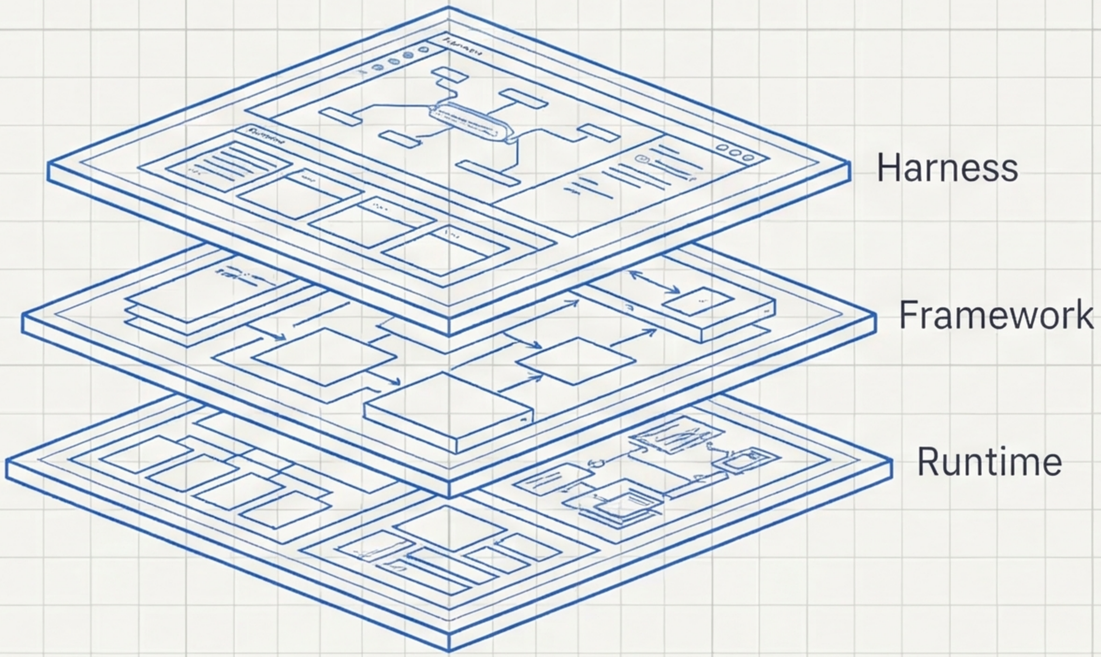
Deep Agents 实战

第 1 讲：Agent 开发的三个层次



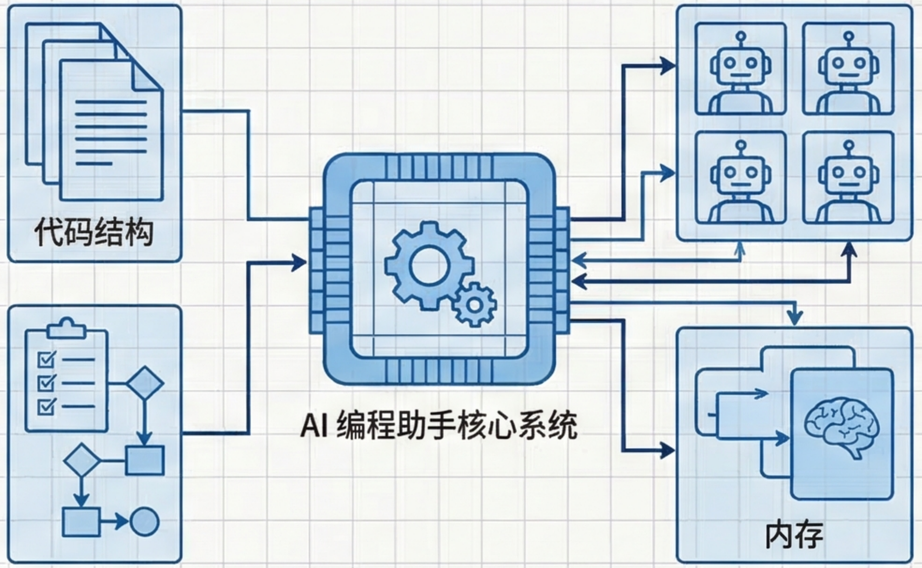
构建 AI 编程助手的困境

你需要的能力远比你想的多

困境（The Dilemma）

- 读取源代码文件、理解代码结构
- 制定修改计划、分步执行、追踪进度
- 委派子任务给专门的 Agent
- 跨对话记住用户偏好
- 每个"认真"的 Agent 都需要这些 → **重复造轮子**

解决方案（The Solution）



