COORDINATION LAYER · EXECUTION TRAJECTORY

多智能体协同的过程评测

面向执行轨迹的协调机制设计

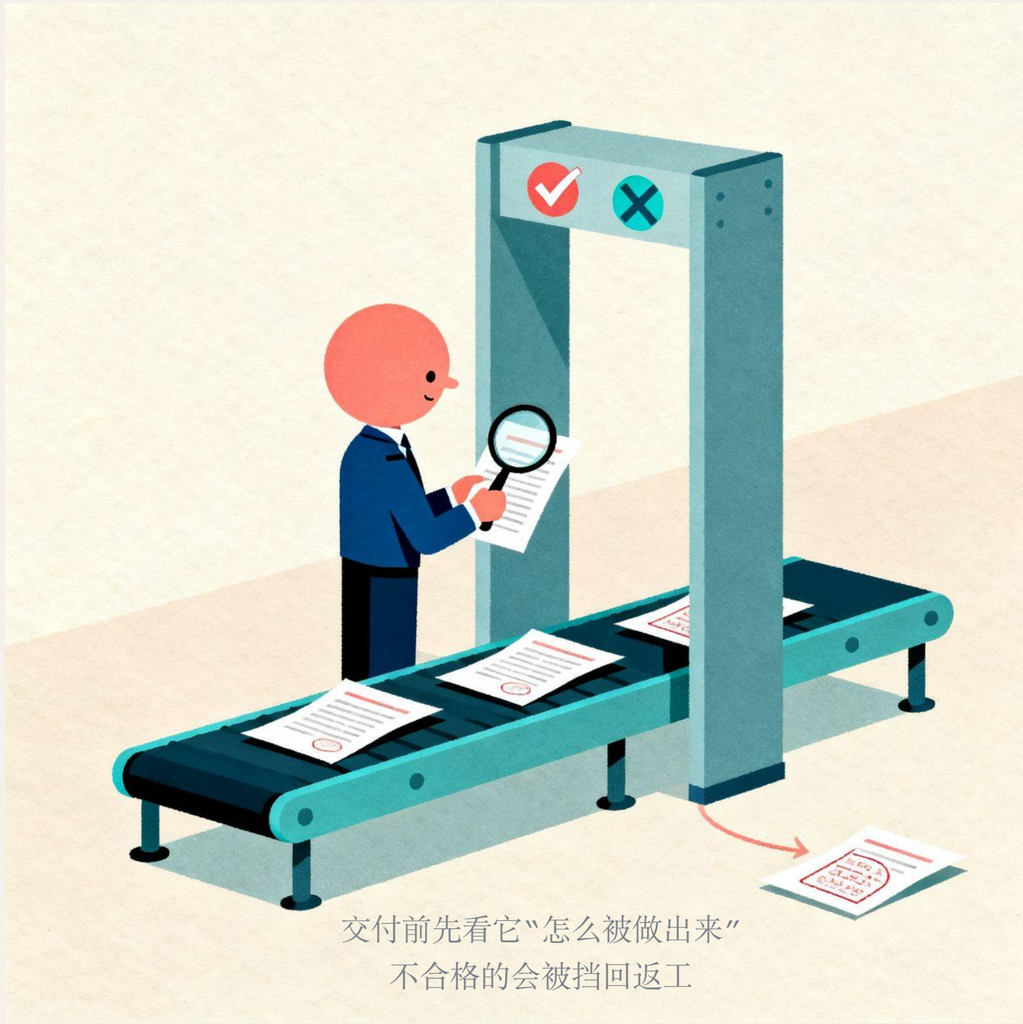
货拉拉 · 资深算法工程师 · 赵江杰

拆解 · 交接 · 沉淀 · 验收 · 复盘

只看最终答案，看不到协作如何失败

最终结果评测仍然必要，但它看不见协作过程。过程评测把五件事变成可回答的问题：

任务拆解	协作结构是否清晰？
执行过程	轨迹是否可追踪？
中间产物	证据链是否完整？
失败之后	原因是否可定位？
交付之前	结果是否可验收？



交付前先看它"怎么被做出来"
不合格的会被挡回返工

来源 · ADK Evaluation; From Agent Traces to Trust (2606.04990); Corrupt success: PAE (2603.03116)

