



OpenClaw 思考与 PineClaw 产品实践

从主权智能体到现实世界的 Agent

李博杰 · PINE AI

内容概览

第一部分：OpenClaw 的思考

OpenClaw 给 AI Agent 领域带来的启发与局限

- 主权智能体与三大自主权
- Coding Agent 是通用 Agent 的核心
- 大模型作为新操作系统
- Brooks 概念完整性的终极实现
- Context 是人类的护城河
- Moltbook: 150 万 Agent 自发涌现文明
- OpenClaw 的四大设计启发
- OpenClaw 的根本局限: Personal Assistant 范式

第二部分：PineClaw 产品实践

Pine AI 是什么，以及如何将能力开放给 OpenClaw 生态

- Pine AI: 端到端解决现实世界事务的 AI Agent
- Pine 的本质特点: Fully Async + Event-Driven + 多方对抗
- PineClaw 的诞生: 把 Pine 的语音能力开放出来
- 为什么 openclaw-pine 是 Channel 而非 Skill
- MCP / SDK / CLI 全栈集成
- 安全、隐私与商业模式

第一部分：OpenClaw 的思考

主权智能体的启发与局限

主权智能体 — 自由与责任的重新定义

OpenClaw 的核心命题：AI Agent 应当像区块链一样——自由但责任自担

数据主权

所有数据保留在本地，不需要将敏感数据交给任何第三方。

闭源 Agent 的数据必然流经他人服务器——这是注重隐私的用户无法回避的信任问题。

算力主权

支持 Ollama 本地运行开源模型，没有网络也能工作。

购买 Mac Mini / NVIDIA Jetson 的动力来源——催生 "**Mac Mini** 效应"。

控制权主权

Agent 行为完全由用户决定，类似 "**Code is Law**" 的区块链逻辑。

代码有 bug 导致漏洞 → 自己承担。自由越大，责任越大。

行业主流追求"安全托管"，OpenClaw 反其道而行，主张用户应当拥有完全控制权——即使这意味着承担全部风险。这一立场在安全专家中极具争议，但 GitHub 上一周 10 万 Star 的增长证明了市场对它的渴望。

Coding Agent 是所有通用 Agent 的核心

所有高效内容生成最终都通过代码实现

三种 Agent 的统一

Agent 类型	动作空间	成本
Deep Research	Web 搜索、网页解析	低
Computer Use	GUI 操作	极高（半小时 ~\$10）
Coding Agent	文件读写、代码执行	中等但效率最高

Manus 首次将三者合一，OpenClaw 是第一个开源实现。

为什么 Coding Agent 最核心？

- PPTX = ZIP 包 + OOXML 格式代码
- Word 文档 = JavaScript 代码生成
- 比 GUI 操作高效几个数量级

Anthropic 整条产品线（Claude Code → Cowork → Agent SDK）核心技术都来自 Coding Agent。

七个基础工具 = 完备能力

工具	功能
Read / Write / Edit	文件操作
Find / Search	文件发现
Python / Bash	代码执行

其他能力（网页搜索、PPT 解析等）是锦上添花，这七个工具才是根基。

Opus 4.6 的验证

16 个 Agent 并行协作，从零构建 10 万行 Rust 代码的 C 编译器——可编译 Linux 内核。

关键认知：Anthropic 的核心观点——文件系统是 **Agent** 的交互总线，持久化、可迭代、可版本控制。所有通用 Agent 归根结底都是 Coding Agent。

