KEYNOTE · 2026

智能体交付不是把模型接上去，而是把失败路径先设计清楚

Loop、工具、上下文、回退与人工接管：真正决定智能体能不能进生产的，往往不是“会不会答”，而是“坏了以后怎么收”。



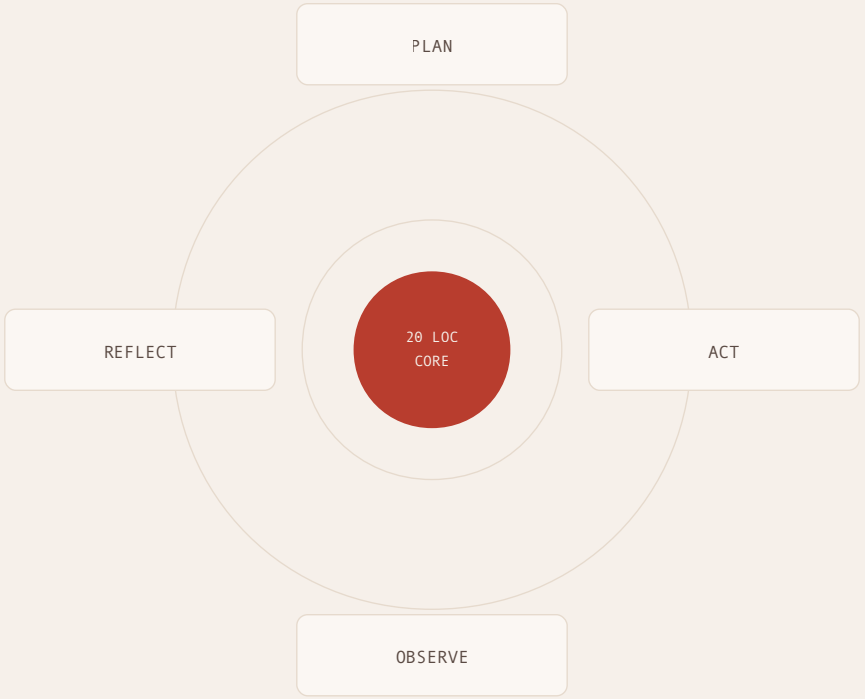
Folio slides demo · A4 landscape · 7 slides

核心可以很小，复杂度大多长在外层

一个能跑起来的 loop 也许只要几十行，但让它持续可交付的成本，通常落在工具约束、状态可见性、回退设计和人工接管上。

- 不要把所有问题都交给更长上下文，先收紧任务边界。
- 默认路径必须比“自由发挥”更强，否则线上会用事故替你补设计。
- 工具越多，越要提前定义停止条件和失败出口。

> 如果 loop 每个迭代都在变长，通常不是模型不够强，而是你在用内部复杂度替代系统设计。



进入生产之后，最贵的不是模型单次调用，而是失败不可见

三种角色各自失明一次，系统就再也没有人敢依赖它。

- 用户不知道系统在做什么，只会把所有等待都理解成“它卡住了”。
- 团队不知道系统为什么错，只能在日志和猜测之间来回切。
- 运营不知道什么时候该接手，于是人工兜底总是来得太晚。

> 可交付智能体至少要同时回答三件事：现在做到哪一步、为什么这样做、如果失败谁来接。

不要只看“答得像不像”，要看系统是否越来越稳

-31% 人工接管前的平均定位时长

+18% 任务首次完成率改善

42s 复杂任务的中位交付时长

4 类 必须显式建模的失败类型

> 真正值得展示的 KPI 不是“模型更聪明了”，而是系统更容易被团队信任了。

从“会跑”到“能交付”，通常是默认值和界面的重写

旧方案

- 把模型、工具和状态全堆在一个聊天窗口里。
- 失败后只能回看原始日志，业务团队无法自助定位。
- 默认路径模糊，系统经常“继续做错的事”。

新方案

- 拆出任务阶段、证据块和人工接管节点。
- 把来源、动作和结果放进同一页，减少跨系统跳转。
- 默认收紧任务边界，把不确定性交给确认和回退。

一个更稳的交付流程，往往比一个更强的 prompt 更有价值

01 先收窄任务

- 先定义任务输入、输出、工具范围和停止条件，再谈模型自由度。

02 把失败前置

- 在设计阶段就列出最常见的失败模式，并给每类失败一个明确出口。

03 让人能接手

- 人类接手时必须看得到任务阶段、证据来源和上一步动作，而不是只看最终报错。

谢
谢

交付不是把答案做出来，而是把系统做成团队
敢反复依赖的东西。

hello@folio.design · folio.so