

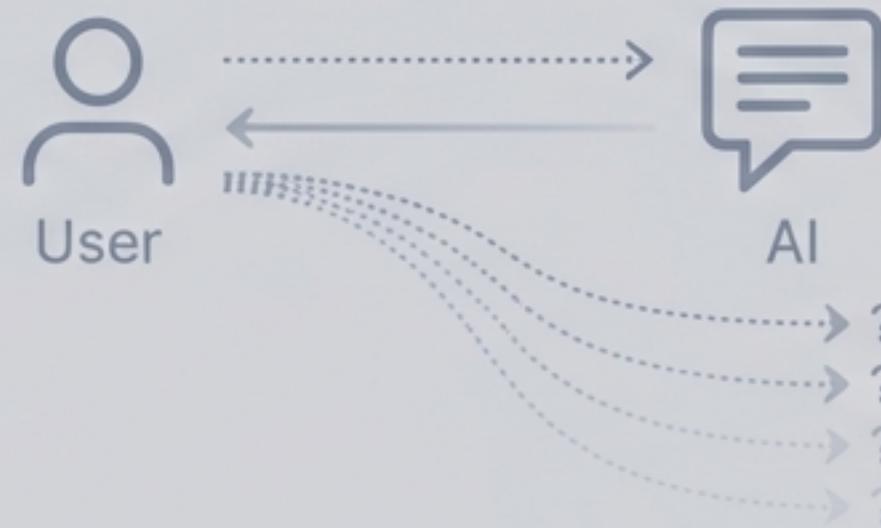
# Long-Running Agent Harness

基于 Eval-Driven Development 的全自动 Claude CLI 开发编排系统

从理论研究到工程实践的自动化开发框架

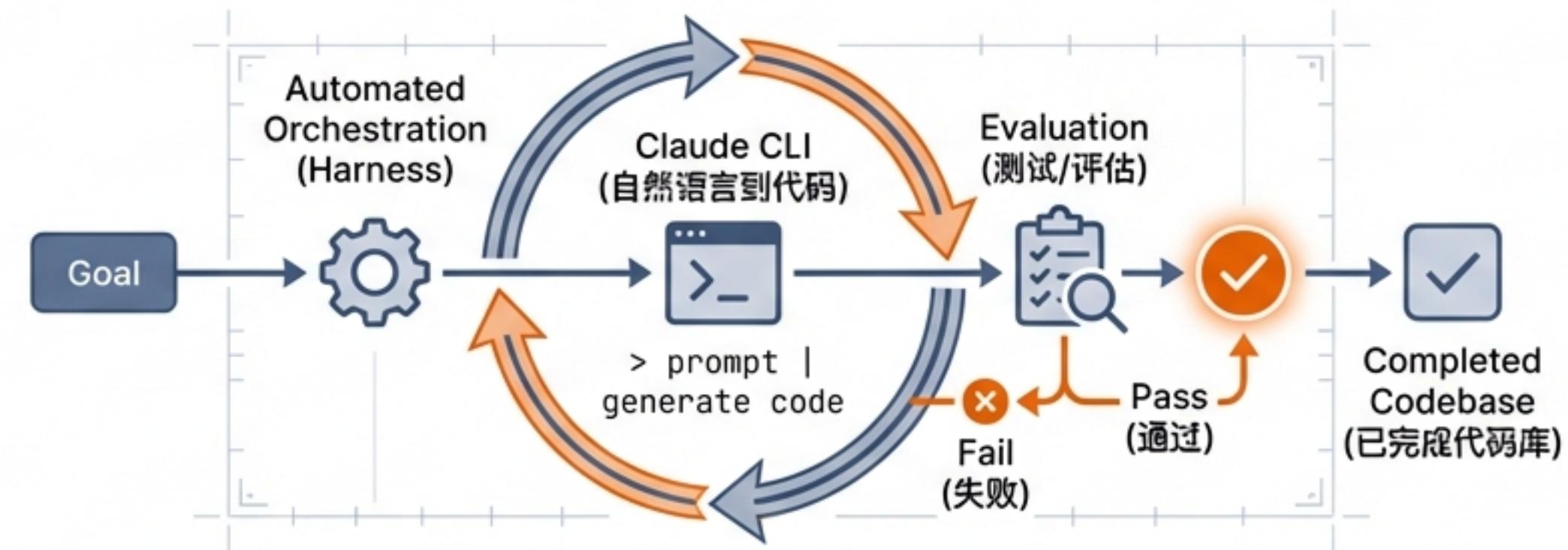
# 项目概述：让 AI 像工程师一样长期工作

## Traditional Chat (传统对话)



上下文易丢失，无法处理长流程，  
缺乏验证机制。

