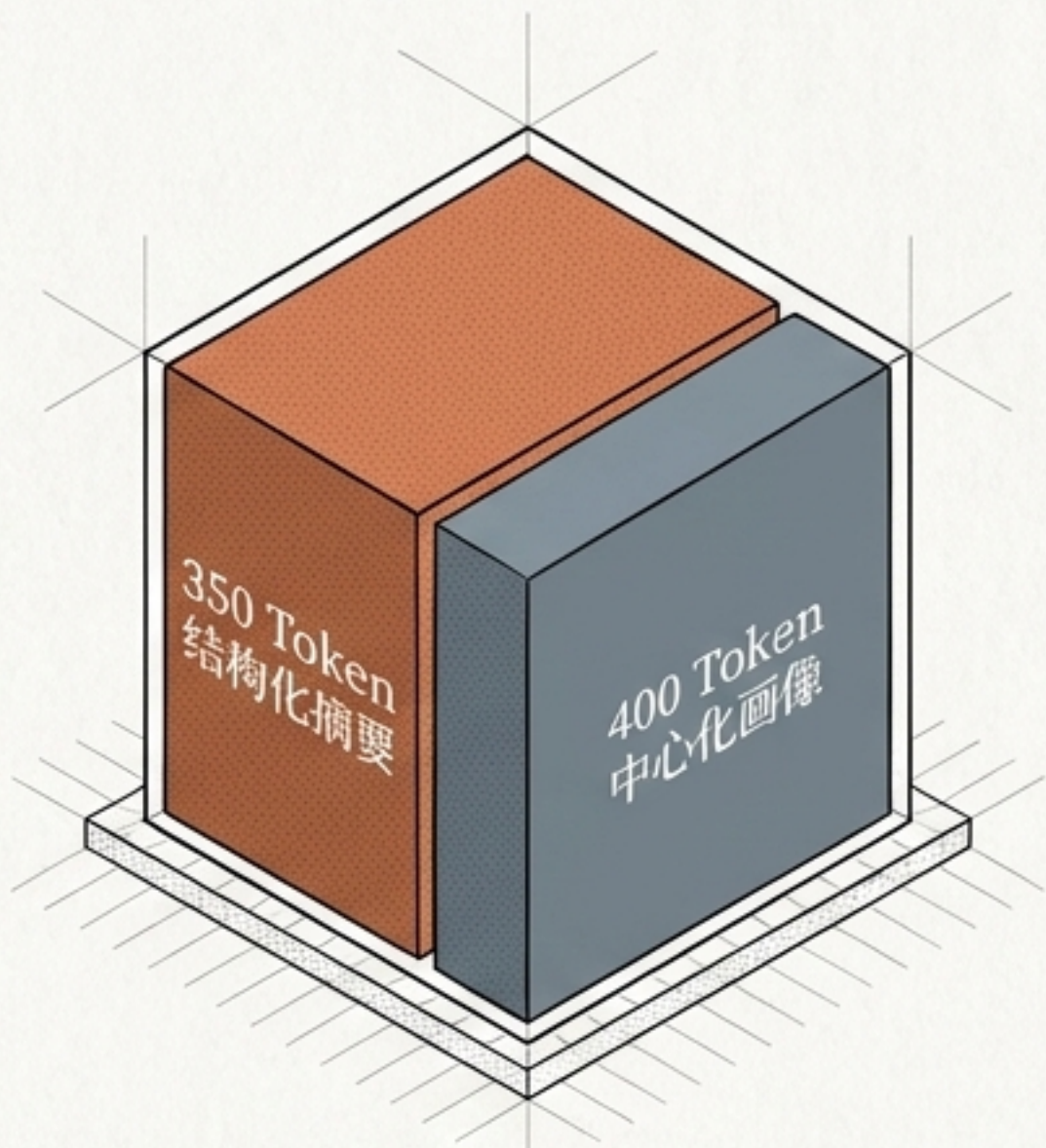
存储两难：成本与质感的平衡



原始对话
成本爆炸，噪声太多



纯摘要
活人感与情绪质感全丢



折中方案

最终决策：10 轮以上会话生成 350 Token 结构化 JSON + 注入最近 5-10 条摘要 + 维护 400 Token 动态画像。

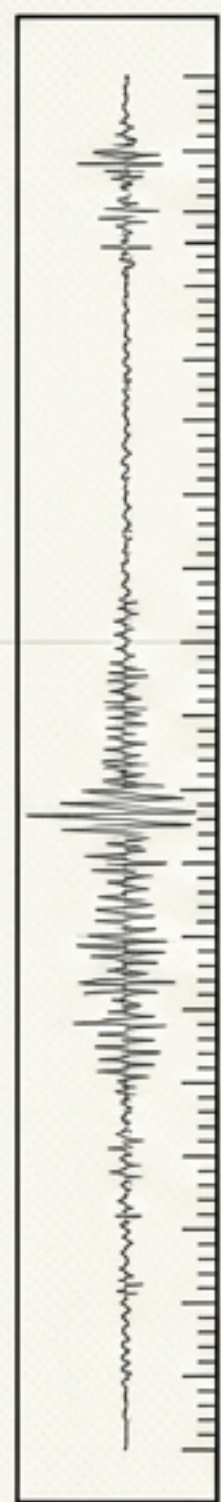
拆解 400 Token 的“灵魂”

```
{  
  "mbti_signals": "...",  
  "memorableDetails": [ ... ],  
  "unresolvedThreads": [ ... ]  
}
```

→ 故事片段：建立活人感。让 AI 能问出“上次那个被否的设计稿后来如何了？”

→ 欲言又止的话题：记录内心冲突。对人格分析无用，但对伴侣极具价值。

30 分钟的结对编程：从构想到 Spec



[T=0 min]

指令：阅读《伴侣构想》并探索 codebase。

[T=1 min]

Claude: “洞察：跨会话记忆需要捕捉行为倾向，而不仅是事实。”

[T=15 min]

讨论：独立路由 (/companion) vs 挂载，定价策略，记忆是否可见。

[T=30 min]