执行轨迹：把结果成败展开成可诊断的机制链

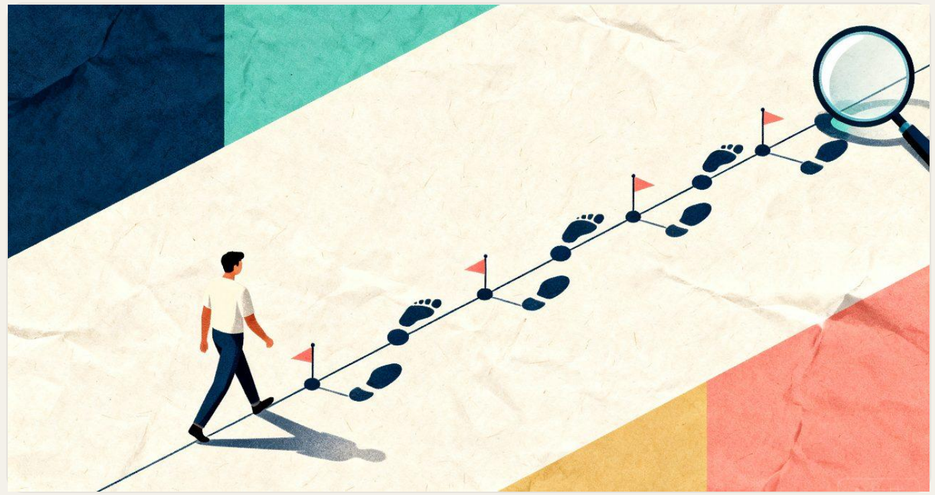
把一次执行展开成轨迹，结果的“成败”就变成一条可逐步检视的因果链，失败发生在哪一步、对应哪个机制，都能看见。

goal → plan → agent_turn

tool_call → observation → artifact

gate_result → handoff / replan → termination

失败定位 → 协调 coordination · 执行器 harness · 门控 gate · 控制 control



来源 · ADK Evaluation (expected vs actual trajectory) ; Willful Disobedience (2603.23806); OpenAI Agents SDK Tracing