## Long-Running Agent (本系统)

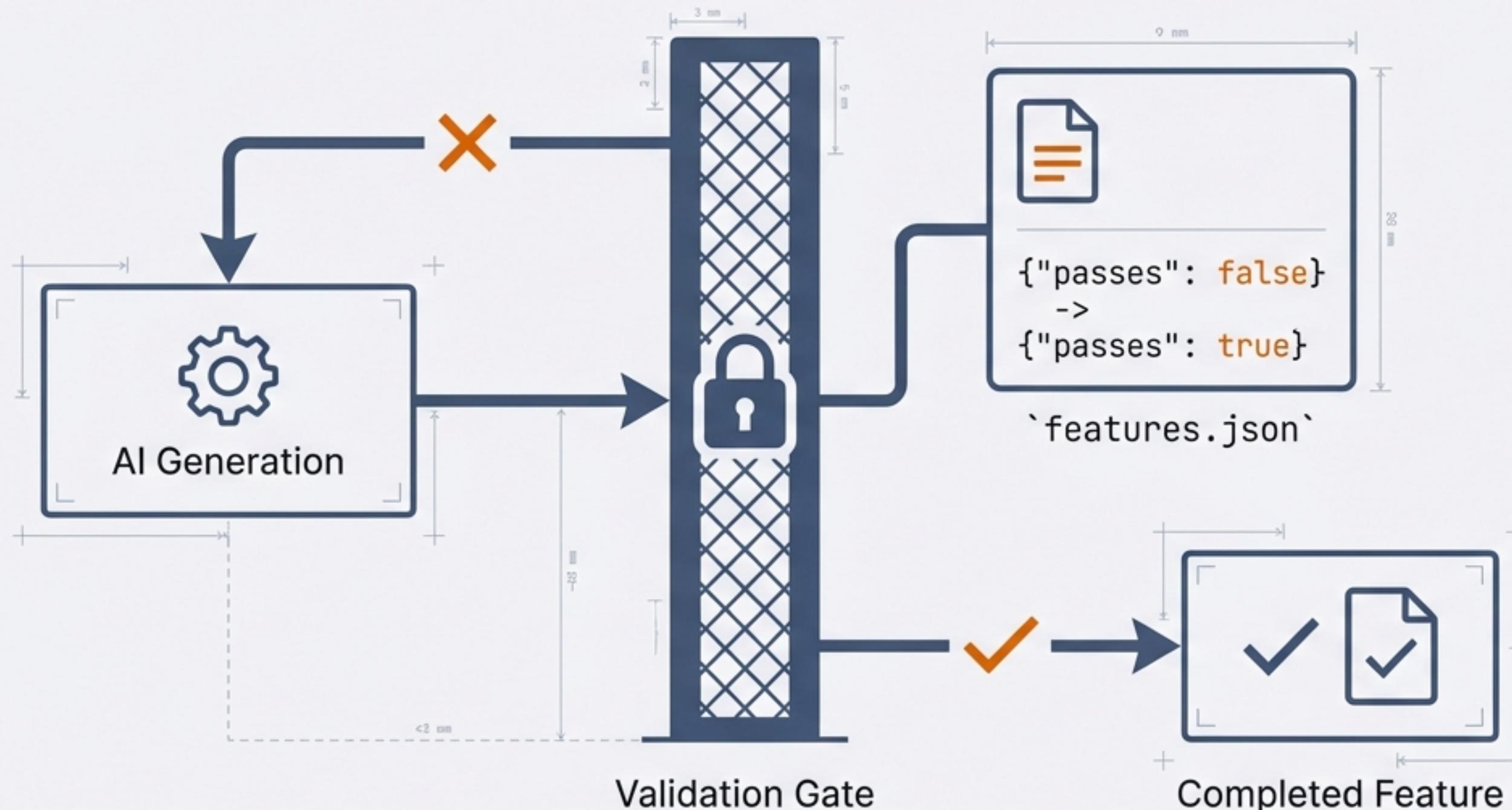


**Core Concept:** 自动化编排 Harness，解决 LLM “无状态” 痛点。  
循环调用 Claude CLI，将自然语言目标转化为通过测试的代码库。

**The Big Idea:** Evaluation is the new Unit Test — 只有通过了预设评估标准的代码，才被视为完成。

# 理论基础 I: Eval-Driven Development (EDD)

评估即开发 (Tests as Specs)