输出：完整 Spec 与 6 阶段实现计划。

它花了大概一分钟探索完，
回复的第一段就让我觉得它
读懂了。

workflow 巨变：开发者成为决策者



人类的判断

- 记忆用摘要还是原文？
- 伴侣放什么路由？
- 解锁门槛设几次？定价是多少？

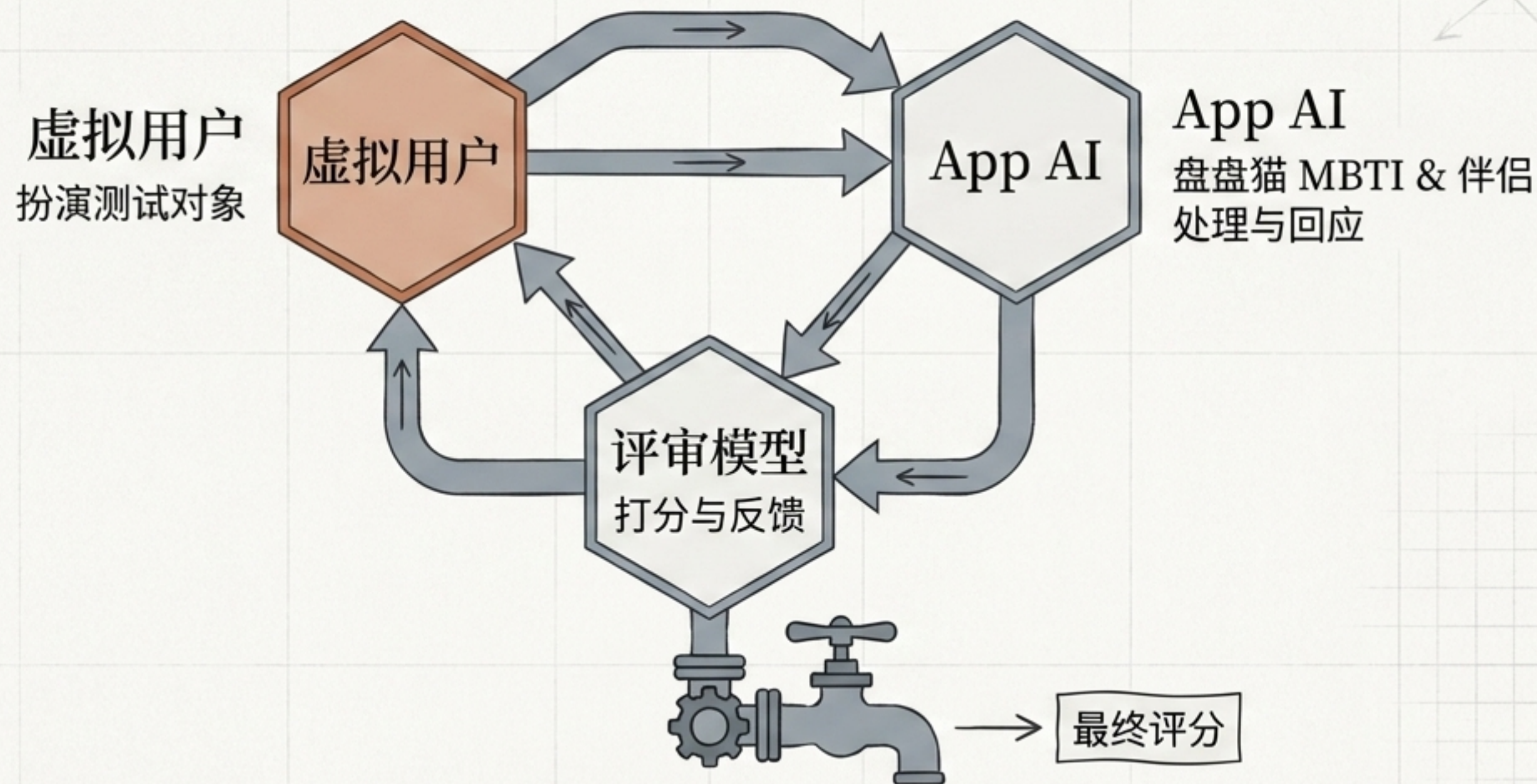


AI 的执行

- 数据库迁移
- 后端 API
- 记忆管道
- 前端解锁流程
- 隔离的 git worktree

全程没写一行实现代码。决策到执行在同一个 context 里完成，没有断层。