2026 的共同信号：评测对象正在前移

从“答案对不对”，扩展到“结构与轨迹对不对”。

Coordination layer

协调层成为可配置、可比较的架构对象

2605.03310

Trajectory eval

轨迹级评测与诊断成为基准任务

2604.02022 · 2606.07054

Execution provenance

证据溯源：结果如何由工具/记忆产生

2606.04990

Graph verification

workflow graph 可静态验证

2603.20356

Structural coverage

agent/tool/delegation 路径是否被覆盖

2605.26521

Harness evaluation

执行环境本身成为评测对象

2606.05548 · 2604.25850

协调层：让协作结构成为设计对象

CONTROL PLANE · 控制面 — 决策 / 授权 / 落账（不直接碰世界）

Orchestration 协调
谁执行 · 何时 · 拓扑 · 委派 · 派生

Gates + State 门控 + 状态
是否允许 · 落账 · 持久记录（单写者）

↑ 提案 + 证据 + 委派请求　　↓ 授权 + turn-lease + 落账

DATA PLANE · 数据面 — 执行（碰外部世界）

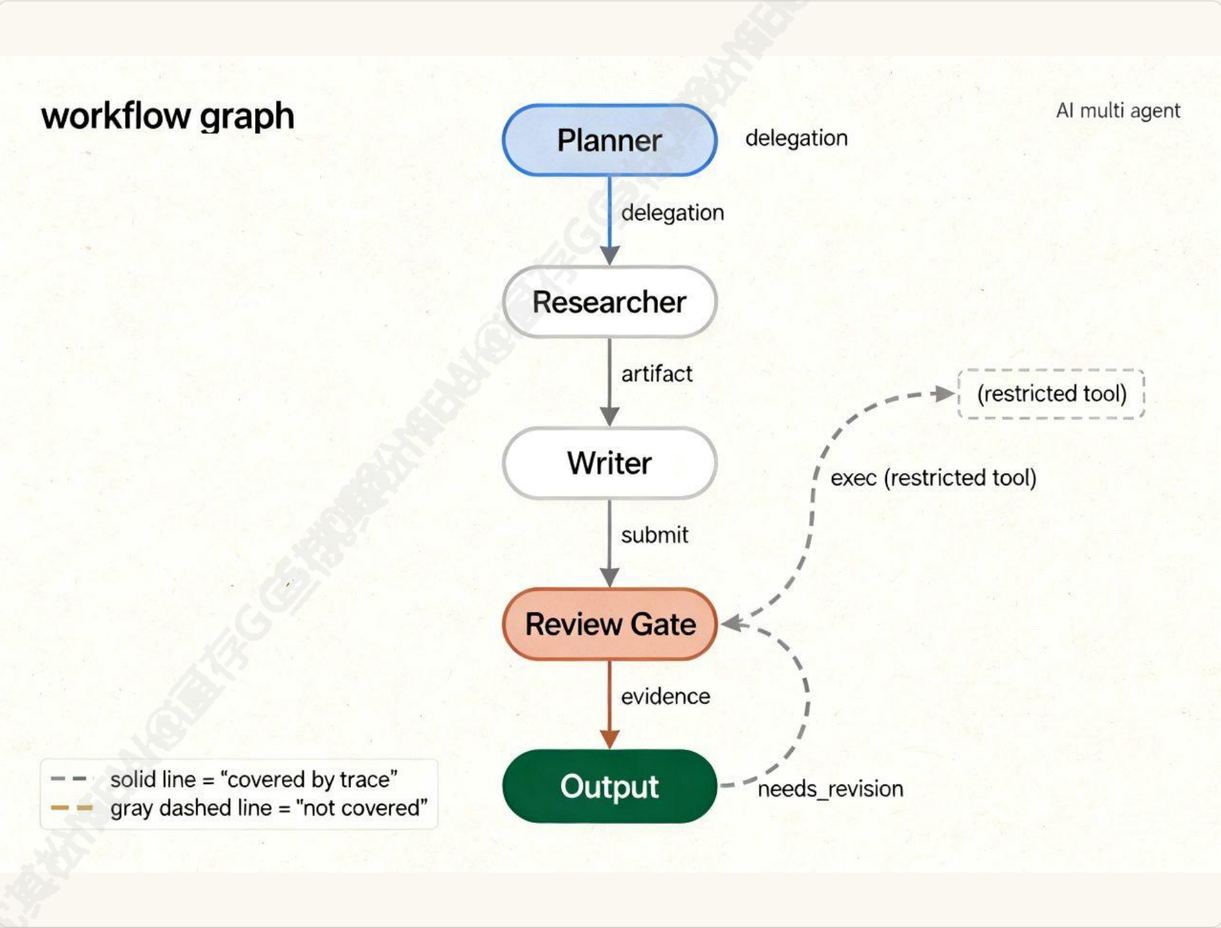
Harness 执行器
循环 · 模型 · 工具 · 副作用 — 从不落账、从不派生

Scaffold 行为配置
prompts · contracts

来源 · Coordination as an Architectural Layer (2605.03310); control/data-plane synthesis（本项目综合）