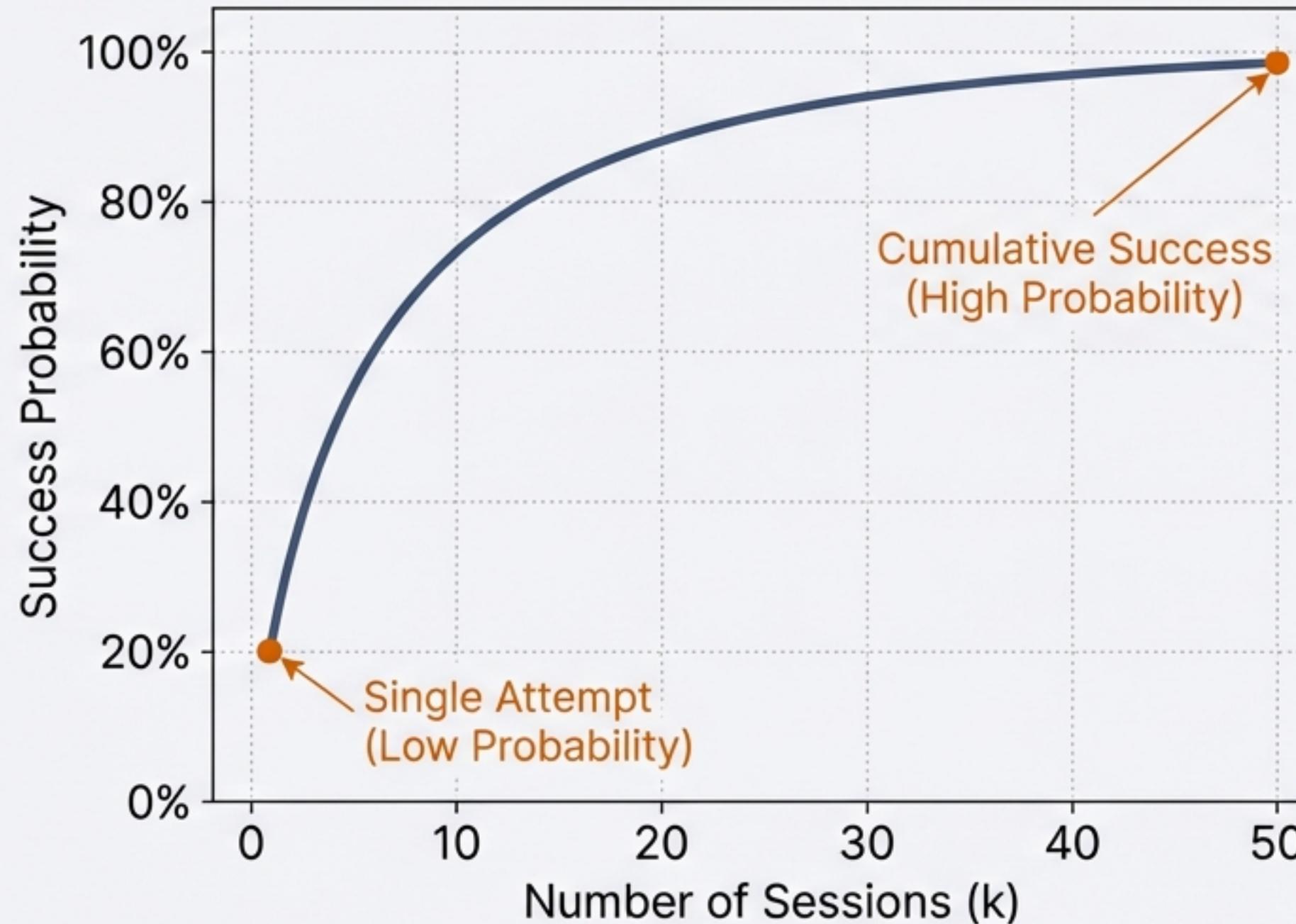


**Core Mechanism:**  
在编写代码前定义成功标准。`features.json`是确定性的验证门控。

**Impact:**  
只有通过具体测试验证，Feature 状态才会翻转。消除“幻觉”和逻辑错误的中间状态。

# 理论基础 II：Pass@k 可靠性度量

Chen et al. (2021) Evaluating Large Language Models Trained on Code



## Insight:

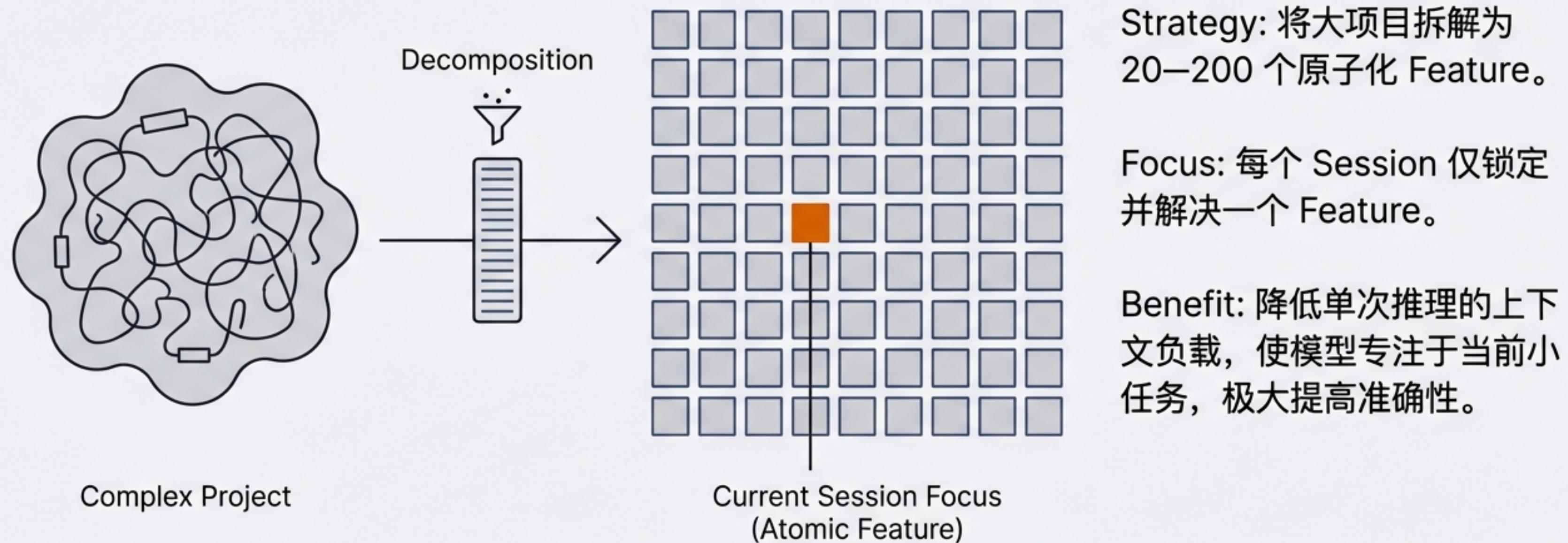