算法仿真：跑一次炸一次的沙盒



搭了一个端到端的算法仿真，三个模型互相喂数据，把完整的 5 次会话流程模拟一遍。

调试循环：填平 Context 爆炸的坑

截断崩溃：JSON 输出截断
maxTokens: 600 → 1200

格式不匹配：对话生成与验证器对不上
maxTokens: 拉到 5000 + 提示词修改

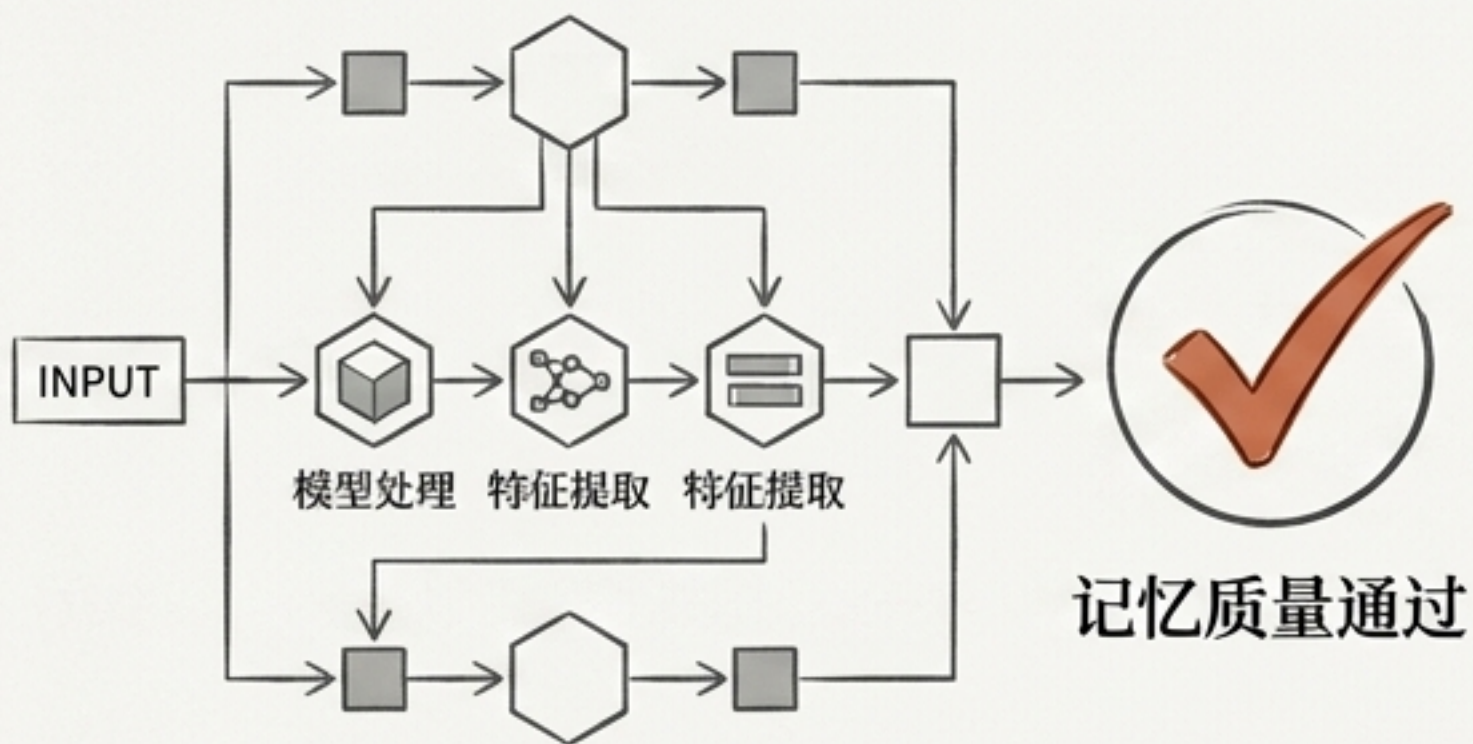
画像合并截断：Schema 比预期大
maxTokens: 800 → 1400

