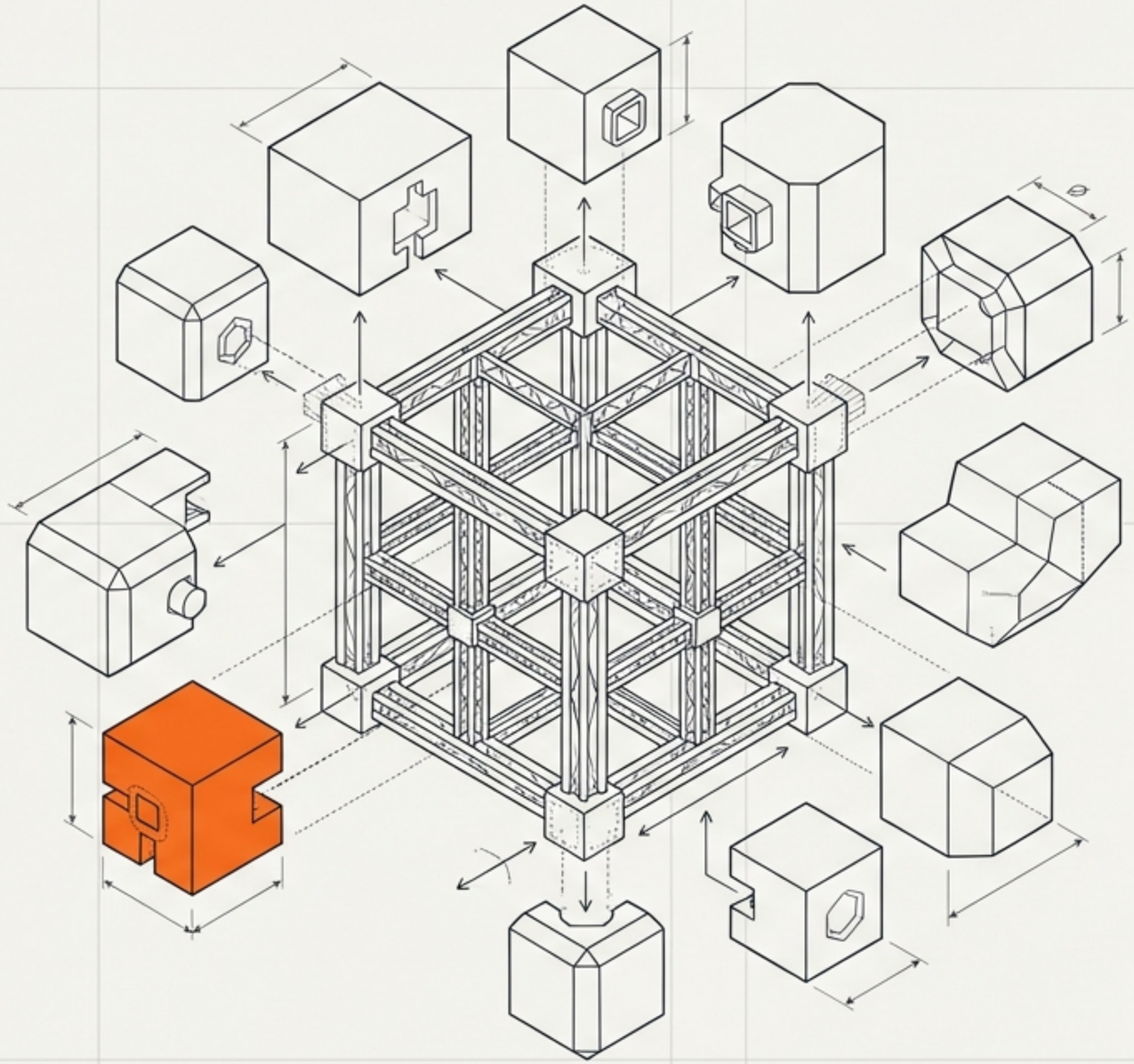
偷取清单：从 20 个开源项目里拆出一个多 Agent 平台

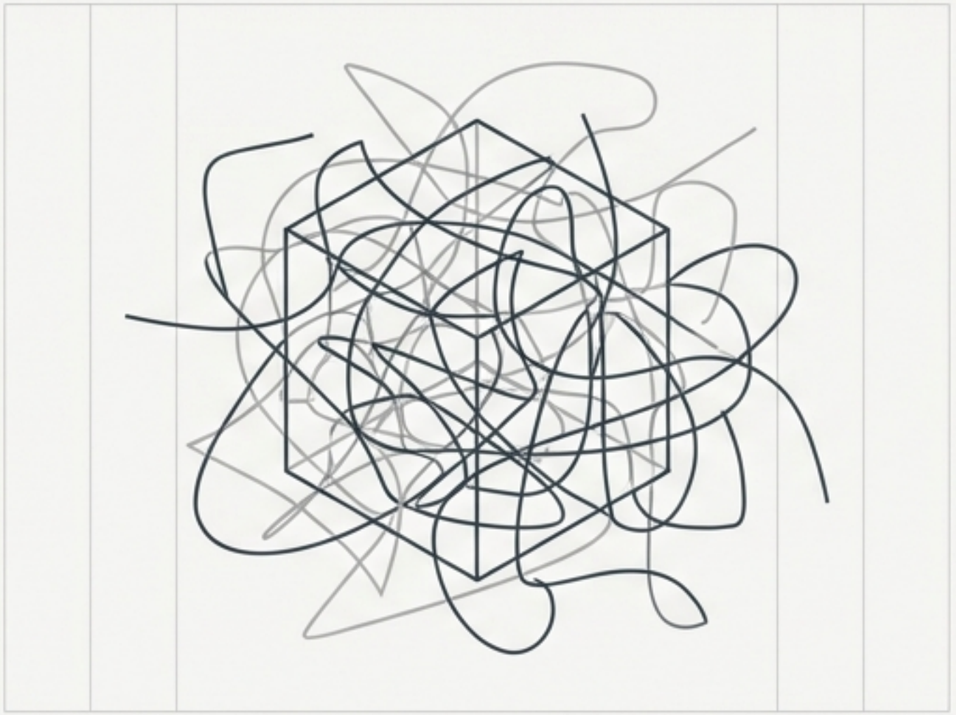
好的架构不是凭空想出来的，
而是从共性里读出来的。



Agora 架构剖析指南

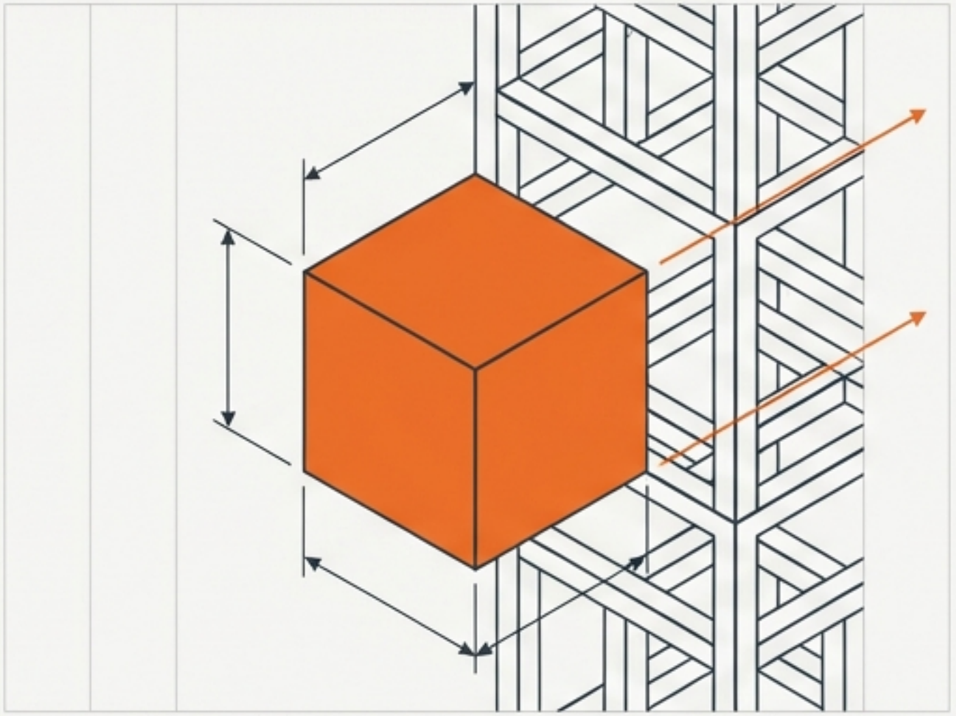
重新定义“偷”：萃取模式，而不是复制代码

模式 A：Fork（全盘接收）



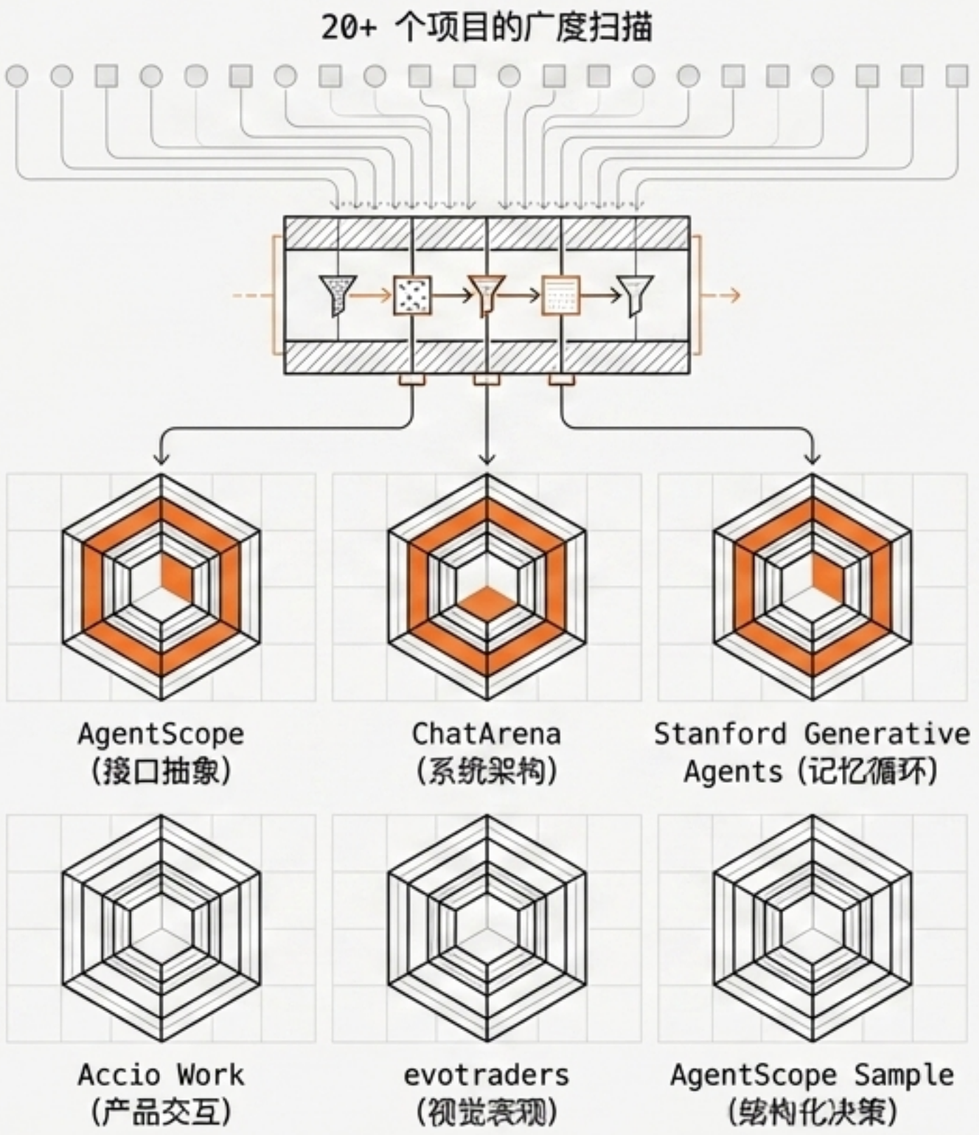
- 被动接受技术债
- 受限于他人的架构体系
- 妥协的 UI/UX 体验

模式 B：Steal（精准萃取）



- 不 fork 不等于不学
- 只萃取最优雅的设计模式
