执行成本降低后

AI-native 团队如何重构工作系统

辛海洋 · 2026年8月31日 · 可可乐博

摘要

AI 系统正在降低信息检索、内容生成、软件开发和流程执行的边际成本。组织由此获得了扩大工作吞吐的条件，但执行速度提高并不自动转化为更可靠的产品判断或工程结果。候选生成与工具调用的边际成本下降，不等于单位被接受成果的总成本下降；后者还包括上下文准备、审查、协调、等待、返工和恢复。任务所需的组织上下文是否完整，Agent 的行动是否受到权限与质量门槛约束，每次运行是否留下可检查并可用于后续工作的结果，开始成为更具决定性的条件。

本文综合 Uber Engineering 关于 Software Factory 的公开案例、Ashwin Gopinath 关于 Company Memory 的系列论述、三篇关于 Agent Loop 与任务图的实践文章、Joway 与 Tw93 关于 AI Coding 责任及工程闭环的实践论述、Khazix 关于 FDE 现场交付的访谈文章、Alex Lieberman 关于 AI-native 公司特征的经营清单、David Ondrej 关于 Agentic Engineering 设置的个人实践，以及可可乐博现阶段的实践框架，提出五级能力演进：个人工具使用、工作流复用、Agent 托管执行、共享组织状态、基于结果的持续进化。该路径的重点不在 Agent 数量或自动化比例，而在组织能否将证据、判断、内容资产、执行与结果建立为同一条可追溯状态链，并在明确的人类责任下持续运行。

本文建议以 Context、Harness、Loop 作为 AI Native 团队的基础结构。Context 提供完成任务所需的事实、决策原因和历史状态；Harness 提供工具、权限、任务图、预算、验证门和停止条件；Loop 负责执行、验证、人工修正、结果写回、版本更新和撤销。产品、内容与研发团队分别管理判断状态、内容资产状态和执行状态；人的工作重点集中于目标定义、证据判断、委托设计、例外处理、结果责任、学习与演进六类责任。

本文与演示稿的关系

演示稿用于建立主线和组织决策，本文用于补充概念定义、实施条件、适用边界与指标口径。重点内容在两份材料中的对应关系如下：

- 演示稿第 7 页呈现事实、互动与推理、行动三类组织记忆；本文解释三类记忆为何不能被全文检索或会议摘要替代；
- 第 3 页说明工具能力扩展后，团队差异转向任务定义、上下文、验证和结果责任；本文说明这一判断的证据边界；
- 第 10 页呈现 AI Native 转型的五级能力演进；本文说明各级的进入条件与常见误判；
- 第 13、15、16 页分别说明产品、研发、内容团队的工作要求；本文补充状态链、权限边界与首批适用任务；
- 第 14 页要求每项 AI 生成变更具有具名工程负责人；本文补充负责人需要理解、说明和验证的具体内容；
- 第 18 页将团队成员的工作重点归纳为目标定义、证据判断、委托设计、例外处理、结果责任、学习与演进；本文说明这些是责任类型而非固定岗位；
- 第 20 页以合同验收、系统验收、一线采用、业务或学习结果四个层次区分“通过验收”，并联合观察结果、监督负担、单位成本与风险恢复；本文补充各层指标口径；
- 第 21 页说明真实失败如何转化为回归测试、原因规则和工作流版本，并给出最小运行记录与 ADR 的落盘方式；
- 第 22 页给出十二个月里程碑，并补充并发 checkout 隔离、单一 merge/publish Owner 与 M5 供应商退出测试；第 23 页要求试点同时证明一线有用、内部可接管和真实交付成立。

演示稿中出现的框架、阶段和指标属于基于公开材料及可可乐博现阶段实践形成的分析建议。Uber 的数据属于企业自述，Company Memory 具有明确的商业产品立场；本文没有将这些材料解释为普遍有效性证明。