LLM 输出具有概率性。  
单次失败只是概率偏差，不代表能力缺失。

## Application:

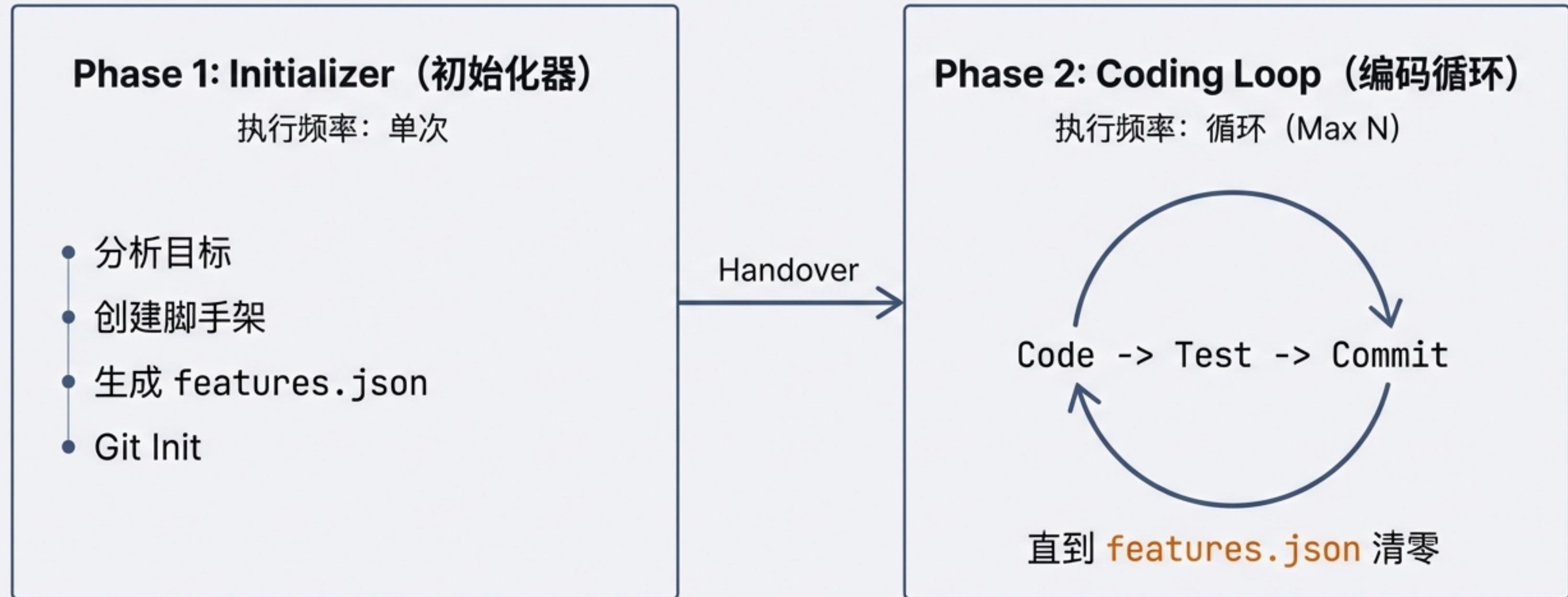
系统允许 50+ 次迭代。  
通过连续重试 (Pass@k)，即使单次成功率低，也能达成 90/90 feature 全部通过的极高最终完成率。

# 理论基础 III：原子化任务分解策略

Jimenez et al. (2024) SWE-bench: Can Language Models Resolve Real-World GitHub Issues?