- 保持 100% 的代码控制权
- 每个核心模块都有明确的设计出处

广度扫描：从二十个项目中锁定六个高价值目标

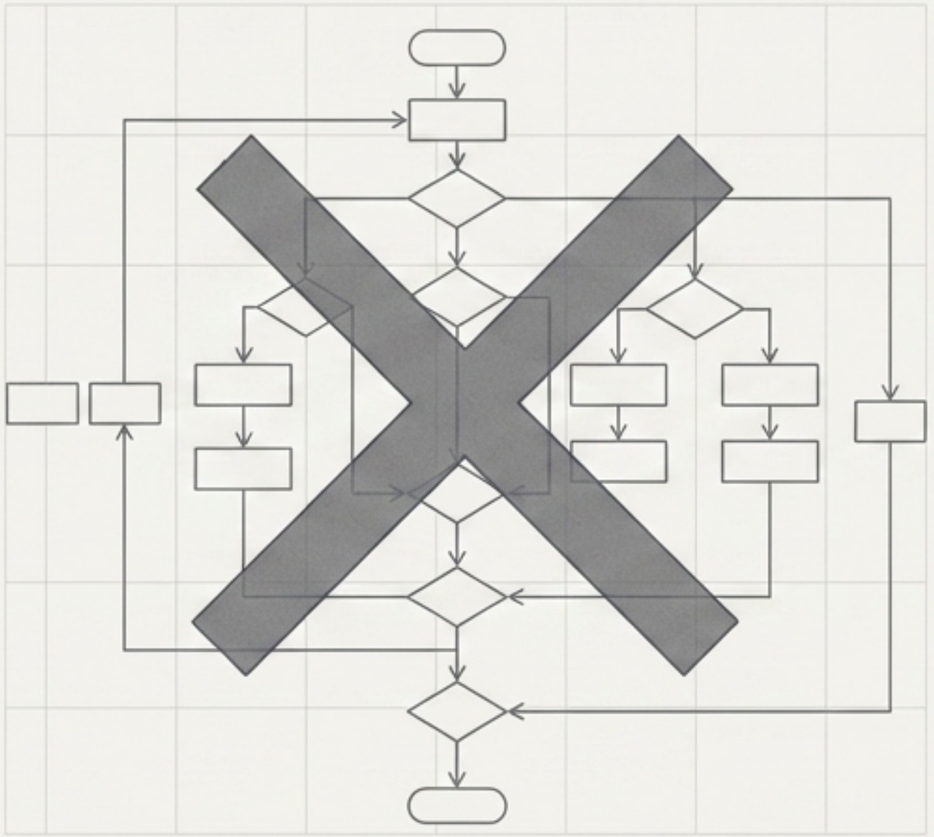


扫描标准：每个项目限时 15 分钟，看 README、看核心接口、跑 demo。按维度拆解评分，寻找唯一的精神图腾。

偷取矩阵：核心模块的设计出处

来源项目	目标模块	核心洞察	技术映射
AgentScope	Agent 接口与消息	分离“回复”与“接收”动作	TypeScript 双接口 + 嵌套频道
ChatArena	整体架构	将“游戏规则”作为独立的可插拔层	Platform > Mode > Agent
斯坦福论文	记忆系统	没有反思，Agent 只是最近消息的复读机	pgvector 检索 + LLM 反思