统一的集成化系统：
能力模块化、可复用、无需重复建设

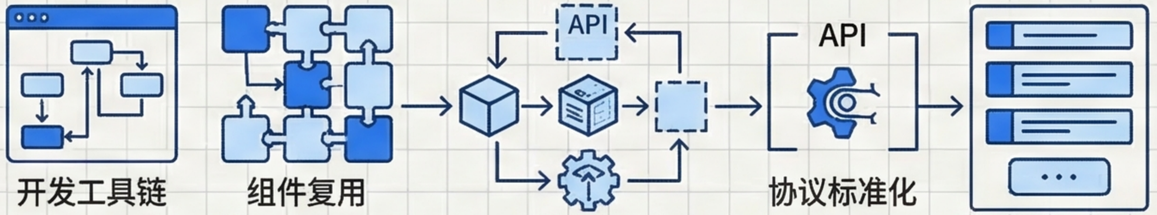
Agent 开发的三个层次

Runtime → Framework → Harness

上层 Harness：Agent 怎么开箱即用

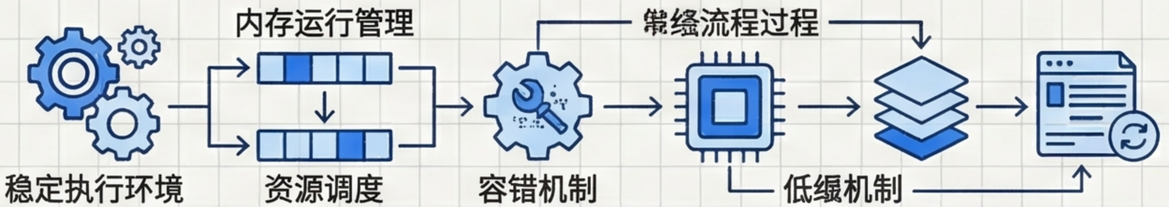
构建于

中间层 Framework：Agent 怎么快速开发



构建于

底层 Runtime：Agent 怎么可靠运行



底层：LangGraph（Runtime）

Agent 世界的"操作系统"