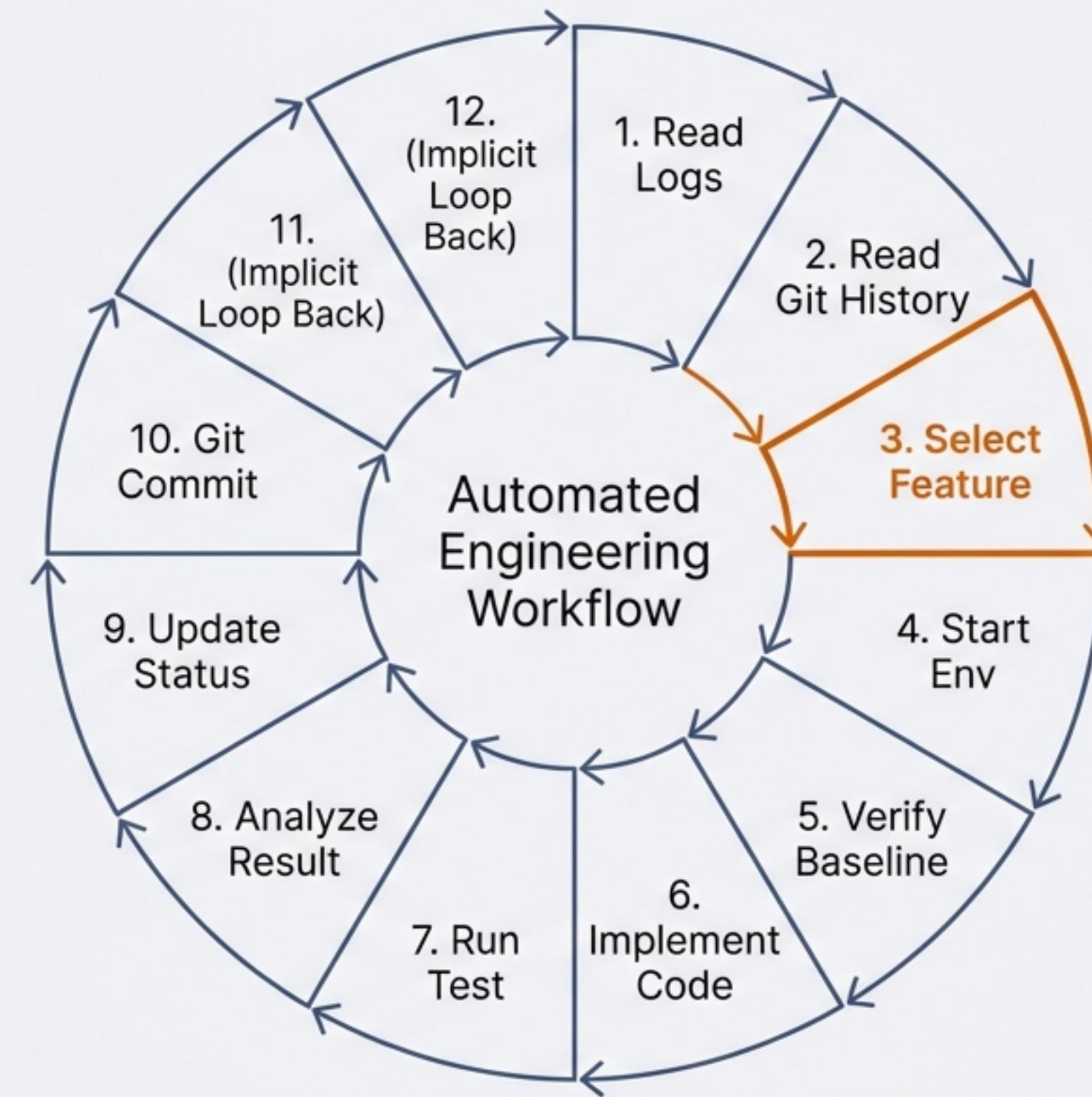


# 系统架构：两阶段自动化流程



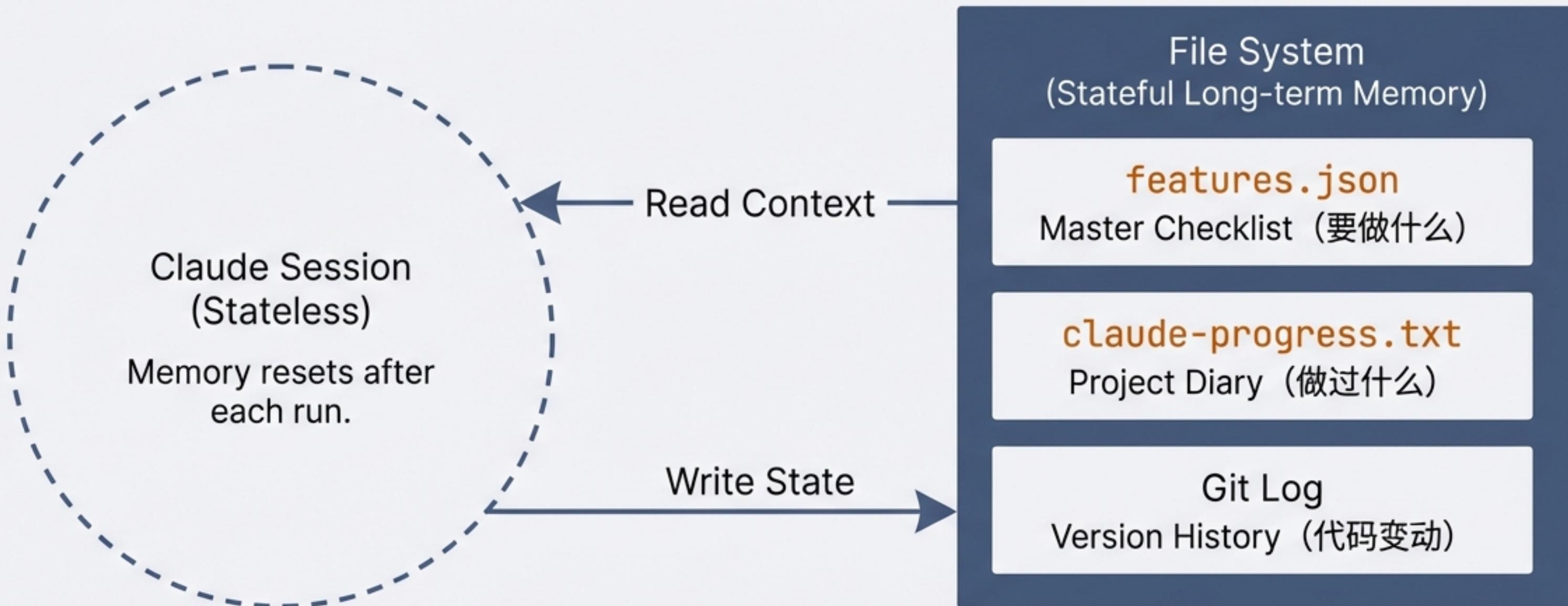
Phase 1 建立地基，Phase 2 负责在此之上不断迭代构建。

# 核心机制：12步 Session 闭环



这是一个严密的工程化流程。每一步都有据可查，每一步都在向目标推进。

# 状态持久化：跨越 Context Window 的限制



即使 Claude 本身是无状态的，Harness 确保了整个开发过程是'有状态'且连续的。

