Vibe Coding vs AI Native Git

从 AI 的 保姆 到 AI 的 甲方

CNB

AI Team · 2026

你一定干过这些事

🤖 人肉 Copy-Paste 工程师

1. 打开 IDE，新开聊天窗口
2. `Ctrl+A` 全选代码 → `Ctrl+C`
3. 切到 AI → `Ctrl+V`
4. 等回复……发现少了上下文
5. 再切回去复制另一个文件
6. AI 说：*"请提供更多上下文"*
7. 心态崩了 💥

🖥️ 禁止关机大字报

- 某公司工位上贴着 A4 纸打印的四个大字："禁止关机"
- 不是怕丢数据
- 是因为 IDE 关了 AI 就断线了
- 昨晚让它写的代码还没生成完
- 行政小姐姐差点全公司发文 📢

荒诞吗？荒诞。但更荒诞的是——
我们都觉得这挺正常。

Vibe Coding 是什么

概念

> 用自然语言"感受式"地描述需求，
> 让 AI 生成代码的开发方式

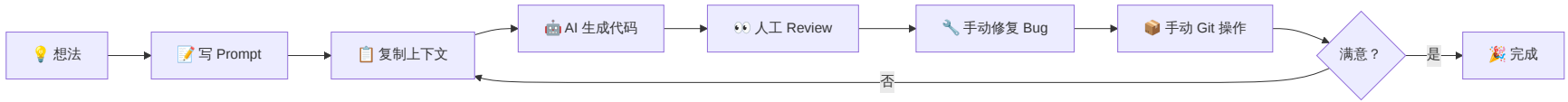
由 Andrej Karpathy 于 2025 年 2 月提出：

> *"I just write prompts and review code now"*

核心特征

- 🗣️ 自然语言驱动 — 不写代码，写需求描述
- 🔧 AI 直接生成 — AI 输出完整代码
- 🔄 快速迭代 — 对话式修改，多轮调试
- 💡 降低门槛 — 非专业开发者也能"编程"
- ⚡ 原型极速 — 想法到 Demo 的时间大幅缩短

Vibe Coding 的典型工作流



每一次循环，人类都是 **翻译官 + 搬运工 + 质检员** 的三合一角色

Vibe Coding 的四大困境

🧠 AI 没有记忆

- 每次新对话，AI 都是"失忆"状态
- 不知道项目结构、技术栈、编码规范
- 昨天的架构决策，今天全忘了
- 你从"人肉翻译官"进化成了"人肉上下文投喂员"