持久化执行

崩溃后从断点恢复

流式输出

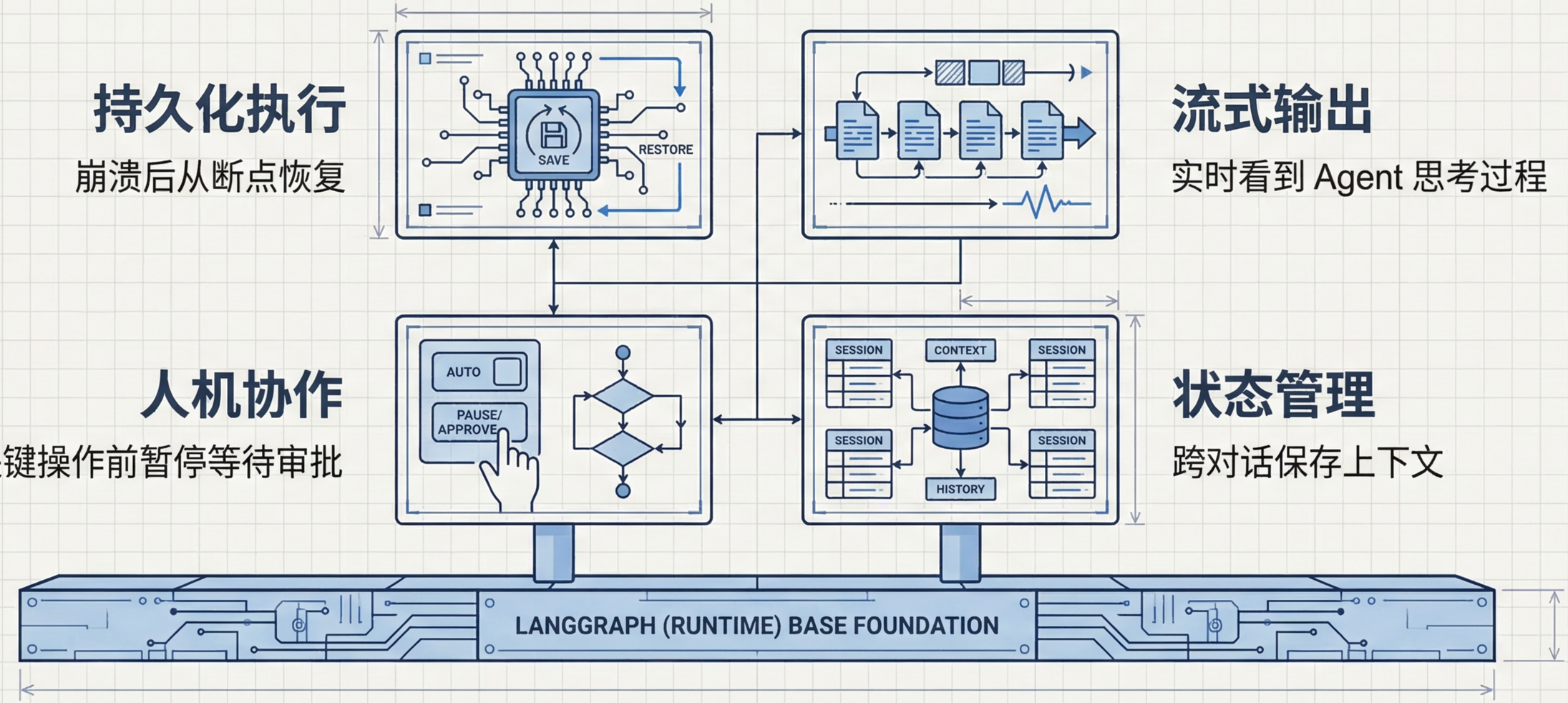
实时看到 Agent 思考过程

人机协作

关键操作前暂停等待审批

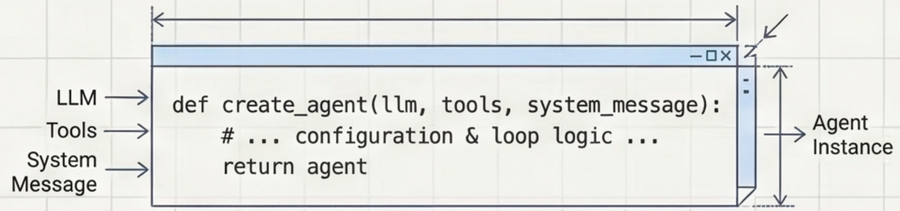
状态管理