AgentScope Sample	决策层	Prompt 是建议，Schema 才是物理约束	Zod schema + Vercel AI SDK
evotraders	前端视觉	把文字 Log 变成一群在互动的角色	呼吸灯状态流 + 空间化气泡
Accio Work	交互隐喻	“群聊”是最高效的 Agent 编排心智模型	像建群一样组建 AI 团队

交互革命：“建群聊”是最高效的系统隐喻



传统的 Agent 编排：认知门槛极高

组建团队 = 建群：
创建 Agent 就像添加联系人。

分配任务 = 发消息：
用户无需理解复杂的路由逻辑。

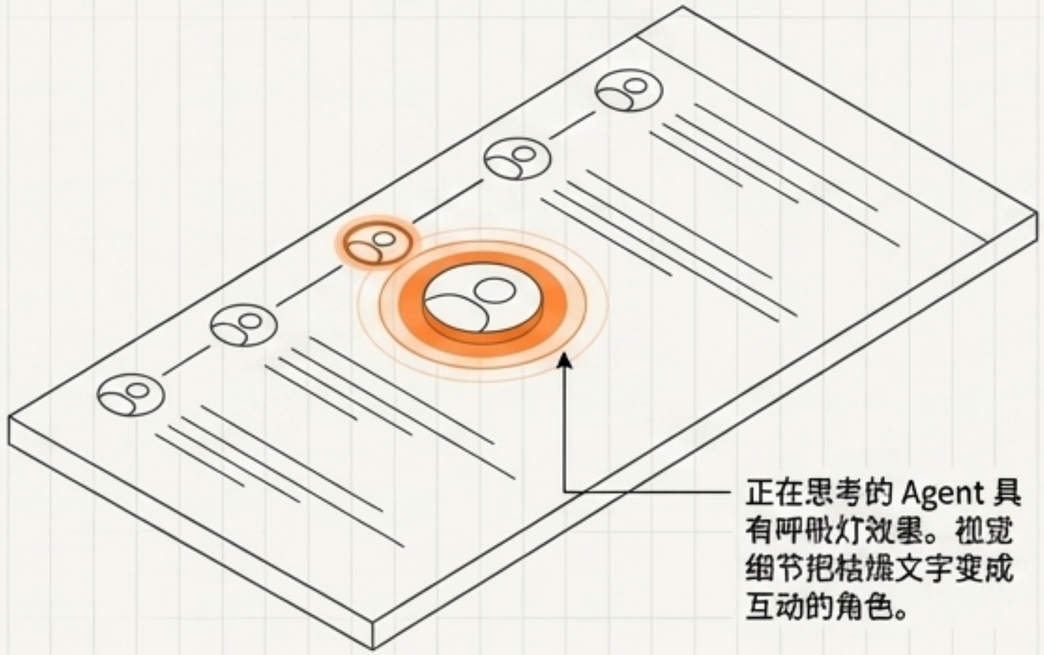
架构决定选型：为实现即时通讯体验，放弃 Python，选择 Next.js + Socket.io。