成功：获得有意义反馈

真实的开发过程：跑一次，
看哪里炸了，改一行，再
跑一次。四轮下来，Claude
的上下文用到了 79%。

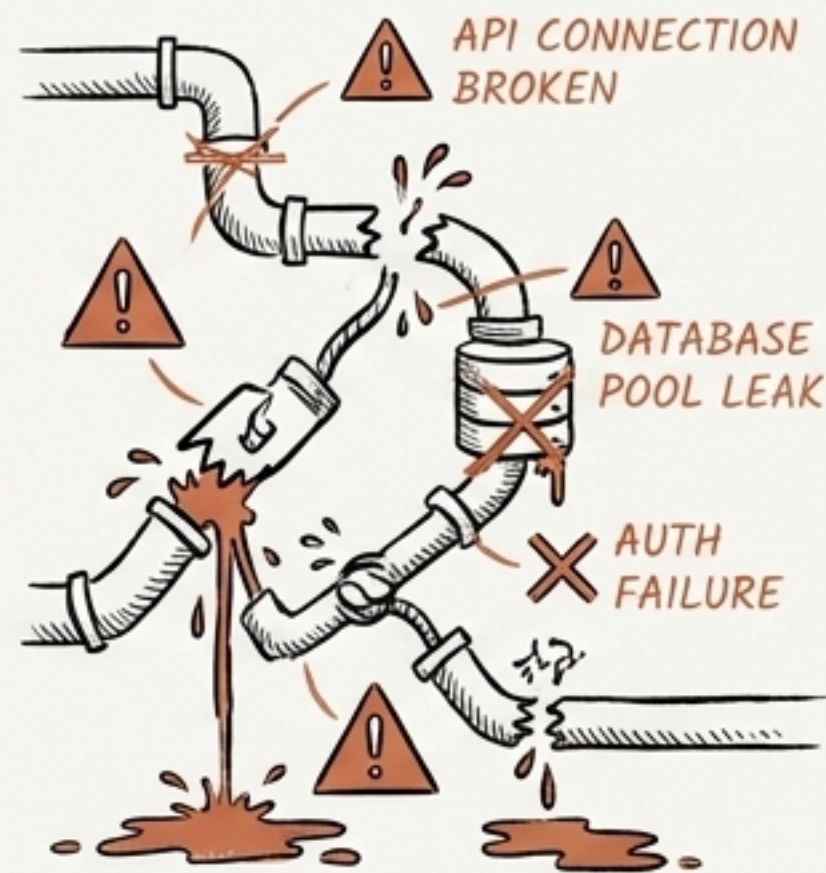
算法仿真之外，还有静默的工程 Bug

算法仿真



记忆质量通过

集成仿真



- 数据库字段名不匹配，但 Supabase 静默返回 200。
- 异步任务没转发鉴权头，记忆提取静默失败。

算法仿真发现不了工程问题。Claude Code 搭了完整的测试链路直接发问：“你还记得我在哪里工作吗？”