跨对话保存上下文



中间层：LangChain（Framework）

构建在 LangGraph 之上的高层框架



create_agent() API

模型抽象：统一的 LLM 调用接口

工具接口：标准化的工具定义

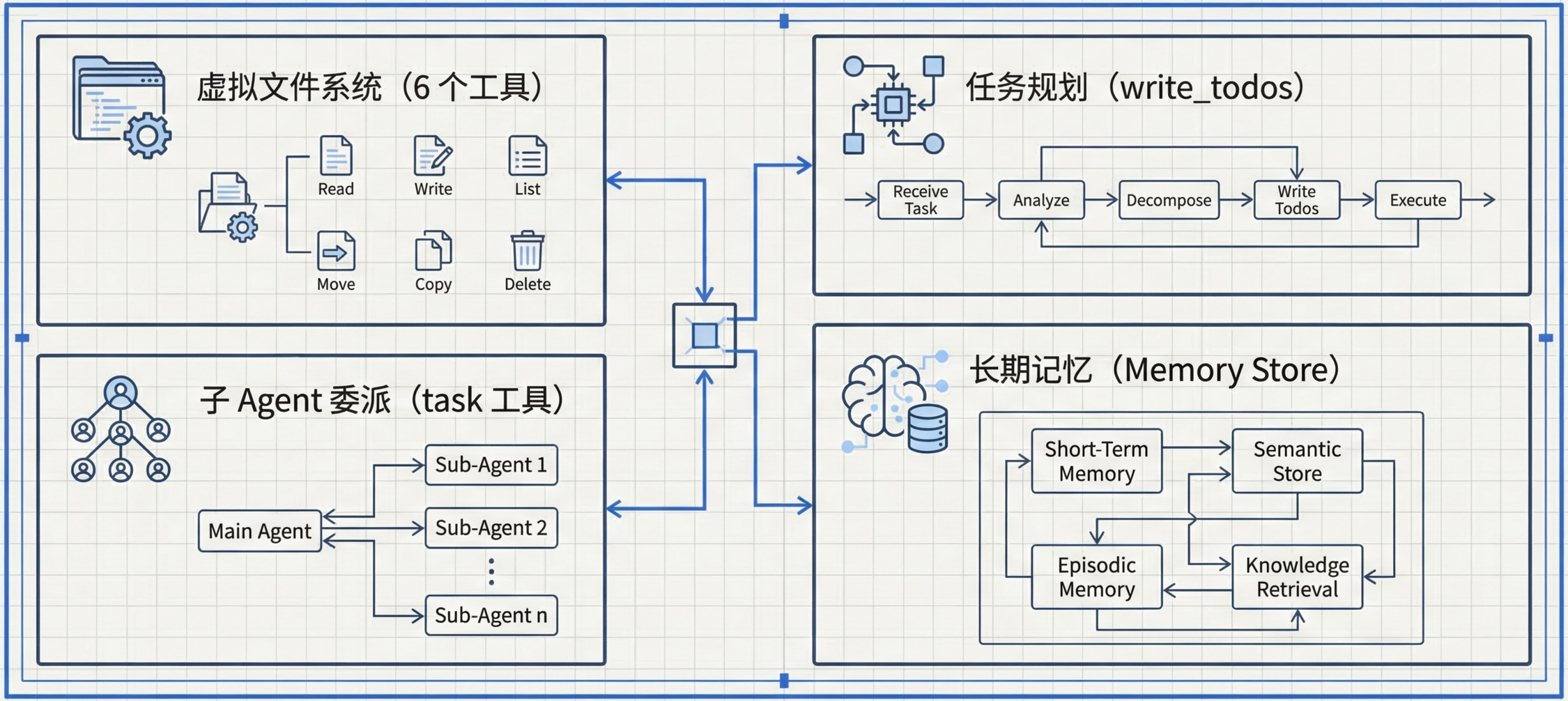