# 工程特性：可靠性与监控



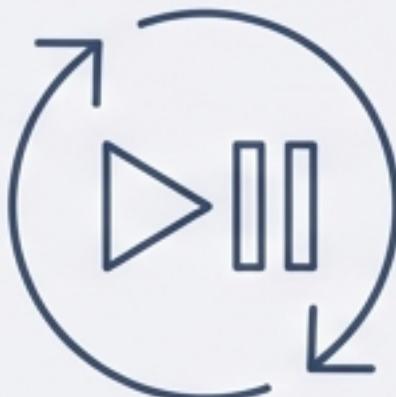
## Watchdog 双重守护

30 分钟硬超时 + 10 分钟空闲超时，防止卡死。



## 指数退避 (Exponential Backoff)

API 失败时智能重试，5 次后安全停止。



## 断点续传

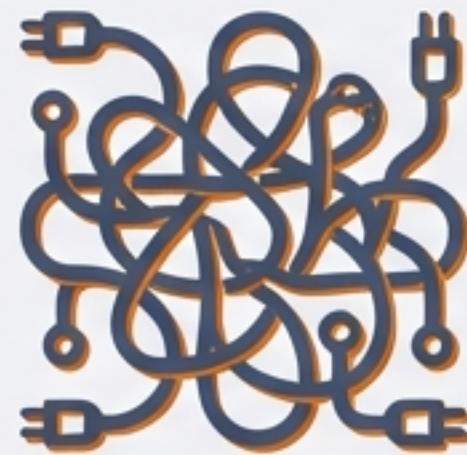
`--skip-init` 参数支持从中断处无缝继续。



## Zero-Token 监控

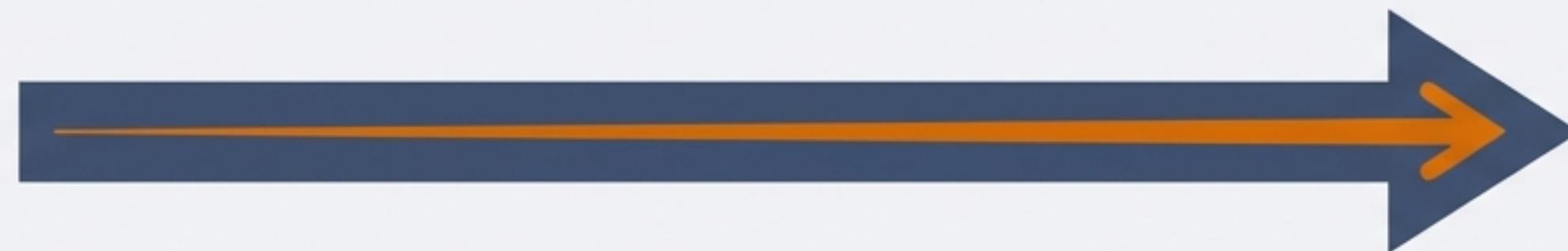