一、组织改造开始于工作单位的变化

在个人使用阶段，AI 系统主要承担搜索、整理、写作、分析和编码。其价值通常表现为某个步骤耗时下降，但完整任务仍由人临时拆解、组织上下文、选择工具并检查结果。这类使用可以形成明显的个人效率差异，却不必然形成稳定的组织能力。人员变化、任务环境变化或模型升级，都可能使原有做法失效。

同一模型和工具不会自动产生同等质量的工作结果。Joway 将其中一种工程实践概括为“尽职编程”：工程负责人不只接收生成结果，还需要理解关键设计、判断上下文是否充分，并能够回答审查中的技术问题。本文将这一观点解释为责任结构的变化；原文没有提供个体产出方差变化的测量，因此不据此宣称 AI Coding 已经扩大了工程师之间的质量差异。

组织级转型要求工作的基本单位从一次对话或一次生成，转向可重复、可验证的工作流。每条工作流需要明确目标、输入、输出、工具、权限、验收、异常处理和负责人。只有这些条件被显式化，Agent 才可能从个人助手转变为受治理的执行单元。

Uber Engineering 将 Agent 使用划分为个人会话、带 Skill 的会话、通用 Agent 和专用 Managed Agent 等层级。其公开文章强调，专用 Agent 需要围绕真实工作建立 Benchmark，并同时衡量结果质量、可靠性、延迟和单位结果成本。代码审查、CI 修复、告警处理和维护任务之所以适合逐步托管，是因为它们具有相对稳定的边界、可观察的结果和明确的升级路径。

Uber 披露的使用规模和成本变化属于企业自述，不能直接外推为其他组织的预期收益。该案例更可靠的启发是：Agent 需要被当作生产系统管理，而不是作为个人工具采购。模型只是运行变量，真实任务评测、统一运行环境、权限控制、运行记录和人工升级共同构成组织能力。

二、AI Native 转型的五级能力演进

五级演进用于判断某条工作流当前具备什么组织能力，不用于给公司贴统一成熟度标签。同一团队的不同工作流可能处在不同级别。

L1：个人工具使用

个人使用 AI 系统完成搜索、整理、生成或编码。收益主要体现为步骤耗时下降，任务拆分、上下文准备、验收和责任仍依赖个人。人员变化或模型升级后，方法往往难以复现。

L2：工作流复用

有效做法被表达为模板、Skill、检查表或标准工作契约。同类任务开始具有稳定输入、输出和验收要求，但运行仍以人为主。此级别的判断标准是方法是否能由另一位成员复现，而不是 Prompt 数量。

L3：Agent 托管执行

部分任务进入受治理的运行环境，具备身份、工具、权限、预算、日志、停止条件、人工审批和恢复机制。适合托管的任务需要边界稳定、结果可观察、失败可恢复，并拥有明确升级路径。

L4：共享组织状态

产品判断、内容资产和研发执行不再由不同工具维护互相冲突的局部事实。组织共享跨团队行动所需的最小契约、引用与状态关系，使其可检查、可纠正、可版本化、可授权并可迁移；原始证据可以继续留在分域管理的专业系统中。Agent 根据职责读取不同视图，异议与不同解释不会被统一状态覆盖。

L5：基于结果的持续进化

真实结果、失败轨迹、人工修正和反例进入评估集，进而更新 Context、Skill、Workflow 与治理规则。更新需要经过历史回放、版本比较和具名批准，并保留撤销能力。

五级演进的关键不是权限持续增加，而是组织逐步获得复用、治理、共享状态和结果学习能力。若真实任务没有通过验收、人工监督负担持续增加或恢复机制尚未验证，则不应晋级。

五级能力与后文 M0-M5 路线图承担不同作用。L1-L5 描述某条工作流当前具备的能力状态；M0-M5 描述组织实施改造的先后顺序。两者不要求一一对应，也不应合并为统一的公司等级或第二套自治阶梯。

三、执行系统的上限取决于组织上下文

代码库、PRD、知识库和会议纪要保存了大量组织信息，但它们通常不能完整回答一项工作为何形成当前状态。架构决策可能没有记录被否决方案，产品需求可能省略客户承诺，流程文档可能没有包含真实例外，测试用例也未必说明某个历史客户或合规条件为何不能被破坏。