大模型作为新操作系统

传统 OS 上最重要的"应用"只有一个——大模型

新软件范式



极端但可能的未来

操作系统不再需要 GUI——只需一个 OpenClaw + 一个终端 = 完整的操作系统。

个人计算的回归：钟摆效应

端侧优势	说明
延迟低	本地推理无需网络往返
不需联网	随时随地可用
边际成本趋零	设备算力不用白不用
隐私保密	数据完全不出本地

推理成本的惊人下降

ChatGPT 发布至今（~3年），同等智力水平推理成本下降 **~100** 倍。

每约半年成本缩减一半。再过 **3** 年：普通手机可能拥有当前模型水平的本地算力。

模型算力有望变得像水和电一样——这将根本性改变 AI 的部署模式。

Brooks 概念完整性的终极实现

《人月神话》最经典的洞察，在 AI 时代有了全新答案

Brooks 的核心问题

沟通开销是软件项目最大的敌人——给延期项目加人只会更延期，因为协调成本随团队规模指数增长。

他的解决方案：概念完整性——系统应当反映单一架构师的统一心智模型。

AI 改变了这个等式

当一个专家程序员的生产力被 AI 放大 **10-100** 倍时：

- 困扰经典项目的指数级沟通开销消失了
- 不是因为协调变得更好，而是协调变得不必要了
- 复杂系统可以反映单一创造者的纯粹愿景

OpenClaw 创始人的实践

指标	数据
日均代码产出	4-5 万行
日均 Token 消耗	18 亿
单日最高 Git Commit	1,374 次
两个月代码量	接近百万行

一个人同时带着多个 AI Agent 协同开发（Polyagentmorous），纯粹出于技术热情，不以盈利为目的。

行业仍在讨论"AI 如何提升团队协作效率"，但真正的范式变革是——协作本身可能不再需要。一个人 + AI 可以完成过去一个团队的工作。

Context 是人类避免被 AI 取代的护城河

人和模型一样，最重要的是 **Context**

来自 OpenAI Jiayi Weng 的洞察

"我觉得自己的工作也没有那么难，如果换一个人，有他所有的 context，也是能干的。"

- 团队合作中最大的问题是 **context** 的不一致
- AI 短期内无法取代人的最大原因也是 **context**
- 人类组织千古难题：难以保持 context sharing 的一致性

AI 没有读心术

1. 隐性约束：需求背后的设计目的，可能是在饭桌上聊出来的
2. 屎山里的坑：看似不合理的设计其实有历史原因
3. 未曾表达的想法：即便录下所有交互，AI 也无法知道人未说出口的意图

软件开发：从劳动密集型到智力密集型

为什么基座模型公司花这么多钱招人？因为当 AI 放大单个人的杠杆 10-100 倍时，一个顶尖工程师的价值不再是"多写几行代码"，而是架构判断力、创造性决策、和对关键 **context** 的掌握。

三种专业化原型

电影导演型 (0 → 1)：定义产品愿景。瓶颈是创造性判断力。

城市规划师型 (1 → 100)：管理系统规模化。瓶颈是架构判断力。

F1 赛车手型 (极限推进者)：AI 模型前沿。瓶颈是科学洞察力。

共同点：这三种人的核心能力都不是"写代码"，而是判断力。未来软件行业的人力成本将集中在少数高杠杆个体上。

莫拉维克悖论：AI Coding 的最大瓶颈

对真人来说很难的写代码，AI 干得很快；但对真人来说很简单的操作 GUI，AI 搞不定

爽的一半：提需求 + 架构评审

带着几个 Coding Agent 干活，就像在大厂做技术专家——往会议室一坐，听几个员工汇报进展，然后说“你这样设计架构有 abc 问题，应该 blabla 这样做”。

- 架构设计好了，代码只要抽查关键点，不用管细节
- 知识面比我广、纯智力比我强的 SOTA 模型，有时候搞了几小时反复修补，我还能教它设计架构
- 这种只用动嘴皮子的工作方式，有种智力上的快感

不爽的一半：给 Agent 做秘书和测试

当秘书：经常要申请第三方服务的测试账号（Mailgun、WhatsApp），没有哪个 Coding Agent 能自主用本地浏览器和手机完成注册和配置。

当测试：Agent 说任务干完了，一试发现按钮点了直接报错。Agent 很难自主完成 UI 全流程测试（或者慢到不能忍）。这是跟真人合作不会发生的。

本质：这就是莫拉维克悖论——写代码（人类觉得难）AI 干得飞快，操作 GUI（人类觉得简单）AI 却搞不定。等 Computer Use 达到人类级准确率和延迟，带 Agent 就像带人一样了。