Agent 循环：model → tools → model 循环

中间件：可插拔的行为模块

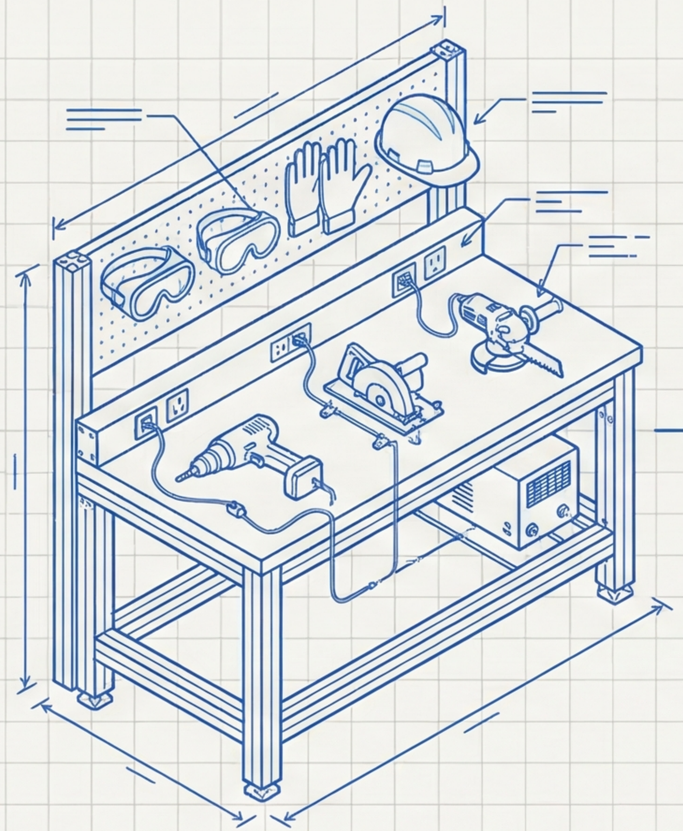
LangGraph (Low-level Runtime & Control Flow)