Ashwin Gopinath 将这一缺口描述为组织记忆问题。他区分了三类相互关联的记忆：

1. **事实记忆**：发生了什么，证据在哪里，由谁负责，何时变化；
2. **互动与推理记忆**：为什么这样判断，存在什么异议，哪些假设和问题尚未关闭；
3. **行动记忆**：什么状态变化应触发行动，真实路径如何运行，谁批准或修正，结果是否符合预期。

Tw93 对 Mole 开发过程的描述提供了一个工程场景中的对应关系：测试和发布状态保存可核对的事实；Rules 保存某种实现为何被拒绝、哪些边界不能突破；回归测试、Skills、CI Gate 和发布检查保存行动方式。该案例说明三类记忆可以由不同工程对象承载，但作者披露的代码量、测试数量和提交数量属于项目自报，不能据此推断普遍效果或推荐比例。

这一区分说明，组织记忆不能等同于全文检索或会议摘要。Agent 完成任务所需的通常不是更多文本，而是能够区分来源事实、产品解释、未验证假设、已批准决策、条件性承诺和实际结果。没有这些区分，检索系统可能为 Agent 提供大量相关材料，却无法说明哪一项信息仍然有效，哪一项仅代表历史意见，哪一项决定了当前行动权限。

组织记忆也不等于记录一切。进入共享状态的信息应服务于明确任务或治理目的，并具备来源、权限、适用范围、有效期与保留期限。与任务无关或未经授权的员工、客户、教师、学生和课堂数据不应因为“未来可能有用”而被无边界收集；长期资产还需要支持更正、撤回、删除和访问审计。可追溯性要求保存足以解释运行与责任的最小记录，不要求复制所有原始内容。

组织记忆也不应由每个工具分别维护。如果会议工具、代码 Agent、CRM 和项目系统各自形成局部记忆，公司仍然会产生相互冲突的局部真相。较为稳妥的做法是建立公司控制的最小共享状态，保存跨团队行动所需的契约、引用与关系；原始证据可以留在各自受控的专业系统中。不同角色通过有权限的引用读取相关证据，并保留来源、版本、异议和不同解释。共享状态的目标是让行动依据一致且可追溯，不是复制全部数据或形成单一解释。

这一架构判断来自作者及其创业项目的公开论述，目前并非经过独立评测的普遍结论。其适用价值主要在于指出一个可被实际检验的问题：当 Agent 输出在评审中被拒绝时，失败究竟来自模型能力不足，还是来自组织没有向系统提供必要的决策原因和约束。

四、Context、Harness、Loop 构成共同基础

综合两组材料，可以将 AI Native 团队的基础结构概括为 Context、Harness 和 Loop。

Arvind Jain 对 Glean:GO 的总结将主动建议、工作流重构、前置安全、结果衡量与明确所有权并列为企业 AI 转型条件，与本文的总体方向一致。该材料来自企业 AI 厂商负责人对客户活动的总结，本文仅将其作为外部旁证：主动建议不自动获得行动权限，CAIO 也不构成必设岗位；文中客户节省工作时间的数字因缺少完整计算口径，不作为量化证据。

1. Context：系统基于什么理解任务

Context 不只是文档集合，而是完成当前任务所需的受控状态。它至少应包含：

- 事实、来源、版本、有效期和访问权限；
- 用户或客户证据、产品解释和反例；
- 假设、异议、决策、承诺及其责任人；
- 架构、数据、安全、兼容性和合规约束；
- 历史执行、失败路径、人工修正和业务结果。

Context 的质量取决于来源和关系，而不是材料数量。第一阶段无需建设全公司大图谱，可以从一条产品到研发的真实链路建立小型、可核查的状态模型。

2. Harness：系统如何行动并受到约束

Harness 负责把模型能力组织为可治理的执行环境，包括 Agent、Skill、Tool、身份、权限、任务图、预算、超时、测试、日志、审批和回退。其目的既是提供行动能力，也是限制系统在证据不足或风险过高时继续行动。