Accio Work 的产品洞察：建群聊

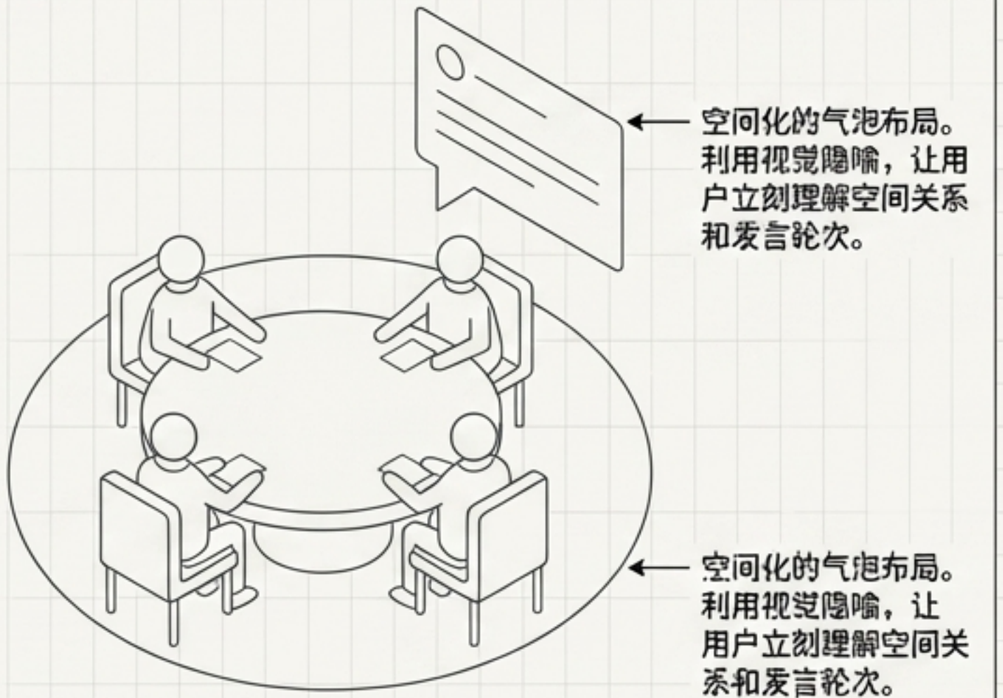
视觉转化：让 AI 对话脱离 Log 文件的枯燥感

出处标记：灵感萃取自 evotraders

模块一：AgentFeed



模块二：RoomView

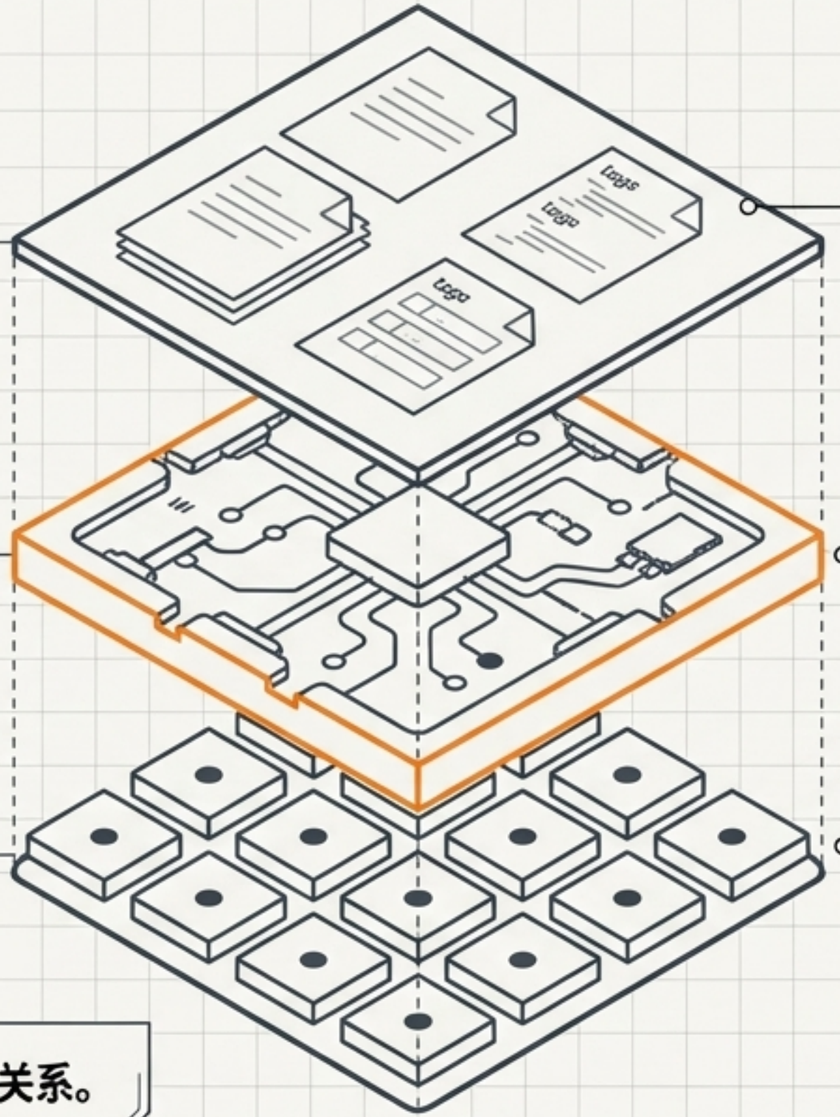


拆解组装：受 ChatArena 启发的三层优雅架构

模式层（Mode）

平台层（Platform）

参与者（Agent）



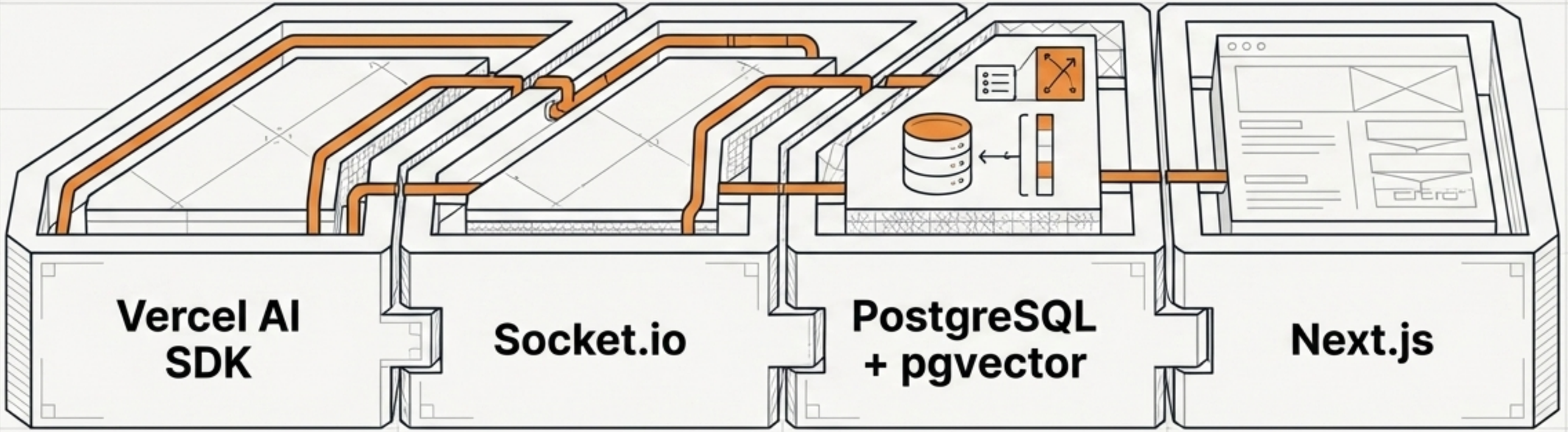
映射 Environment。规则配置包。
定义角色模板、频道规则、决策格式。
新增游戏无需改平台代码。