上层：Deep Agents（Harness）

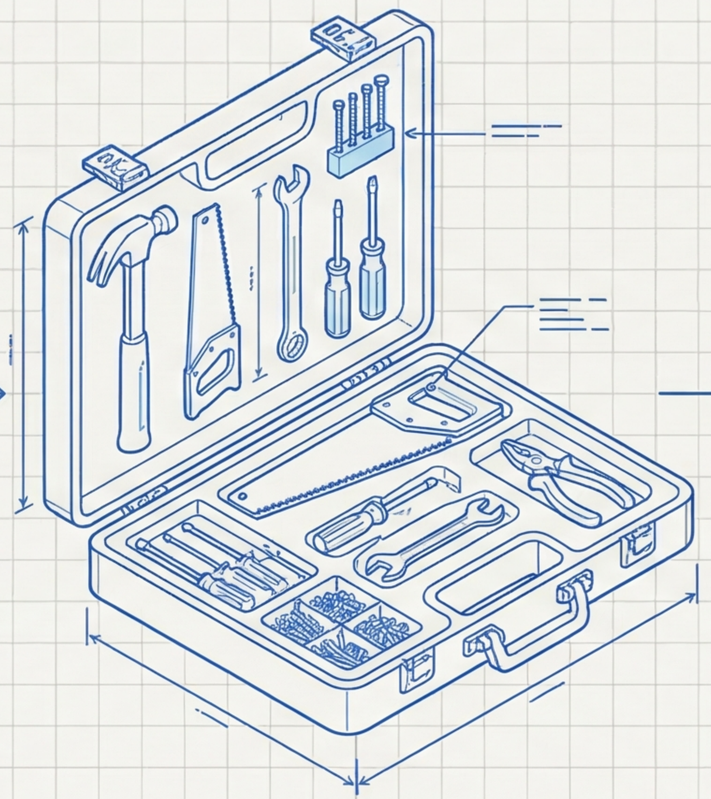
开箱即用的 Agent 套件



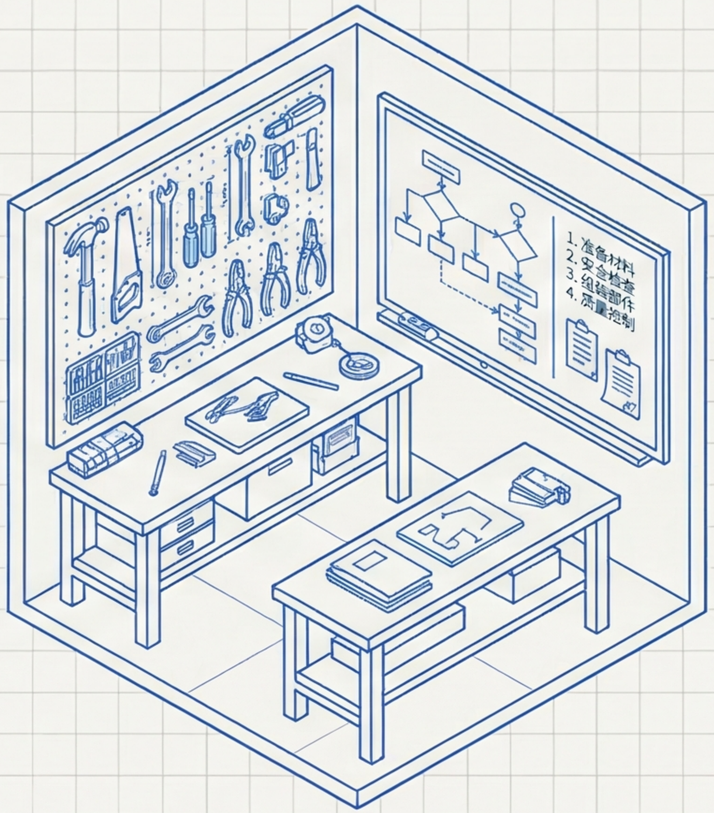
三层架构的直觉类比



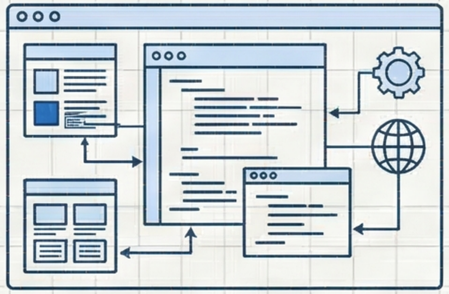
Runtime = 工作台、电源、安全护具（底层基础设施）



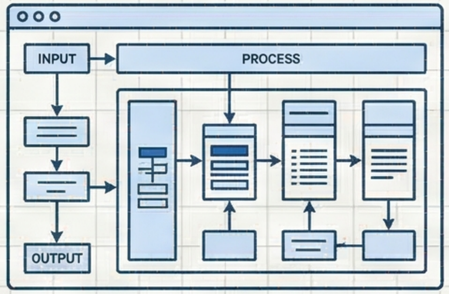
Framework = 锤子、锯子、钉子（标准化工具）



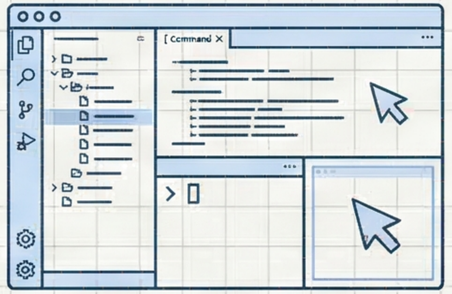
Harness = 装好的工具间，工具挂墙上，流程贴白板（开箱即用）



Claude Code



Manus



Cursor