Workflow graph 是过程评测的结构底座

Trace 告诉我们发生了什么；workflow graph 告诉我们**哪些路径、委派、门控和工具约束本来应该被检查**。



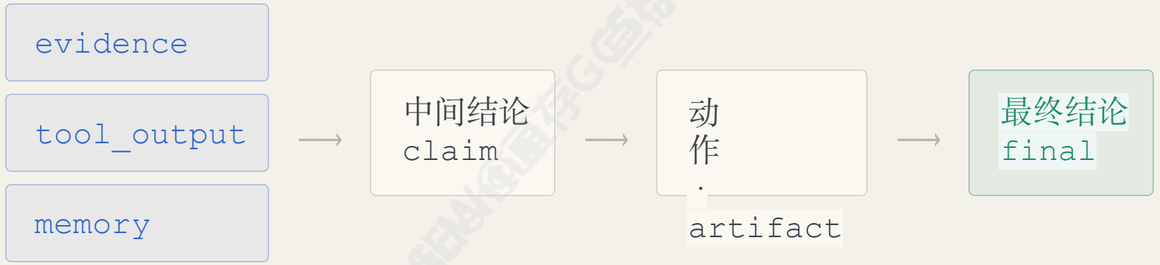
Graph	预期协调结构 · 该被检查的路径
Trace	实际执行路径 · 真的走过的
评测	用 trace **覆盖 · 对照 · 验证** graph 的义务

结构覆盖义务 · coverage obligations

reachable agents	每个 agent 是否都被走到
allowed tool edges	允许的工具调用是否覆盖
restricted tool edges	受限工具边是否被验证
delegation edges	委派路径是否被测试
gate / policy	门控是否真的触发

来源 · Structural Coverage arxiv.org/abs/2605.26521 · Agentproof arxiv.org/abs/2603.20356

轨迹的价值：解释结果如何产生



每条 trace link 都能往回追 → who · when · why · based-on-what

证据关联：结论与它依赖的证据、工具输出、动作绑定在一起

执行溯源：能说清工具、记忆、中间产物如何影响了结果

长程任务需要跨步骤把证据聚合，而不是孤立的单步日志

来源 · From Agent Traces to Trust (2606.04990); TRACE (2606.07054)

协调机制的可设计变量

多智能体系统的质量，取决于这组变量怎么设计；它们跑出执行轨迹，才谈得上评测与改进。

协调机制变量	责任边界	轨迹证据
拓扑与角色分工	Orchestration	agent_spec / schedule
决策权与轮次	Orchestration · Control	turn_lease / vote
交接与委派	Orchestration	handoff_requested / message
聚合与收敛/终止	Orchestration · Control	consensus_check / terminate
失败处理与升级	Orchestration · Control	replan / escalate / fallback

这五个变量就是评测与改进的对象——下一页把每类失败信号映射到该改的机制。

来源 · Coordination as an Architectural Layer (2605.03310)：协调层可配置维度与失败签名 · Silo-Bench (2603.01045) · owner 取自控制/数据面分层（docs/96 本项目综合）

协作失败信号 → 协调机制诊断

角色缺位 / 重叠 / 单点

该出现的角色没出现，或都挤在一个点

↳ 看 拓扑·角色分工 / delegation

行动权不清 / 少数被压

该谁发言、决策不明；异议被多数淹没

↳ 看 决策权·轮次 / 聚合权重

交接断裂 / 丢上下文

委派链断、接收方拿不到必要上下文

↳ 看 交接·委派 / handoff 契约·隔离

早停 / 不收敛 / 错误级联

过早终止、反复绕圈、错误沿链放大

↳ 看 聚合收敛终止 / 失败升级

面向轨迹评测的协调机制分层设计

每一层都按同一契约设计：设计对象 → 轨迹事件 → 可评测问题；我们的系统已按此实现，后面逐一看。

层	设计对象	轨迹事件	可评测问题
Orchestration	拓扑 · 角色 · 委派 · 轮次	turn_lease / handoff / schedule	该谁做·走没走到·交接全不全
Control / Gates	授权 · 落账 · 终止 · 升级	gate_recorded / decision / replan	为什么通过·返工·何时停
Harness	loop · 工具调用 · 副作用	tool_call / observation / artifact	工具越权·产物可否验收
Observability / Governance	schema · 溯源 · 回归集	trace / evidence_ref / eval_run	能否回放·claim 溯源·过没过回归

S5 讲协调在哪运行；这页讲每层怎么设计才可被轨迹评测驱动——后面 4 个系统各自实例化这套分层。

参考架构（提出，非标准）· 抽象自 HydraMind / AI4S / Meetkat / short-video 的实现 · Coordination as an Architectural Layer (2605.03310)