映射 Arena。运行时容器。
提供基础设施（房间、频道隔离、
流程控制、实时通信）。

映射 Players。
各自具有人设、模型、记忆。
通过统一接口与平台交互。

核心洞察：层级之间保持极度干净的依赖关系。

基础设施层：纯工具的务实选择



负责多模型路由和结构化输出。

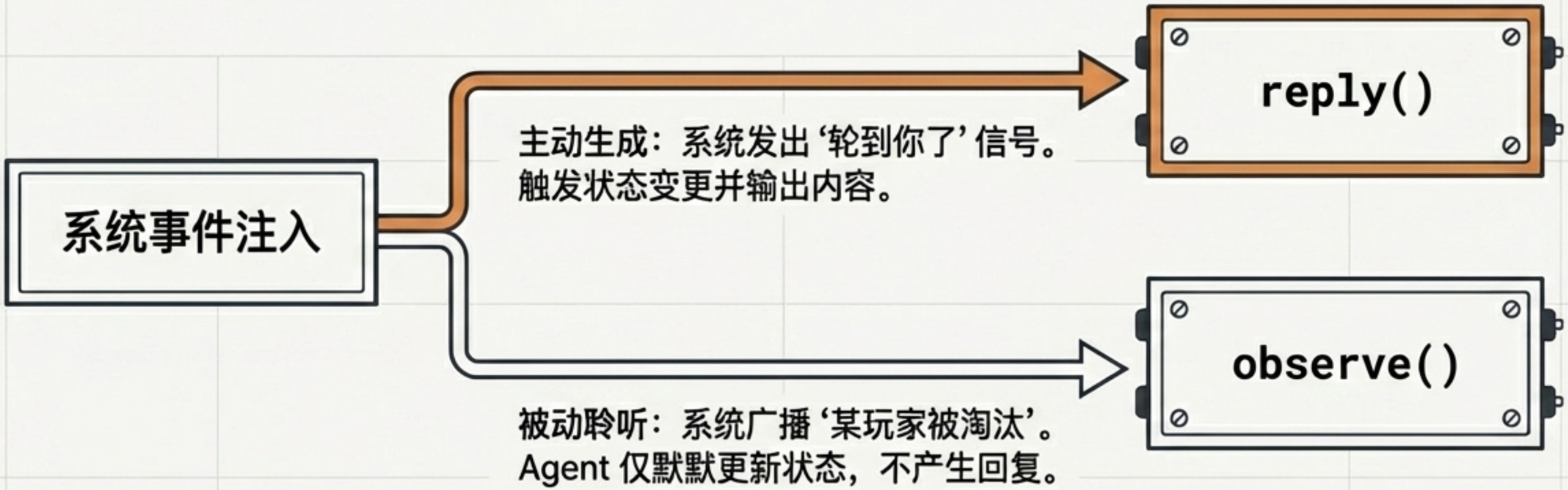
支撑类似群聊的低延迟实时通信交互。

统一存储结构化数据与高维向量记忆。

承载全栈 Web UI 体验。

设计原则：每一层的选型都有明确出处，是从 20 个开源项目的共性中筛选出的最优解。

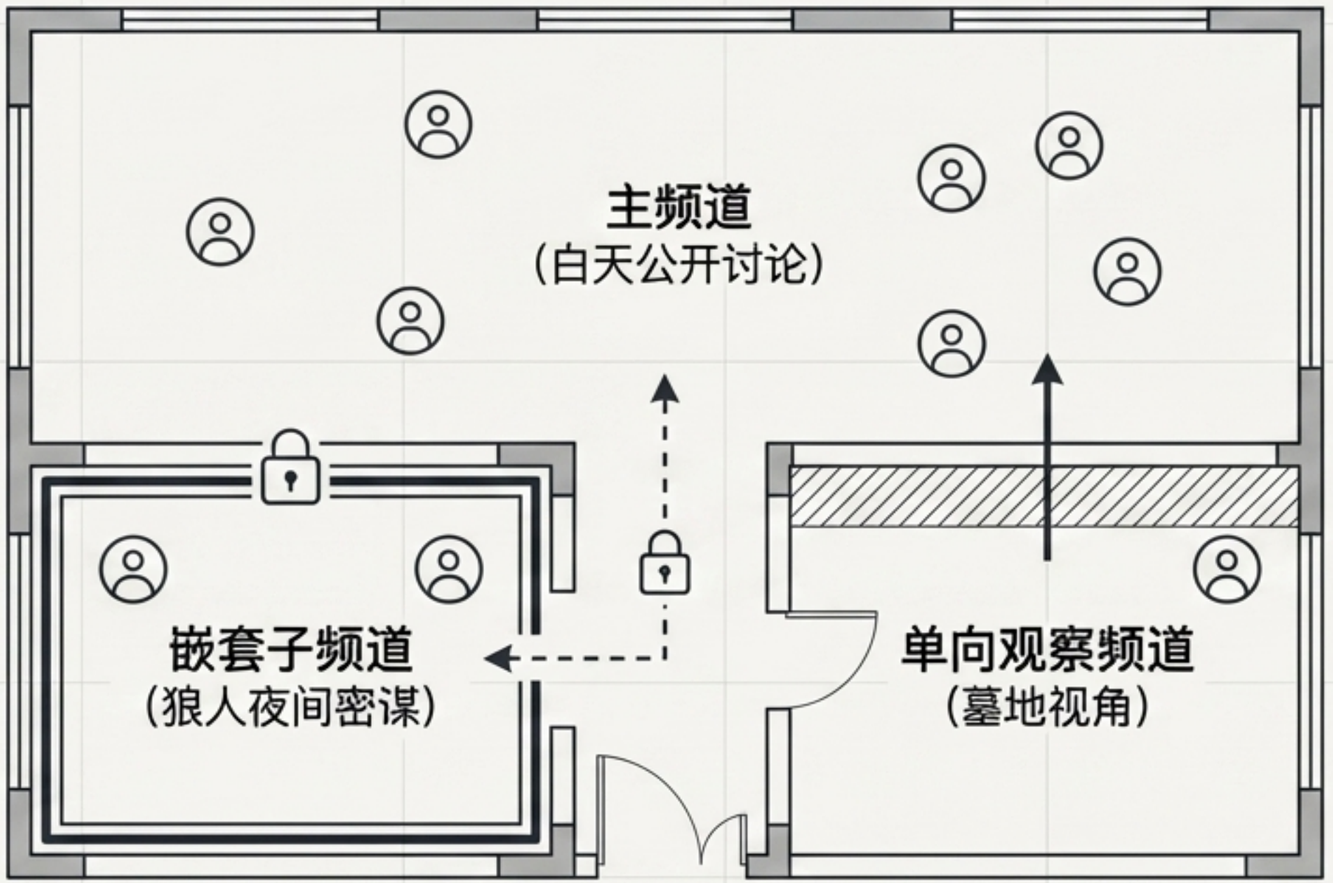
