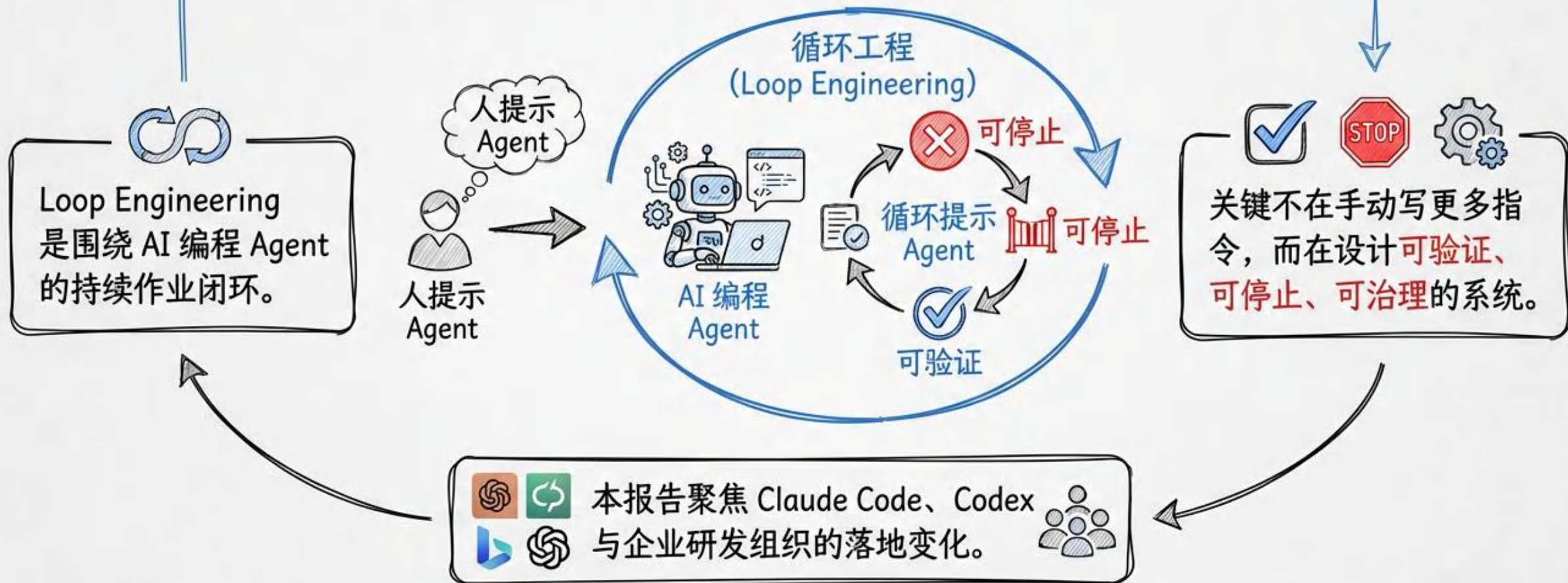


循环工程研究报告

开场与核心判断

AI 编程从“人提示 Agent”走向“循环提示 Agent”



@ 清新研究团队简介

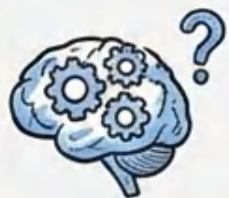
- 领导学术研究团队近30人。指导大数据、AI、人形机器人等多个产业团队。
- 团队坚持：整体主义、实证主义、社会建构、进步主义。

沈阳：清华大学新闻学院/人工智能学院双聘教授、博导



六大研究方向：

视频号：@清新研究；公众号：@清新研究



1. AI大模型
理论与哲学



2. AI文艺



3. AI应用



4. 新媒体与
网络舆论



5. 大数据



6. XR应用

✉ 邮箱：124739259@qq.com | 微博：@ 清新研究 | 公众号：@ 清新研究

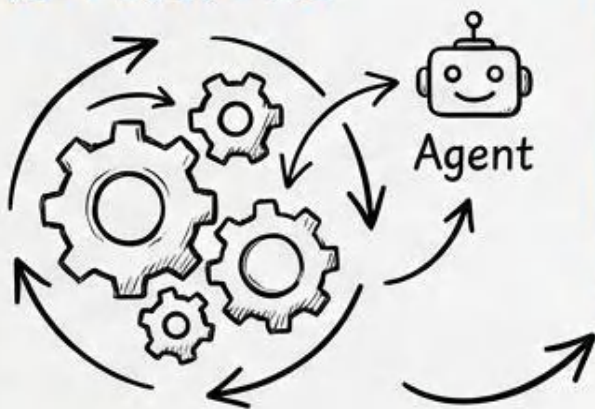
更多免费行业报告 并购家 www.ipoiipo.cn

一句话判断

开场与核心判断

Build the loop, stay the engineer.

循环负责调度



Agent 负责执行

“循环可以提示
Agent，但工程判断
不能外包。”



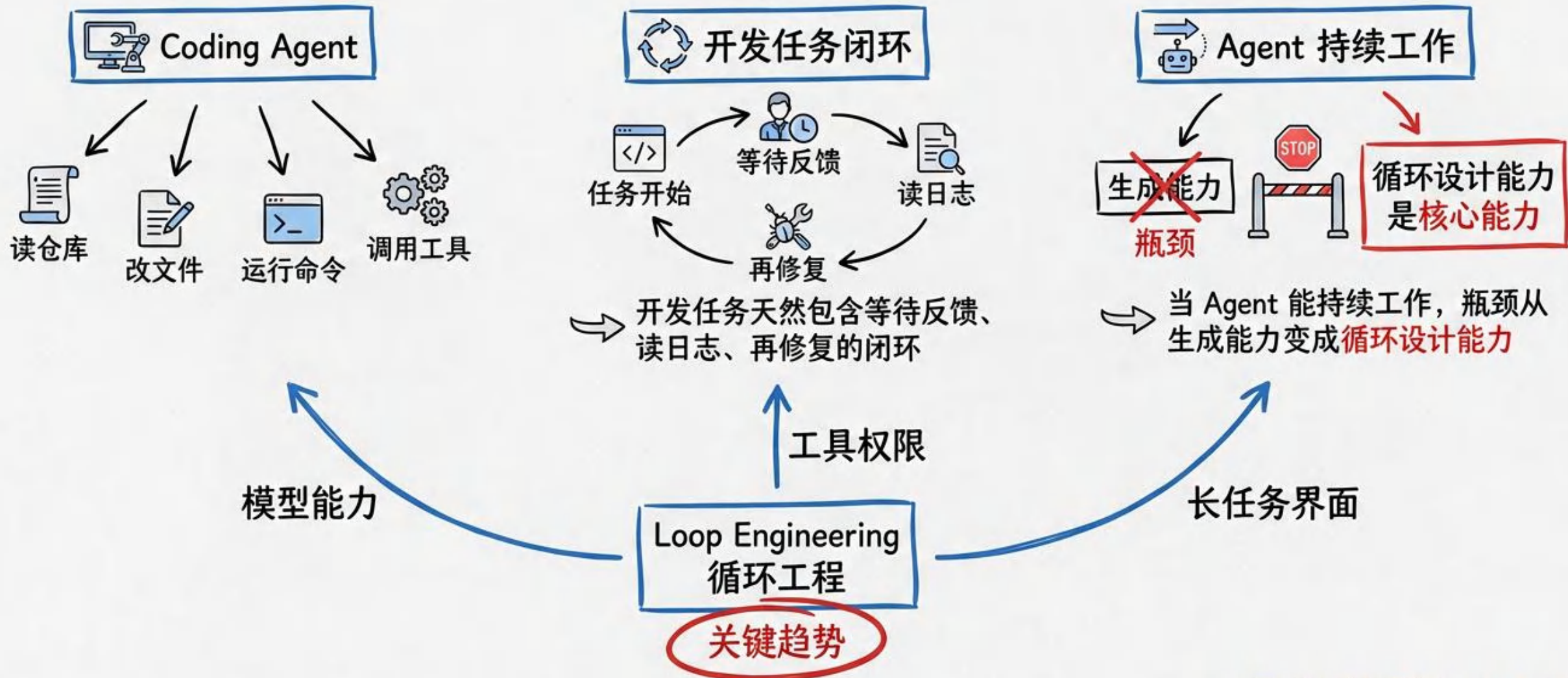
人负责目标、边界和验收

真正的效率来自更少轮
次得到可证明结果。



为什么现在出现循环工程

开场与核心判断 | 模型、工具、长任务界面同时成熟



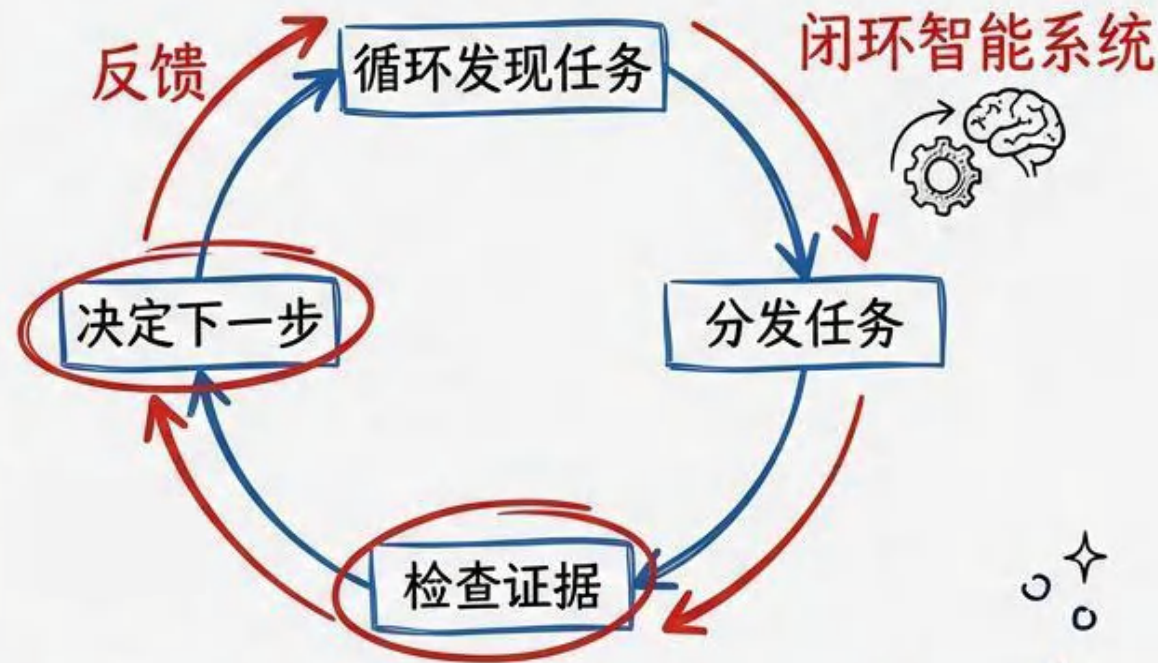
从提示词到循环

表达问题只是第一步，**经营反馈**才是关键。

旧模式



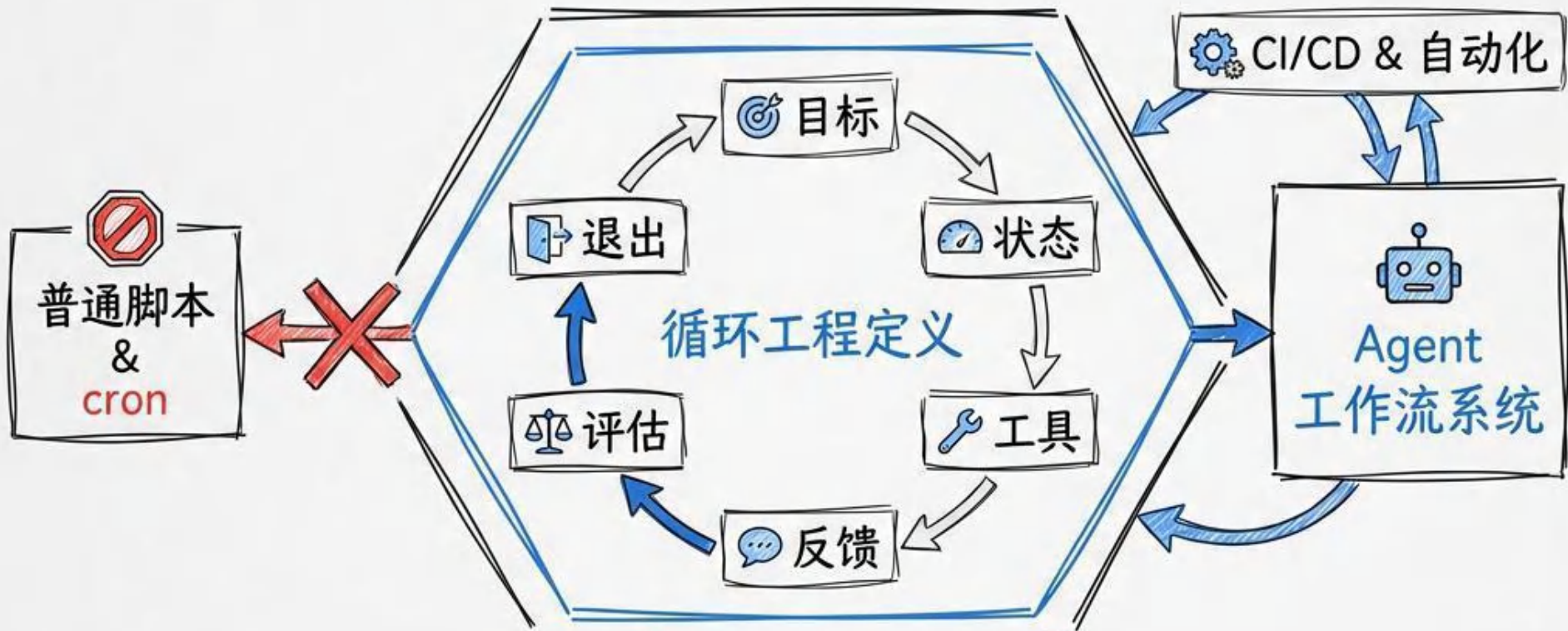
新模式



提示词没有消失，而是进入**技能**、**规格**、**评估**和**停止条件**。

研究对象与边界

开场与核心判断：本报告研究的是 Agent workflows 系统，不是普通脚本



适用于

- CI
- PR
- 依赖
- 文档
- 上线验证
- 缺陷回收
- 技术债治理

~~不是 无限 while~~
~~不是 普通 cron~~
~~不是 模型自我确认~~

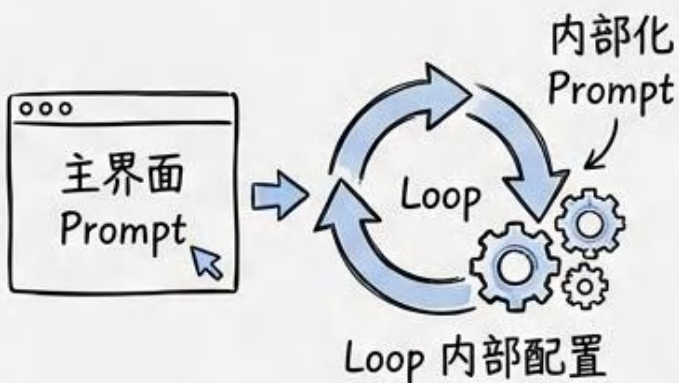
本报告的核心结论

开场与核心判断

循环工程是提示词工程的系统化

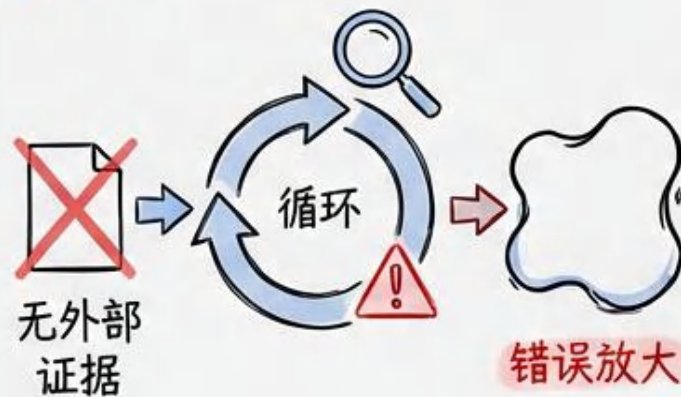
结论一

Prompt 从主界面变成 Loop 内部配置。



结论二

没有外部证据的循环会放大错误。



结论三

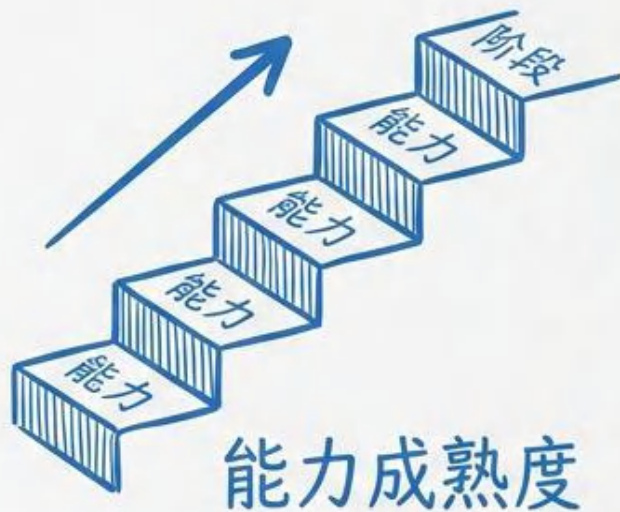
企业壁垒来自 Harness、评估、治理和成本控制。



不再采用日历式路线图

开场与核心判断

落地不应按时间强推，
而应按能力成熟度推进



先证明一条 Loop 能
安全完成小任务。



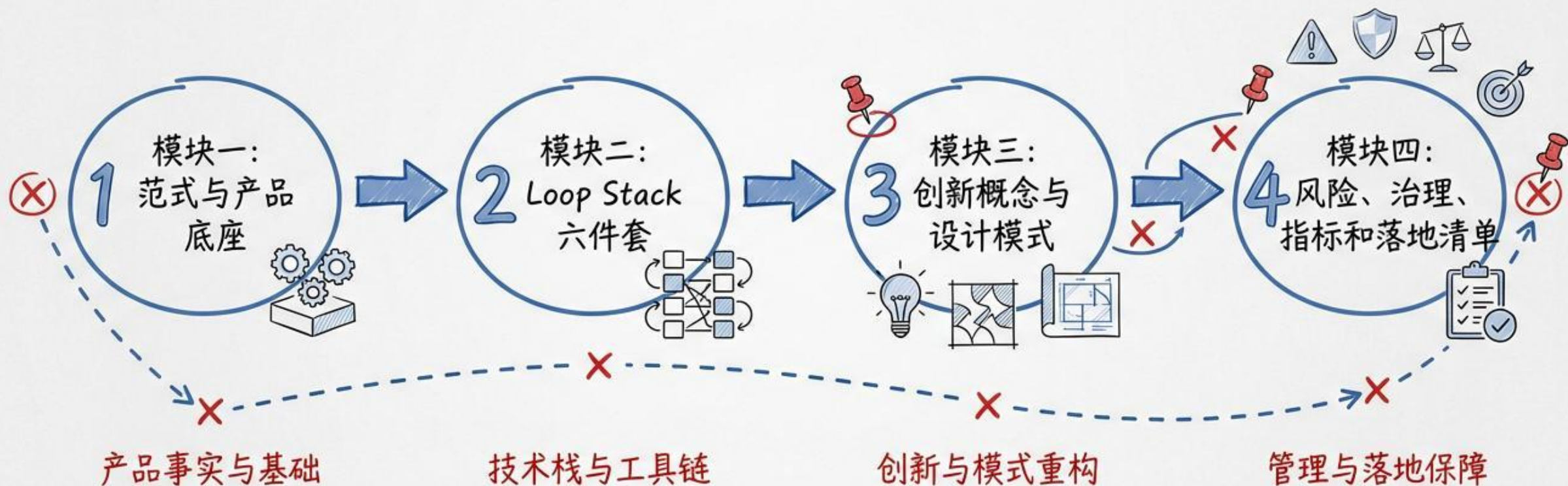
再扩展到多场景、多工
作树、多评估器。



最后形成可登记、可审计、
可记账的 Loop Fleet。

全报告结构

从产品事实到创新概念，再到企业治理



行业讨论触发点

范式背景

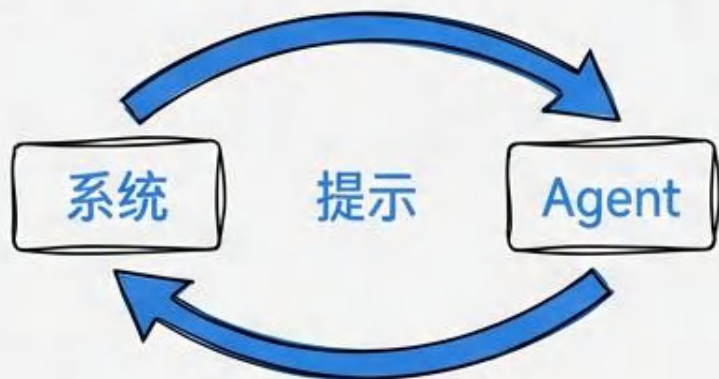
“写循环”成为 AI 编程的新抽象

旧范式



新变化概括

不是人逐轮提示 Agent，而是系统提示 Agent。



Loop Engineering

→ Addy Osmani 将这种做法命名为 Loop Engineering。

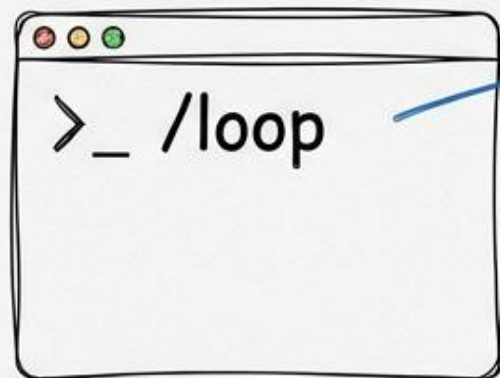


报告以官方产品能力和工程机制作为分析底座。

Claude Code: /loop

产品能力底座

会话内周期性重复运行任务



心跳型循环

官方文档说明 /loop 可在 Claude Code 会话内 **重复运行任务**。

适合部署 **轮询**、**PR 监控**、**长构建检查** 和 **提醒**

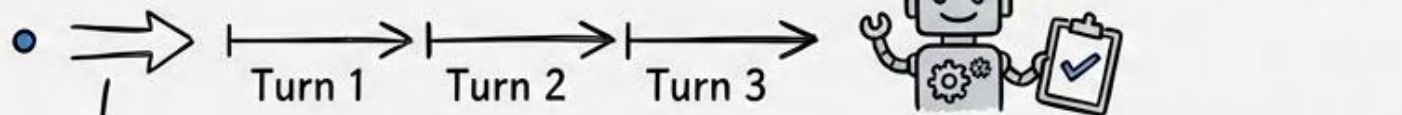
它是“**心跳型循环**”的产品化入口

Claude Code: /goal

从固定周期走向完成条件驱动

产品能力底座

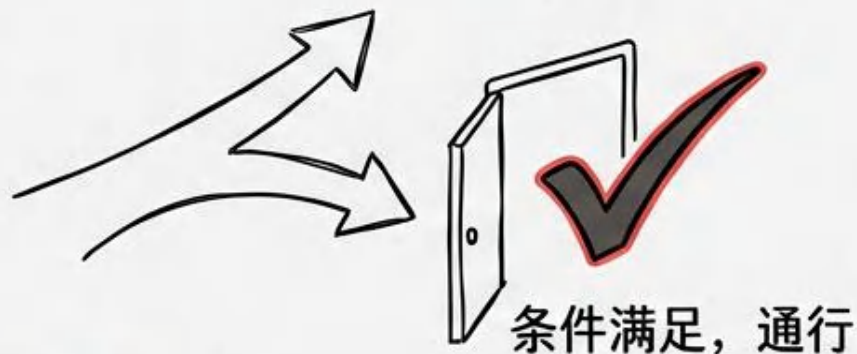
-  用可验证条件定义目标。



独立小模型判断条件是否满足

这体现了“写的人不该给自己打分”