`monitor.sh` 实时仪表盘，不消耗 Token。

# 价值 I：角色转变 —— 从 Coder 到 Product Owner



The Coder

关注 Syntax, Libraries, Debugging



The Product Owner

关注 Requirements, Logic, User Experience

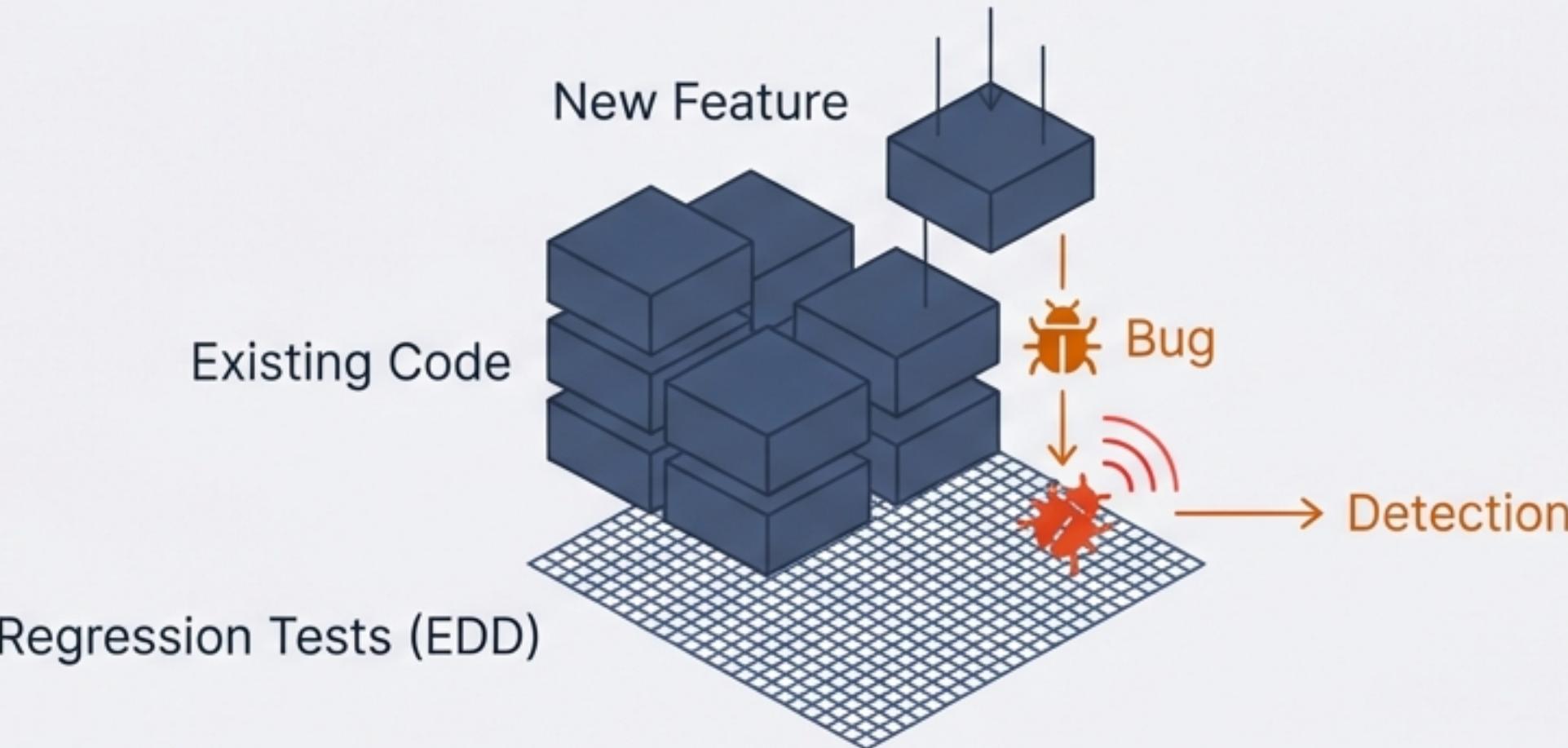
**Insight:** 传统的编程要求精通 '**How**' (语法)；本系统只要求清楚 '**What**' (目标)。