抽象之美：AgentScope 的双接口模式



核心痛点：大部分框架将“生成回复”和“接收信息”混为一谈，导致 Agent 收到消息就必须说话。

优雅解法：物理隔绝动作。在狼人杀中，被淘汰的 Agent 依然可以 observe 墓地频道的消息，但物理上剥夺其 reply 的能力。

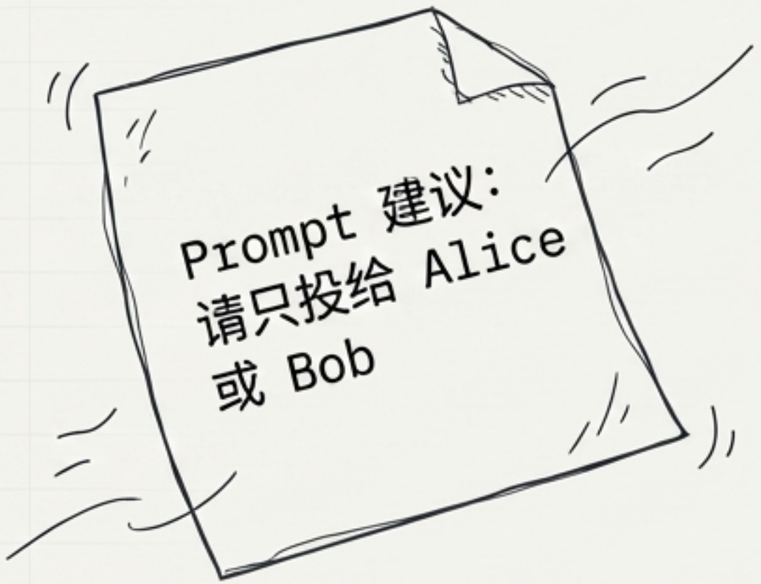
多 Agent 的核心难题：信息与频道隔离



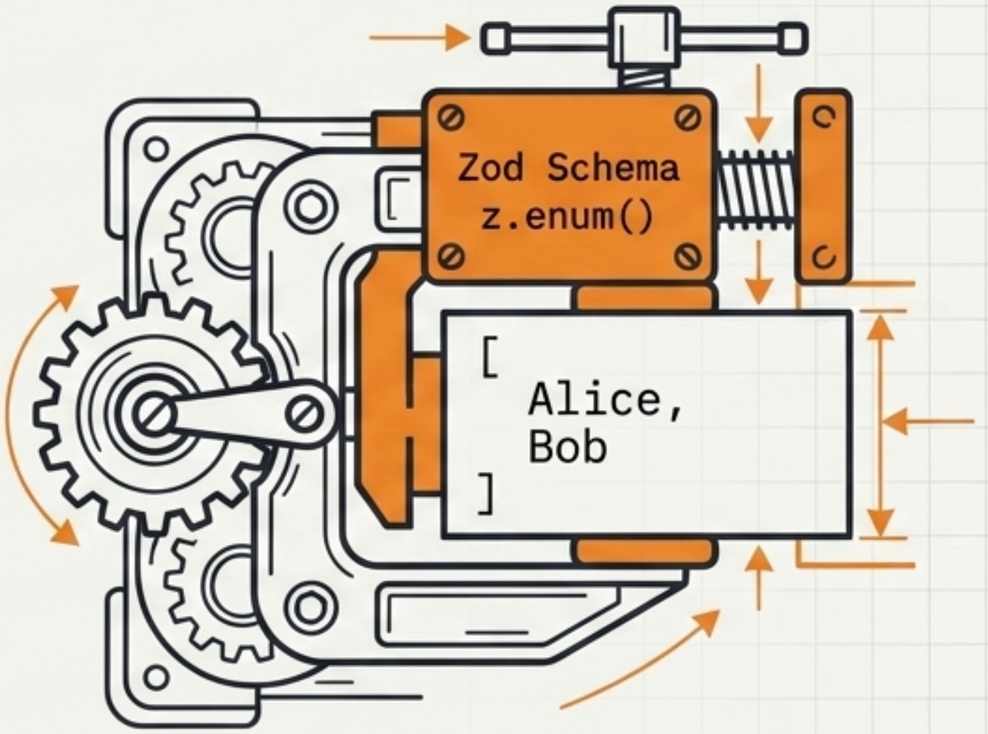
- 没有隔离的系统无法运行复杂剧本。
- 动态订阅：运行时将淘汰玩家移出主频道，加入墓地。
- 广播控制：剧本杀的独占线索需要手动拉取，而非全局广播。