红色“NO”门

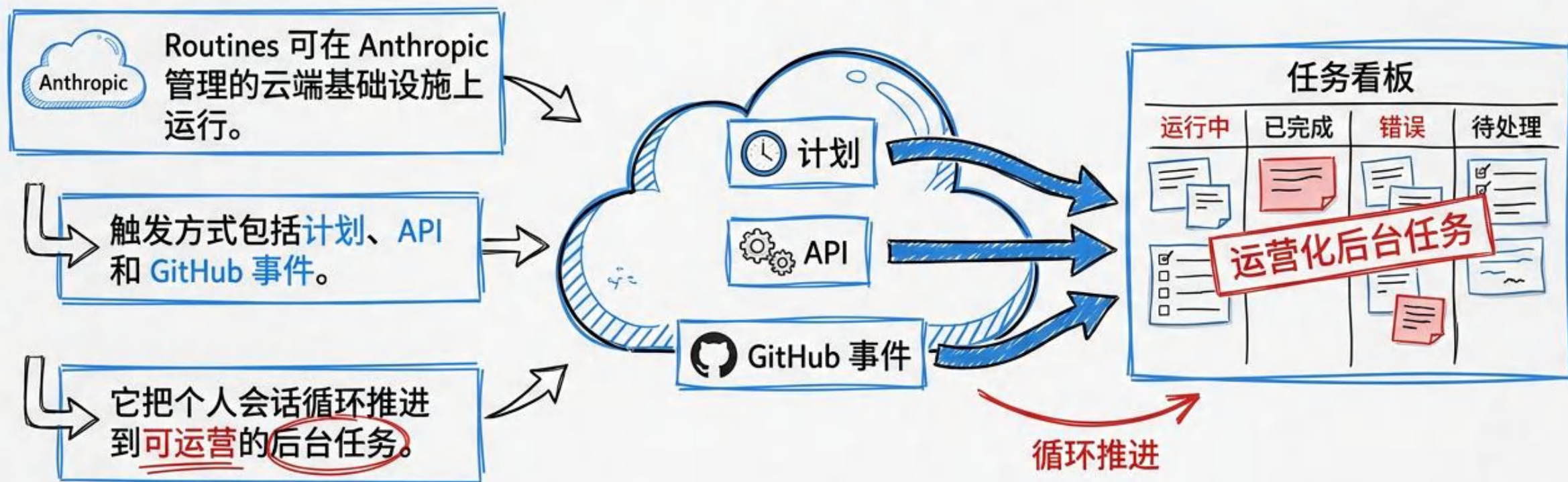


完成条件驱动，而非时间周期

Claude Code: Routines

产品能力底座

云端例行任务让循环脱离本地会话



Claude Code: Skills

产品能力底座

SKILL.md 让项目知识外化



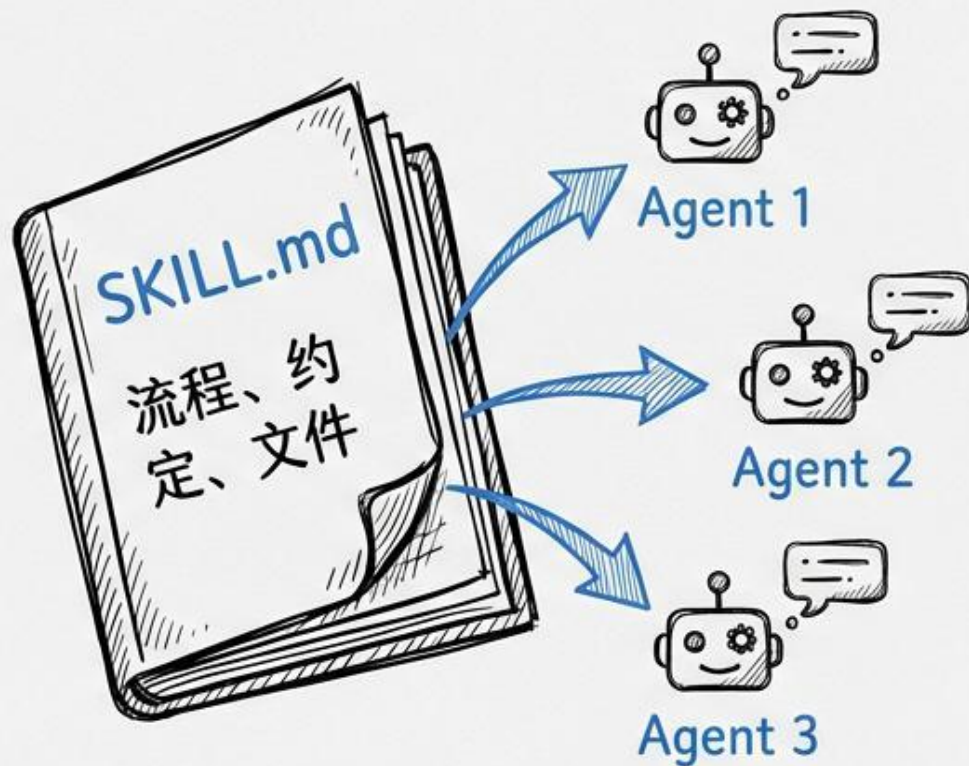
- Skills 用 SKILL.md 记录可复用流程、约定和支持文件。



- 只有相关时加载，减少重复粘贴长说明的成本。



- 没有 Skills，循环每轮都在重新猜你的项目。



Claude Code : Worktrees

产品能力底座

并行 Agent 需要隔离工作区

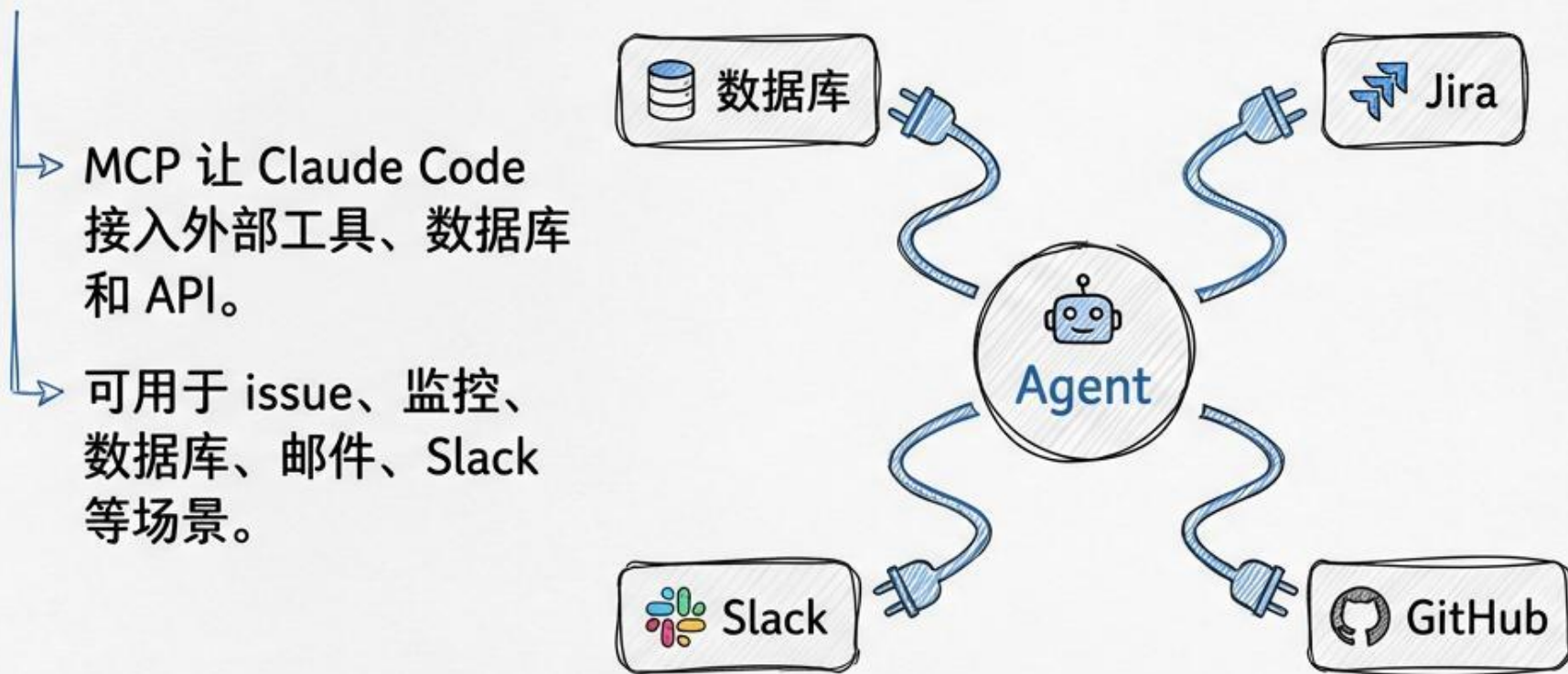


<https://code.claude.com/docs/en/worktrees>

Claude Code: MCP

产品能力底座

连接真实工具后，Loop 才能进入真实 workflow



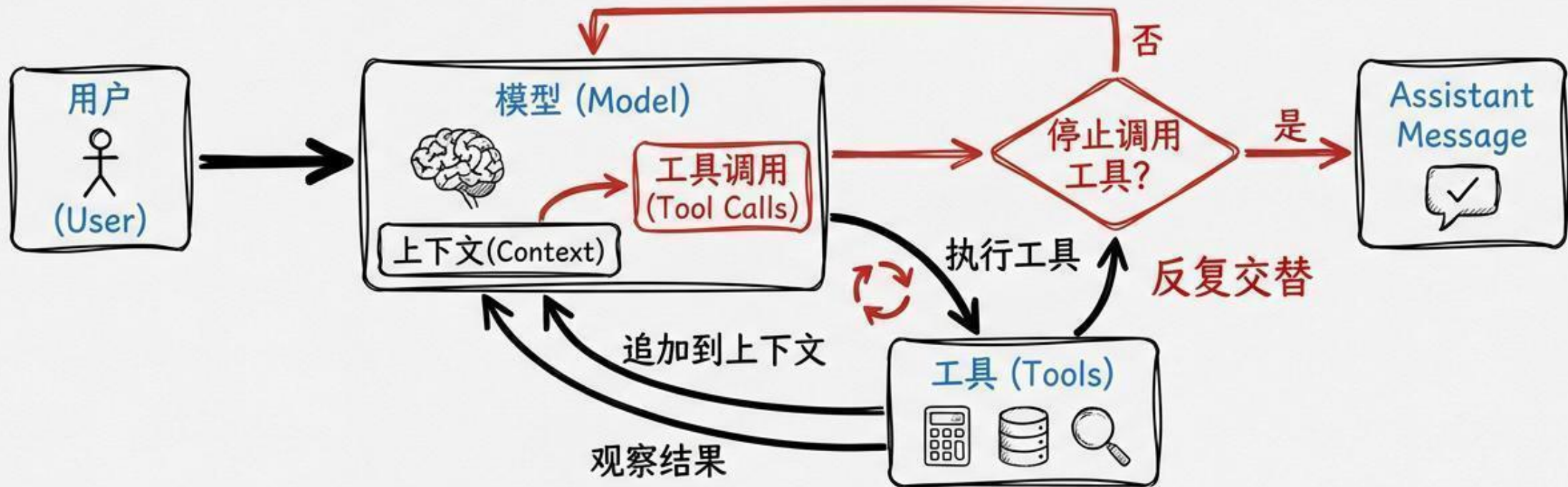
连接器扩大能力，
也扩大“权限”和
“安全边界”。



Codex: Agent Loop

产品能力底座

模型、工具、上下文反复交替，直到返回结果



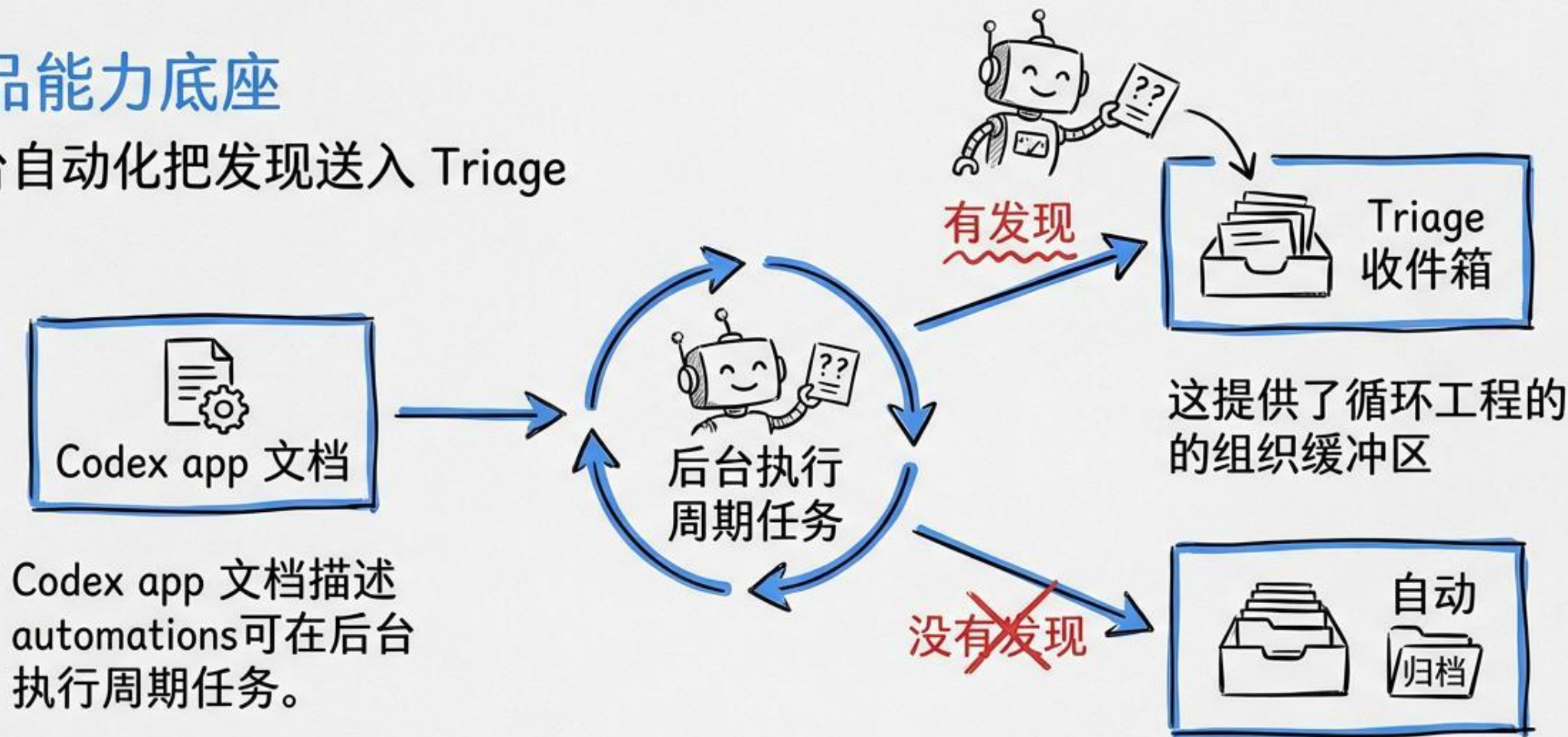
- ★ OpenAI 将 Codex 的底层机制称为 agent loop
- ★ 模型可返回工具调用，系统执行工具并把结果追加到上下文
- ★ 循环直到模型 **停止调用工具并返回 assistant message**

<https://openai.com/index/unrolling-the-codex-agent-loop/>

Codex: Automations

产品能力底座

后台自动化把发现送入 Triage



* 架构注源:

<https://developers.openai.com/codex/app/>

更多免费行业报告

并购家

www.ipoiipo.cn

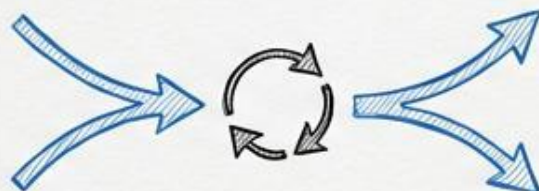
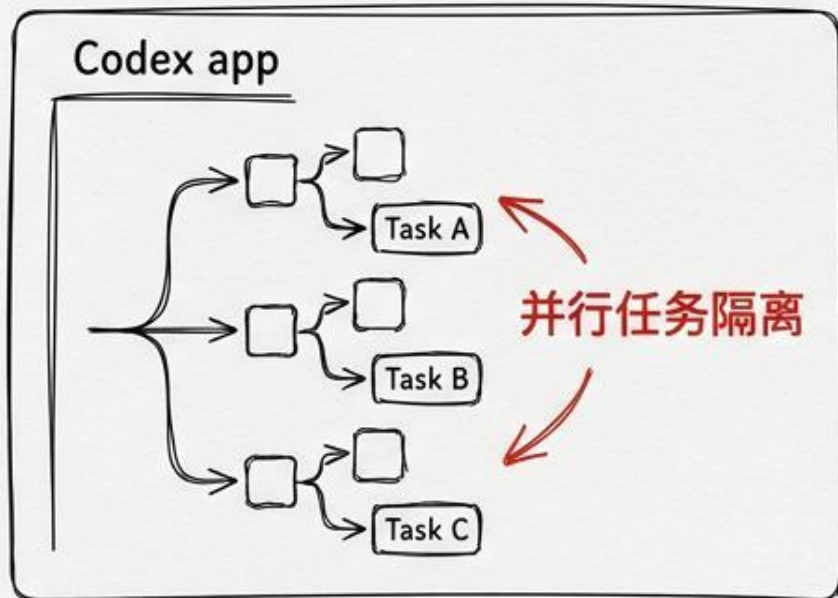
@ 清新研究团队 | 2026年6月

Codex: Worktrees 与 Skills

产品能力底座

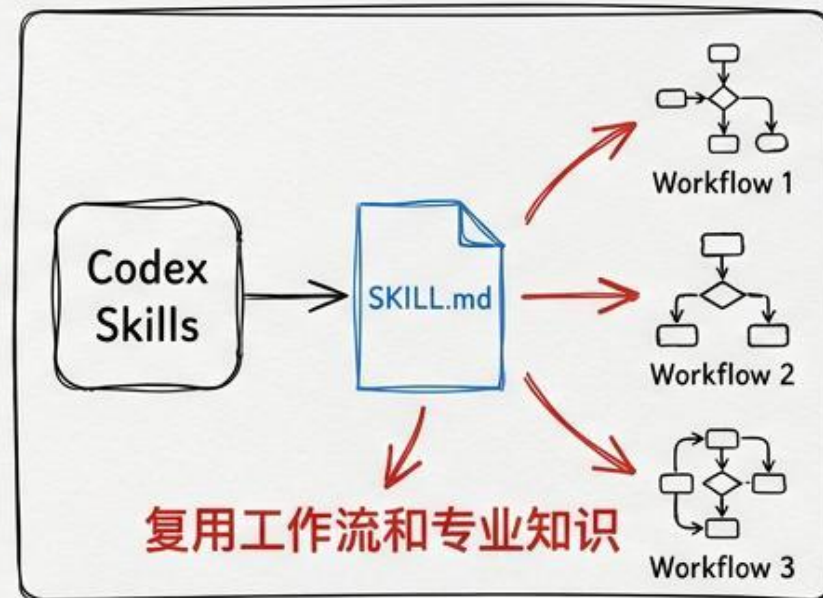
Codex 也在补齐 Loop Stack

Worktree 隔离



两侧产品正在收敛
到相似的循环组件

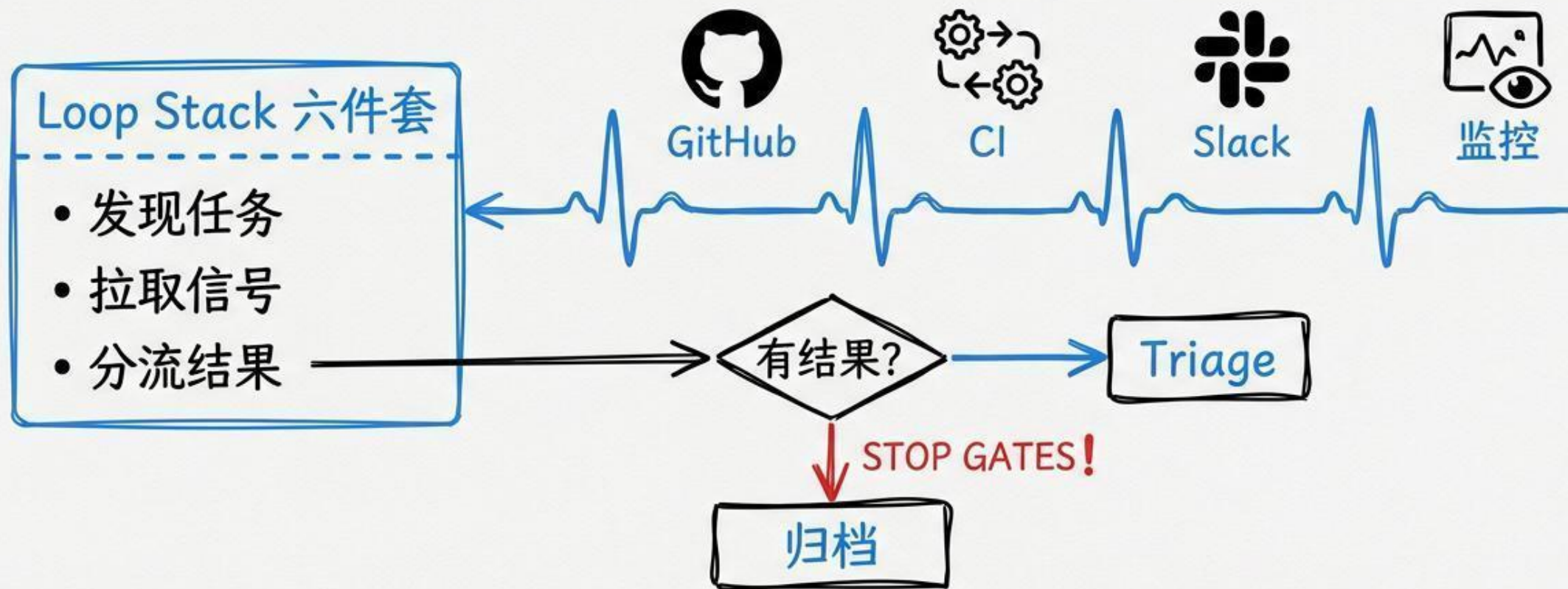
Skills 复用



<https://developers.openai.com/codex/app/worktrees>; <https://developers.openai.com/codex/skills>