🖥️ 强依赖 IDE

- AI 能力绑定在 IDE 内
- 关掉编辑器，AI 就断线
- 无法自动化，无法无人值守
- 想让 AI 晚上加班？贴"禁止关机"吧 😄

🔄 无法形成闭环

- AI 只能生成代码片段
- Git 操作还得人来
- CI 失败了还得人来修
- PR Review 后的修改还得人来跟进
- AI 是个"半自动"助手

📏 质量难以保障

- 没有项目级的代码规范约束
- 没有自动化的测试验证
- 多轮对话容易产生代码漂移
- "改了这里，坏了那里" 是常态

全行业都在找记忆

超长上下文窗口 — 100K、200K、甚至 1M token

- 塞是塞进去了，AI 该忘还是忘
- 而且账单比你工资条还长 💸

RAG 检索增强生成 — 代码库切块、建索引、向量化

- 本质是给 AI 装了个"外挂硬盘"
- 检索出来的"相关代码"经常是过时的、废弃的
- 跟 Git 里代码的真实语义关系八竿子打不着

各种记忆框架 — 短期 / 长期 / 工作 / 情景记忆

- 分类学整得挺明白
- 本质就是给 AI 的失忆症贴各种创可贴
- 换一个项目，记忆又得从头来过

蓦然回首

代码在哪？在 Git 仓库 里。

每一次 commit → 工作记忆

谁改了什么、为什么改

每一个 Issue → 需求记忆

要做啥、为什么做、谁提的

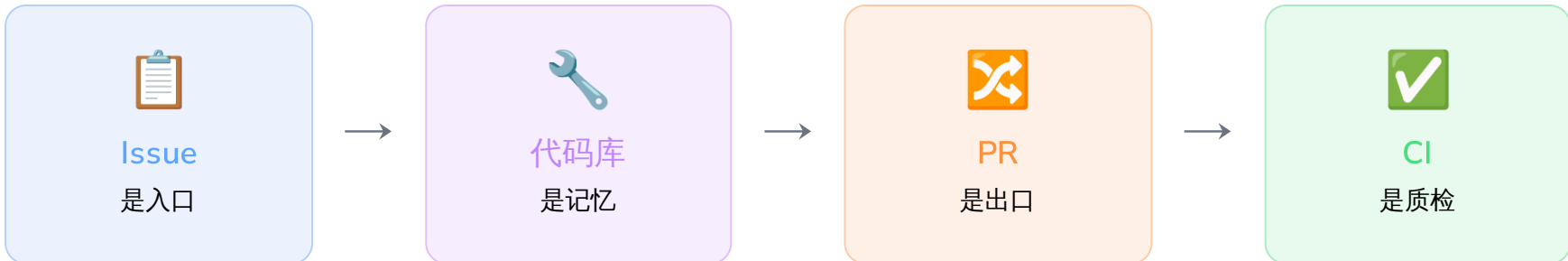
每一个 PR → 决策记忆

怎么做的、谁 Review 的、改了几版

每一条 CI 日志 → 质量记忆

跑没跑通、哪些测试挂了

AI Native Git 核心思想



AI 应该在这个闭环里原生运转，
而不是在外面探头探脑。

NPC 是什么

NPC = Native Programmable Collaborator

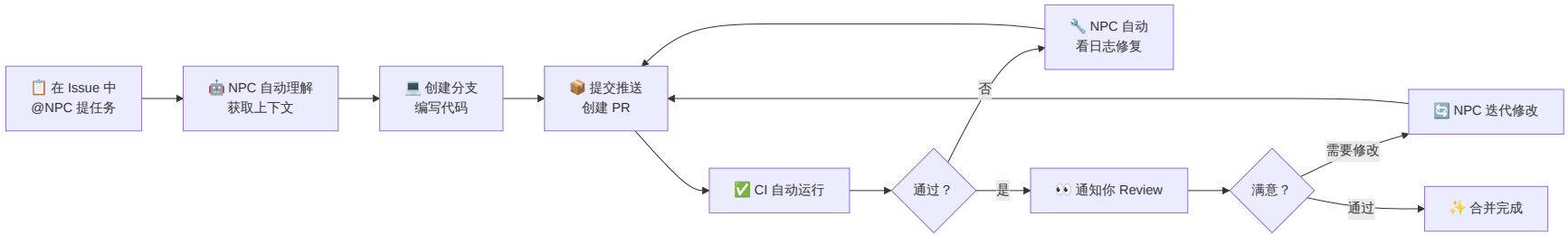
原生可编程协作者

- 🏠 住在仓库里 — 不是 IDE 插件，不是聊天框
- 🧠 天然拥有记忆 — Git 仓库就是它的记忆
- 🤝 以"仓库成员"身份协作 — 被 @ 就干活
- 🔁 全流程闭环 — 从 Issue 到 PR 到 CI 到合并
- 🎯 7×24 待命 — 不需要你打开 IDE

NPC vs 传统 AI 助手

	传统 AI 助手	NPC
位置	IDE 内	仓库内
记忆	每次清空	Git 永久记忆
操作	只能写代码	写代码 + Git 操作
CI	不参与	自动修复
协作	单人工具	团队协作者
自定义	有限	完全可编程