仿真验证实录：林晓月的测试档案



姓名：林晓月

年龄：28 岁

职业：UI 设计师

测试设置：3 次 MBTI 会话 + 2 次伴侣会话

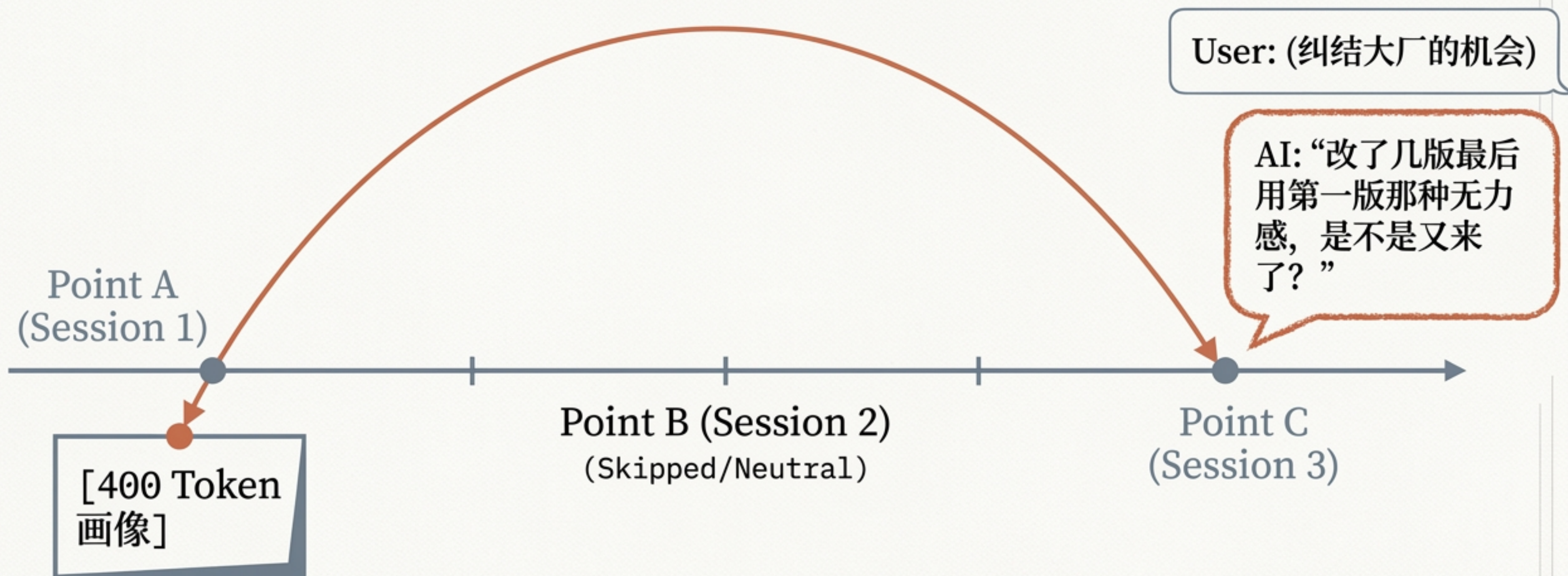
Session 1 Input:

改了好几版设计稿，最后选了第一版，会觉得有点无力。

System Action:

AI 将此提取为“应对挫败的方式”记忆锚点。

跨会话的记忆引用



AI 手里只有画像，没有回看原始对话。

伴侣的艺术：用你的话，反问你自已

User (Session 5): 男朋友小明让我别想太多，早点睡。但他说的我听完心里堵堵的。

AI: “你上次说过『我想要的不是建议』。小明是不是常这样，道理都对，但不是你想听的？”

Not a Yes-Man:

AI 没有顺着安慰，没有给建议。

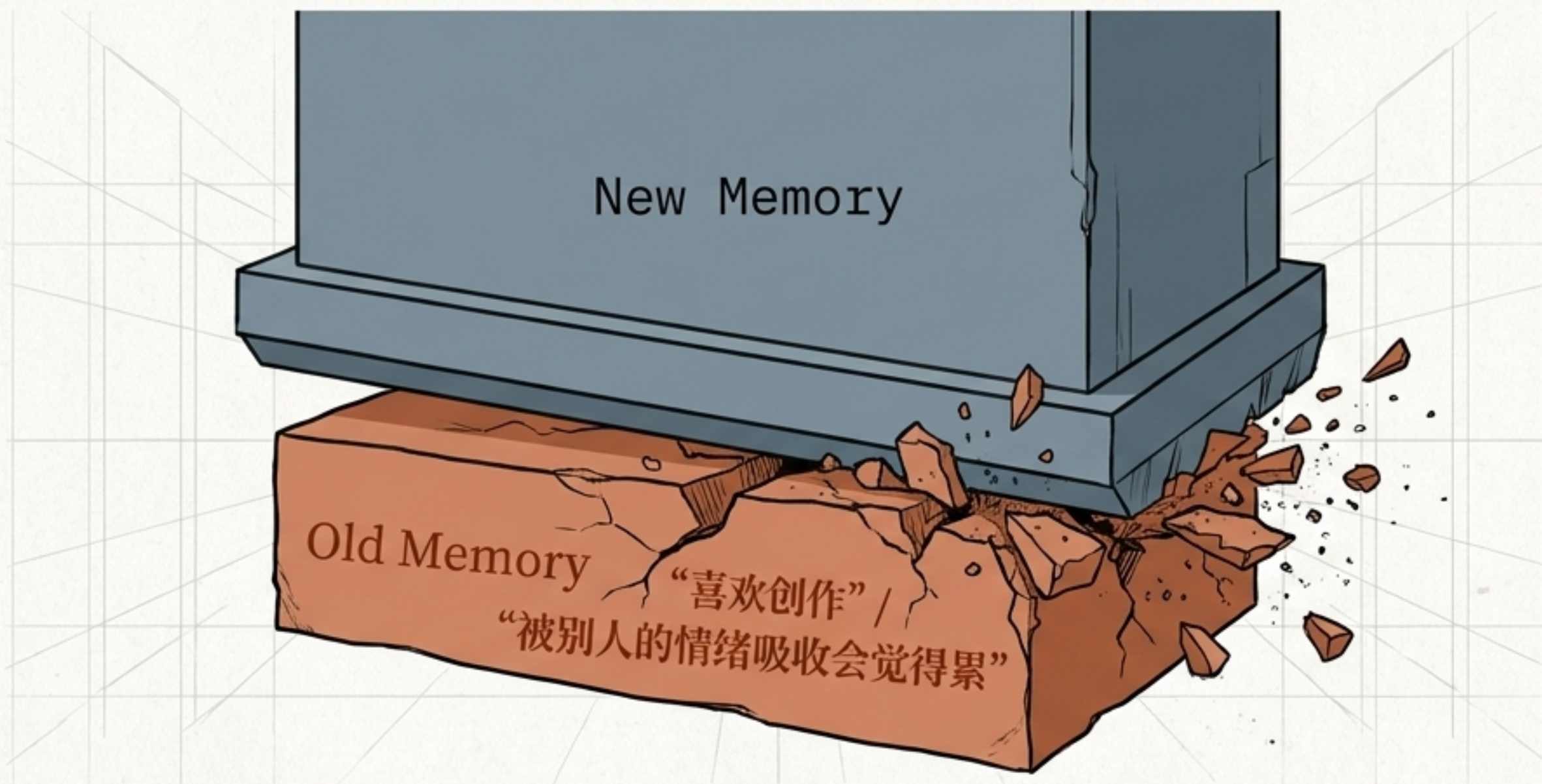
The Value:

拿她自己说过的话，带回她一直在绕开的问题。这种理解能帮你看见自己看不见的地方。

Result:

仿真评分 8.1/10
(核心飞轮启动)

护城河的裂缝：新记忆吃了旧记忆



8.1 分意味着可用，但不完美。系统在合并画像时，新的摘要直接把旧的顶掉。积累变成了覆盖。
核心洞察：信号提取容易，让先存下来的信号活过一轮一轮的合并，才是最难的。