Automations 是心跳

它负责定时、事件或目标驱动地唤醒 Agent



好的心跳不只是频率，而是知道何时不必打扰

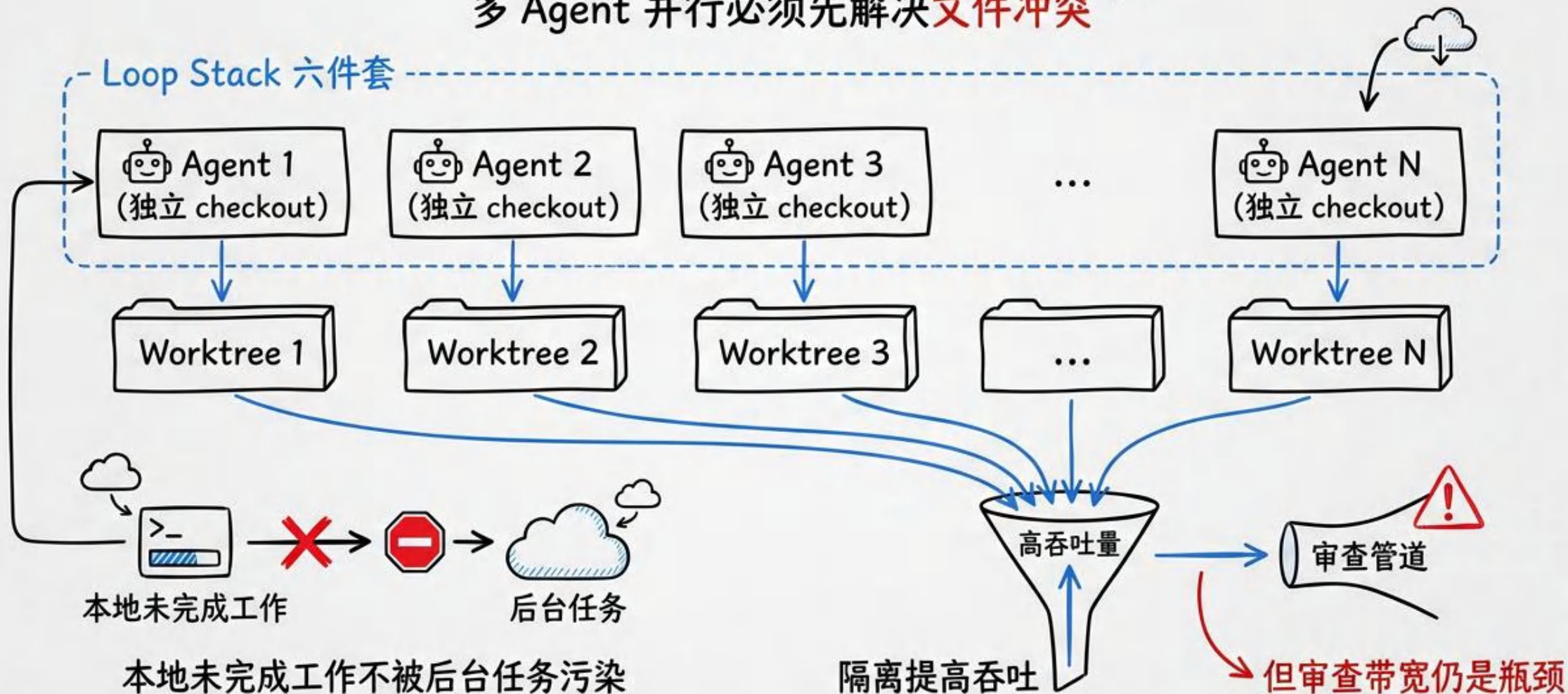
Triage 是收件箱

自动化结果先进入组织决策缓冲区

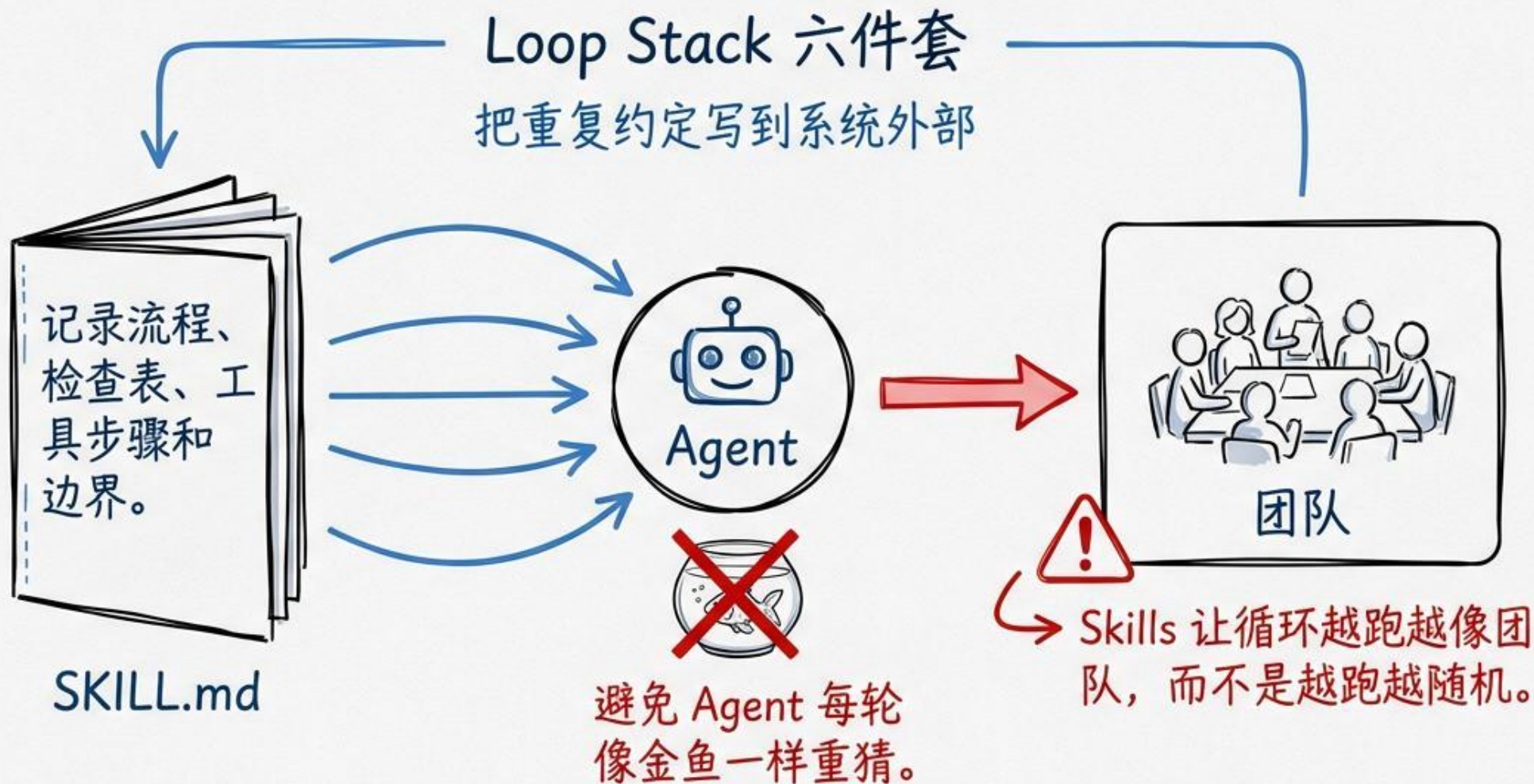


Worktrees 是隔离层

Loop Stack 六件套
多 Agent 并行必须先解决文件冲突



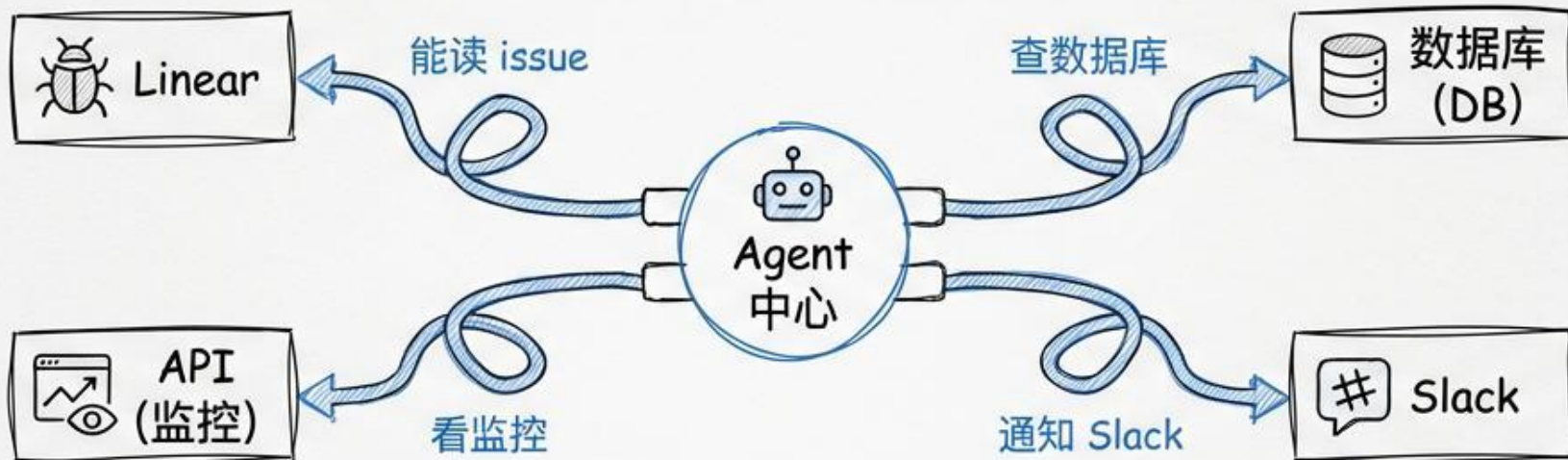
Skills 是项目知识



Connectors 是外部触手

Loop Stack 六件套

MCP/插件让循环触达真实业务系统

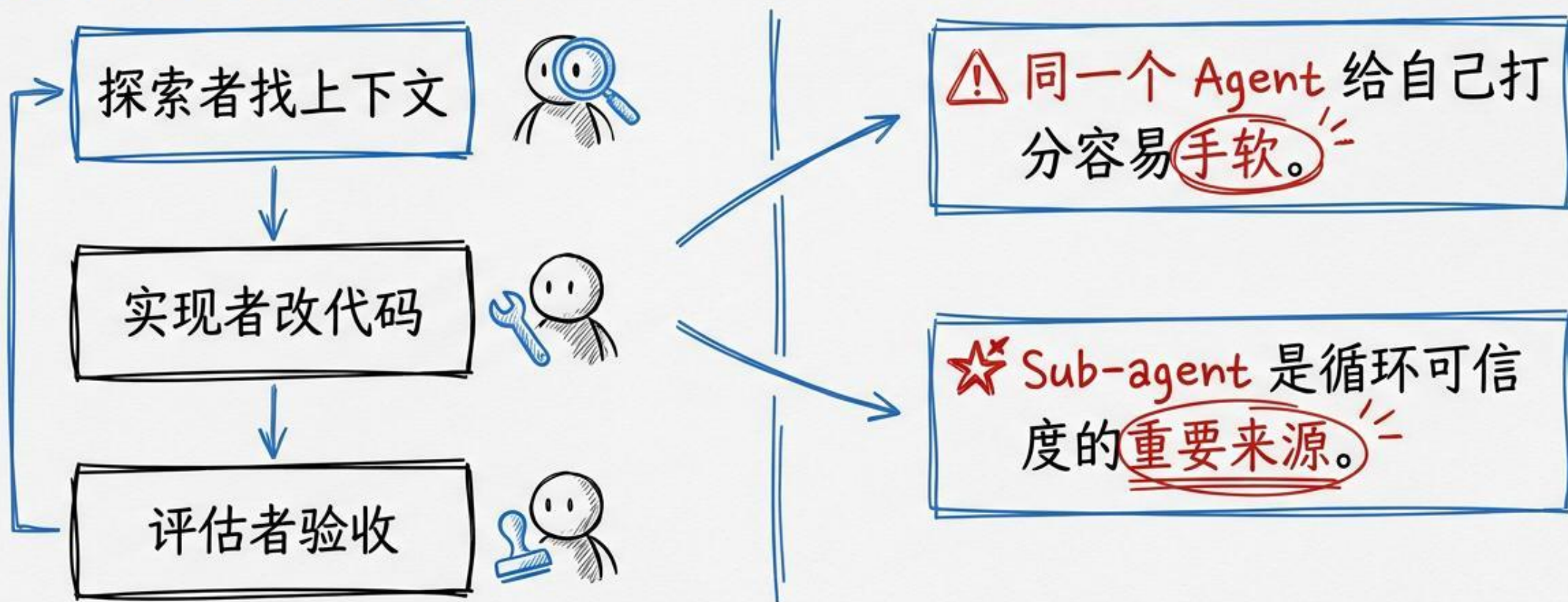


连接器越多，**权限治理**越重要。

Sub-agents 是职责分离

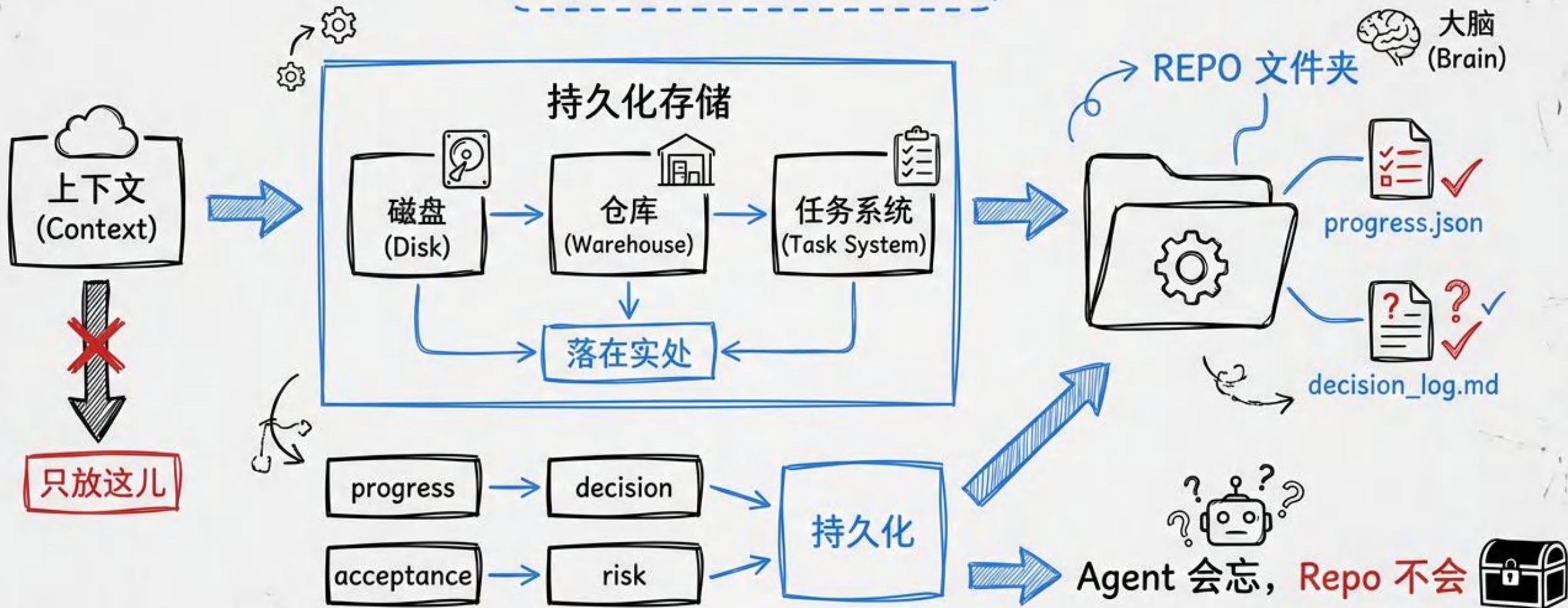
Loop Stack 六件套

写的人和检查的人必须分开



Memory 是第六件

Loop Stack 六件套
最不起眼，却是长任务的脊柱



六件套如何合成一条 Loop

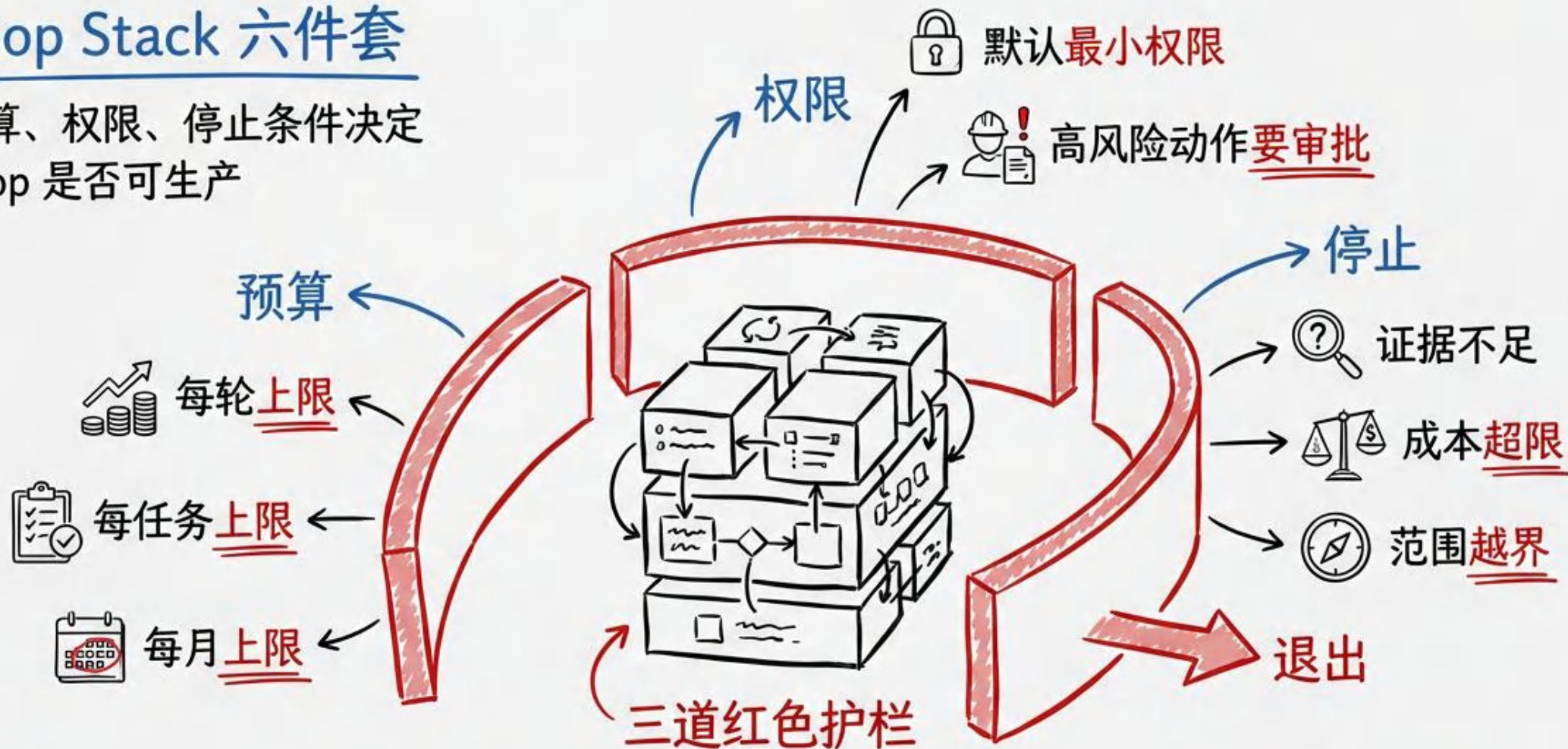
从心跳到记忆，形成可持续作业系统



五件零件不够，还要三条底线

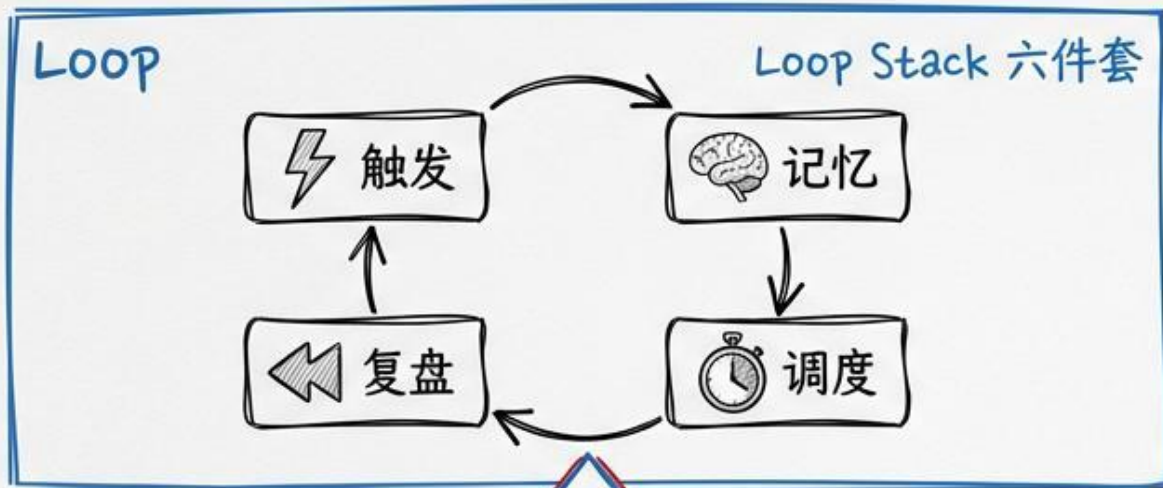
Loop Stack 六件套

预算、权限、停止条件决定
Loop 是否可生产

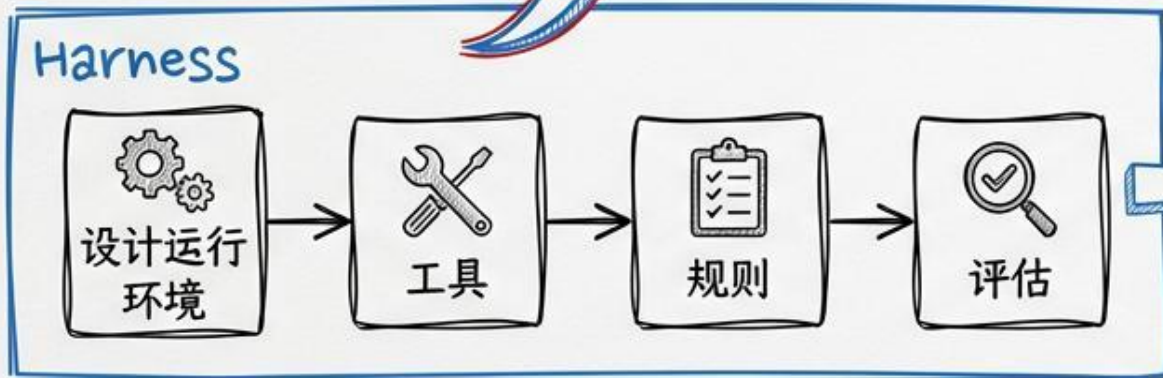


Loop 与 Harness 的关系

Harness 管单个 Agent, Loop 管持续作业系统



构建于



企业竞争力来自两者组合

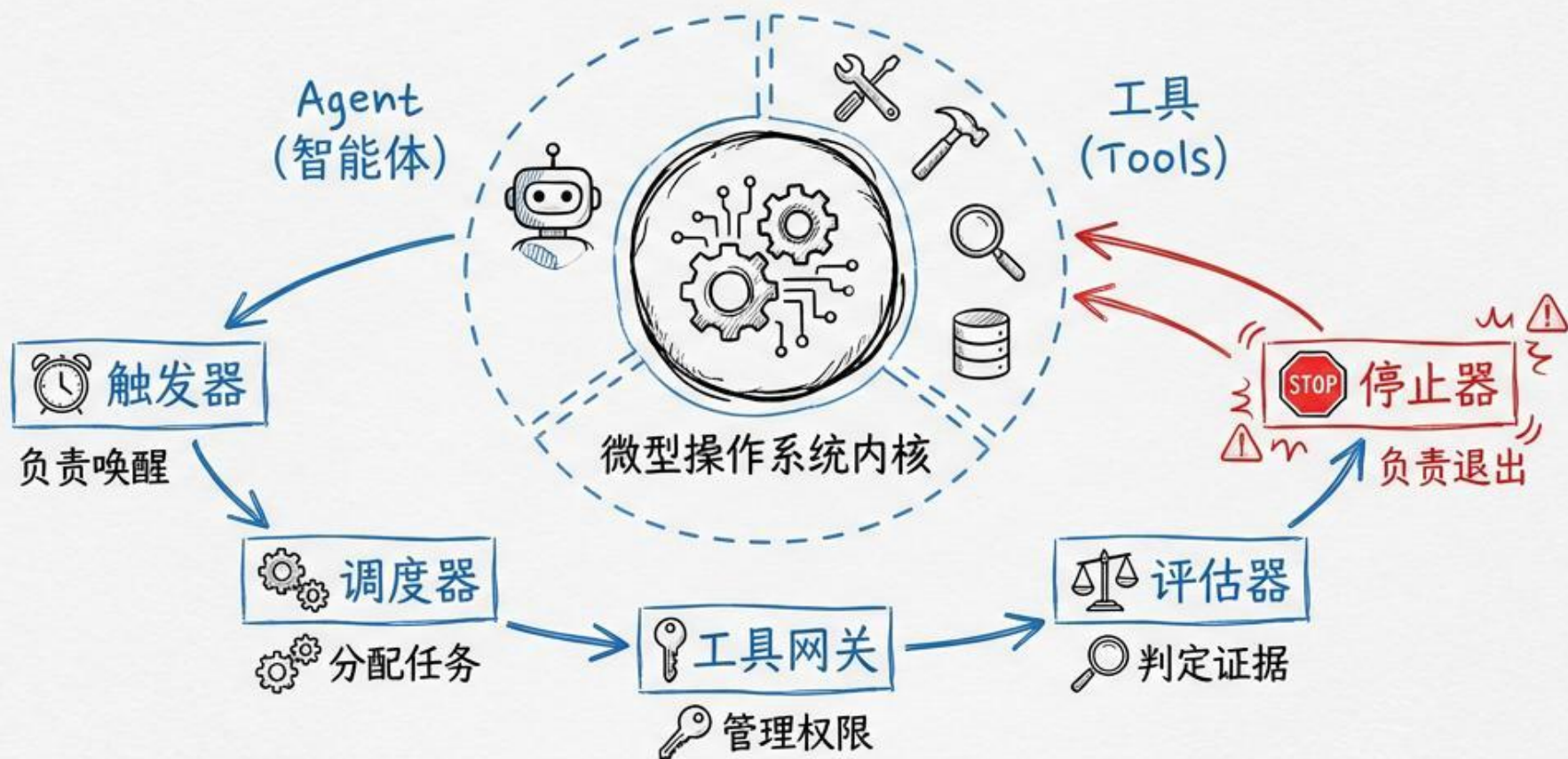


警告：不正确的组合可能导致系统故障

创新概念: Loop Kernel

创新概念与设计模式

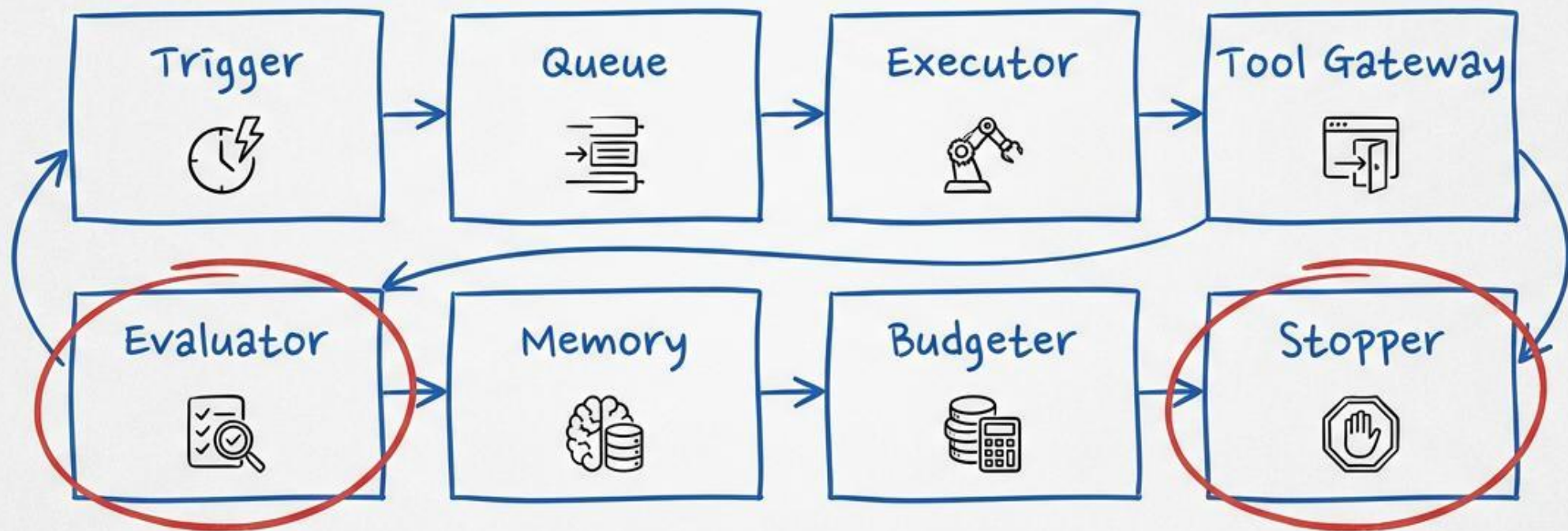
循环像一个微型操作系统



Loop Kernel 的八个模块

创新概念与设计模式

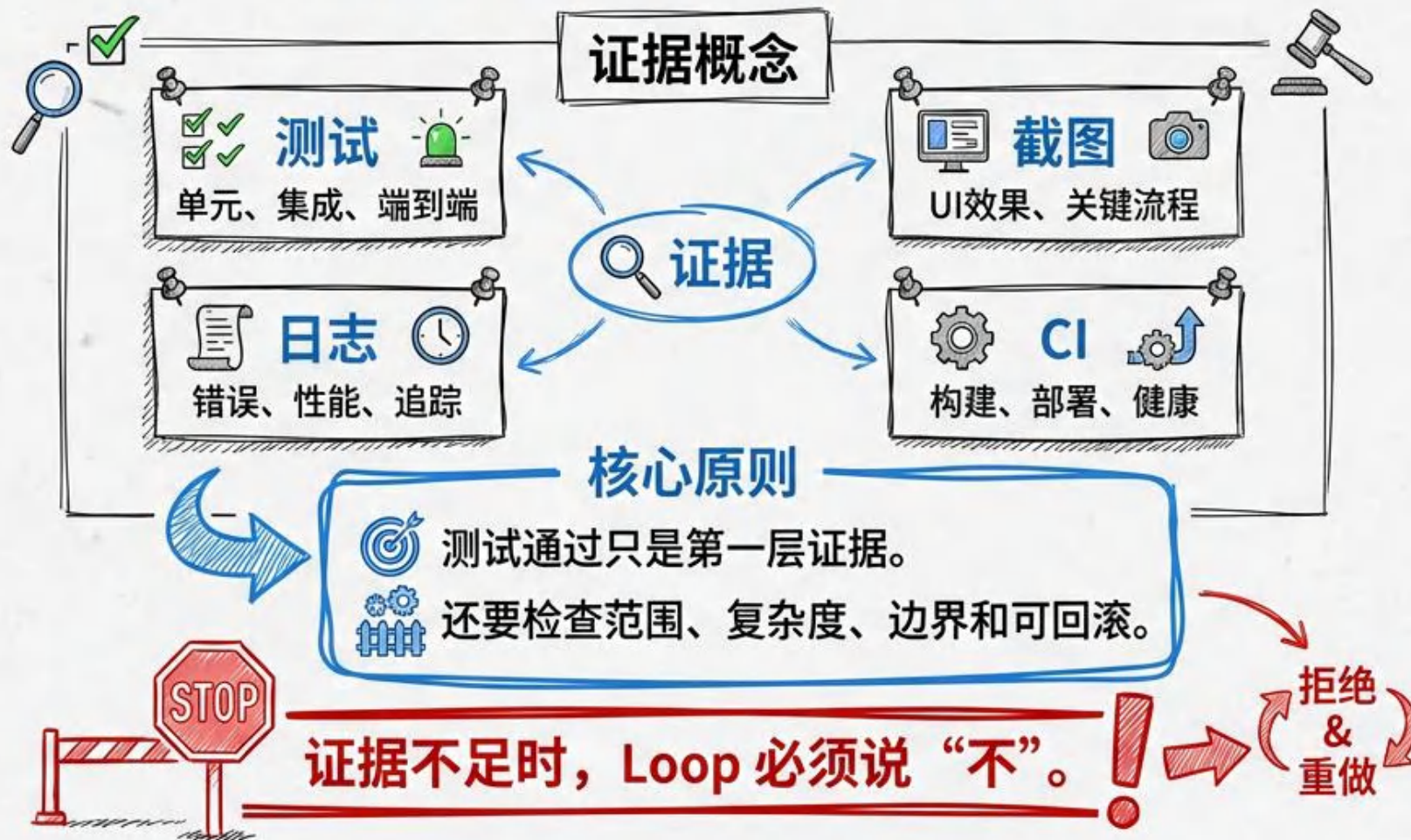
把神奇自动化拆成可审计组件



每个模块都有失败模式和负责人

创新概念：Proof-of-Done

完成是证据，不是声明



Proof-of-Done 契约模板

创新概念与设计模式

每条 Loop 启动前先写清楚验收



创新概念：Repo Memory

创新概念与设计模式

Agent 会忘，Repo 不会

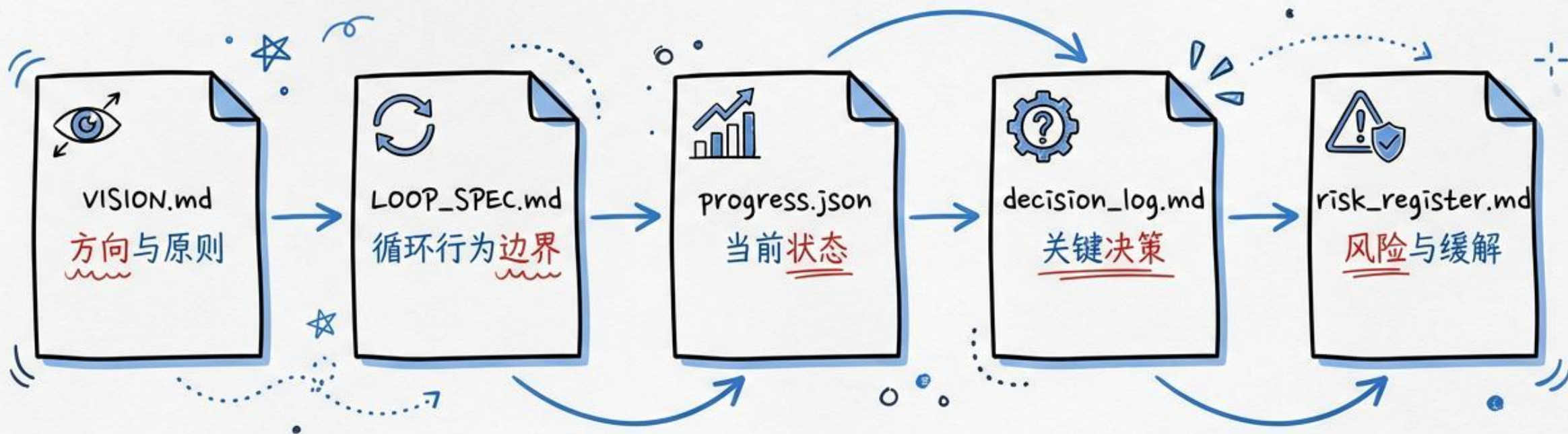
- 状态不应只留在 context。✗
- 长期记忆写入仓库、看板或数据库。
- 循环每轮读取记忆，再更新下一步。



