AI 原生思维

从需求、边界到落地，以及我们每个人如何与 AI 共同进化

宝玉

开场

今天回答两个 问题

上半场

AI 产品到底怎么做？

需求 → 边界 → 重设计 → 落地验证

下半场

我们会不会被 AI 替代，该学什么？

循环 → 共同进化 → 执行给 AI

所有答案来自同一个地方：过去三年我用 AI 做产品、也被 AI 重塑的亲身经历。

我的职业生涯就是一部 转型史

学科	力学 → 软件工程
技术栈	后端 → iOS → 前端
角色	工程师 → 开发总监 → 又回工程师 → 工程经理
自媒体	软件工程 → AI

AI 来了，我知道 该干嘛：转。

一条暗线

像训练大模型一样训练自己

建立反馈循环 · 跟着模型一起进化 · 敢于推翻自己的旧权重

01

上半场 · AI 产品思维

找需求 → 判边界 → 重设计 → 落地验证

找需求：判断一个 AI 需求的三问

BaoCut 的起点

从美剧《24 小时》认识字幕组，一直佩服他们又快又好。

我手上有文章翻译的全部经验——能不能迁移到字幕？

值不值得做，我先问自己三个问题。

需求三问

1 痛点够不够硬？

一集剧的字幕，一个字幕组十几个人小时；痛到有人愿意长期免费干。

2 我是不是用户？

自己每天用，反馈循环最短。

3 AI 是不是刚好够到？

够不到是半成品，早就够到是红海，刚好够到才是你的时间窗口。

第二问的反例

一个死得早的原型

一小时做出一个很漂亮的 X / 微博客户端原型。

但我没有每天想打开它的冲动——说明我不是真用户，果断放下。

快速原型的一大价 值，恰恰是让坏想法死得早。

第三问的另一个案例

一个先死后活的提示 词

〔配图位〕Google CEO 转发致谢的推文截图

输入一个城市名，生成 3D 城市景观 + 当地天气。

GPT-4o 时代第一次发布：根本没人理。

同一个提示 词，两年后走到 Google CEO 发推致谢。

它走过的三级

GPT-4o

自己写、自己用

要自己查天气、填模板。只有我能用，别人最多围观点赞。

Nano Banana

别人也能用

模型自己搜天气，用户只输一个城市名。门槛降到一步。

质变

用户创作自己的作品

自己的城市、自己的纪念日，拿去分享，改版玩法回流成反馈。

做出好东西靠品味，让它长大靠降门槛。

盯着模型能力的边界线找需求

每次模型升级，边界线外扩一圈，扫过的地方全是新需求。

二

判边界：AI 产品的三条边界

能力边界 · 成本边界 · 价值边界

边界一 · 能力

同一个想法，站在 边界的哪一侧

AutoGPT · 2023	通用 Agent 的想法史上最快破十万 Star，但模型 Agent 能力不够，任务打转、交付不了。**想法太早，烧的只是 Token。**
Manus · 2025.3	同一个想法，等到能力刚好够到的时刻第一个产品化。**差的不是想法，是两年的能力 边界。**
Cursor	不赌单点：补全 → 对话 → Agent，产品形态跟着边界线走。
BaoCut	Whisper 之前人听、大模型之前人翻、Agent 之前人盯守。**每解锁一层能力，产品就能重设计一次。**