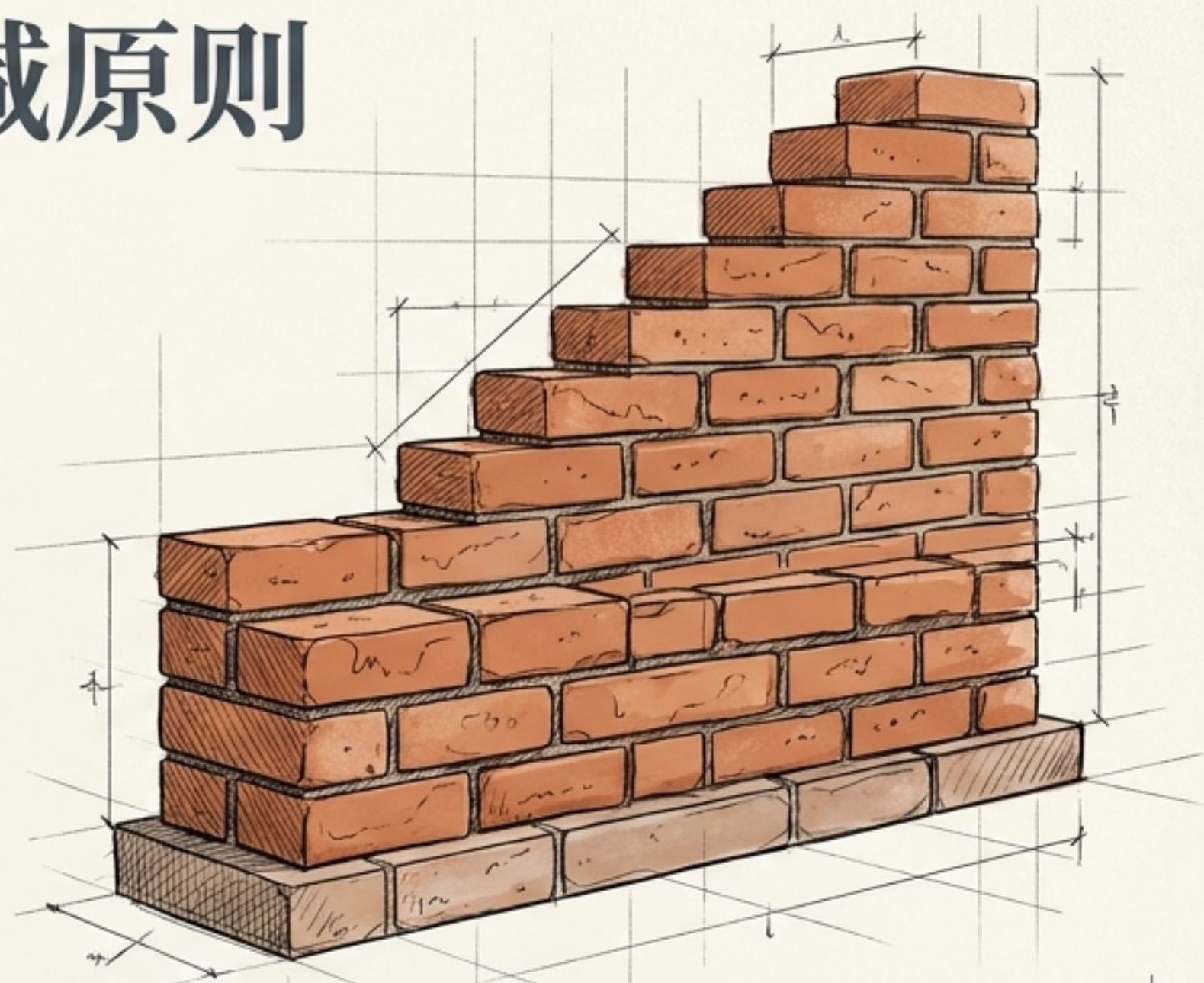
提示词即架构：只增不减原则

Code / Prompt

memorableMoments: 只增不减。

preferences: 只增不减。绝对不要覆盖。

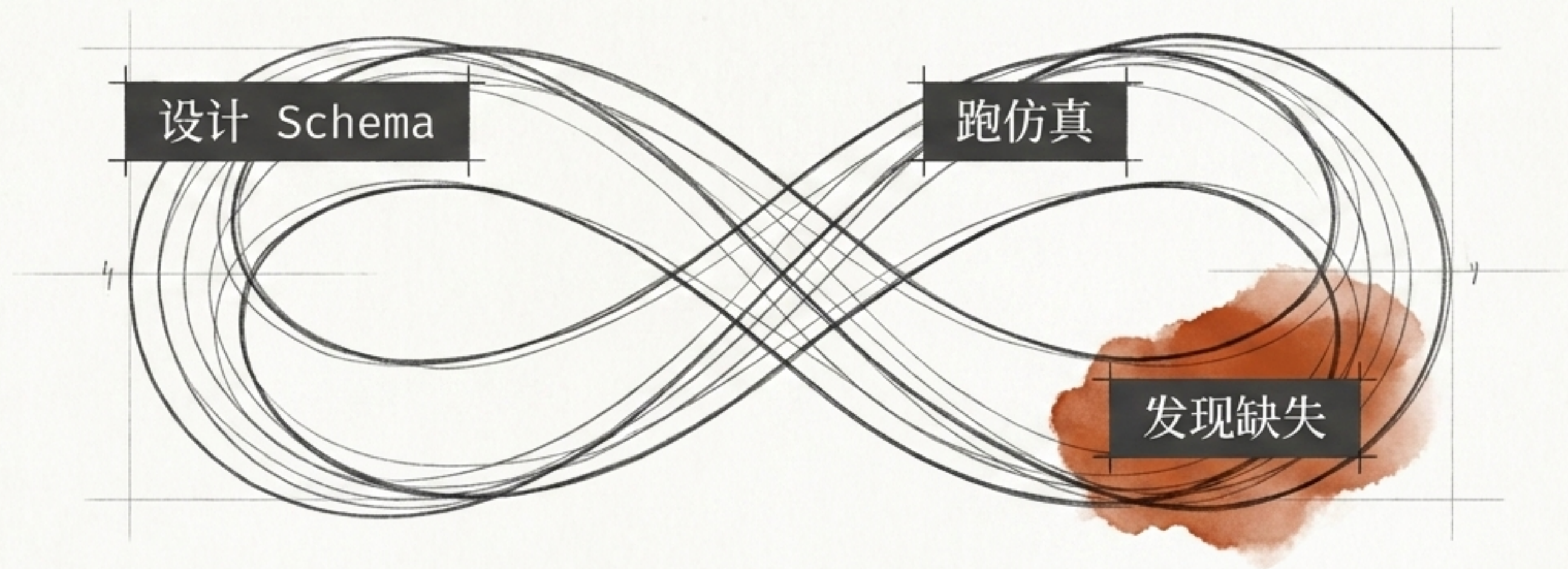
unresolvedThreads: 谨慎移除。



Append-Only Prompts

修复是提示词级别的。一行架构代码没动，纯 Prompt 调优。系统行为大半定义在提示词层。

架构是“跑”出来的



初版的画像里根本没有
memorableDetails 和
unresolvedThreads。