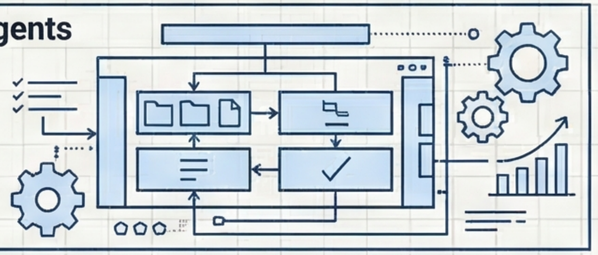
Claude Code、Manus、Cursor 的共性：

- ✓ 文件系统操作
- ✓ 子任务委派
- ✓ 任务规划能力
- ✓ 上下文管理策略

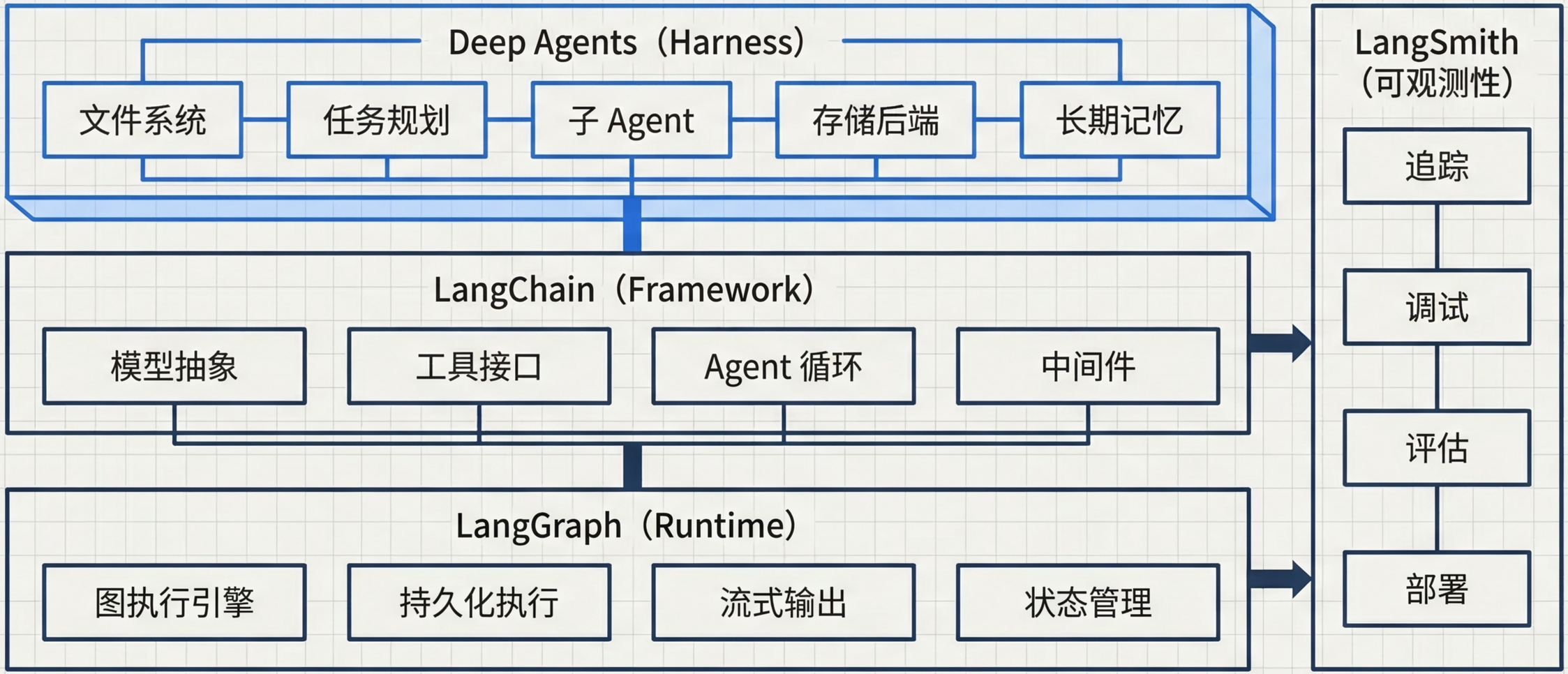
为什么需要 Harness?

成功的 Agent 产品都长得差不多

Harness 的价值：
把验证过的模式固化下来



Deep Agents 技术全景



本讲回顾

- Runtime → Framework → Harness，自底向上

- Harness = 验证过的 Agent 模式的固化

- Deep Agents 的核心：Context Engineering

- 下一讲：Context Engineering 与竞品对比

下一讲预览：Context Engineering 与竞品对比