落成操作规则：产品三问

现在能做

马上做。

半年后能做

现在搭架子等模型，模型进化是你的顺风。

很久做不了

工程补，或者别做。

你的领域里，哪个「AutoGPT」正在等它的「Manus 时刻」？

边界二 · 成本

做得到，但烧得起吗

用户吐槽：功能是好，Token 烧太快。

让 Codex / Fable 反复分析性能——多开 workers、预热，方案头头是道，改了就是不明显。

我自己分析数据包，根因是**让模型输出的 JSON 太长** 。翻转取舍：模型输出简单纯文本，复杂解析交给程序。

AI 能在框架内优化到极致；跳出框架、质疑约束本身的，是人。

翻转取舍之后

33 → 12

模型调用次数

31 → 18

单集处理分钟数

42 分钟

7 小时访谈完整润色

成本优化不是上线后的运维，是产品设计的一部分。

边界三 · 价值

该交给 AI 吗

有些环节交出去，产品就失去了灵魂。

我自己在写作上栽过这个跟头——全套 AI 写作试过，最后全部收回来人肉写。

到底什么该交、什么不该交？有一个判断标准，是我今天最想让大家带走的一句话——下半场揭晓。

三条边界都在动

能力边界	随模型升级外扩
成本边界	随价格下降外扩
价值边界	由你自己划——标准下半场给

方案要给边界的移动留出空间。

三

重设计：不要让 AI 模仿人

用一个做废了一半的产品来讲

它差点进我的「半成品 坟场」

Whisper 时间戳对不准。

视频编辑器工作量巨大。

业余时间老被打断。

卡死在：人工校 对绕不开界面，界面又做不 动。

破局：原型先行

高保真原型

不写代码，分钟级做出可交互原型，马上发现问题马上改。

MVP 决策

只做 Mac，只做最简单的播放和字幕。

用完全不熟的 Swift

AI 实施、我指挥验收。正因为不熟，才能真正放手。

最小验证不只让好想法长得快，更让坏想法死得早。

V1 · 2023—2024

模拟人类字幕组

听写 Whisper → 翻译 LLM → 人工校对 → 时间轴 → 压制

LLM 自作主张合并字幕、时间戳乱、人工校对依然繁重。

V1.5 用工作流编排和算法打补丁，有效，但——

在旧框架里打补丁，收益递减。

V2 · 推倒重来

以终为始的三连追问

为什么对不齐？

语序不同，srt 级对不上但句子级可以——为什么非要先拆 srt？因为在模仿字幕组。

为什么翻译质量差？

分页割裂上下文——先整体分析、提术语表，程序注入全局信息。

为什么这么多人工？

每步给 Agent 验证工具，Agent 自己验收，人只做最后兜底。

V1 与 V2 的对照

V1 · 模仿人	V2 · 围绕 Agent 重设
人操作	**Agent 操作**
凭经验验证	**给 Agent 工具去验证**
模仿人类工作流	**围绕 Agent 重设计**
图形界面	**cli 和文本**
数据库	**HTML / JSON**

三条设计准则

1 让 Agent 操作它最擅长的格式

选型先问：这个格式模型见得多吗。

2 给设计规范，不给模板

通用方法终将战胜专家捷径。模型每升级，产品不改一行代码自动变强——这就是「搭架子等模型」。

3 把 App 做成 Agent 的插件，不在 App 里内置 Agent

依据不是趋势报告，是两个已经发生在我自己身上的变化。

变化一 · 用户习惯

打开的不再是 App，是 Agent

买一台吸树叶的机器：我没有逛电商，先让 Agent 帮我找。

吹叶和吸叶模式的参数根本是两回事、裸机不含电池的真实性价比、电池生态锁定——分析深度远超我自己刷商品页。

当购买决策发生在 Agent 的对话里，你的商品详情页就不再是给人看的页面，而是 Agent 的数据源。

变化二 · 操作主体

人不操作了，Agent 操作，人只确认

我不再自己用 Google Slides 做 PPT，而是让 Agent 写——当然，大纲是我的。

BaoCut 是我自己写的 App，连我自己都不打开它的界面了：发一个链接给 Agent，它调用 cli 转录翻译，我只看结果。

作者本人都绕过了自己的图形界面，这就是入口迁移最诚实的证据。

〔实拍图位〕左：Agent 跑 BaoCut 工作流；右：网页版嵌在 Agent 环境里供验收

入口迁移的第三次

第一次

门户 → 搜索框

谁不在搜索结果里，谁消失。

第二次

PC → 手机 App

没有 App 的产品淡出视野。

第三次 · 现在

App → Agent

用户不再「打开你」，而是让 Agent「调用你」。

你不在 Agent 够得着的地方，你就不存在。

我踩过的坑，Figma 给出了同一个答案

帮朋友做过从 Figma 稿生成 UI 代码的 Agent，功能完整——没人用：开发者的场景在本地编辑器里，工作流是断的。

改成 Skill，让开发者自己的 Coding Agent 调用，问题消失。

一年后 Figma 官方做了同一个选择：不在产品里内置写代码的 Agent，而是开放官方服务，让 Cursor、Claude Code 这些用户自己的 Agent 直接读设计稿、连设计系统。