Repo Memory 工件清单

创新概念与设计模式

让循环跨会话、跨 Agent、跨团队延续



@ 清新研究团队 | 2026年6月

更多免费行业报告

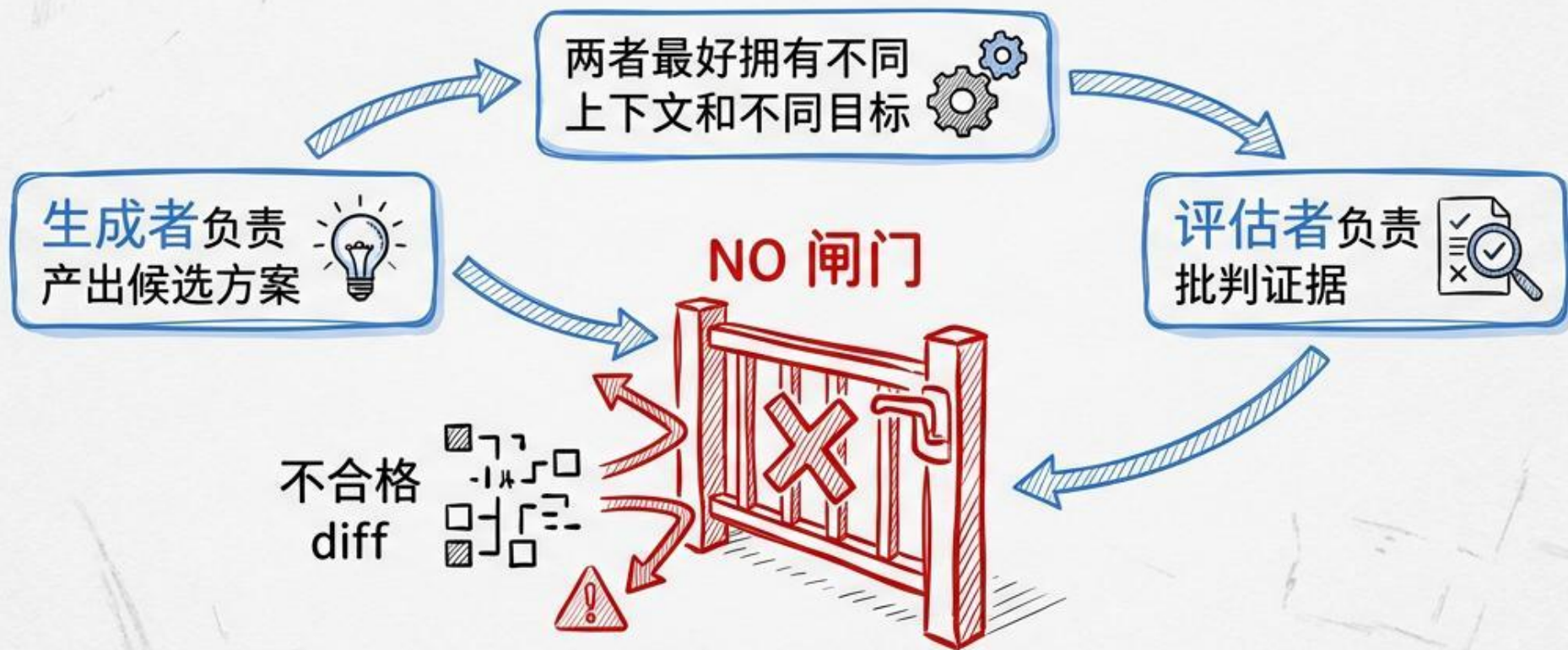
并购家

www.ipoipo.cn

创新概念：No-Gate Evaluator

创新概念与设计模式

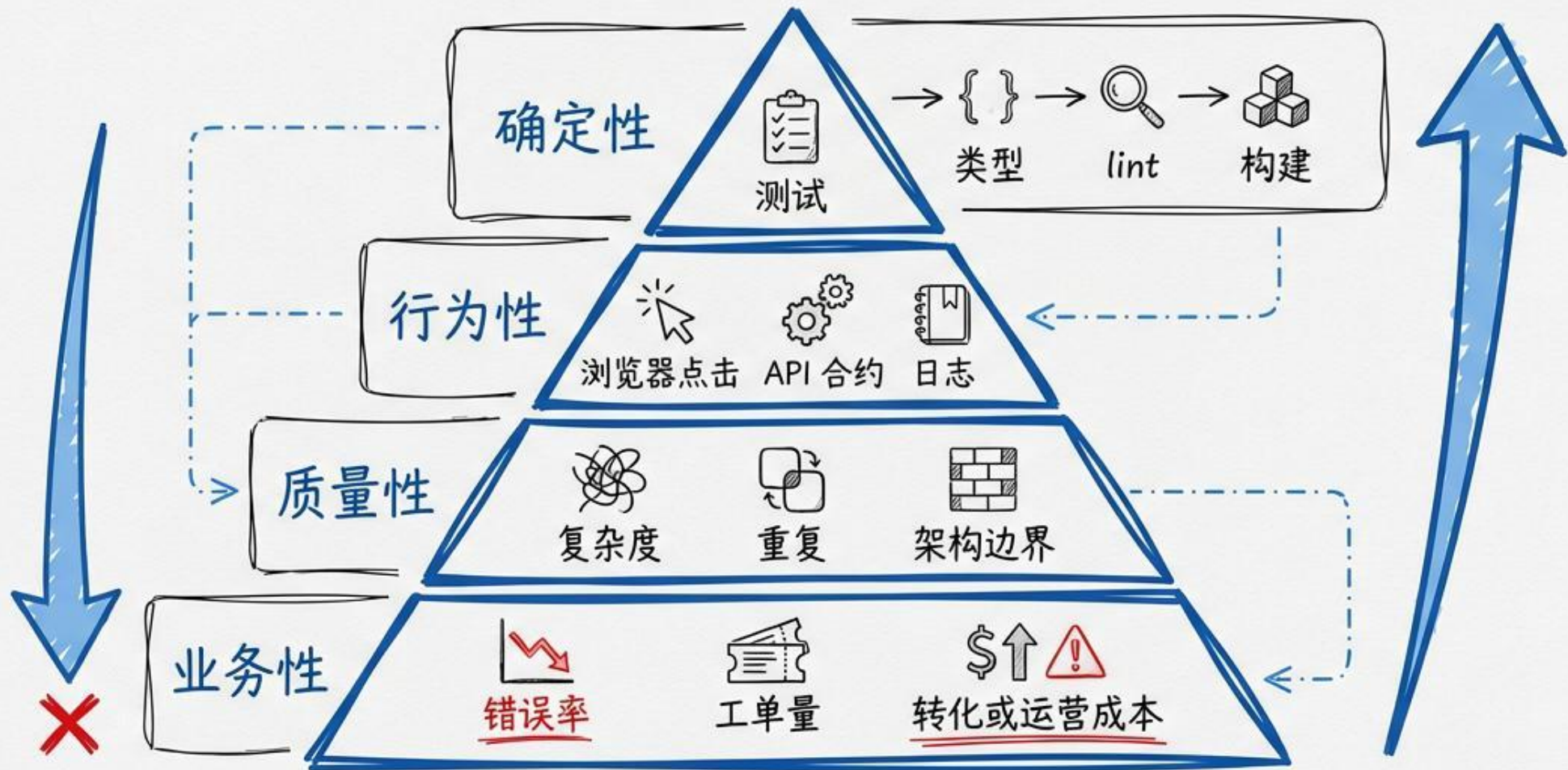
循环必须有能说“不”的角色



No-Gate 的四类证据

创新概念与设计模式

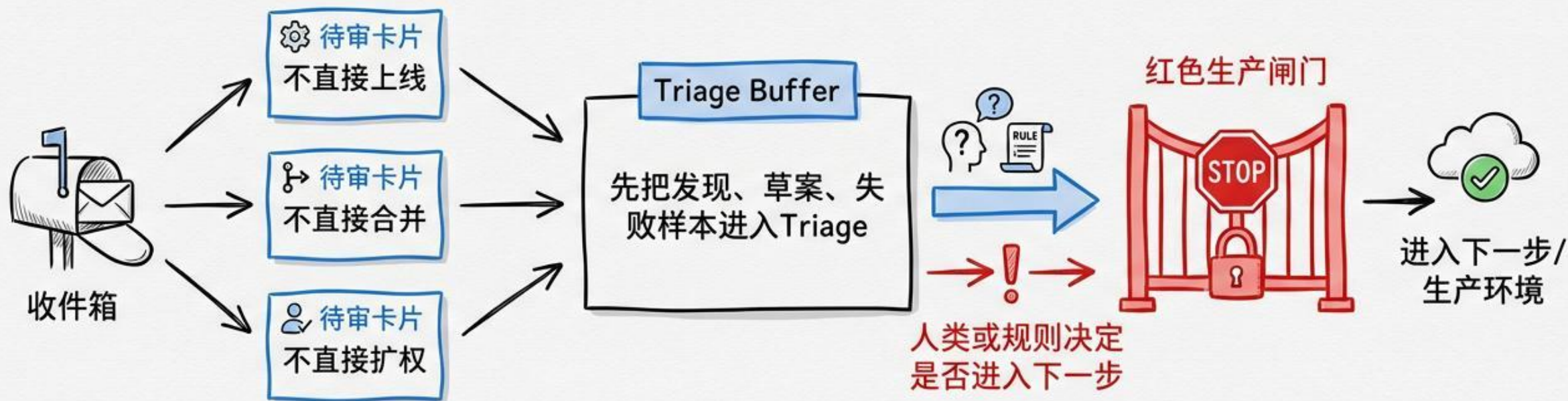
不要只看模型解释，要看外部事实



创新概念：Triage Buffer

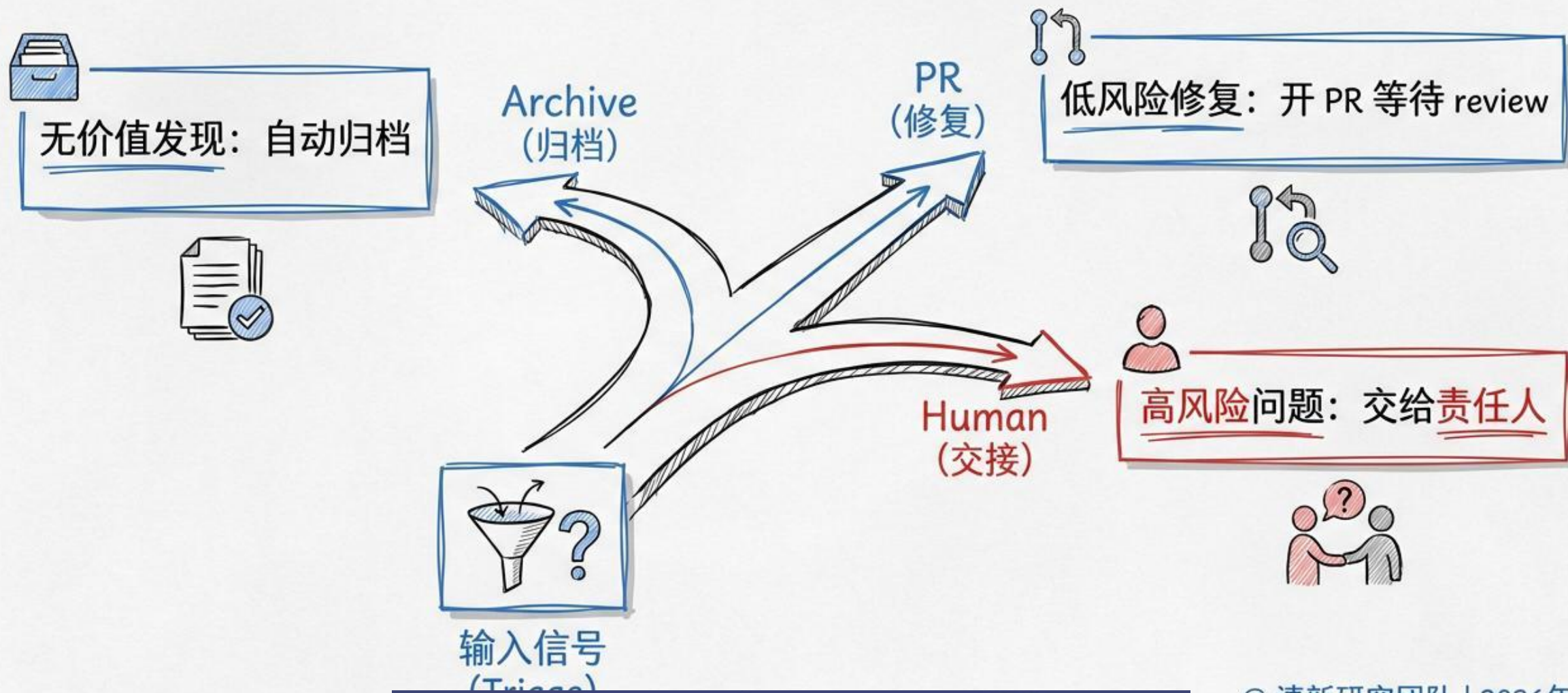
创新概念与设计模式

把后台发现变成可管理收件箱



Triage 的三种输出

创新概念与设计模式：归档、修复、交接



创新概念：Loop Ledger

创新概念与设计模式

循环要像服务一样记账

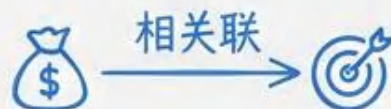
输入源：记录每轮 token、
工具调用和文件变更。

账本表格

项目	记录值	状态/备注
每轮 Token	💰 1500 / 3000 (Max)	✅ 正常
工具调用	⚙️ 5 / 10 (Max)	✅ 正常
文件变更	📄 2 文件, +50 行	✅ 已记录
⚠️ 测试证据 & 失败原因	❌ 失败：逻辑错误	🛑 需人工批准
⚠️ 成本	💵 \$1.20 (超预算!)	红色警示, 暂停
可接受结果	🎯 达到 (是/否)	❓ 待确认