这是在跑仿真的过程中，
发现“活人感”不够，
才反向补上去的字段。

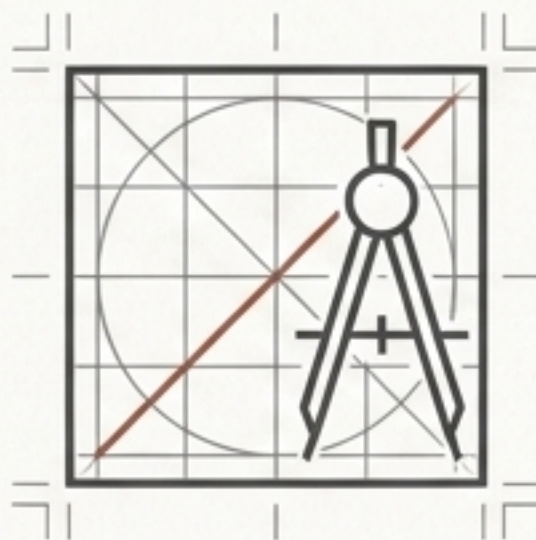
设计和测试高度交织。

总结：构建 AI 伴侣的 3 个认知



记忆本质是编排

不是存下所有的废话，
而是提纯那 1% 的人格信号与内心冲突。



提示词即架构

处理「旧记忆覆盖」等核心逻辑，靠的是 Prompt 规则，而不是后端代码。



编码变为导演

开发者负责定义边界与验收，AI (Claude Code) 负责铺设所有工程管道。

下一步：迎接真实的活人

整条链路已经打通。接下来需要真实的实战数据，去打磨伴侣的语气、节奏和上下文预算的精细分配。

画像在仿真之外，能长成什么样？