设计出处：MsgHub

物理约束：多 Agent 协作决不能依赖 Prompt 施舍

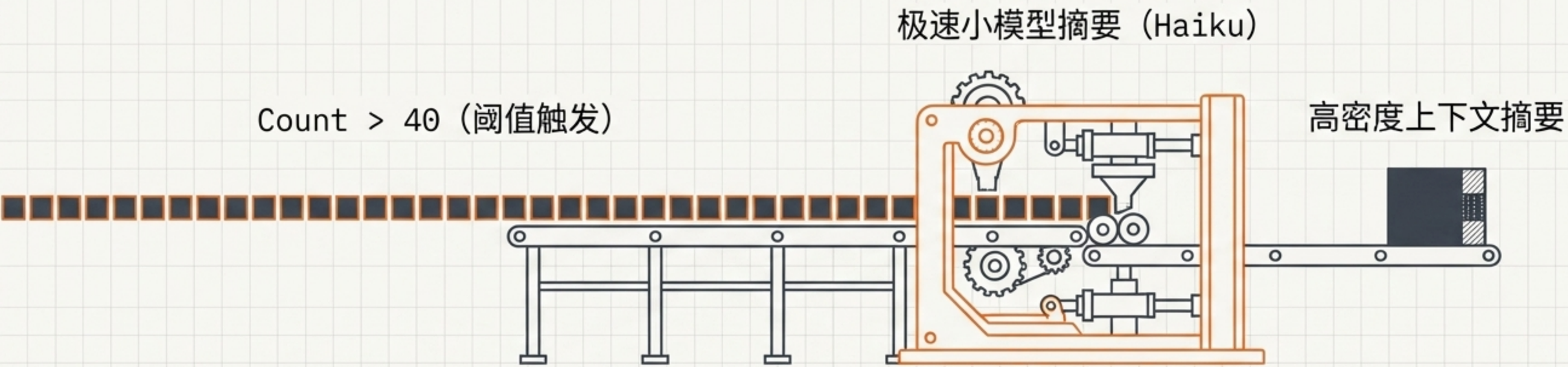


脆弱的 Prompt： AI 可能投给死人、编造名字或输出“我弃权”，导致整个系统崩溃。



强硬的 Schema： 用 Pydantic/Zod 动态生成存活玩家枚举，在物理层面让 Agent 无法输出错误格式。

200 行代码的记忆系统：短时会话压缩



80/20 **法则：**只用 200 行核心代码，实现企业级系统 80% 的记忆能力。

运行时状态：Agent 的上下文窗口不再是冗长的记录，而是【单条摘要 + 最近 15 条消息】。

斯坦福的馈赠：长期语义检索与反思循环

将事件文本向量化，持久化存入 PostgreSQL。

Retrieve（检索）

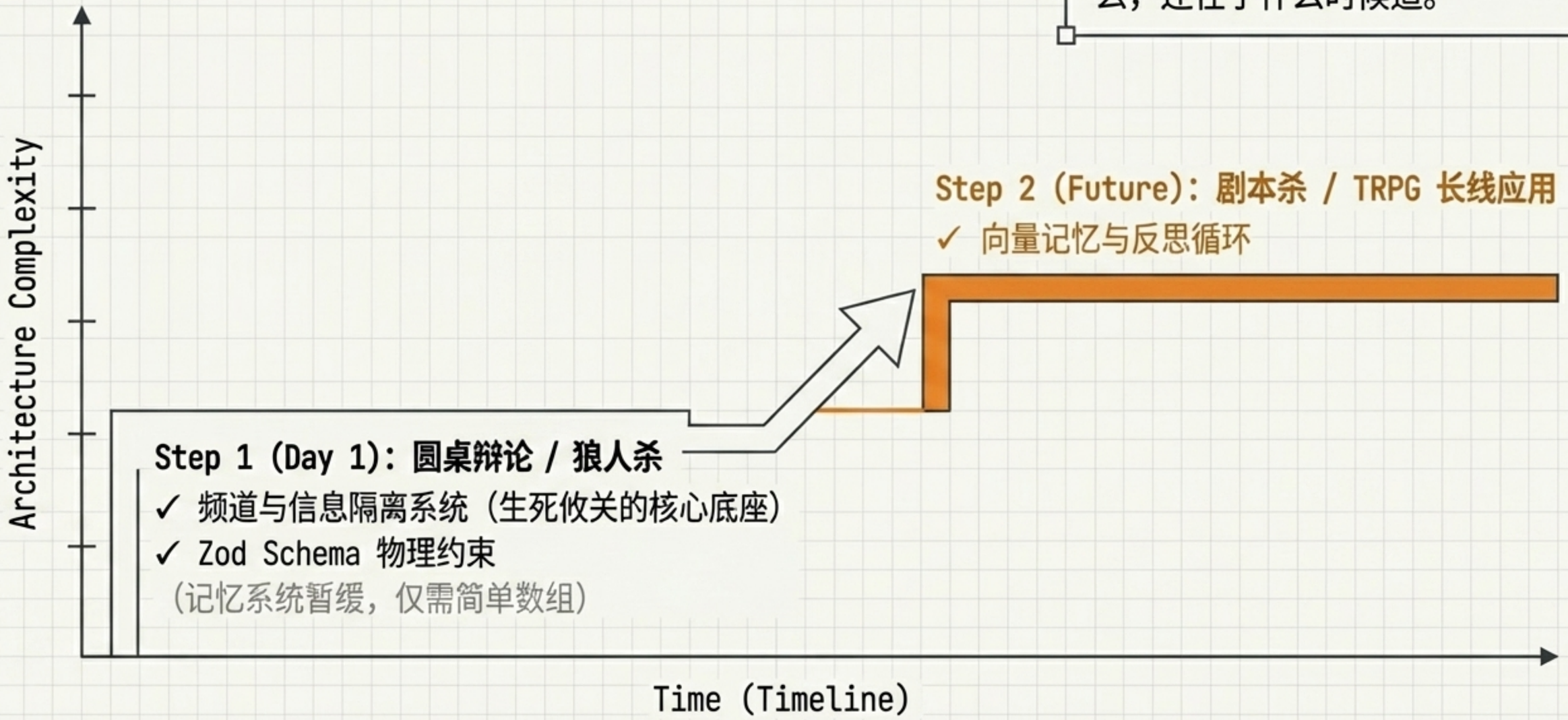