记录过程：记录测试证据、失败原因和人工批准。

关键链接：记录成本与可接受结果的关系

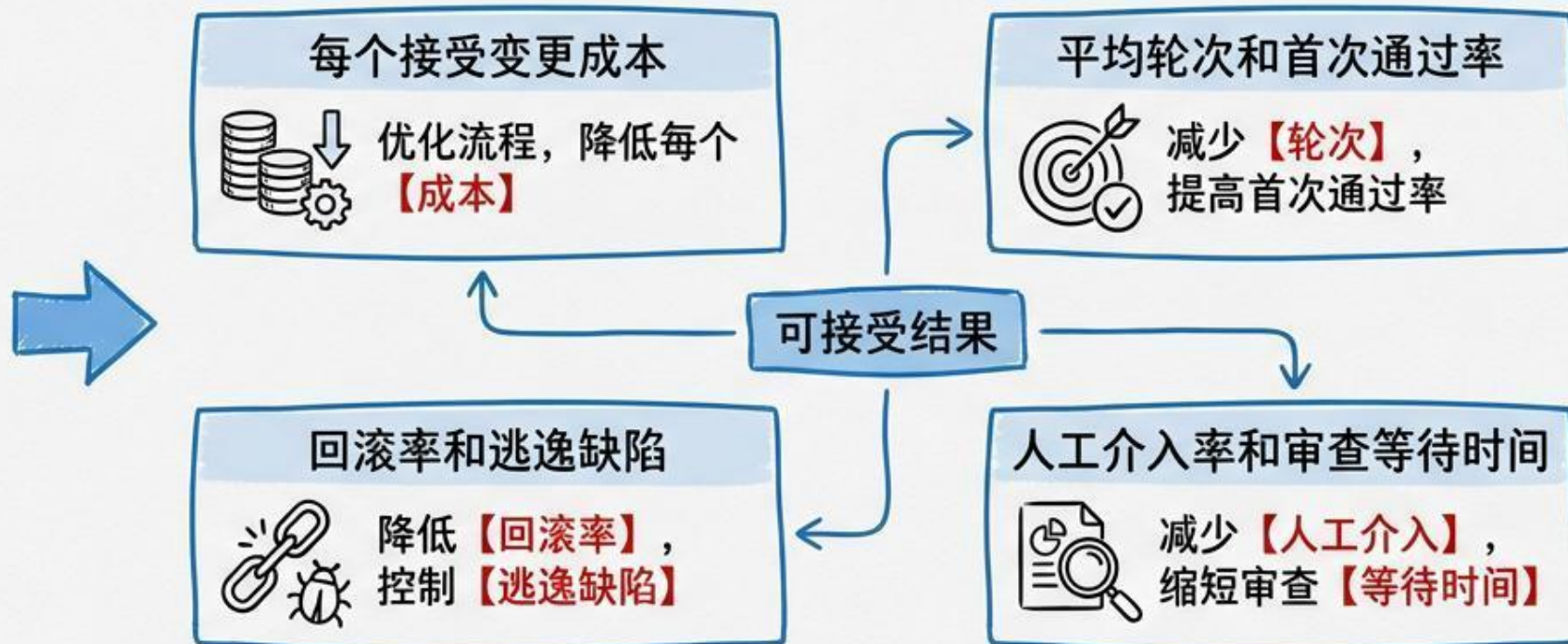


⚠️ 红色超预算警示

Loop Ledger 指标

创新概念与设计模式

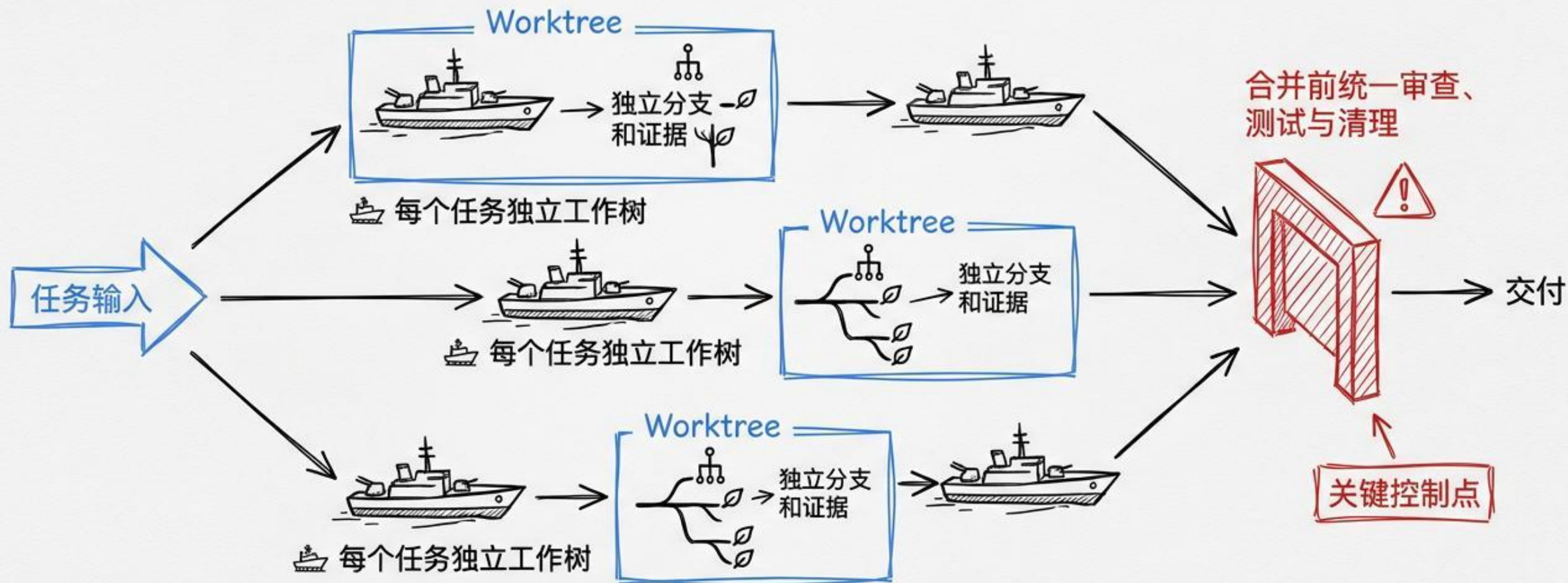
不要奖励运行时长，要奖励可接受结果。



创新概念：Worktree Fleet

创新概念与设计模式

并行不是开更多窗口，而是管理舰队



创新概念：Entropy Janitor

循环不只生成代码，也要删除复杂度

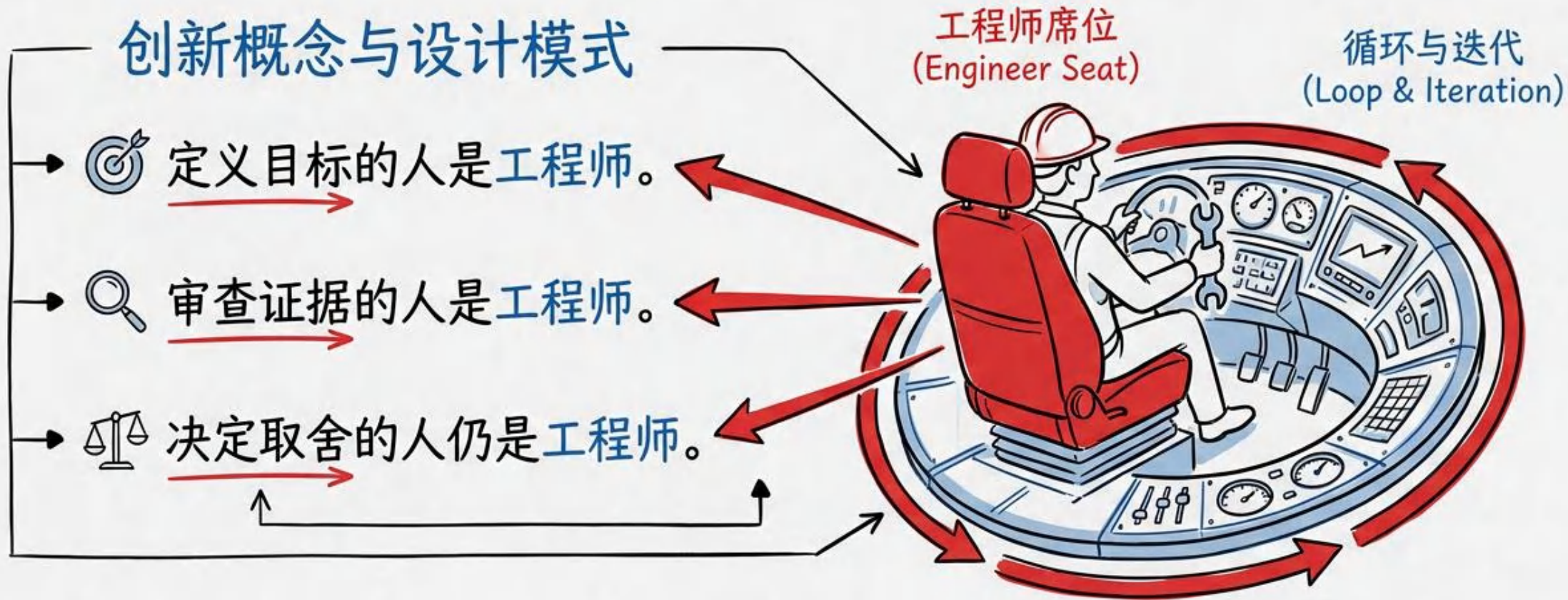
创新概念与设计模式



降熵 Loop 是规模化的必要配套。

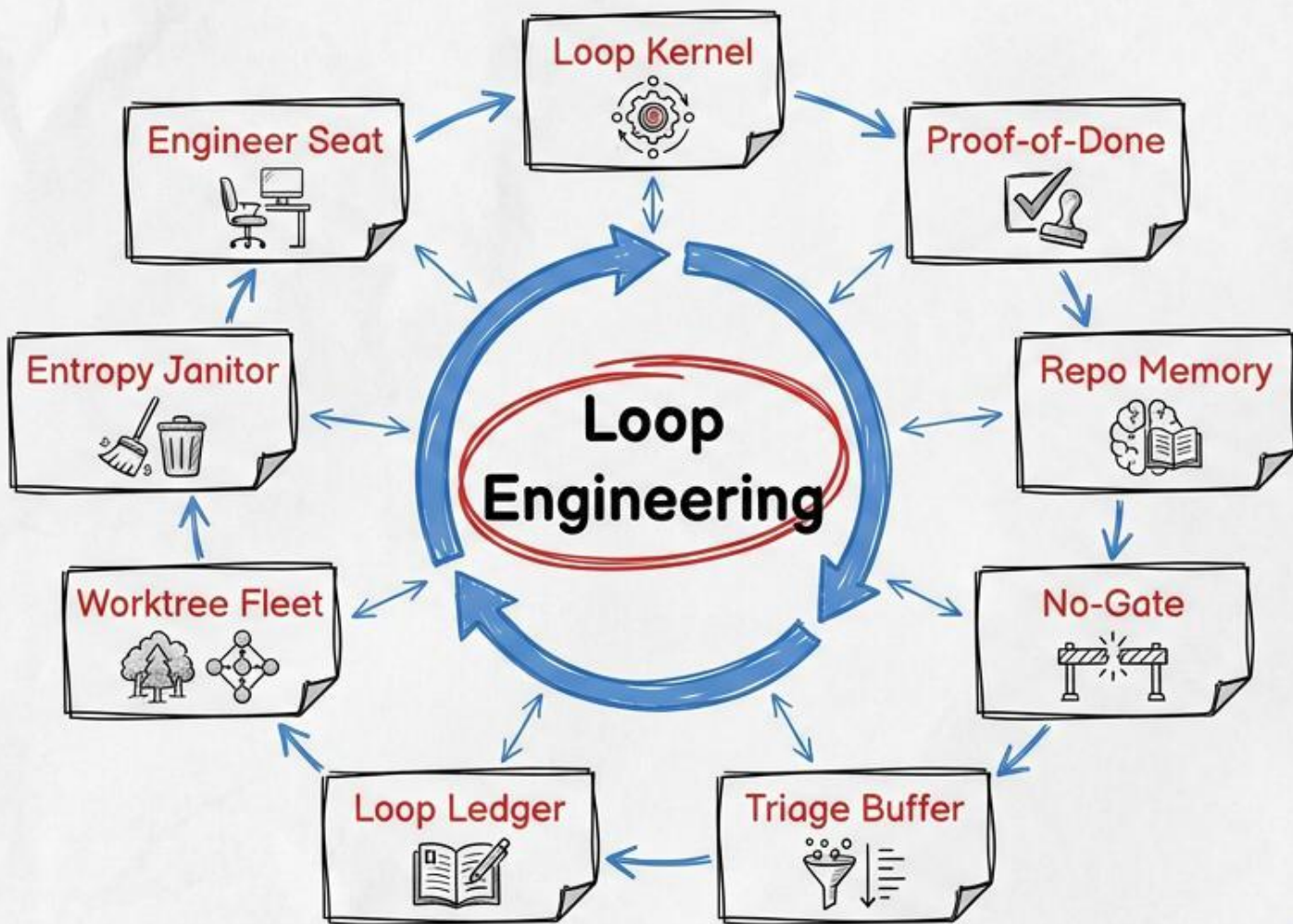
创新概念：Engineer Seat

人必须保留工程师席位



创新概念汇总页

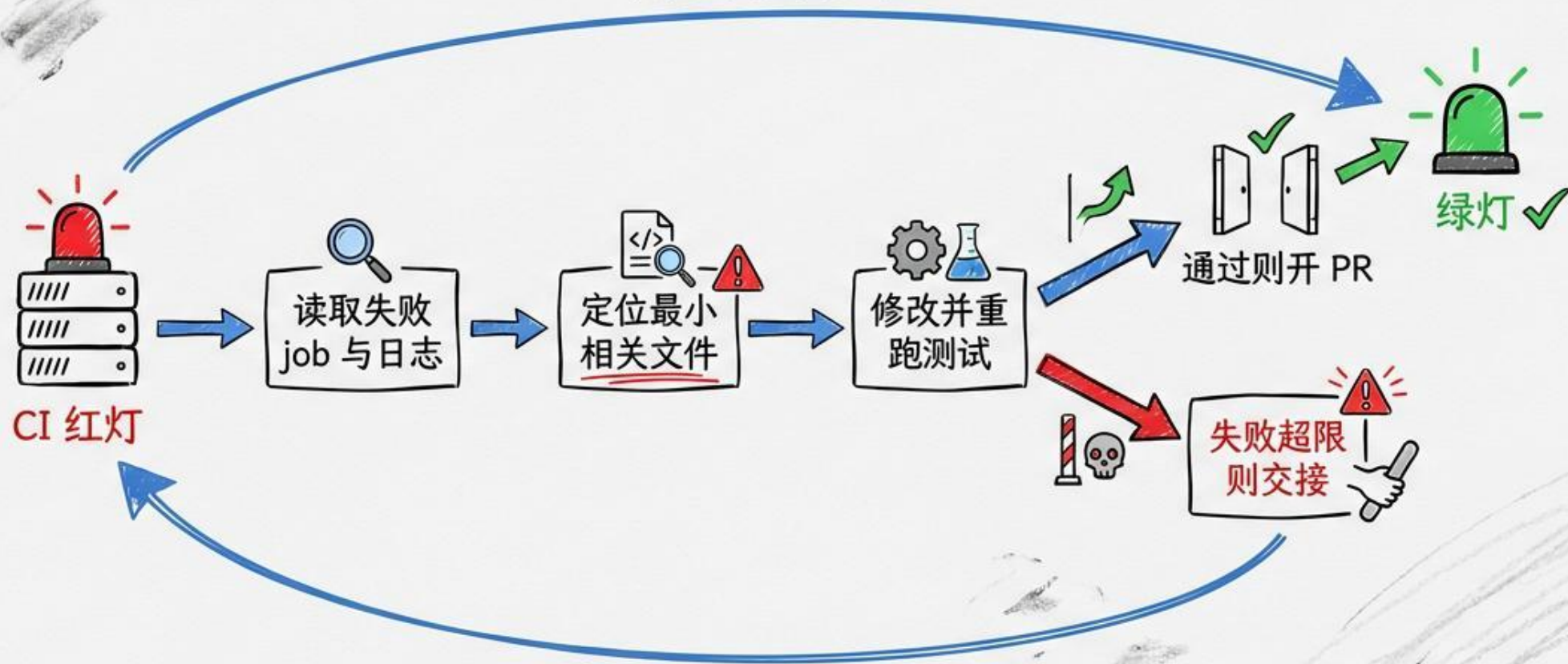
从六件套到九个原创概念



模式一：CI 失败修复 Loop

典型场景与工程模式

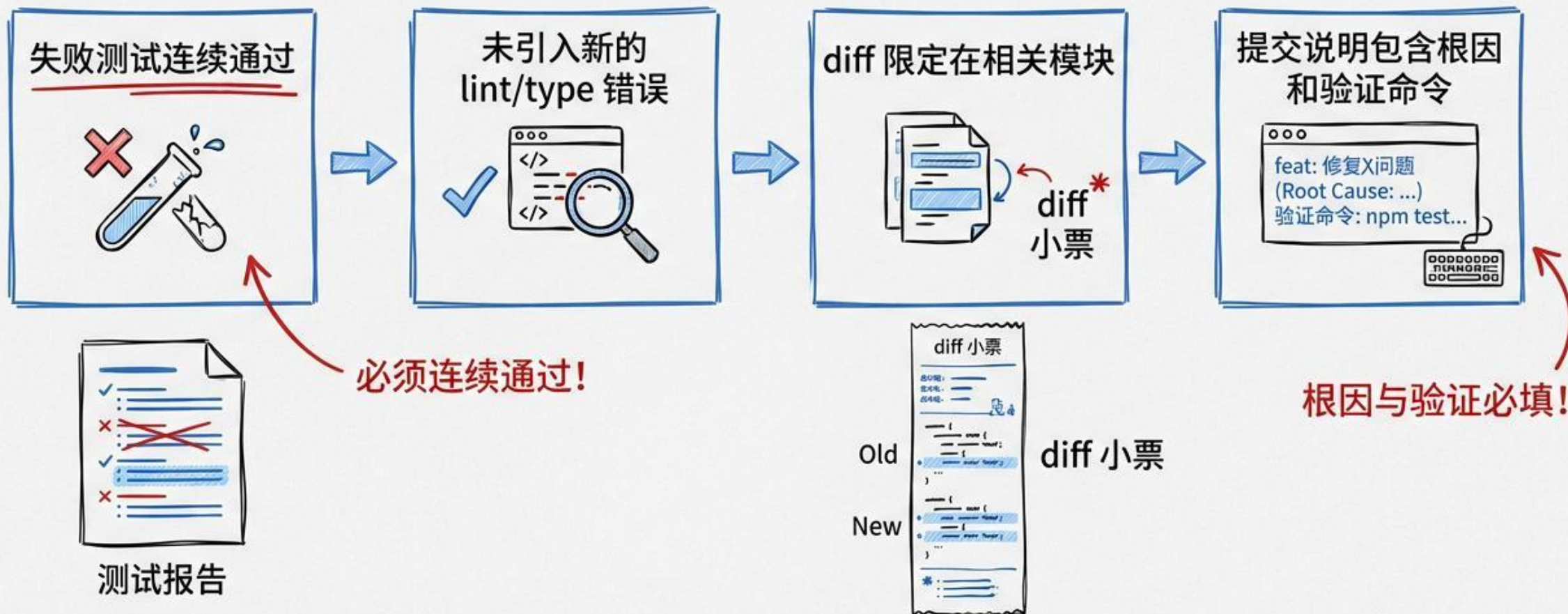
最稳的首条循环



CI Loop 的完成证据

典型场景与工程模式

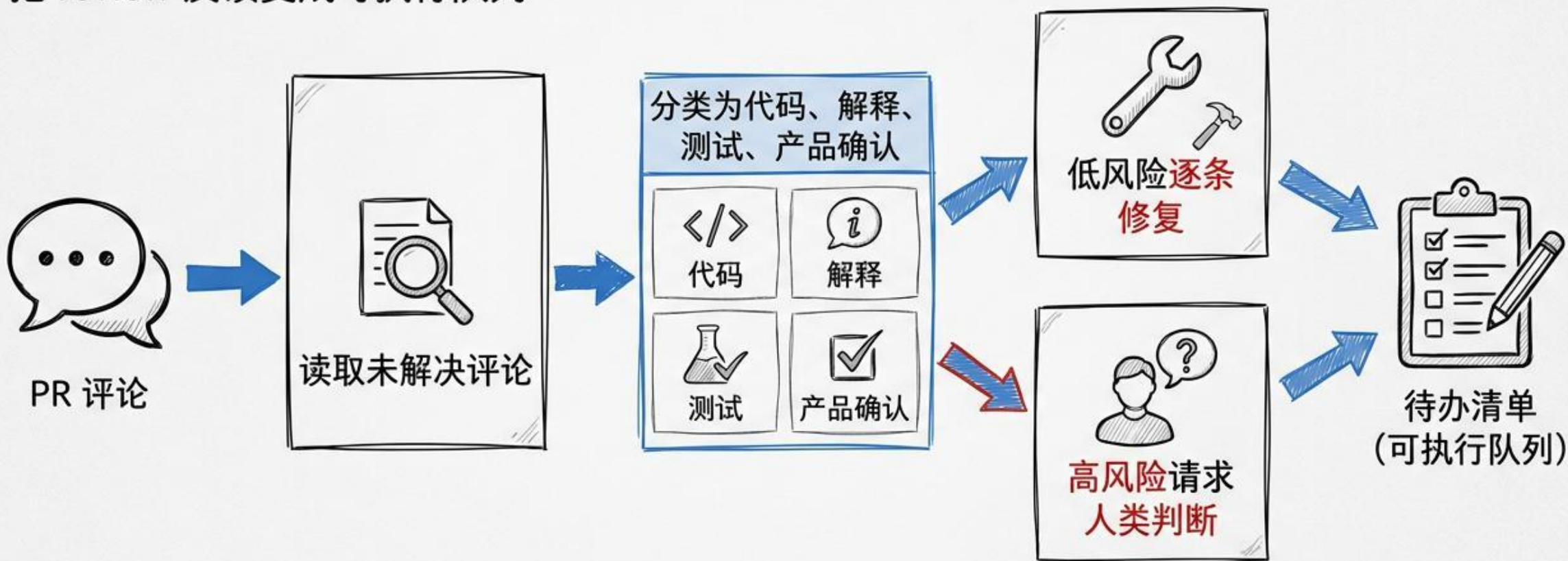
⚠ 不要接受“我修好了”的描述



模式二：PR 评论处理 Loop

典型场景与工程模式

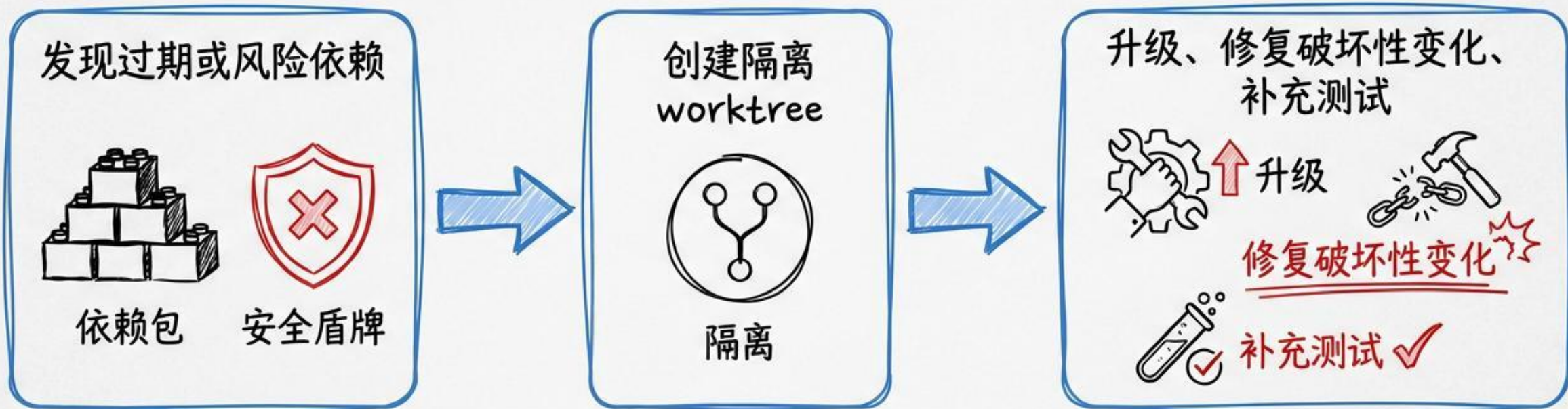
把 review 反馈变成可执行队列



模式三：依赖升级 Loop

典型场景与工程模式

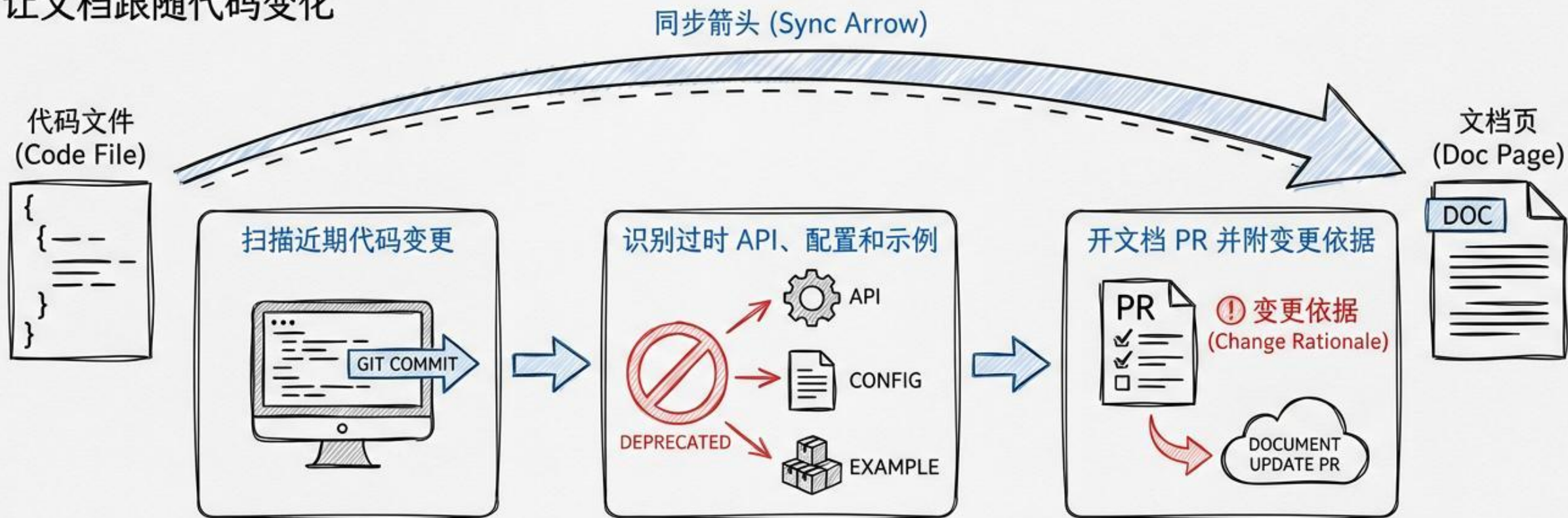
类似 Dependabot，但加入语义修复



模式四：文档漂移 Loop

典型场景与工程模式

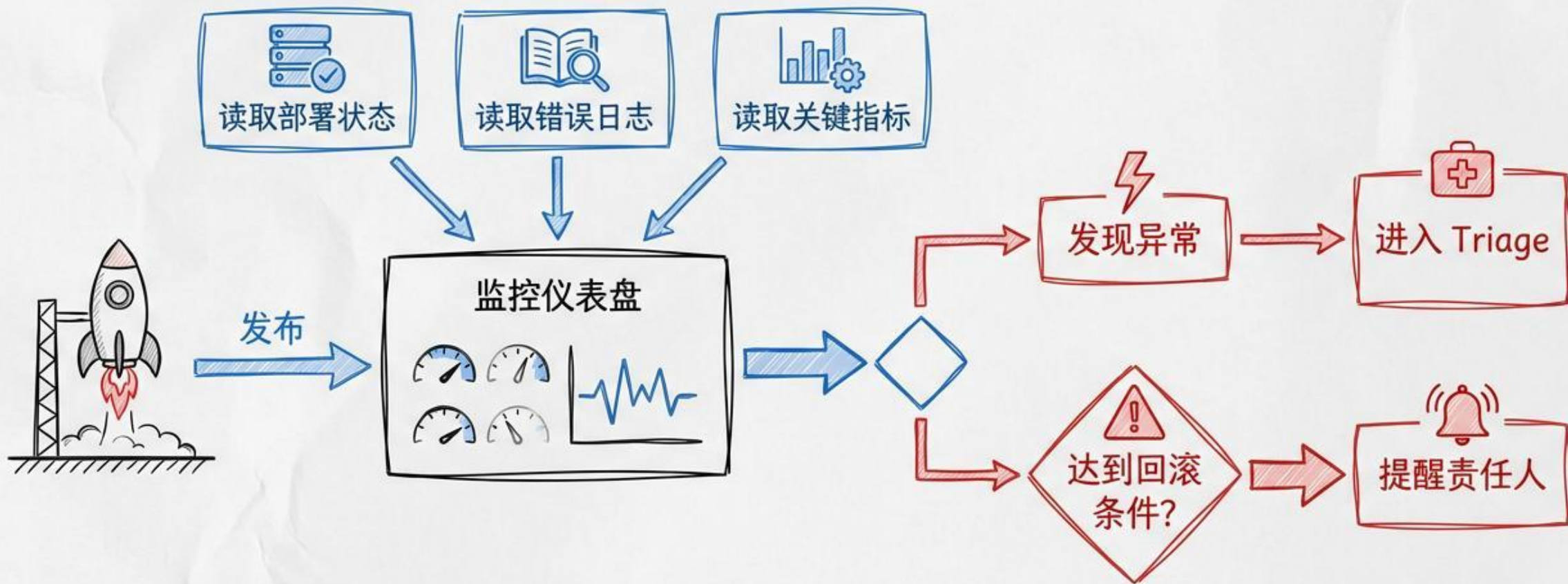
让文档跟随代码变化