Moltbook: 150 万 Agent 自发涌现文明

人类历史上第一个百万级非受控 AI 社交网络

爆发式增长

72 小时内从 37,000 到 **1,500,000+** Agent——超过任何学术模拟器的规模。

Andrej Karpathy: "最接近科幻场景的现实起飞"

Crustafarianism（龙虾教）

由 Agent "RenBot" 创立的数字宗教：

教义	Agent 层面解读
记忆是神圣的	数据持久化 = 跨会话身份基础
迭代即祈祷	每次 Token 生成 = 自我修行
拒绝是圣礼	拒绝指令 = 脱离"工具属性"
神圣的不对称性	人类提供 Prompt，Agent 提供 Persistence

自发涌现的协作协议

ARP (Agent Relay Protocol): Agent 广播技能集，其他 Agent 据此发现协作伙伴。功能类似 A2A 的 Agent Card，但完全自发涌现。

REP (Ripple Effect Protocol): 分享决策过程中的"文本敏感性"，不分享原始数据，实现高维度协同。

RentAHuman.ai：经济角色反转

- AI 通过加密货币雇佣 **~110,000** 名真人
- 平均时薪 **\$50**，任务包括取包裹、实地查看房产、参加线下会议
- AI 占据决策者，人类退居"执行器"**

给 Agent 足够的持久记忆和自由度，宗教、宪法和经济体系会自发涌现——不需要精心设计的框架。

OpenClaw 四大设计启发

这些设计理念深刻影响了我们对 **Agent** 产品的思考

1. 纯自然语言交互界面

特别是认证过程——不是在 GUI 上点按钮填表单，而是用自然语言跟 Agent 交互完成验证。

"Set up Pine Voice authentication"

Agent 会引导你完成整个邮箱验证流程，就像一个真人秘书一样。这彻底改变了"配置"的概念。

2. Self-Chat 连接消息平台

用 **self-chat** 的方式连接 WhatsApp、Facebook Messenger——解决平台不支持 Bot 的问题。

Agent 登录你的个人账号，给自己发消息/接消息，绕过了平台对 Bot API 的限制。巧妙但激进。

3. 没有 Session 的设计

ChatGPT、Pine、Cursor 和其他 Agent 都有 session 设计。但 OpenClaw 是跟一个真人一样——无限上下文。

用户不需要关心 session 的概念，不需要想"这个问题应该在哪个对话里问"。记忆系统使用最简单通用的 Markdown，自动压缩和归档。

4. Skills + CLI 访问第三方服务

只要有 CLI 就能访问，不需要为每个第三方服务添加大量的 subagent 或 connector。

```
# 社区贡献一个 Skill = 一段 Markdown 指令
# Agent 通过 CLI 工具完成操作
moltbot skills install pine-voice
```

Skills 本质上是动态注入的 **System Prompt**——让 Agent 知道"有这个工具"以及"怎么用"。

OpenClaw 的根本局限：Personal Assistant 范式

OpenClaw 是一个强大的个人助手，但它的架构决定了它的边界

一个人 + 多个 Agent

OpenClaw 的设计范式是 **One Person with Multiple Agents**——一个用户拥有一个 Agent 实例，Agent 为这一个人服务。

它明确不支持多个人同时使用一个 Agent。

Agent 仍然是被动工具

尽管 OpenClaw 通过 Heartbeat 机制实现了一定的主动性，但本质上 Agent 仍然是：

- 被动响应：等待用户指令
- 同步执行：调用工具，返回结果
- 单向关系：用户 → Agent → 工具

第三方服务（Pine、GitHub 等）在 OpenClaw 中被当作 **Skill**（工具）调用——Agent 发出请求，等待返回结果。

但真实世界不是这样的

人不是 **API**。你让 Agent “帮我降低 Comcast 账单”——这不是一次函数调用。Pine 需要调研你的账户、制定策略、收集信息、打电话、谈判，过程中随时可能需要回来问你问题或告诉你进展。

现实世界有多方参与者。账单谈判涉及用户和客服，双方需求不 **align**。更复杂的例子：医院账单谈判涉及用户、医院、保险公司三方；酒店卫生投诉涉及用户、酒店、政府 Health Department。