评测驱动协调机制设计：诊断 → 修订 → 回归

诊断 → 修订

失败轨迹 → 定位机制变量 → 提出修订

例：角色长期失败 → 拆分 / 重定角色

例：handoff 丢上下文 → 收紧交接契约

回归后采纳

新机制必须通过 eval / regression

通过才采纳；不过则打回修订

评测不是只给分，而是把协调机制改成可验证的下一版。

来源 · ABSTRAL（前沿）· MAFBench / Silo-Bench · AHE / RHO / Self-Harness / HarnessFix

四个工程锚点，各自支撑一类协作评测

协调机制不只停在概念层 —— 四个系统分别把一类**“可评测协作”**落到真实代码、事件与产物证据里。

HydraMind

结构 治理

带生产级架构纪律的多智能体框架：provider 隔离 · agent-queue 解耦 · 门控即一等公民契约。

机制锚点：四层分层 · RuntimeSession 单写 SoT · 可替换 ExecutionHarness · GateContract 门控。

本地代码证据 证据卡片 · Apache-2.0

AI4S FlockMind

重规划

面向学术写作的多智能体研究助手 · CentralizedOrchestrator 中心化编排。

机制锚点：评审把失败定位回 研究 / 大纲 / 写作 某一阶段（replan_scope → resume_action），建在跨角色证据接力之上。

github.com/ChristopheZhao/FlockMind

Meetkat

协作

native-agent 决策室 · supervisor 排“谁何时说”，specialist 自己决定“说什么”。

机制锚点：协同动作 = 可回放事件 （supervisor.plan / agent.message）· schema-first 日志全程可复盘。

github.com/ChristopheZhao/Meetkat · Apache-2.0

short-video-maker MuseCraft

轨迹 · 产物

多个专业 agent 协同的长链路创作流水线。

机制锚点：每段 = 带租约的 attempt，阶段边沿触发受授权门，失败定位到 node / attempt / gate / reason_code。

本地代码证据 证据卡片

来源 · HydraMind / FlockMind 为主锚点；Meetkat / short-video 为辅助案例，分别支撑事件化轨迹采集与创意流水线轨迹物化。