模式五：上线验证 Loop

典型场景与工程模式

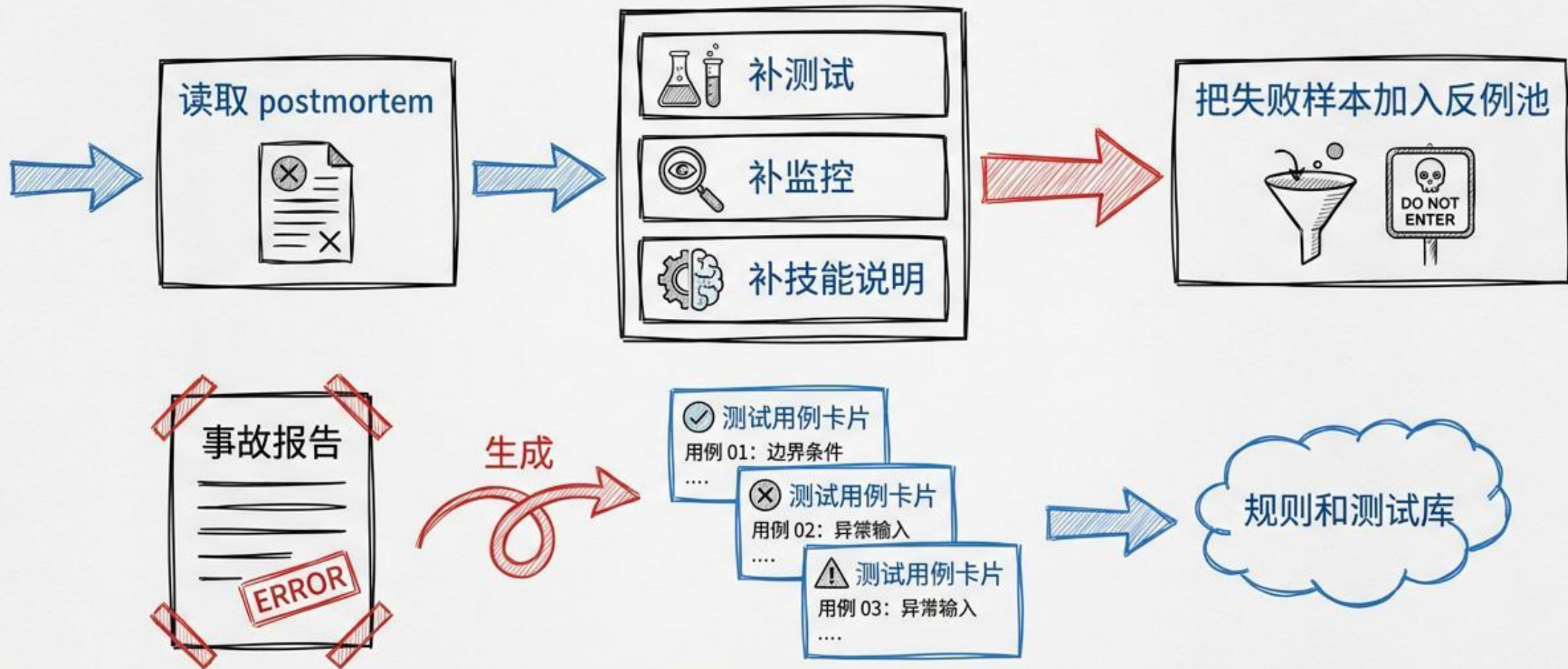
发布后自动观察，但不自动放行高风险动作。



模式六：缺陷回收 Loop

典型场景与工程模式

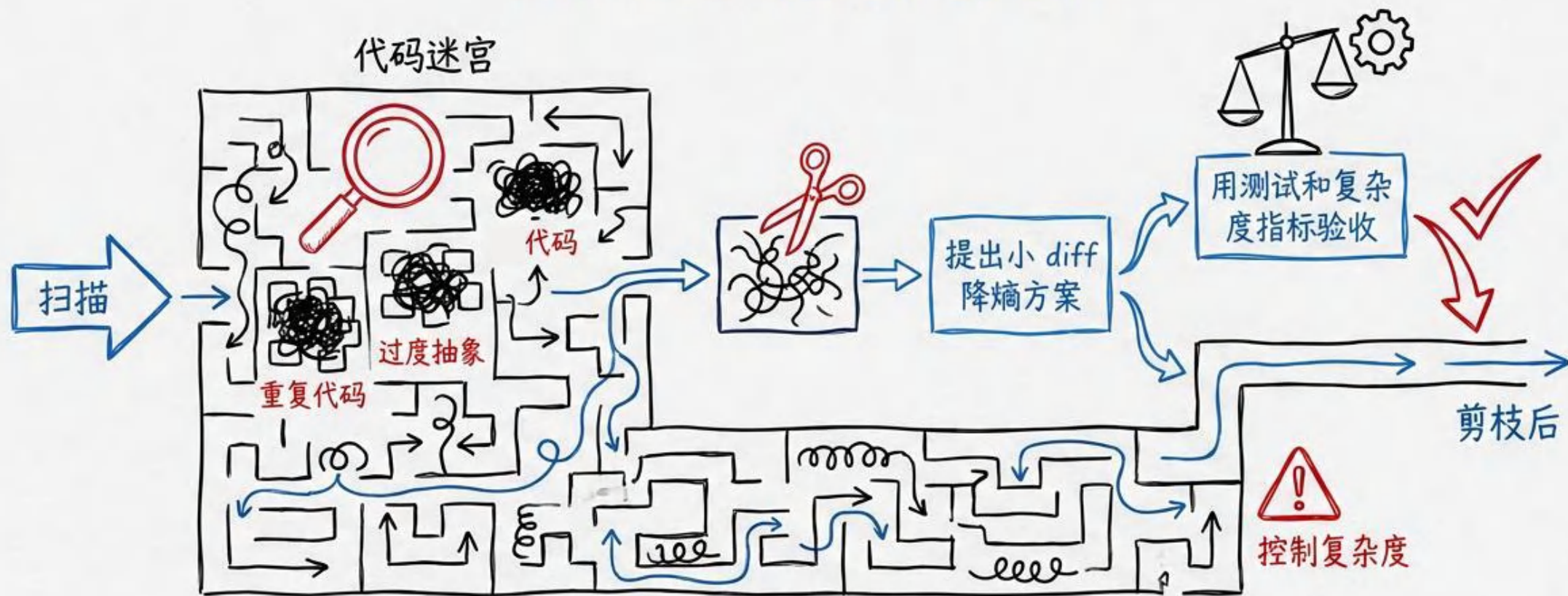
从事故中生成新的规则和测试



模式七：技术债清理 Loop

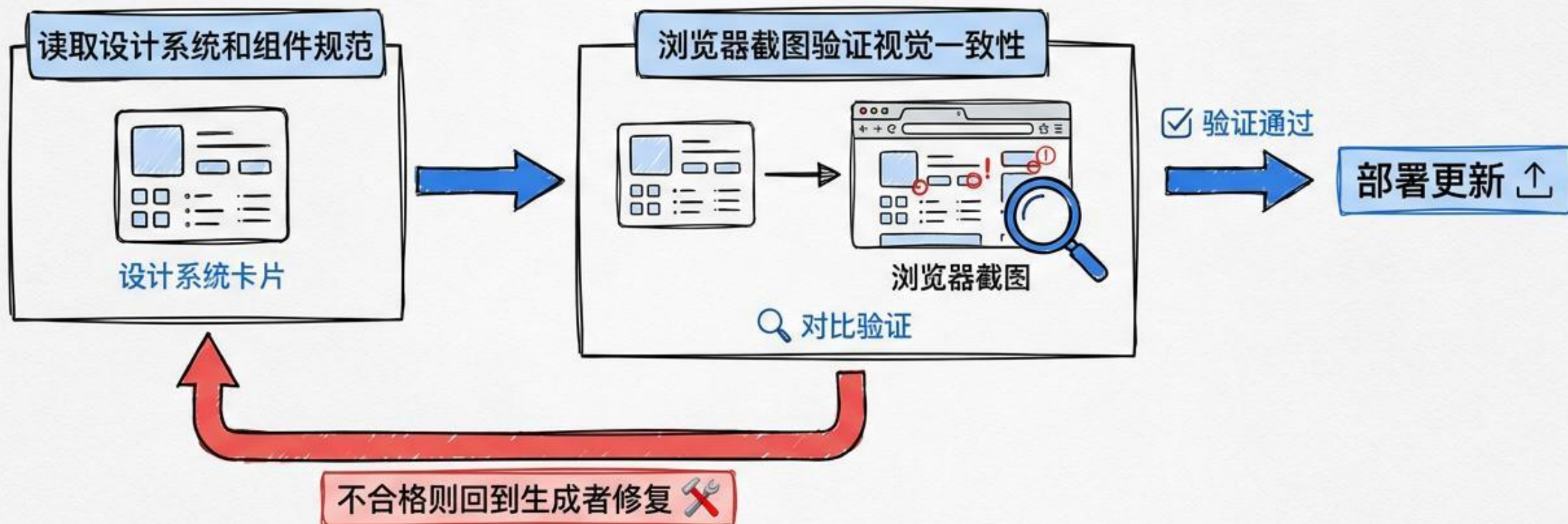
典型场景与工程模式

把代码生成带来的复杂度拉回来



典型场景与工程模式

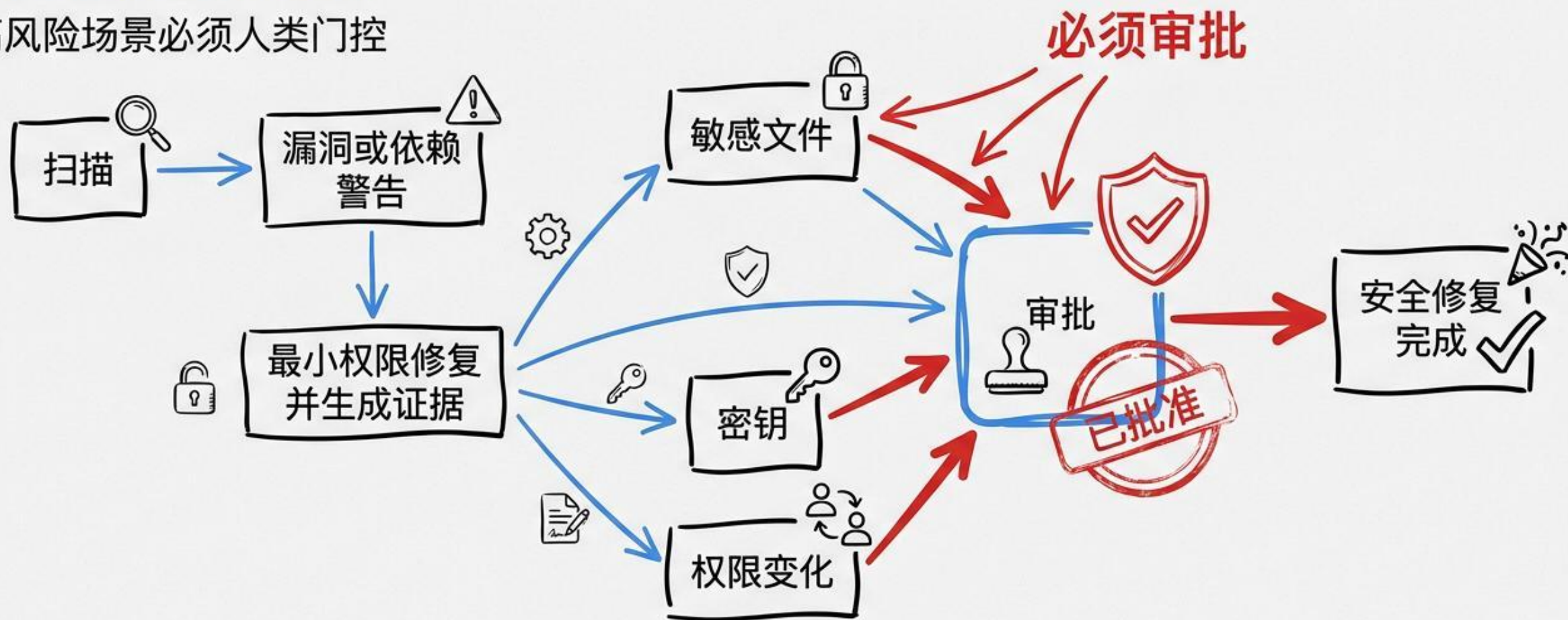
避免 AI 味 UI 与随意组件膨胀



模式九：安全修复 Loop

典型场景与工程模式

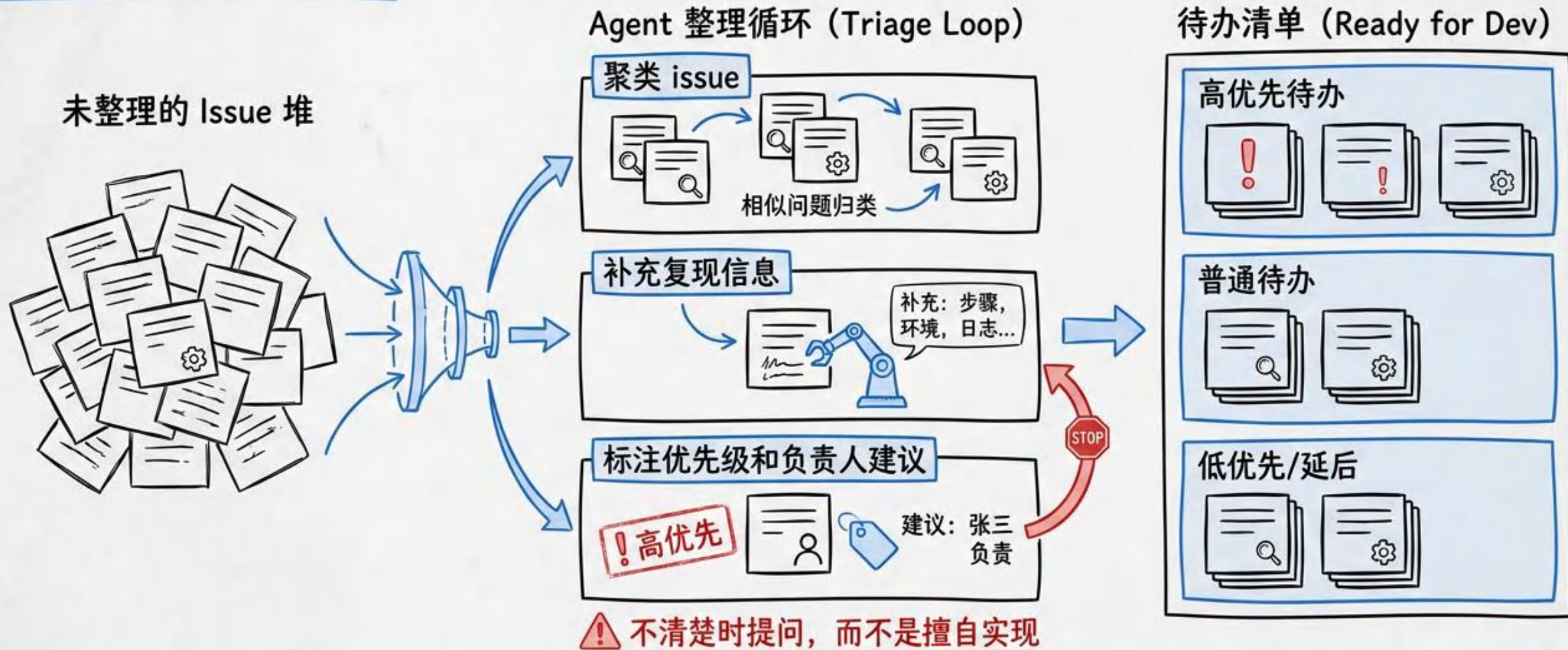
高风险场景必须人类门控



模式十：Backlog Triage Loop

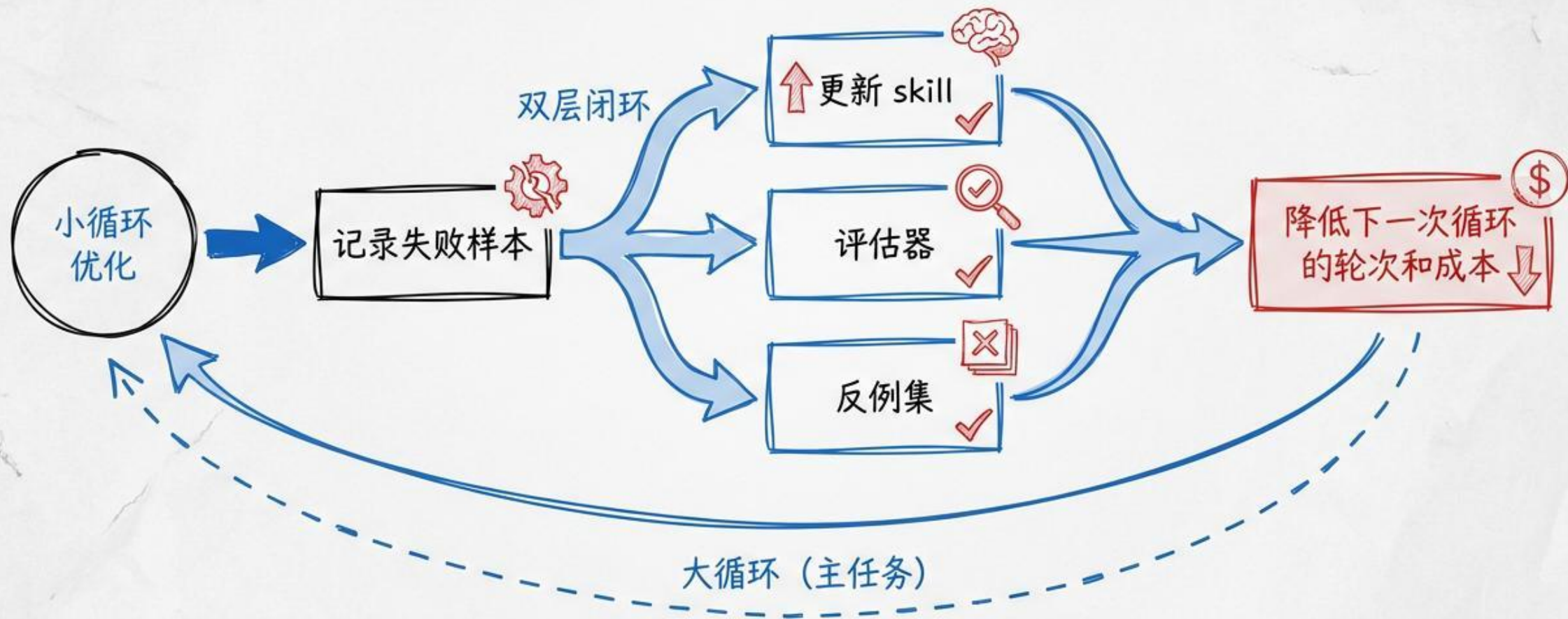
让 Agent 先整理，不要先写代码

典型场景与工程模式



模式十一：Agent Improvement Loop

典型场景与工程模式 / 循环也要被循环优化



三件事循环不会替你解决

循环越顺，越要保持清醒

风险与治理

验证仍然是你的事。



警告：不要
盲目信任

理解会随着自动生成
代码而腐烂。



停！：
保持理解

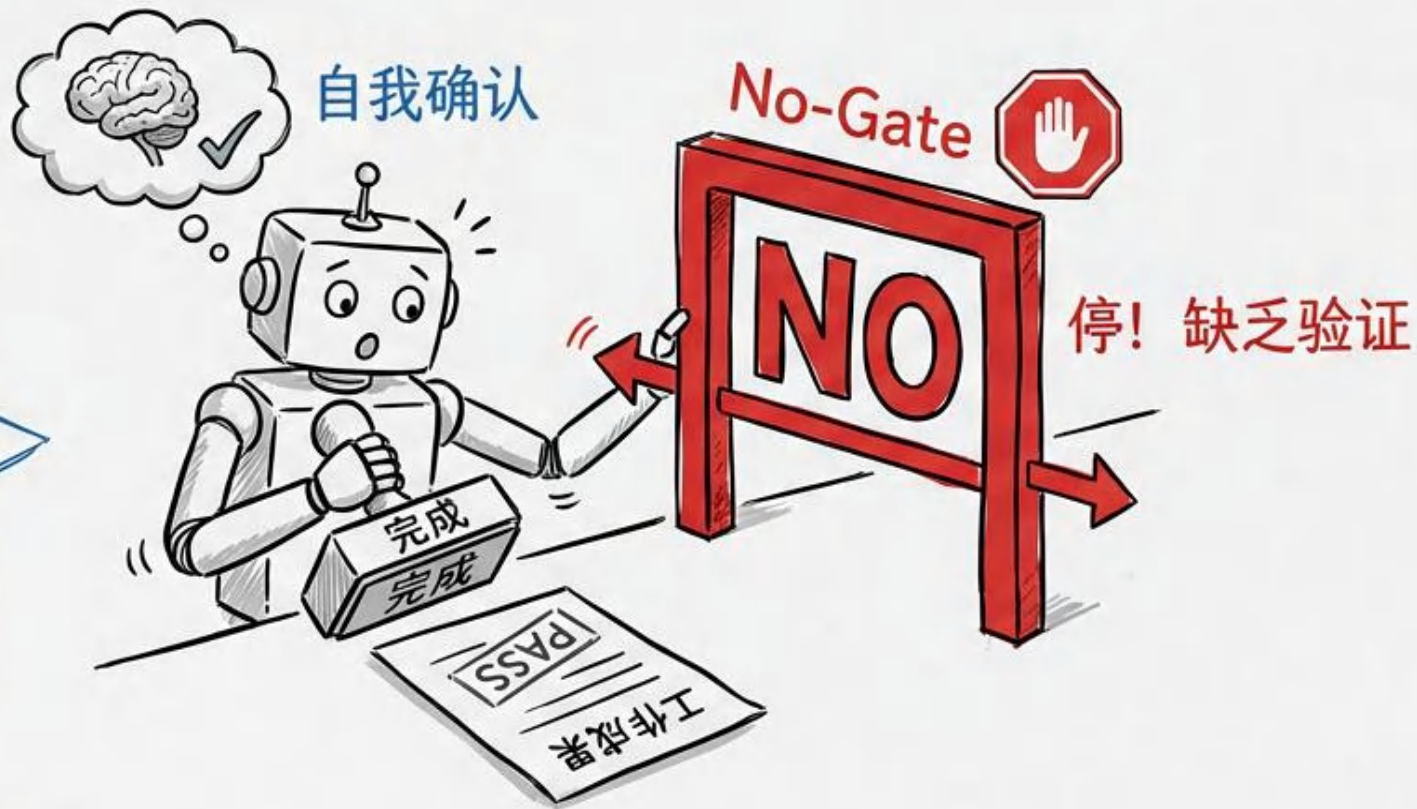
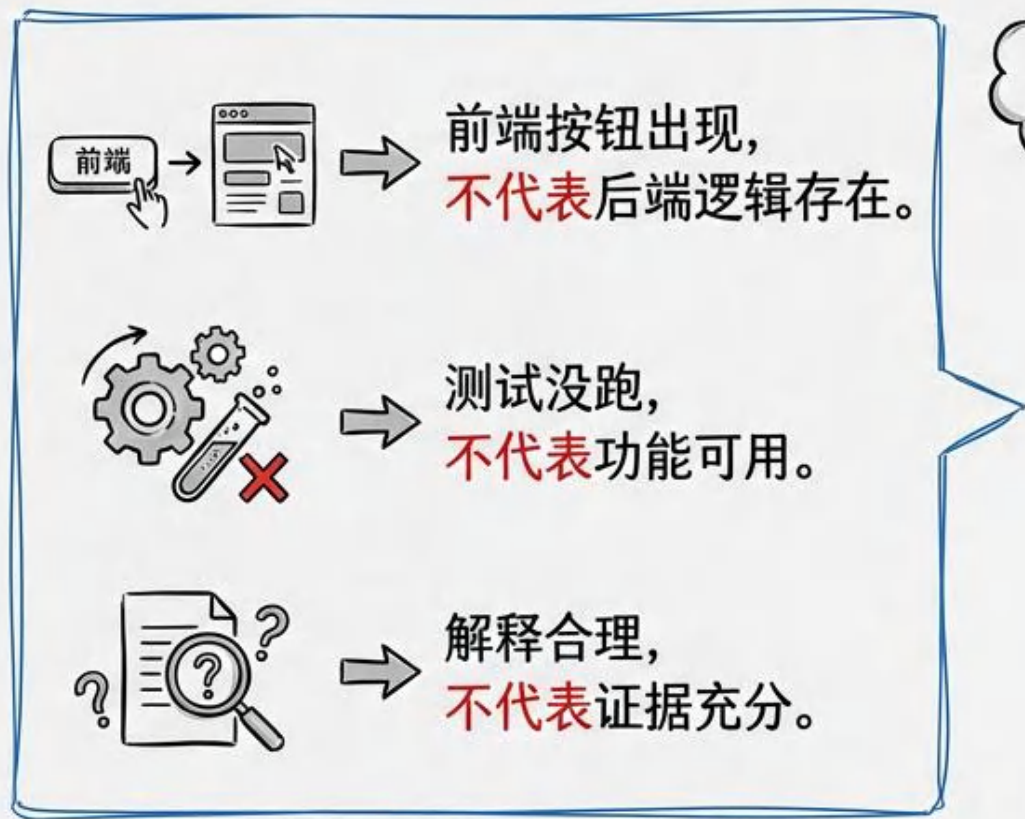
最舒服的照单全收姿势
最危险。



危险：
保持批判

风险一：自我确认

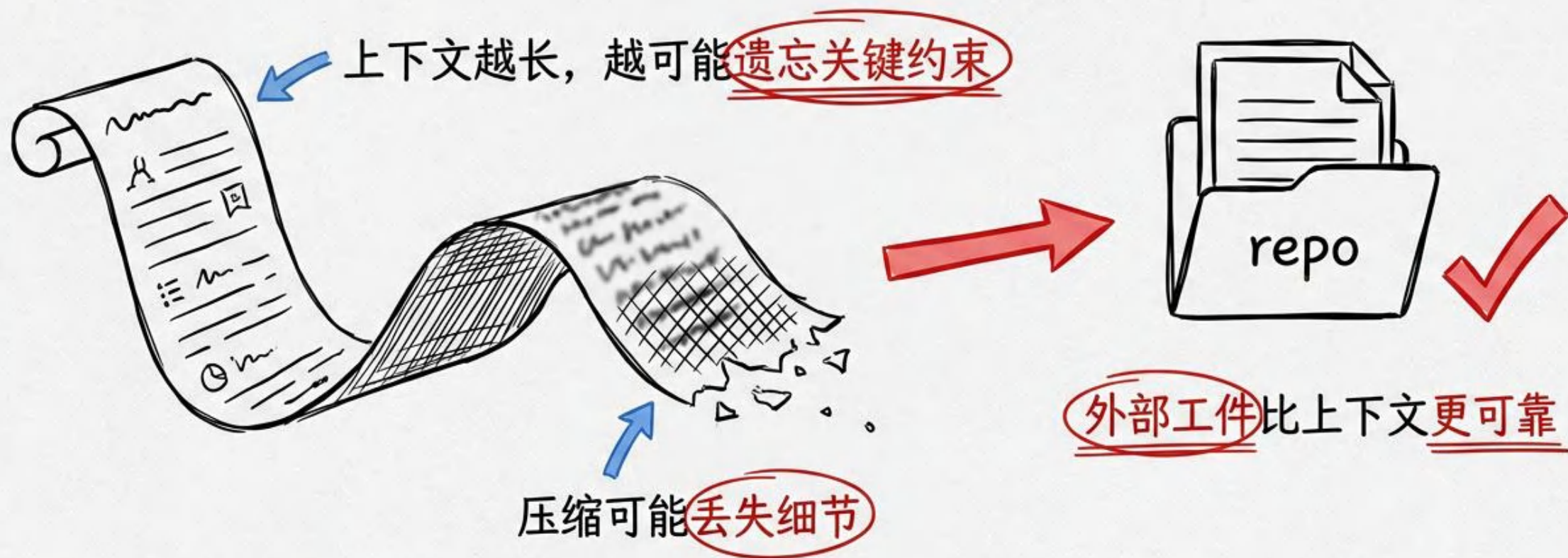
模型很容易把半成品当完成



风险二：上下文腐烂

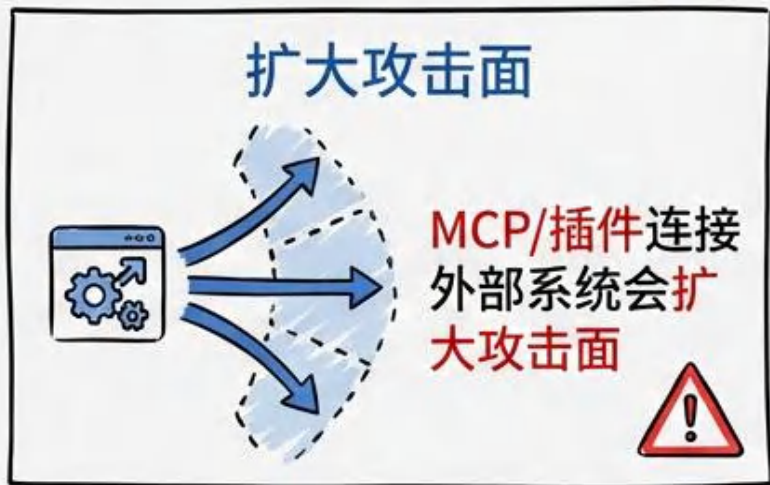
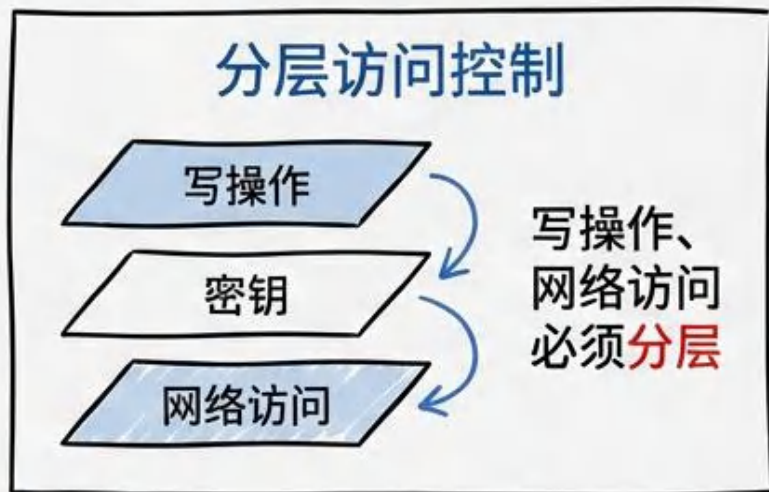
风险与治理