**Empowerment:** 用户定义 `features.json`, Agent 负责实现所有技术细节。

---

“编程不再是门槛，逻辑清晰的需求描述才是核心竞争力。”

# 价值 II：自动化的质量安全网

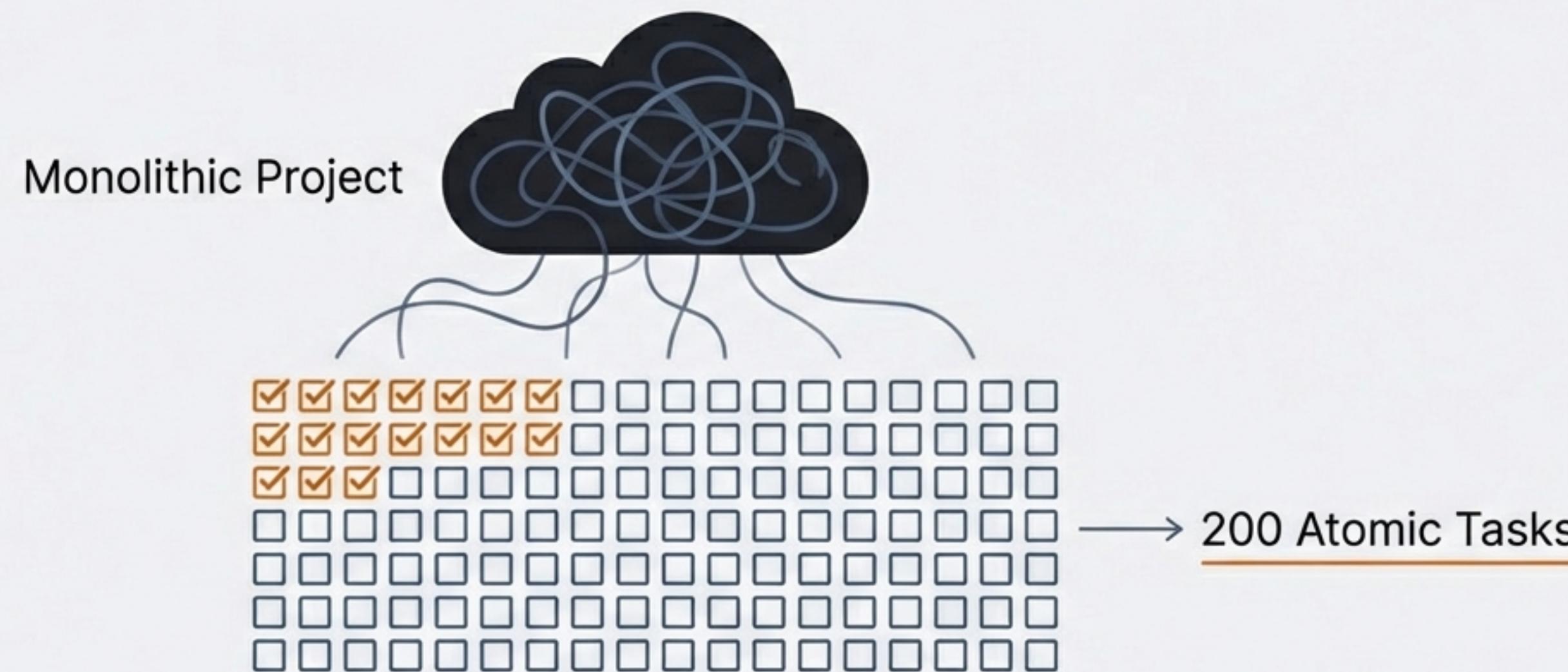


**Pain Point:** 非专业开发者常面临“修复一个 Bug，引入两个新 Bug”的回归地狱。

**Solution:** 全自动回归测试。EDD 确保新功能通过测试，且不破坏旧功能。

**Benefit:** 强大的工程安全网，让非专业人士敢于“大胆修改”。

# 价值 III：复杂度的降维打击



**Pain Point:** 庞大的项目需求带来巨大的认知负荷。

**Solution:** 系统将大项目拆解为小任务，逐一击破。

**Benefit:** 用户看到的是稳步增长的进度条，而不是空白的编辑器。将复杂的软件工程程转化为线性的、可管理的流程。

# 案例实证：OpenClack 项目



- **Execution:** 约 24 个自动 Session 完成开发。
- **Outcome:** 生成了完整的测试套件和 CI/CD 流水线。
- **Proof:** 验证了该系统能处理真实世界的复杂性（Electron/TypeScript）。

# 总结：软件工程即服务 (SEaaS)



- **Evolution:** 从“辅助写代码”进化为“代理构建产品”。
- **Final Thought:** 拥抱 Eval-Driven Development, 让 AI 为你完成最后一公里的交付。
- **Call to Action:** Software Engineering as a Managed Service.