搜索时代拼 SEO，App 时代拼买量——

Agent 时代，做好被调用就是新的 SEO

你的产品，是抢着当入口，还是做好被调用？

四

落地验证：从一条 GitHub Issue 到功能上线

用户想把转录丢给家里有显卡的电脑跑。三问过完：值得做。但我没有打开编辑器。

五关

01	**可行性**	Agent 给出四个方案，我扫一遍拍板。模式：Agent 分析罗列，人判断拍板。
02	**设计文档**	AI 原生开发里文档是一等公民：给人确认修改，给 Agent Session 之间传上下文。
03	**高精度原型**	需求、原型、UI 三合一。原型阶段改布局是一句话的事，开发完再改成本指数级上升。
04	**实现**	文档 + 原型交给 Agent，按里程碑推进，自己跑测试截图验证。人主要在等。
05	**测试**	把自己当普通用户，凭直觉乱用，问题扔回给 Agent。

一个诚实的反例

「本地模型辅助拆分字幕」，可行性分析也做了，做完体验很糟，砍掉——浪费好几天和一堆 Token。

可行性分析是期望 值的赌注：不保 证不看走眼，但没有它，坑只会更多。

「那你 review 代码了吗？」——这个问题先按下，下半 场有专门的交代。

到底什么变了

流程没变，角色变了

五关一个没少；执行主体全是 Agent，人从一线程序员变成技术总监。

确认可以合并，不能省略

加速确认，不跳过确认——人的判断力是流程里不可替代的部分。

瓶颈转移到代码两侧

左边设计确认、右边测试部署还在人类速度运行。精力花在流程设计上，比花在 prompt 优化上回报大得多。

上半场落地地图

找需求

三问

判边界

三条线

重设计

三准则

落地五关

人在关键路径确认

持续进化

边界线动了，重新过一遍这张图

这张图不是方法论推导出来的，是一步步踩出来的——产品要进化，做产品的人呢？

02

下半场 · 我们每个人如何与 AI 共同进化

建立循环 → 共同进化 → 执行给 AI，思考留自己

五

建立快速反 馈循环

GPT-3.5 时代

翻译提示词的六步进化

01
设定角色

02
抽卡多翻几次

03
让 AI 一次出几版

04
意外发现第二版几乎总更好

05
正好学了 CoT，恍然大悟

06
先直译、再意译——广为流传

分享之后，读者建议加「反思」，演化成直译 - 反思 - 意译—— **一个人的迭代到第六步就停了，分享 让循环接上了别人的反馈。**

循环分层

周

写作 · 学习 · 分享

天

想法 · 代码 · 验证

分钟

原型 · 验证

把「翻译」换成你手里的代码、设计稿、周报、运营文案——每个人都有自己的那个提示词。

关键不是意志力，是循 环自己会转。

那个广为流传的提示词，今天已经没法用了。

废掉它的人，是我自己

o1 发布，模型自己会思考，提示词里的 CoT 和它打架，效果反而变差。改成一句话：用中文重写这篇文章。

一个提示词的三条命

人用的提示词 → 跟着推理模型改 → **Agent 用的技能**

和模型共同进化，第一步是敢于否定自己的旧答案。

给个人的产品三问：这事 AI 现在能做吗？将来能做吗？多久？——每半年重问一次。

七

把执行交给 AI，把思考留给自己

我现在不 review AI 写的代码了

2000 年开始写代码，26 年——这句话说出口比听起来难得多。

中间发生了什么

抵触期	不信任，但立规矩刻意多用，避免偏见做出错误判断。
o1 之后	写码接近超过我，架构不行——我设计，AI 写，我 review 每一行。
Claude Code	真的理解项目了——还是 review。
Fable 5 之后	各方面碾压。两层验证够了：Agent 自验证 + 我黑盒测试。金融核心逻辑另说。