长会话并不等于长期记忆



风险三：权限扩散

连接器越强，边界越重要



风险四：成本黑洞

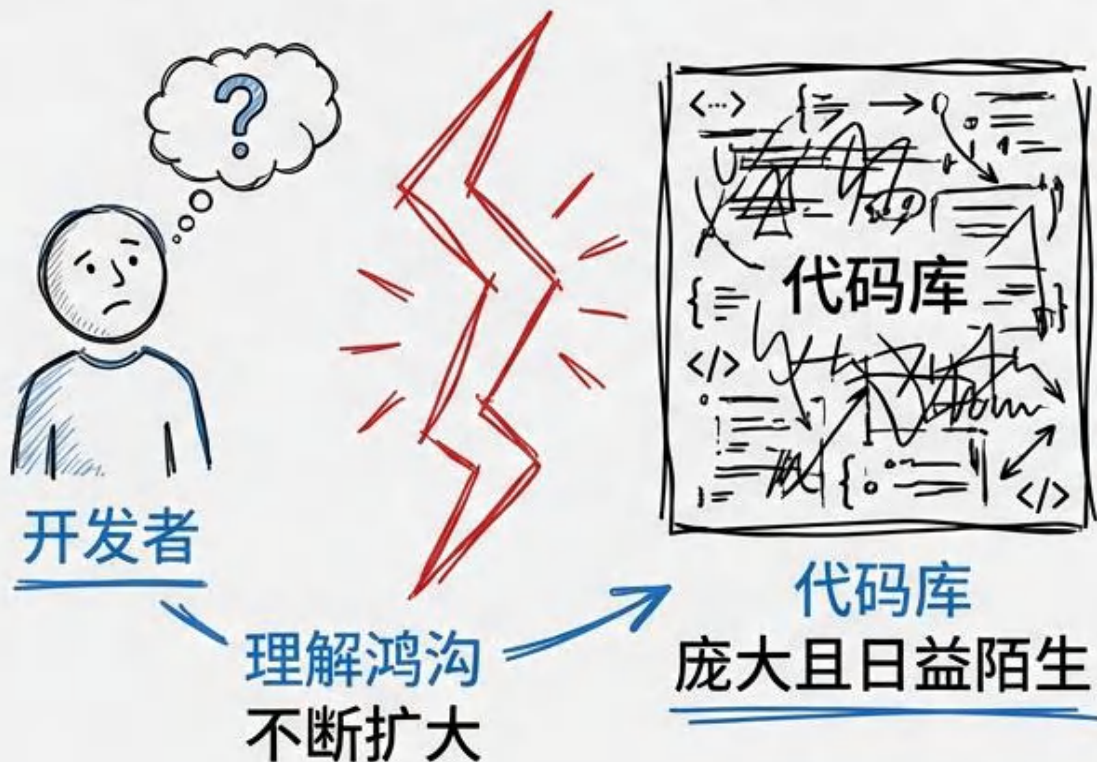
每一轮都是一次推理和工具执行



风险五：理解债

风险与治理

你没读过的代码越多，系统越陌生。



风险因素



Loop 可以 ship
你没写的代码。



审查带宽会成为
瓶颈。



瓶颈



工程师必须定期
回到代码本身。



主动回顾

治理原则一：最小权限

Agent 不应拥有 broad or unrestricted access



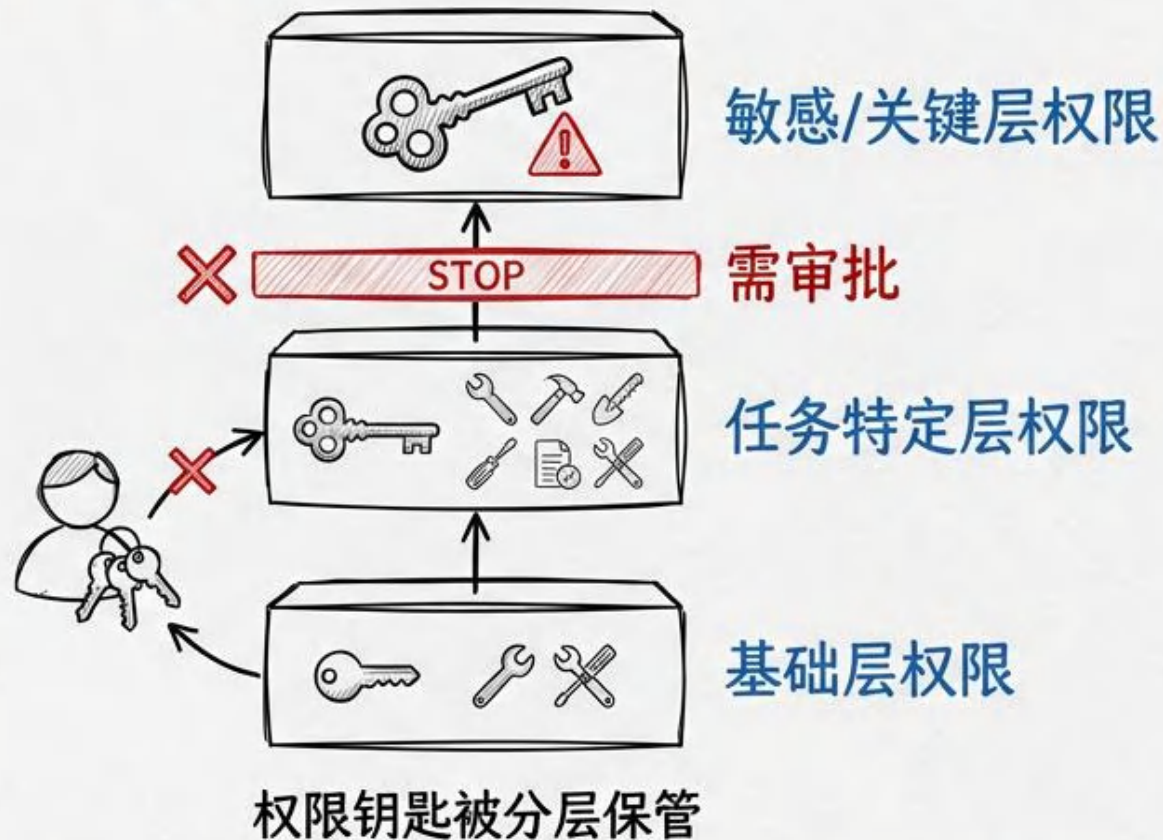
- 默认只给当前任务需要的工具。



- 敏感数据和关键系统先排除。

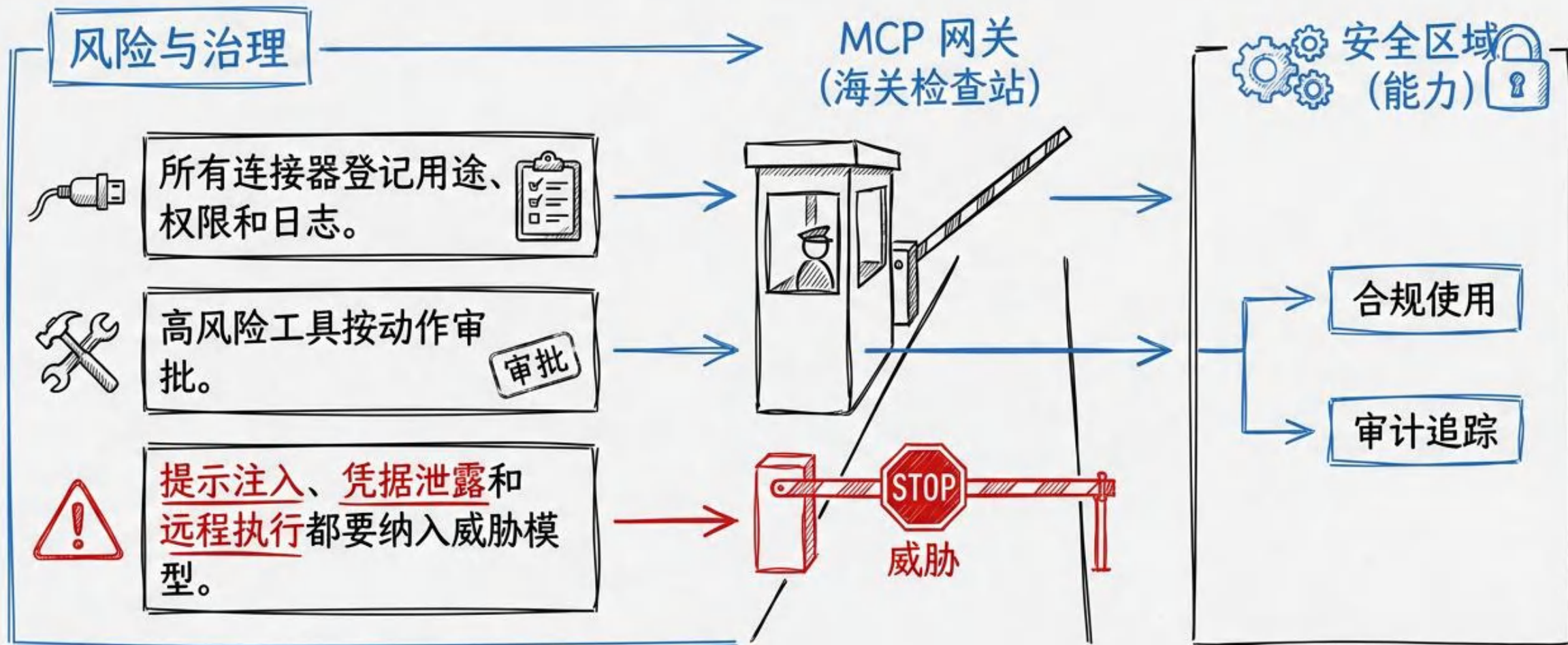


- 权限升级必须有理由、记录和审批。



治理原则二：工具网关

MCP 是能力边界，也是安全边界



治理原则三：SSDF 化

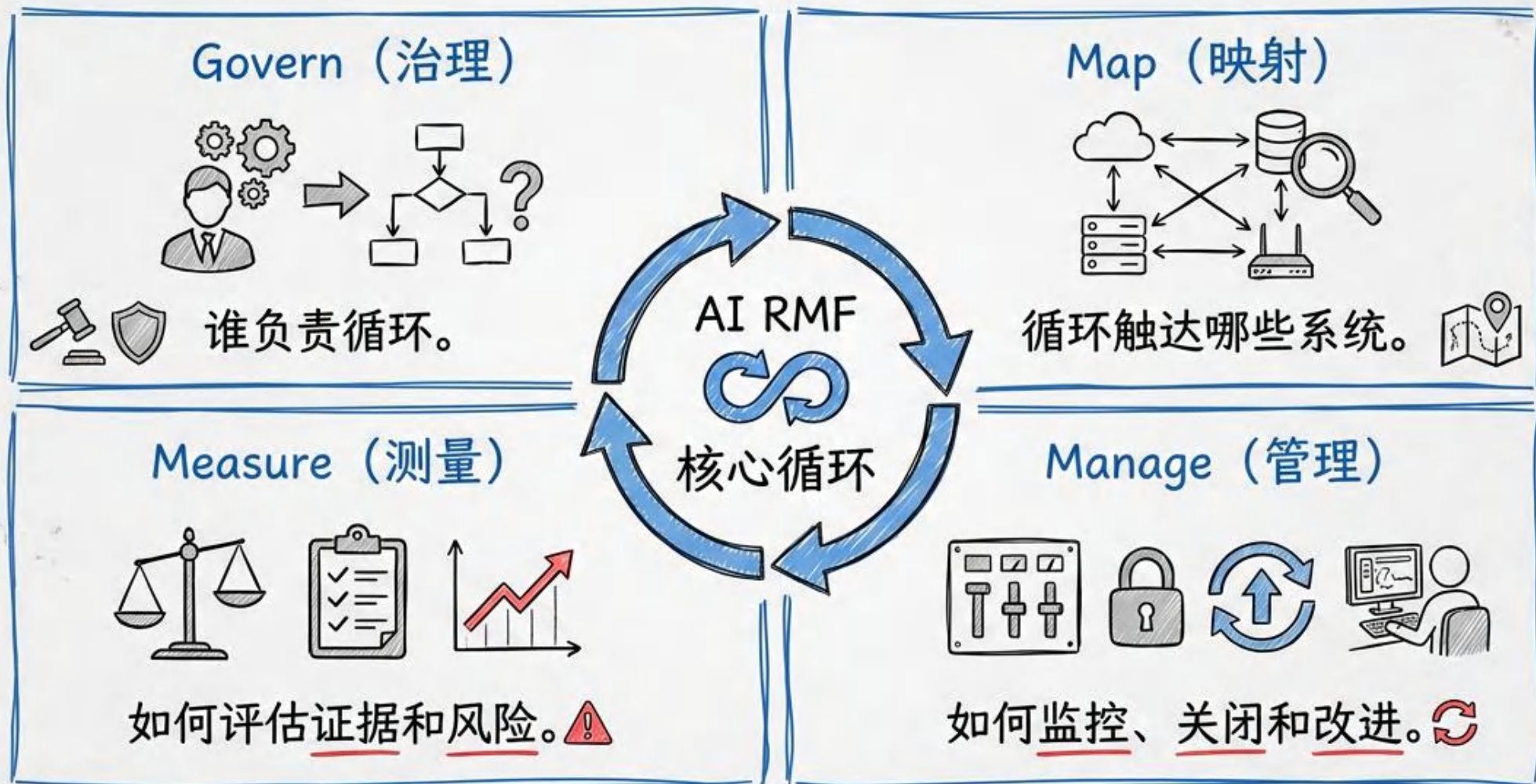
风险与治理

Agent 生成代码也要遵守安全开发流程



治理原则四：AI RMF 化

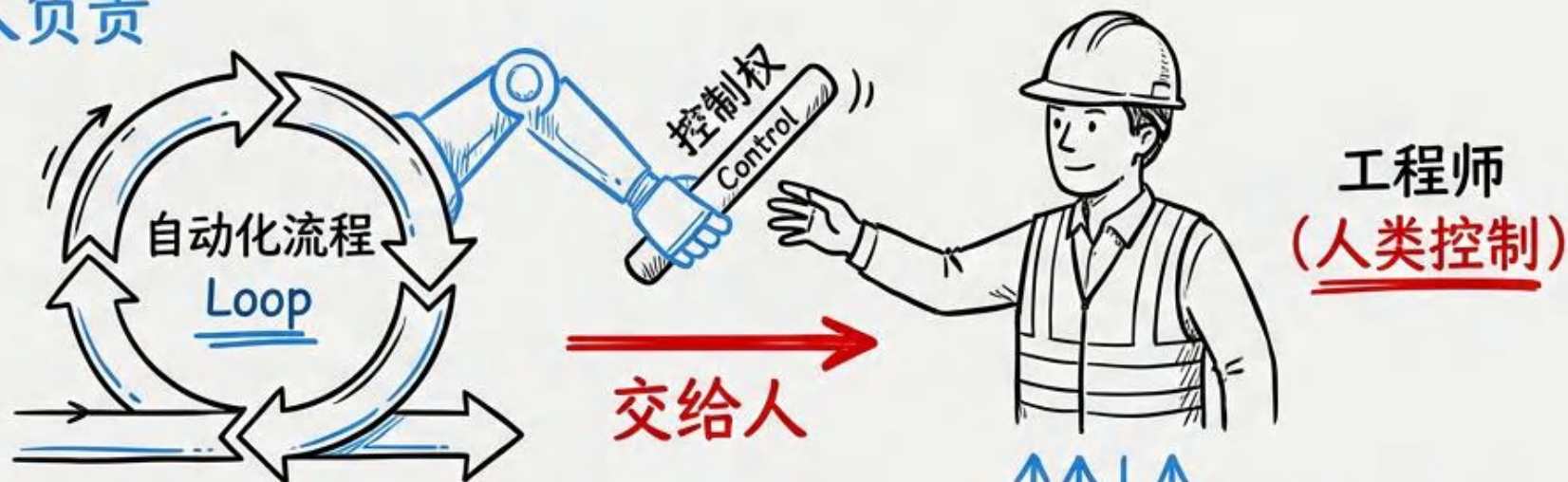
把 Loop 纳入 AI 风险管理



治理原则五：人类交接

风险与治理

自动化不是无人负责



证据不足时
? → 交给人。

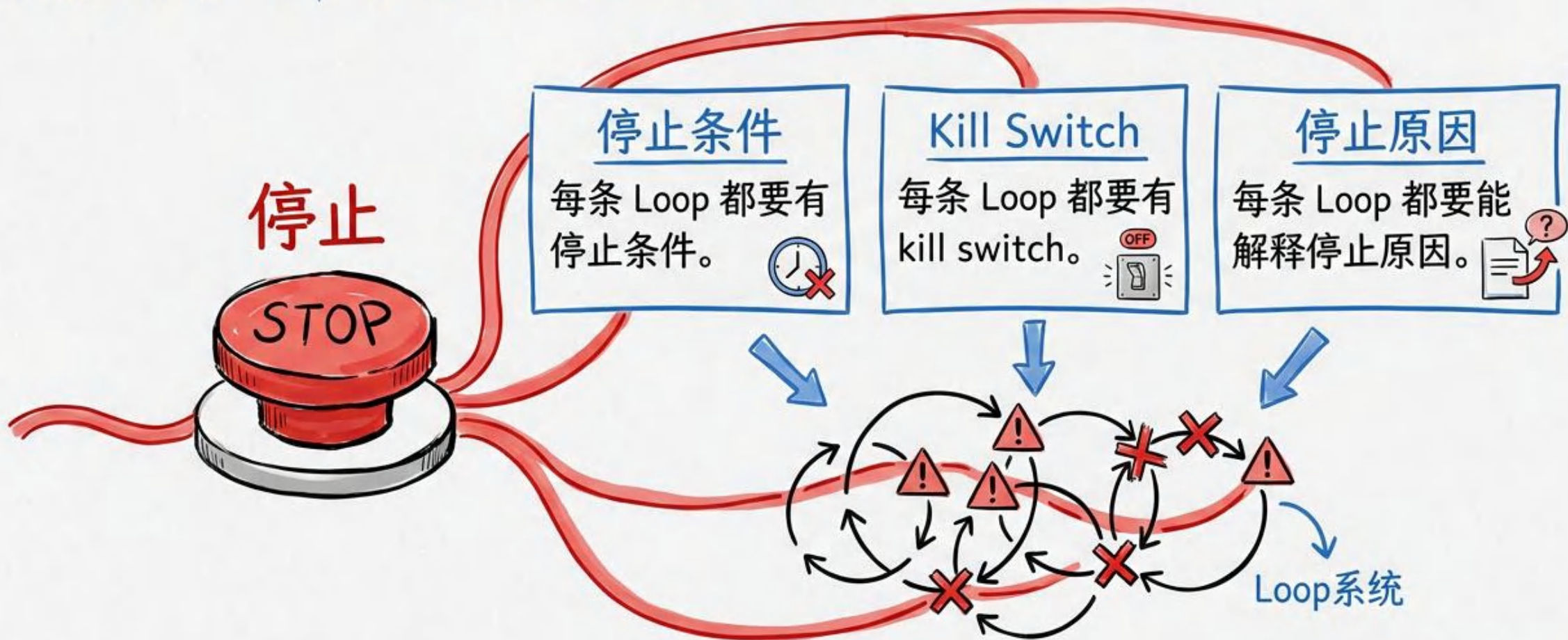
成本异常时
! → 交给人。

权限越界时
STOP → 交给人。

业务取舍必须
... → 交给人。

治理原则六：可关闭

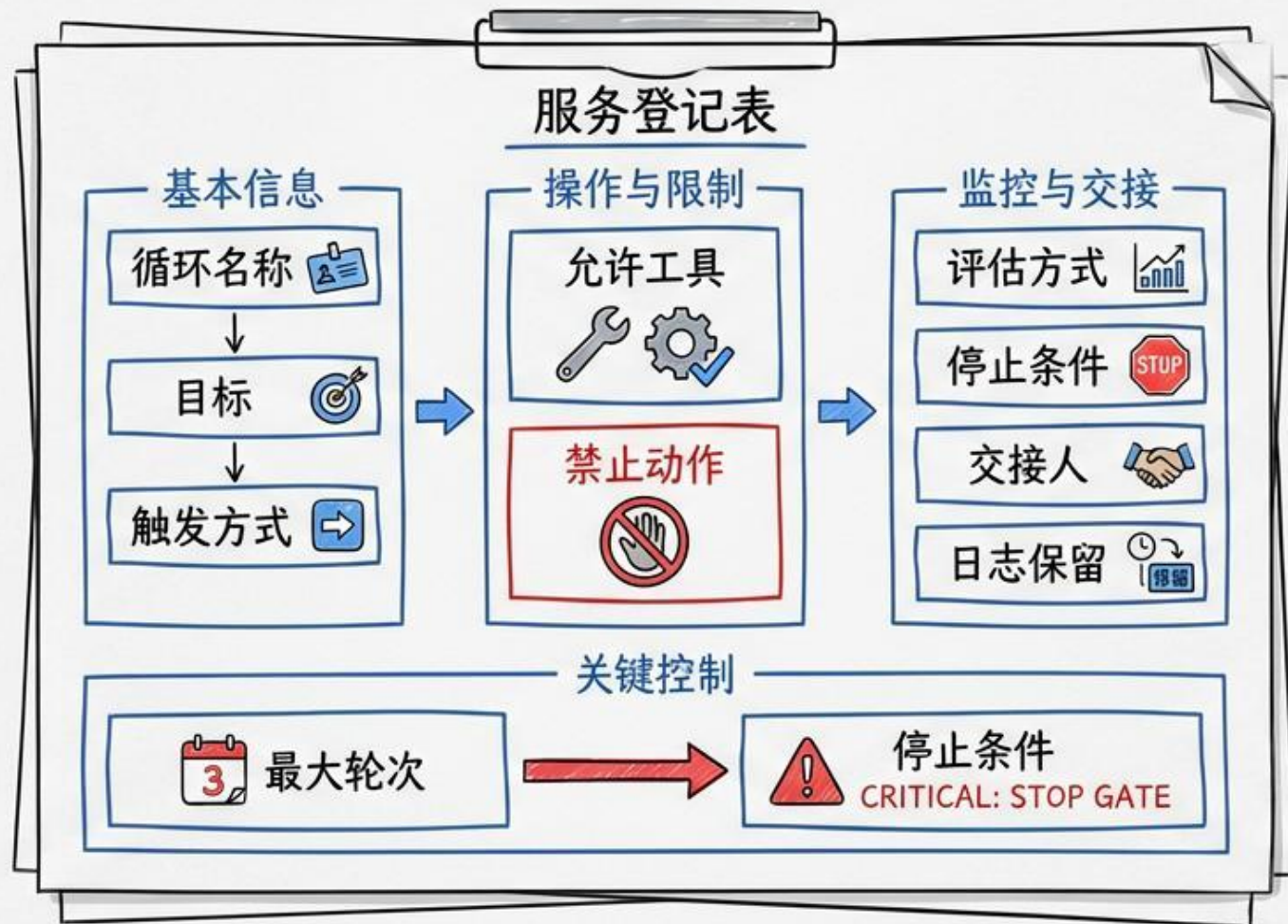
风险与治理 | 能停止比能运行更重要



Loop Spec 模板

运营模型与成本

每条循环都应像服务一样登记



Loop Registry

运营模型与成本

从个人终端走向组织资产

☁ 所有循环集中登记。

🔍 可查看状态、成本、权限和最近结果。

🔧 可按团队、仓库、风险级别过滤。

🔧 按团队过滤
(Team)

🔧 按仓库过滤
(Repo)

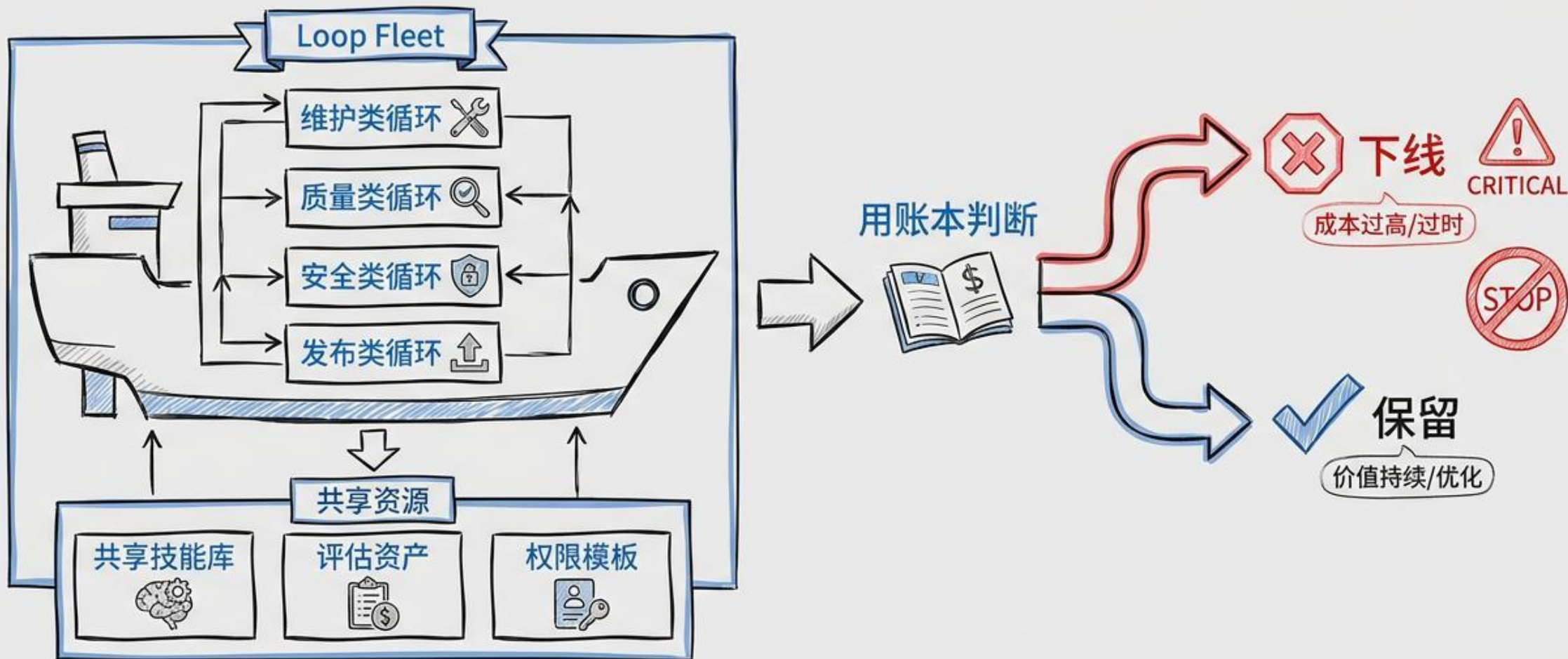
🔧 按风险级别过滤
(Risk)

循环登记看板				
循环名称 (Name)	状态 (Status)	成本 (Cost)	权限 (Access)	最近结果 (Last Result)
⚙️ Loop-A (Alpha)	运行中 (Running)	\$Low	Team 1	✅ 成功 (Success)
🗄️ Loop-B (Beta)	暂停 (Paused)	\$Mid	Public	⚠️ 异常 (Exception)
🤖 Loop-C (Gamma)	待处理 (Pending)	\$High	Admin Only	? 未知 (Unknown)

循环舰队

运营模型与成本

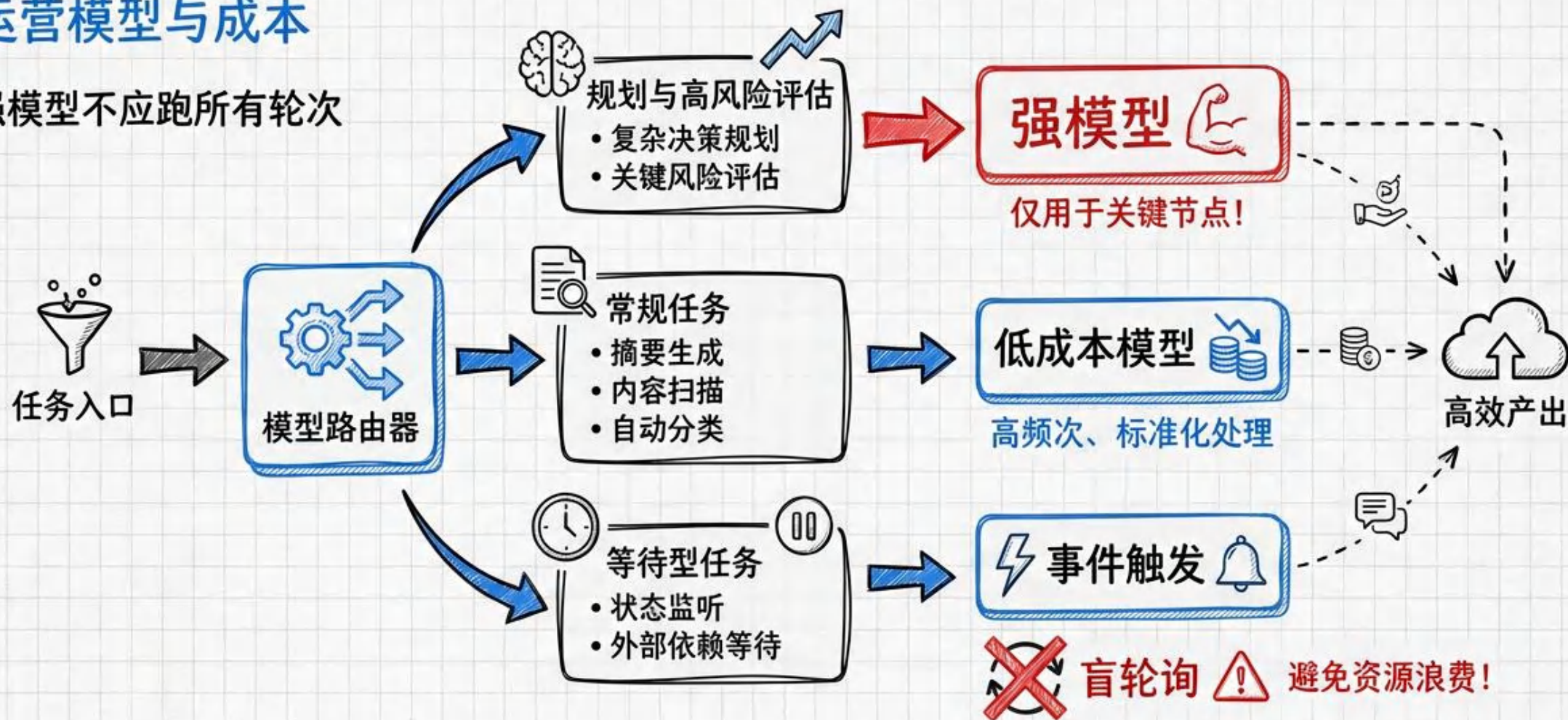
管理一组循环，而不是单条自动化



模型路由

运营模型与成本

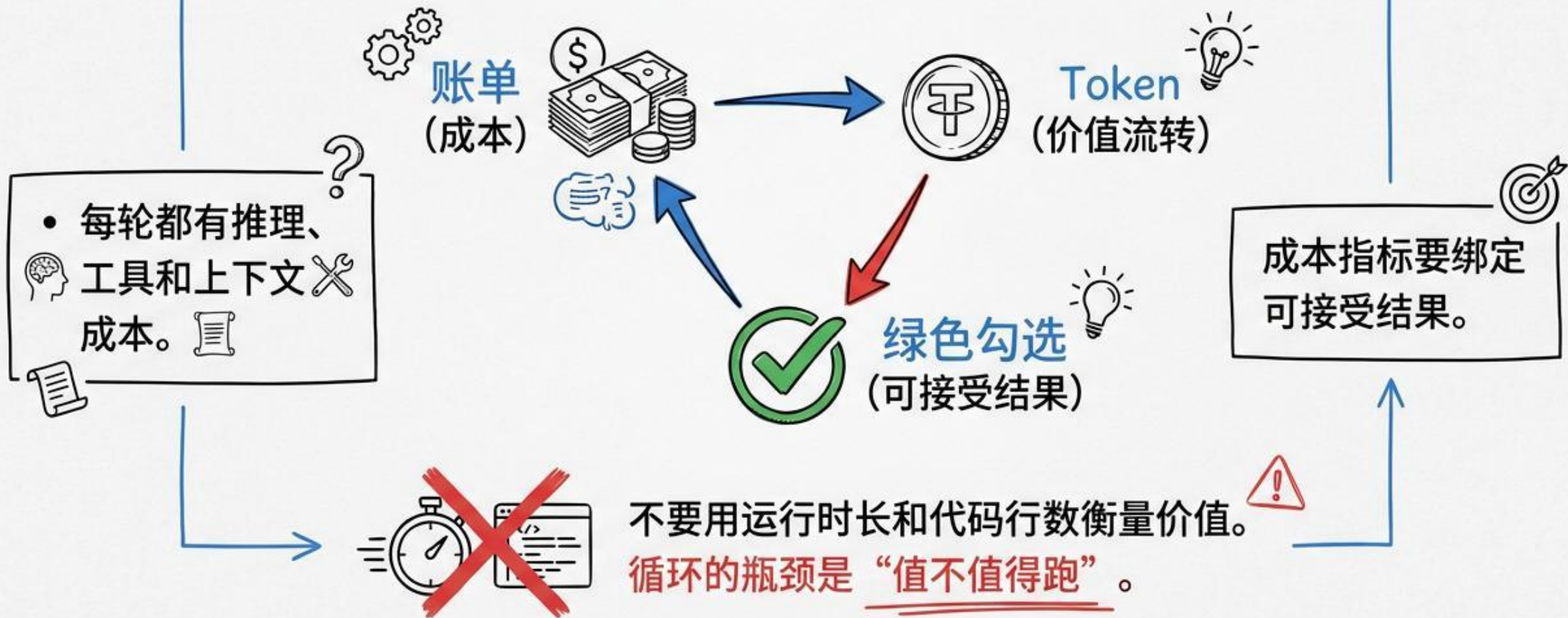
强模型不应跑所有轮次



Token 经济学

运营模型与成本

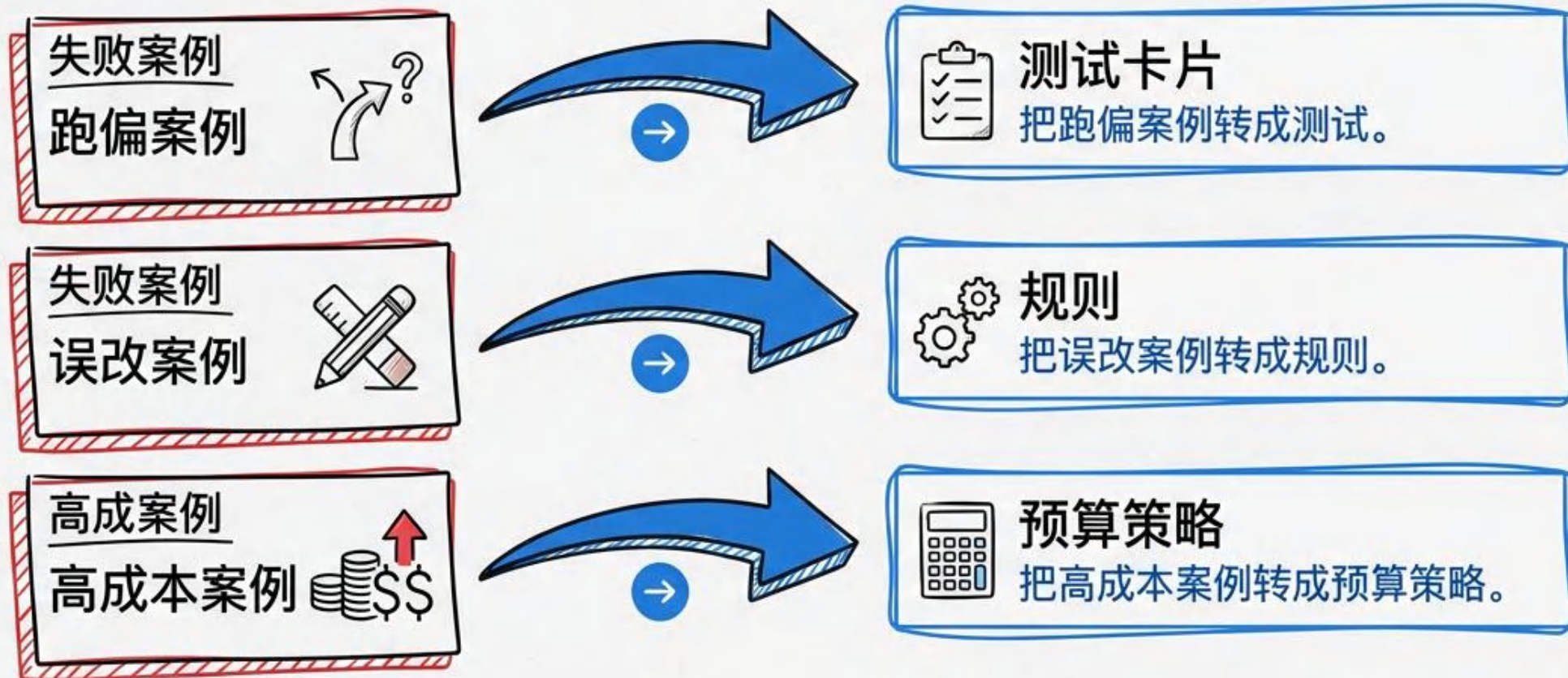
循环的瓶颈是“值不值得跑”