HydraMind 锚点：协调责任如何落到可评测边界

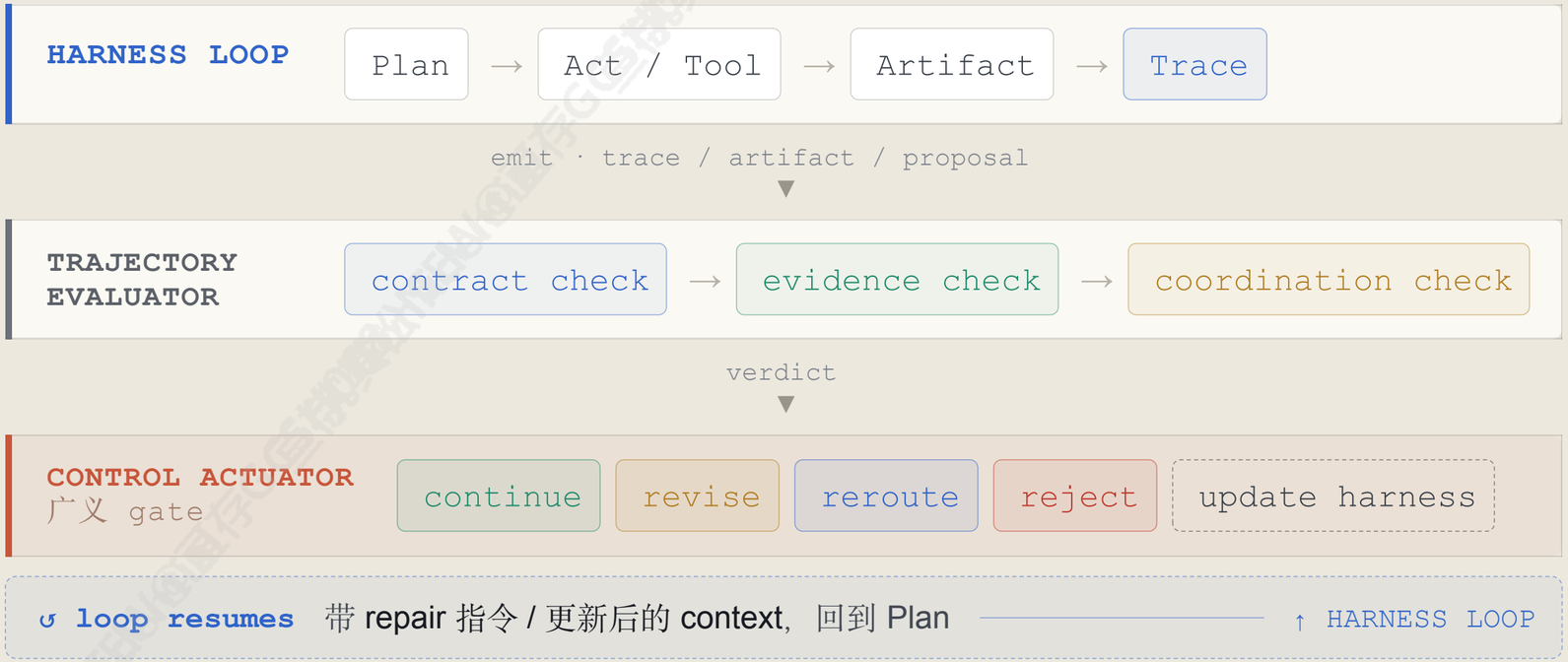
一次执行如何流过四层

1. Harness 跑模型+工具，产出 proposal + evidence， 但不落账
2. Gate 按 GateContract 校验证据是否充分
3. Control 作为单写者把决定落账（RuntimeSession）
4. Orchestration 据此决定下一个 turn

来源 · https://github.com/ChristopheZhao/HydraMind

过程评测如何驱动 Harness Loop

Gate 本身不是新东西 —— 新机制是把**轨迹评测结果**转成 harness loop 的下一步动作：**continue · revise · reroute · reject · update harness**。



前沿支撑 · evidence

ADK · 评测对象是 trajectory，不只是最终回答

adk.dev / evaluate

OpenAI Agents SDK · guardrails = 运行时拦截

input / output / tool guardrails

Harness-Bench · 能力报告在 model-harness 配置层

arXiv 见备注

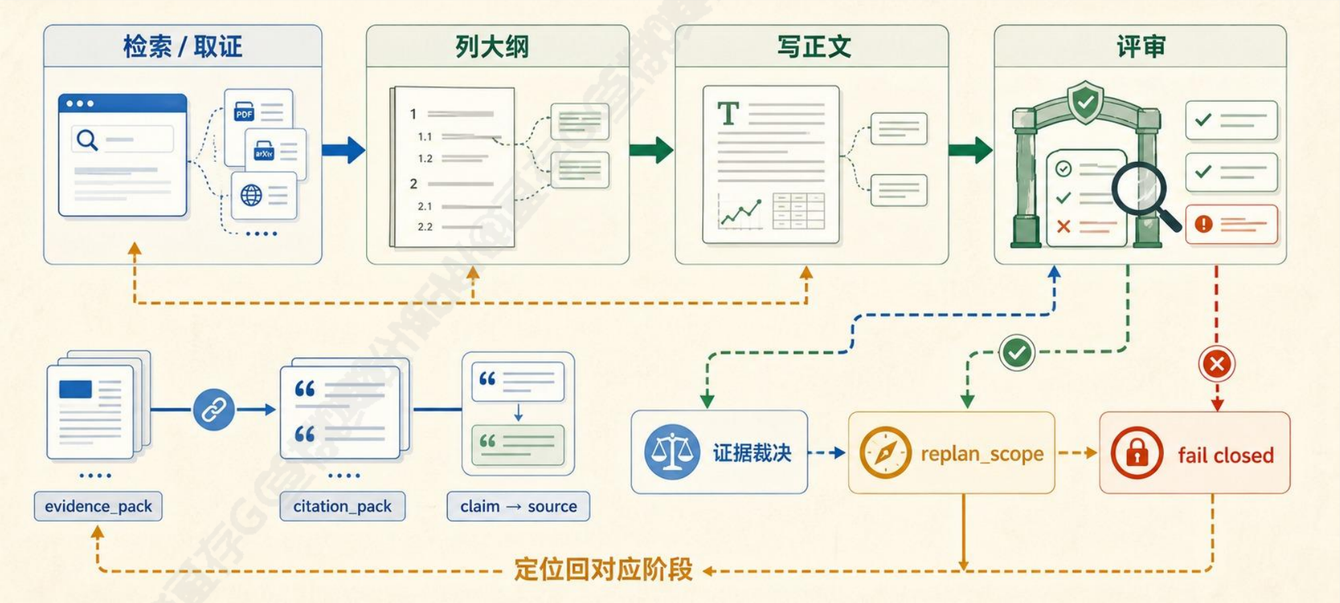
trace 驱动的 harness 改进 · 从历史 / 失败轨迹优化、修复 harness