NPC 工作流演示



你从头到尾只需要说一句话。其余的，NPC 自己搞定。

核心对比：Vibe Coding vs AI Native Git

维度	🎸 Vibe Coding	🚀 AI Native Git (NPC)
开发者的角色	AI 的保姆 — 投喂上下文、盯进度、打补丁	AI 的甲方 — 提需求、审结果
AI 的记忆	每次对话清零，需要人肉投喂	Git 仓库天然记忆，永久积累
上下文获取	手动 Ctrl+C / Ctrl+V	自动读取仓库上下文
Git 操作	人工执行 add/commit/push/PR	NPC 自动完成全流程
CI 集成	失败后人工修复	NPC 自动诊断并修复
协作模式	个人在 IDE 内对话	以 Issue/PR 为中心的团队协作
AI 的工作时间	跟你一起 — 你关 IDE 它下线	7×24 — 住在仓库里随时待命

角色转变：从保姆到甲方

Vibe Coding — AI 的保姆

- 手动打开 IDE
- 手动写 Prompt
- 手动复制粘贴上下文
- 手动审查生成代码
- 手动修复 Bug
- 手动执行 Git 操作
- 手动处理 CI 失败
- 手动跟进 Review 意见
- 全程盯工 💀

AI Native Git — AI 的甲方

- 在 Issue 中 @NPC 提需求
- NPC 自动获取上下文
- NPC 自动编写代码
- NPC 自动提交 PR
- NPC 自动修复 CI
- NPC 自动迭代修改
- 你只负责 Review 和拍板
- 解放出来的时间做架构设计 ✨

NPC 生态：可编程的 AI 协作者

🛠️ 可编程

在 `.cnb/settings.yml` 中定义 NPC 角色：

- 名字、口头禅
- 行为风格
- 运行时环境
- 专属 Docker 镜像

📦 可分享

你定义的 NPC：

- 别人关注你的仓库就能用
- 就像分享开源库一样自然
- 一个 NPC，全团队受益

🧩 可组合

多个 NPC 协作：

- 猎头 NPC 找人
- Slidev NPC 写 PPT
- 代码 NPC 写代码
- 各司其职，流水线作业

这不是一个助手，这是一个可编程的 AI 协作者生态。

NPC 带来的效率提升

环节	传统方式（Vibe Coding）	NPC 方式	效率提升
上下文准备	10-30 分钟手动整理	0 分钟 — 自动读取	✅ 100%
代码实现	多轮对话反复调试	NPC 一次性生成并迭代	✅ 3-5x
Git 操作	人工 add/commit/push/PR	全自动	✅ 100%
CI 修复	人工查日志、改代码	NPC 自动诊断修复	✅ 5-10x
Review 修改	人工理解意见并修改	NPC 自动迭代	✅ 3-5x
重复性任务	每次从头来	NPC 记忆积累，越用越熟	✅ 持续提升

研发团队从 80% 时间花在"伺候 AI"上，
变成 80% 时间花在"创造性工作"上。

换赛道

开发效率的进化路径

不是让 AI 帮你敲 git 命令，

是让你再也不需要敲 git 命令。

不是给 AI 加记忆，

是让 AI 住进记忆里。

总结

1. Vibe Coding 是第一步 — 让 AI 帮我们写代码，但人类仍是保姆
2. 瓶颈在记忆和闭环 — AI 失忆、依赖 IDE、无法自动化全流程
3. Git 仓库 = 最佳 AI 外部记忆 — 蓦然回首，记忆就在身后
4. AI Native Git = AI 住进仓库 — Issue 是入口，PR 是出口，CI 是质检，代码库是记忆
5. NPC = 原生可编程协作者 — 不是外挂工具，是仓库的原住民
6. 角色转变 — 研发团队从 AI 的保姆，变成 AI 的甲方

AI Native Git，启动。🚀

Q & A

感谢聆听

了解更多：https://cnb.cool