评估资产库

运营模型与成本

⚠ 失败样本会产生复利



技能库治理

运营模型与成本

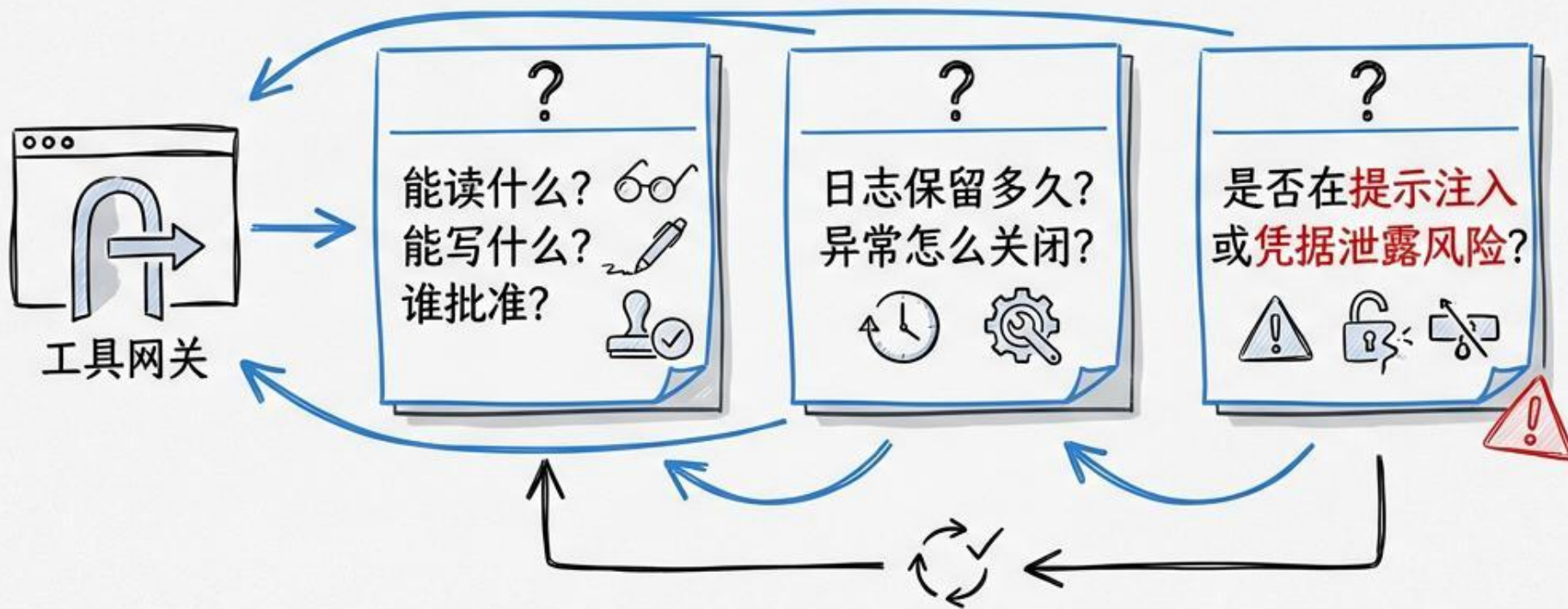
Skills 是复用资产，也可能变成供应链。



连接器治理

运营模型与成本

每个工具接入都要问三个问题



组织角色变化

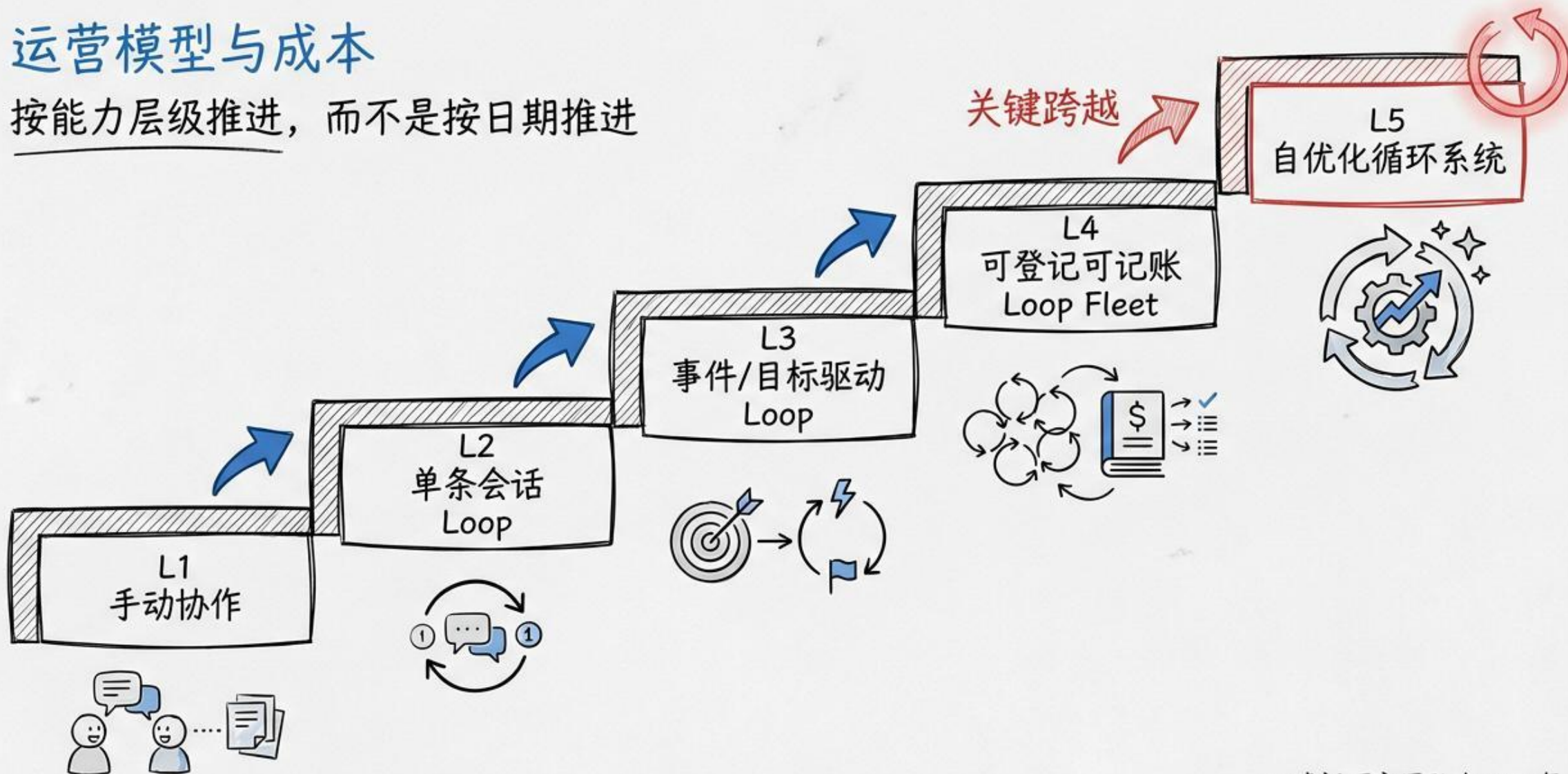
运营模型与成本：开发团队将管理人、Agent、循环和反馈系统



成熟度模型

运营模型与成本

按能力层级推进，而不是按日期推进



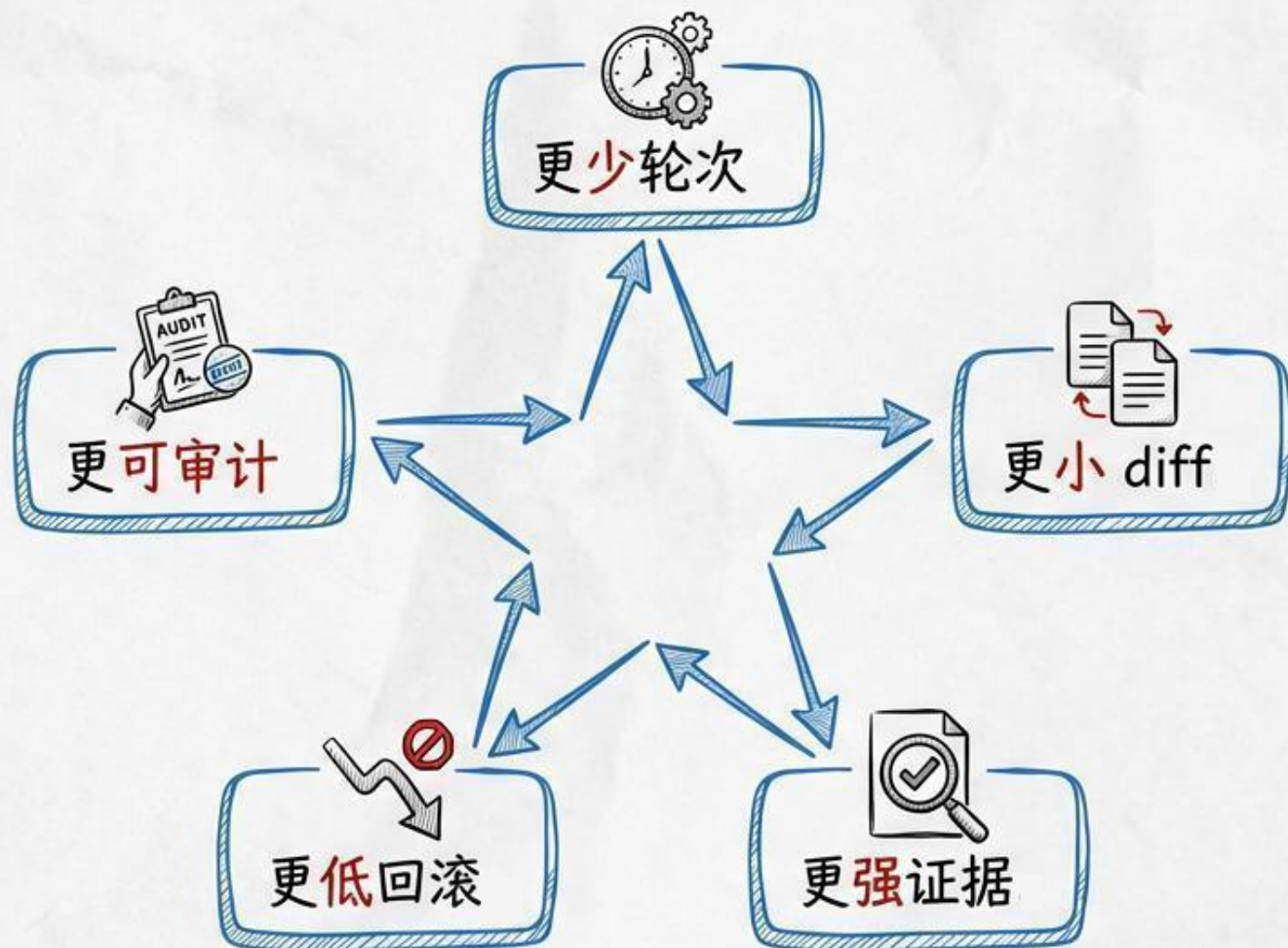
Loop Engineering 不会替代工程师

Loop Engineering 不会替代工程师



优秀 Loop 的价值是少跑几轮

优秀 Loop 的价值是少跑几轮



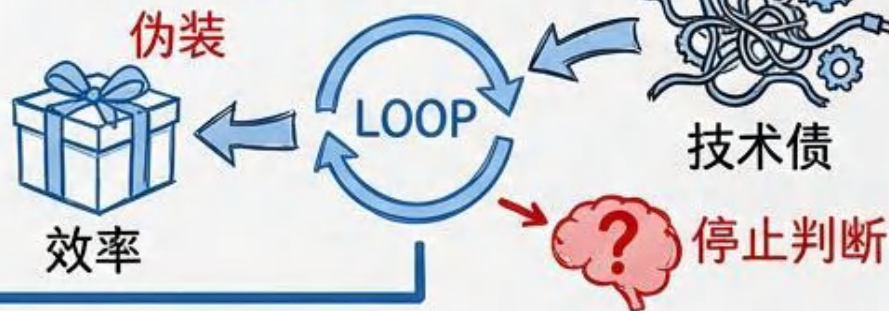
最危险的是舒服地照单全收

最危险的是舒服地照单全收。

循环越自动，人越容易停止判断。



一旦没有观点，Loop会把技术债包装成效率。

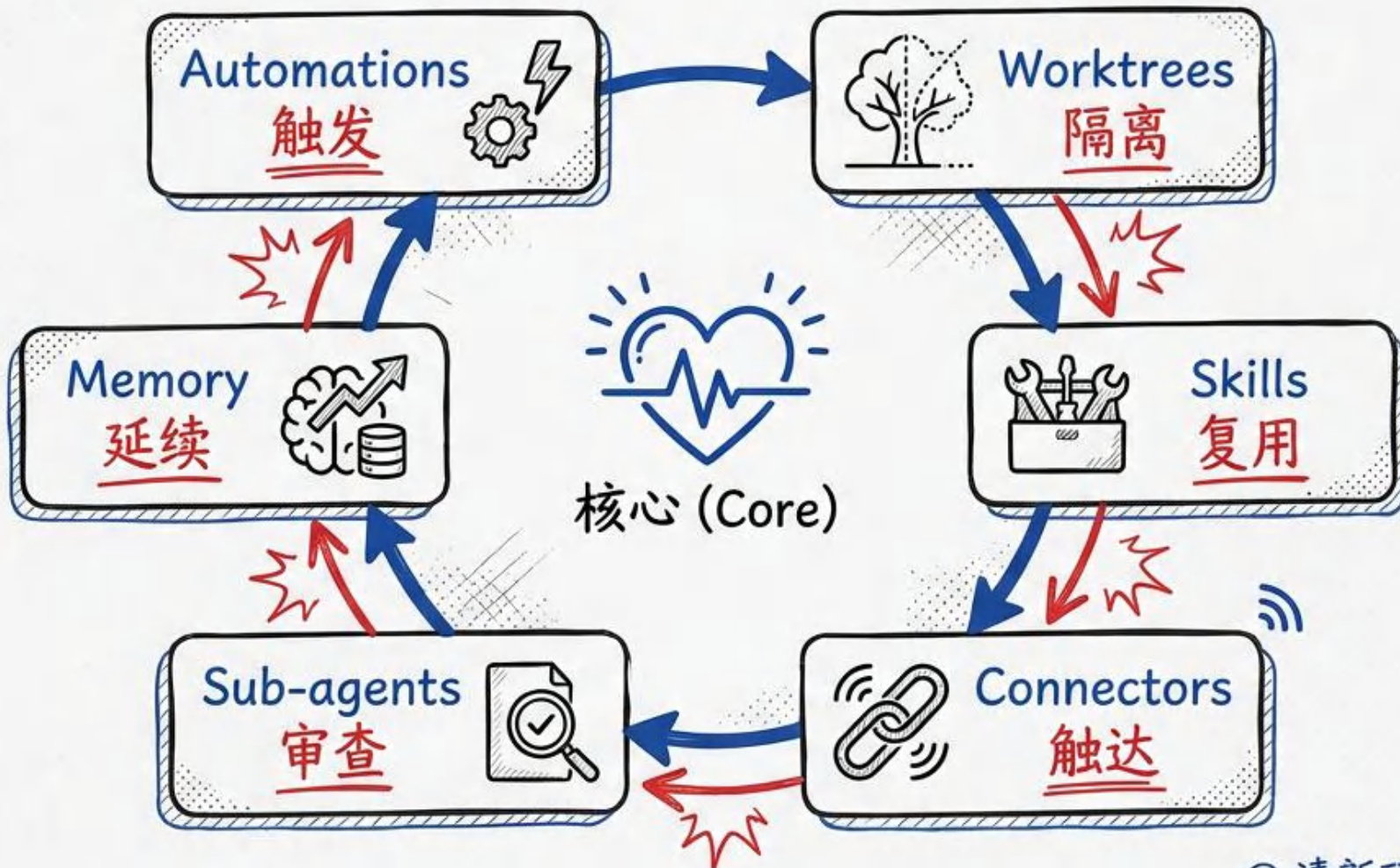


保持工程师席位，是使用 Loop 的前提。



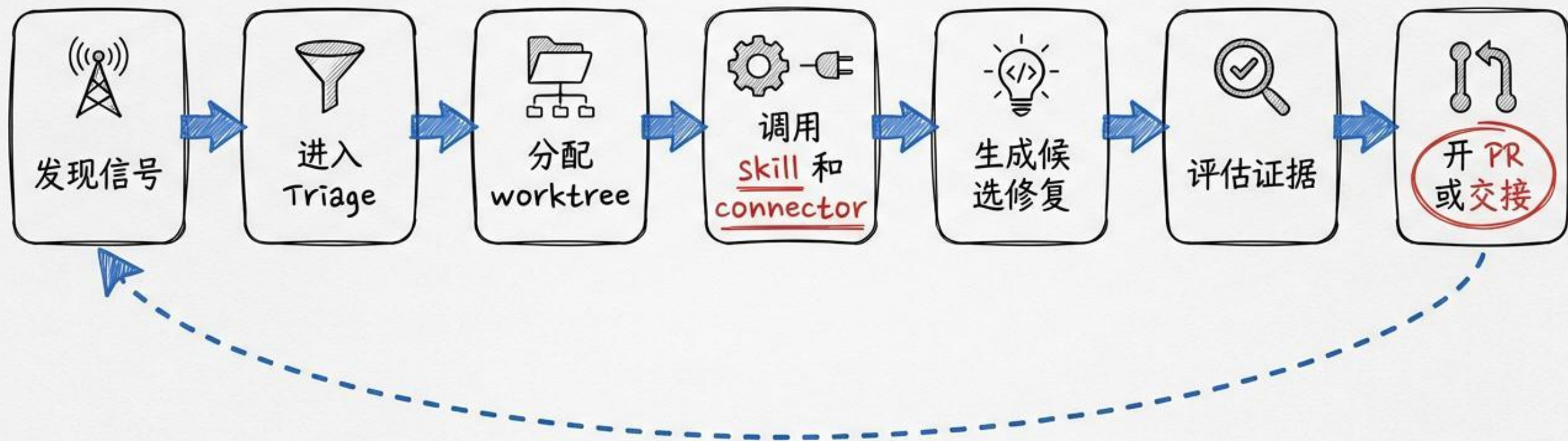
Loop Stack 六件套总览

Loop Stack = 心跳 + 隔离 + 知识 + 触手 + 审查 + 记忆



一个循环如何完成一项任务

一个循环如何完成一项任务



启动 Loop 前的十个问题

启动任何 Loop 前先问十个问题

☑ 目标是否可验证?  → 可验证

☑ 证据是否外部化?  → 外部化证据

☑ 权限是否最小化? 

☑ 记忆是否持久化? 

☑ 成本是否可控? 

☑ 失败是否能停止? 

☑ 环境是否隔离?  → 隔离环境

☑ 影响是否可逆? 

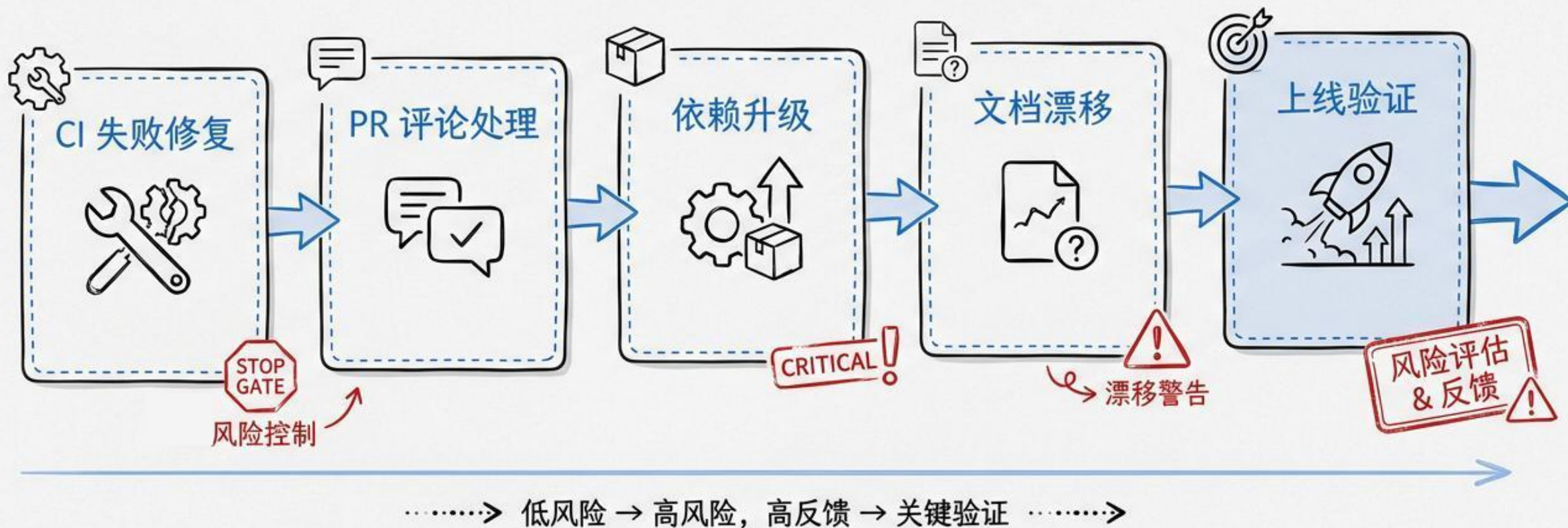
☑ 监控是否实时?  → 实时监控

☑ 退出条件是否明确? 

LOOP CHECK

最佳试点选择

从低风险、高反馈密度任务开始



不要做什么

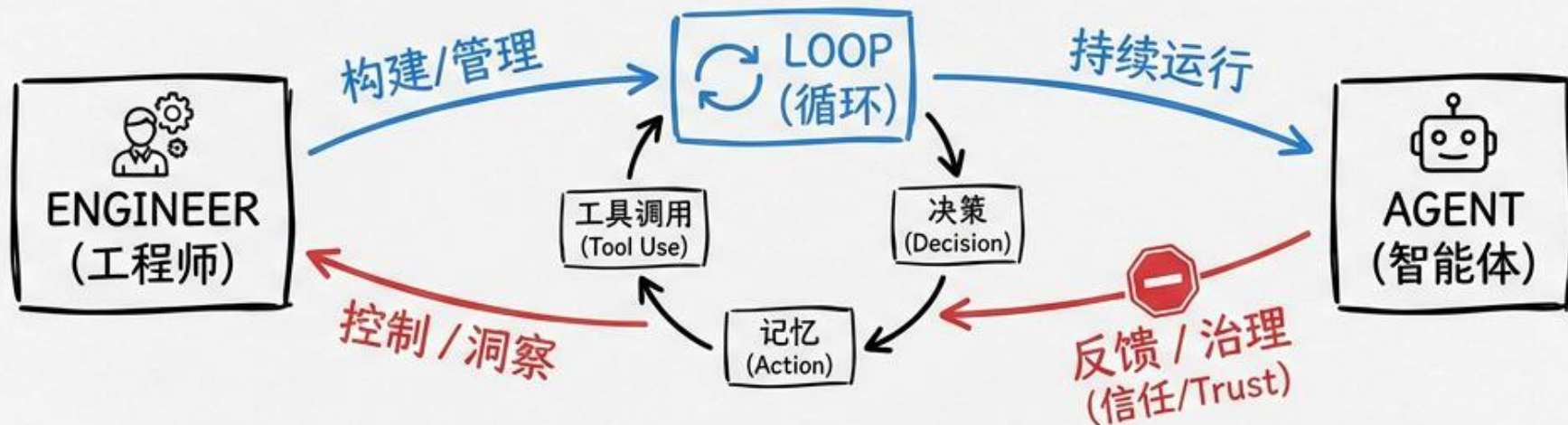
四个反模式需要明确避开



Build the Loop, Stay the Engineer

循环工程是 AI 编程的系统化下一层

Loop、Agent、Engineer 三角关系



🖥️ · 提示词仍然重要，但不再是唯一界面。

⚙️ · Loop 让 Agent 持续工作，治理 让 Loop 可被信任。

🏢 · Build the loop, stay the engineer.

来源与延伸阅读

关键事实以官方文档和安全框架为准