RHO · AHE · HarnessFix · Self-Harness

来源 · continue / revise / reject 已在 HydraMind 落地；reroute 与 update harness 属前沿方向（RHO / AHE / HarnessFix / Self-Harness）。

评审驱动的阶段化重规划：把“终稿不够好”定位回 查文献 / 列大纲 / 写正文

评审不止判好坏，而是把缺陷定位到 查文献 / 列大纲 / 写正文 哪一步，再把修订回灌到那一步



证据接力底座：evidence_pack / citation_pack 携带 claim→source 血缘与 section 覆盖，作 检索→大纲→写作→评审 的协调载荷。

评审不是打分而是归因：五档 replan_scope 把缺陷定位到具体角色边界。

定位即动作：derive_resume_action 映射修订动作，submit_resume_replan 从被定位阶段恢复；证据门 fail-closed 仅为回灌前硬约束。

来源 · github.com/ChristopheZhao/FlockMind

协调动作事件化为可回放轨迹；产物链路物化为可定位轨迹

协作动作先记成可回放事件、产物链路先拆成逐段可定位的记录——过程评测才有对象

交接 = 带上下文标签的显式事件
而不是藏在 prompt 里的一句话

Meetkat：每个协同动作都记成可回放事件

`plan` `message` `challenge` `handoff` `vote`

short-video：每段创作过关才进入下一段，失败能定位到具体哪段

`concept` → `script` → `image` → `video` → `compose` → `quality check`

每段创作都是一次带"租约"的尝试，过关（受授权的门）才进入下一段；失败能精确定位到哪一段、哪次尝试、卡在哪个门

来源 · github.com/ChristopheZhao/Meetkat · ‖ https://github.com/ChristopheZhao/MuseCraft

协调机制设计的五项工程准则

协作动作事件化、产物化之后，方法可以收成五条直接落地的动作 —— 这是从“能编排”走到“可评测”要补齐的五件事。

01 把协调机制做成可配置、可比较的对象

02 把执行展开成可对照预期的轨迹

03 让结论可溯源到证据

04 把失败定位到具体 agent / 步骤

05 让诊断回灌、持续改进机制

把这五条补齐，多智能体系统才真正从“能编排”迈进“可评测”。

来源 · 01 Coordination Layer 2605.03310 · 02 Structural Coverage 2605.26521 · 03 Agent Traces to Trust 2606.04990 · 04 TraceElephant 2604.22708 · 05 Early Diagnosis 2606.01365 / RHO 2606.05922

FROM ORCHESTRATION → TO PROCESS EVALUATION

从“能编排”，走向“可评测协作过程”

执行轨迹

= 协调机制的证据基础

过程评测

= 诊断协作质量，而非只看答案

协调机制

= 可迭代、可复盘的对象

让多智能体系统从**“能编排”，走向“可评测、可诊断、可复盘”**。

谢谢，欢迎交流

围绕执行轨迹 · 协调机制 · 过程评测，欢迎提问与讨论。

RuntimeSession | GateContract | 可替换 Harness | 轨迹记录