把自己变成 PM + EM

坐 IC 的位置

AI 是威胁。

坐管理者的位置

AI 是最强下属。

我讲的是代码，但把「代码」换成你手里的 PPT、报表、文案、设计稿，道理一模一样。

Token 比人力廉价，人的位置只能一直上移：上移到定　义要什么、验收好不好。

但写作，我一个字都不敢交 给 AI

试过全套 AI 写作，两个问题：AI 味读者看得出来；更严重的是，失去了借助写作思考的过程。

翻译总结

AI 做

素材整理

AI 辅助

草稿

人肉写

初稿之后

AI 提意见，我手改

上半场欠的那个判断标准，现在还上——

产出是「结果」，还是「你的思考」？

结果可以外包，思考不行。

不要外包自己的大脑。外包大脑的人，最终会被拿回大脑的人替代。

收开场之问 · AI 时代该学什么

三层能力，AI 改不掉，只改变它的杠杆

学习能力

× 循环：以前建不起来的跨界循环，现在转得起来。

解决问题的能力

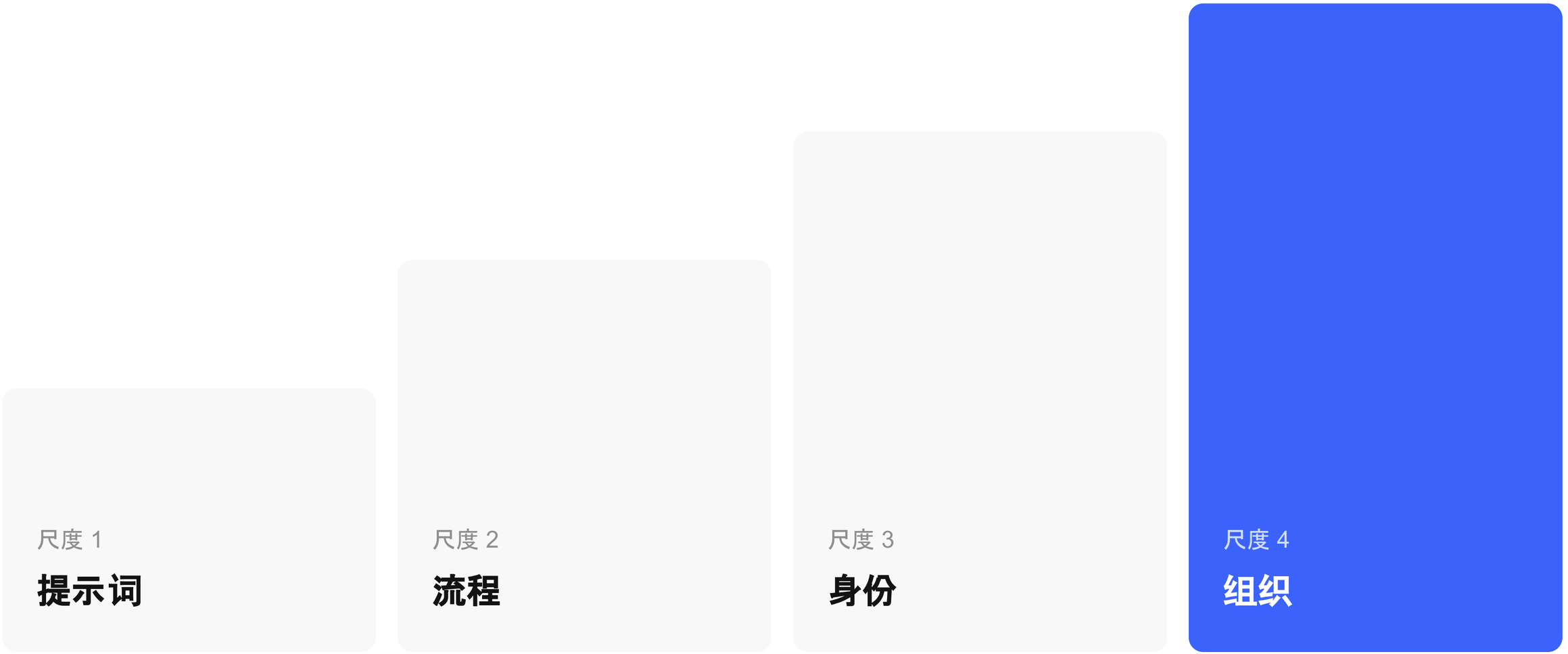
× 共同进化：和模型一起进化的人，先看到边界线扫出的新需求。

影响力

× 执行给 AI：管好一群 Agent，一个人就有一个团队的产出。

AI 替代的是执行，放大的是能力 ——站在哪边，你自己选。

推翻的阶梯



同一个动作，尺度越来越大——组织的旧权重，就是它的流程和架构图。

给管理者两分钟

组织的旧权重

现象

每个环节都有提效数字，整体交付没快多少——编码快了，评审排队反而更长。AI 塞进每个格子，摩擦还在格子之间。

提醒

每次交接同时是「传递工作 + 质量关卡」。该投的不是给每个岗位配 AI 助手，而是验证基础设施：自动化测试、预览环境、合规自动检查。

起步

一个能定义问题的人 + Agent + 验证工具，端到端试点；度量换成「想法到上线的时间」——你度量什么，组织就会长成什么样。

老话说知人善用，AI 时代是知 AI、善用 AI。

给你的产品：哪一环还在让 AI 模仿人？围绕 AI 重新设计，哪些环节会直接消失？

给你自己：AI 帮你省下来的时间和精力，你打算投到哪一层能力上？

谢谢 · Q&A