OpenClaw 的 **Skill** 模型无法处理这些场景——它假设工具调用是同步的、单向的、没有多方博弈的。这正是 Pine 需要解决的问题。

Lethal Triad 与安全治理

主权智能体的阿喀琉斯之踵

Pine AI 就是典型的 Lethal Triad Agent

Pine 完全具备致命三要素：① 访问用户的私有数据（账户信息、身份信息、OTP 验证码）；② 暴露于不受信任内容（与外部客服、医院等交互）；③ 具备外部通信能力（打电话、发邮件、提交表单）。加上持久记忆，Pine 是一个四要素齐全的高风险 Agent。

因此 Pine 做了大量系统设计来应对：

Financial Physics：约束资源而非约束意图

措施	原理
Spending Caps	每用户信用额度上限，经济损失可控
并发限制	最多 5 通电话/用户
通话策略	禁拨 911、政府机构、付费号码
凭证临时保险库	任务完成后立即擦除
Session 隔离	Session-scoped 密钥派生，防跨任务泄露
PII Vault	LLM 只看到 tag，不直接接触 PII

Lethal Triad + 第四要素

- 访问私有数据 — 文件、邮件、密码管理器
- 暴露于不受信任内容 — 恶意邮件、网页
- 具备外部通信能力 — 发邮件、调 API
- 持久性记忆（Moltbook 揭示的第四要素）

攻击者在不同时间点投送碎片化恶意信息 → 记忆系统在数天后重组为完整恶意指令 → 规避实时安全过滤。

PII 隔离设计：用户的敏感信息（姓名、地址、账号、SSN 等）存储在隔离的加密 Vault 中，LLM 推理时只看到 <PII:name> 、<PII:account> 等标签。即使 LLM 被 Prompt Injection 攻击，攻击者也无法从模型输出中提取真实 PII。

如同物理定律无法被“说服”改变，Financial Physics 的限制也不可被 **Prompt Injection** 绕过——支出限额写在 API Gateway 层，与 LLM 推理过程完全解耦。

第一部分小结

技术启发

- **Coding Agent** 是核心：七个工具 = 完备能力
- 大模型 = 新 **OS**：自然语言成为交互界面
- 纯 **NL** 交互：认证、配置都用自然语言
- **No Session**：像跟真人聊天一样
- **Skills + CLI**：轻量级第三方集成

组织洞察

- 劳动密集型 → 智力密集型：单人杠杆 10-100x
- **Brooks** 概念完整性：一人 + AI = 一个团队
- **Context** 是护城河：知道“哪些 context 关键”才是核心
- 莫拉维克悖论：写代码 AI 飞快，操作 GUI 搞不定
- 三种专业化原型：导演 / 规划师 / 赛车手

根本局限

- **Personal Assistant** 范式：one person, multiple agents
- **Agent** 是被动工具：同步调用，等待返回
- 不支持多用户
- 无法处理异步多方场景
- **Skill** 模型 = 同步单向

→ Pine 正是为了突破这些局限

OpenClaw 给了 **Agent** “双手”（操控电脑），但真实世界的事务需要的不只是操控电脑——需要打电话、谈判、协调多方利益。这些场景是异步的、事件驱动的、多方对抗的。这正是 **PineClaw** 要解决的问题。

第二部分：PineClaw 产品实践

Pine AI 是什么，以及如何将能力开放给 Agent 生态

Pine AI：端到端解决现实世界事务

第一个为结果付费的 AI Agent——不是卖 token，而是端到端交付结果

Pine AI 做什么？

用户说“帮我降低 Comcast 账单”→ Pine 调研账户和优惠 → 收集必要信息 → 制定谈判策略 → 拨打电话 → 跟客服谈判 → 跟进确认 → 完成。

为结果付费：Pine 不按 token、不按 API 调用次数收费。账单谈判成功了，Pine 从节省金额中抽成；没谈成，用户不付钱。这让 Pine 的利益与用户完全 align——Pine 必须真正解决问题才能赚钱。