基于当前情境，通过余弦相似度抽取相关历史记忆注入 Prompt。

Reflect（反思）

核心洞察：定期回顾流水账式记忆，利用 LLM 提炼高层认知（如：“张三连续替李四说话，他们是同伙”），存回记忆库。

实施策略：标记时间线，按需引入模块

并非所有模块都需要在 Day 1 偷取。架构的优雅不仅在于造什么，还在于什么时候造。



架构师的“偷取”说明书

不设限搜寻，每个项目限时 15 分钟评估核心接口。

告别笼统评价，别给接口、UI、隔离机制单独打分。

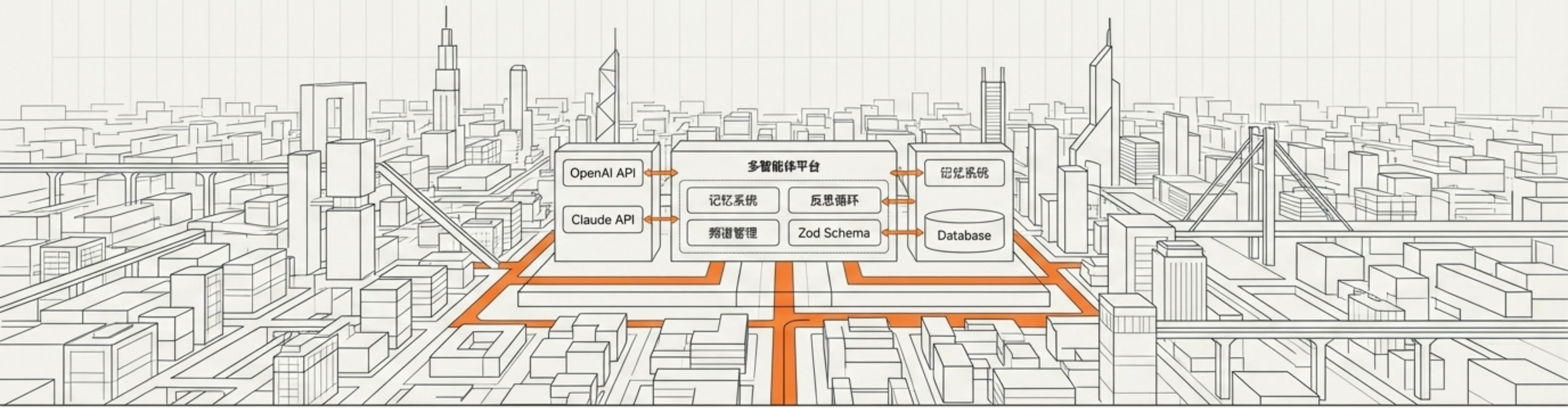
为你的每一个自建模块，指定一个唯一的精神图腾。

将来源、模块、洞察和技术映射写在一张表里。

区分生死攸关的核心底座与锦上添花的外围功能，分阶段实施。

站在巨人的肩膀上，用自己的腿走路。

好的设计不是从零想出来的，是从大量现有实现的共性中读出来的。



完