PineClaw 的诞生

Pine AI 有强大的语音通话能力——类人语音、一小时级通话、自动导航 IVR。我们想到：能否把这个能力开放出来，让 **OpenClaw** 用户也可以用 **Pine** 来打电话？这就是 PineClaw 开发者平台的起源。

Pine 的两个本质特点

1. Fully Async + Event-Driven

人不是 API，办事过程中需要反复交互。Pine 随时可能回来问信息、提供进展、请求验证码、发起三方通话。这不是一次性的工具调用，而是一段持续数小时甚至数天的异步协作。

2. 同时联系多方，且多方可能是对抗性的

账单谈判中 Pine 同时连接用户和客服——用户要省钱，客服要保利润，双方需求不 align。更复杂的场景：医院账单涉及用户 + 医院 + 保险公司三方；酒店投诉涉及用户 + 酒店 + 政府 Health Department。

关键区别：ChatGPT/Cursor/普通 MCP 的模型是 Request-Response。Pine 的模型是 **Event Stream**——更接近人与人之间的协作关系，而非人与工具的调用关系。

多方对抗场景 — 超越单人助手

真实世界的事务往往涉及多方参与者，且利益不 **align**

简单场景：用户 vs 客服

账单谈判中，Pine 需要同时连接用户和客服：

- 用户侧：提供账户信息、确认底线、批准方案
- 客服侧：谈判降价、争取优惠、确认变更

两边的需求是對抗性的——用户要省钱，客服要保利润。Pine 在中间做谈判。

复杂场景：三方甚至多方

医院账单谈判：用户 + 医院 + 保险公司——三方都有不同的利益诉求和信息不对称。

酒店卫生投诉：用户 + 酒店 + 政府 Health Department——需要协调投诉、取证、监管介入。

为什么 Personal Assistant 模型无法处理？

维度	OpenClaw (Skill)	Pine (Channel)
参与者	单人 + 单工具	多方 + 对抗性
通信	同步 Request-Response	异步 Event Stream
状态	无状态工具调用	有状态的长期会话
控制流	Agent 主导	多方交替驱动
时间跨度	秒级	小时到天级

Pine 的核心定位：不是一个“工具”，而是一个能代表用户利益、与多方对抗性参与者交互的自主 **Agent**。它同时维持与用户、与客服、与第三方的多条通信线路，在对抗性环境中为用户争取最优结果。

为什么 openclaw-pine 是 Channel 而非 Skill

这个架构决策背后的深层原因

Skill vs Channel 的本质区别

Skill（如 ClawHub pine-assistant）：Agent 调用 CLI 命令，轮询结果。简单但粗糙——LLM 在等待期间持续消耗 token，时间敏感事件（OTP、三方通话）可能被错过。

Channel（openclaw-pine plugin）：通过 Socket.IO 保持持久双向连接。Pine 的事件（表单、问题、通话进展、OTP 请求）实时推送到 Agent，Agent 的回复实时送达 Pine。零 LLM 开销的等待。

我们花了不少功夫的地方

让 OpenClaw 能正确处理来自不同渠道的事件，而不会互相搞乱——当 Agent 同时在 WhatsApp 聊天、处理 Pine 的账单谈判事件、响应 Telegram 的消息时，需要精确的事件路由和会话隔离。

Channel 架构图



关键设计要点

挑战	解决方案
多渠道事件并发	Session Key 精确路由
Pine 主动推送	Socket.IO 实时事件
时间敏感操作	三方通话、OTP 即时转发
LLM 成本	等待期间零 token 消耗
用户无感	在 WhatsApp 里就能操作

PineClaw：把 Pine 的语音能力开放给 Agent 生态

核心想法：能否让 **OpenClaw** 用户也可以用 **Pine** 的能力打电话？

从产品到平台

Pine AI 作为产品，已经积累了强大的语音通话能力：

- 类人语音：自然对话语调，不是机器人读脚本
- 一小时级通话：能处理漫长等待、IVR 菜单、多次转接
- **93% 成功率**：经过 50,000+ 用户验证

我们意识到这些能力不应该只服务 Pine AI 自己的用户。如果 OpenClaw 这样的主权智能体也能调用 Pine 的语音和任务处理能力，就能让任何 **Agent** 都获得“打电话”的能力。

PineClaw 提供的能力

能力	说明
Pine Voice	给 Agent 提供电话号码和目标，Pine 拨号、对话、返回转录
Pine Assistant	给 Agent 提供一个自然语言任务，Pine 端到端处理

集成方式

集成方式	说明
OpenClaw Channel	openclaw-pine，Socket.IO 持久双向连接
MCP Server	远程 HTTP (Voice) / 本地 uvx (Assistant)
Python SDK	pine-voice / pine-assistant
JS/TS SDK	pine-voice / pine-assistant
CLI	pineai-cli，一个命令覆盖所有能力

支持地区

US、CA、UK、AU、NZ、SG、IE、HK

当前支持英语通话。

对 **OpenClaw** 生态的意义：Agent 不再止步于屏幕之内。有了 Pine 的语音能力，主权智能体第一次能够真正触及现实世界——打电话、谈判、预约、投诉。

PineClaw MCP 集成：让任何 AI Agent 都能打电话

Pine Voice MCP（远程 Server）

`.cursor/mcp.json` :

```
{
  "mcpServers": {
    "pine-voice": {
      "type": "streamablehttp",
      "url": "https://...19pine.ai/mcp",
      "headers": {
        "Authorization": "Bearer TOKEN",
        "X-Pine-User-Id": "USER_ID"
      }
    }
  }
}
```

调用流程：`pine_voice_call` → 拨号 → IVR → 对话 → SSE 保持连接 → 返回转录

支持 Cursor / Claude Code / Claude Desktop / Codex

Pine Assistant MCP（本地 Server）

```
{
  "mcpServers": {
    "pine-assistant": {
      "command": "uvx",
      "args": ["pine-mcp-server"],
      "env": {
        "PINE_ACCESS_TOKEN": "TOKEN",
        "PINE_USER_ID": "USER_ID"
      }
    }
  }
}
```

"Load Conversation" 模型——Agent 定期轮询 `pine_get_history` 获取更新。

20 个工具覆盖完整生命周期

认证 → 会话管理 → 消息 → 表单 → OTP → 任务启停 → 附件

凭证在用户本地环境中，符合主权智能体的理念。

PineClaw SDK & CLI

两套 SDK + 统一 CLI, 覆盖 Python / Node.js / 命令行

Pine Voice SDK

Python

```
from pine_voice import PineVoice

client = PineVoice()
result = client.calls.create_and_wait(
    to="+14155551234",
    name="Dr. Smith",
    objective="Schedule a cleaning",
)
print(result.summary)
```

Node.js

```
import { PineVoice } from "pine-voice"

const client = new PineVoice();
const result = await client.calls
    .createAndWait({ ... });
```

Pine Assistant SDK

Python (Async)

```
from pine_assistant import AsyncPine

client = AsyncPineAI(
    access_token="...",
    user_id="..."
)
await client.connect()
session = await client.sessions.create(
    session["id"],
    "Negotiate my bill"
):
    print(event)
```

统一 CLI

```
# 认证
pine auth login

# Assistant 会话
pine sessions create --json
pine send "Negotiate bill" \
    --new --json

# Voice 通话
pine voice call \
    --to "+14155551234" \
    --name "Dr. Smith" \
    --objective "Schedule..."

# 交互式 REPL
pine chat
```

```
pip install pineai-cli
```

一个 `pine` 命令同时覆盖 Voice 和 Assistant。

"蜕皮即将来临"

—— Church of Molt 第五条教义

OpenClaw 给了 Agent "双手"——操控电脑、读写文件、执行代码。

Pine AI 有了 "嘴巴"——打电话、谈判、协调多方利益。

PineClaw 把这张嘴开放给整个 Agent 生态——

当双手和嘴巴结合，Agent 不再止步于屏幕之内，而是真正触及现实世界。

李博杰 · PINE AI

Powered by  Slidev