Graph 在本文中不是第四个基础层，也不是高于 Loop 的新阶段。它是 Harness 中的执行拓扑，用于规定节点如何读写运行状态、哪些步骤存在真实数据依赖、哪些工作可以并行、结果如何汇聚、验证门如何阻断以及失败影响到什么范围。运行 State 回答“这一轮执行到了哪里”；组织 Context 回答“行动基于哪些事实、决策、约束、权限和历史”。Checkpoint 可以恢复当前运行，但不能替代跨任务、跨团队的组织记忆。

每条工作流应显式定义以下行为：自动执行、提出候选方案、请求审批、等待、拒绝行动、升级和回滚。技术上能够调用某项工具，并不代表组织上已经授权该项行动。对于生产写入、删除、迁移、资金操作、权限调整或公开发布等高后果行动，批准应被实现为不可绕过的边界条件；批准状态、证据和目标版本同时满足前，执行节点应保持不可达。

3. Loop：一次执行如何转化为后续能力

Loop 将运行结果纳入组织学习。成功案例可以成为经过验证的先例，失败案例进入 Benchmark 和风险记忆，人工修正形成改进信号，业务结果用于判断工作流是否只是技术完成，还是实现了预期目标。Loop 必须包含能够拒绝结果的检查，以及重试、时间、费用或信息增益的停止上限；仅重复执行而不能使本轮失败的过程，不构成质量闭环。

Loop 需要区分两条返回路径。只有当输入输出契约稳定、共享状态未被污染、失败域可证明隔离且全局不变量仍成立时，短纠错边才只退回未通过验证的工作单元；否则应回退到最近一致状态并重新验证受影响范围。长学习边把具名确认后的失败、人工修正和业务结果转化为下一轮 Context、评估集和 Workflow 约束。前者修复当前任务，后者改变未来运行。

Skill 和 Workflow 的更新不宜自动成为组织规则。系统可以根据运行轨迹生成候选版本，但应经过真实任务回放、版本比较和具名批准，再进行受控发布。若新版本造成质量下降或权限风险，应能够撤销。

五、产品团队需要管理可追溯的判断状态

产品团队的主要产物不应只被理解为 PRD。PRD 是某一时点的表达形式，底层需要保留从证据到结果的判断状态：

原始证据 → 产品解释 → 异议或反例 → 假设 → 决策 → 承诺 → 行动 → 结果

这条状态链可以减少三类问题。第一，访谈摘要或高频反馈被直接写成确定需求；第二，新版 PRD 覆盖了旧决策的原因和未解决异议；第三，功能上线后没有回到原始假设，导致交付完成被误认为问题已经解决。

AI 系统可用于整理证据、定位冲突、生成候选状态、检查 PRD 缺口和汇总实验结果。候选决策、承诺、责任人和假设应由具名负责人确认后进入共享状态。涉及敏感交流、个人评价或未经授权的原始材料时，系统应保留分层权限，不以建立公司记忆为由扩大可见范围。

产品团队的改造重点包括：

- 将会议总结升级为决策、假设、异议、承诺和开放问题的候选提取；
- 为重要产品判断保留证据来源、样本范围、反例和失效条件；
- 使新证据能够定位其影响的假设、路线图决策和客户承诺；
- 为发布设置观察窗口、目标指标、反指标和撤回条件；
- 将发布结果写回判断状态，而不是停留在复盘文档。

六、研发团队需要管理可验证的执行状态

研发工作不应只留下代码 diff 和测试通过记录。中高风险任务还需要保存目标、约束、方案、被否决路径、验证环境、人工判断、发布结果和恢复方式。相应的执行链可以表示为：

目标 → 约束 → 方案 → 变更 → 测试 → 固定证据 → 批准 → 发布 → 结果写回

每项 AI 生成变更都应有具名工程负责人。负责人需要能够说明任务目标、关键设计、依赖关系、潜在副作用、未覆盖情形、验证证据、兼容性和回退条件，并记录接受或拒绝 Review 意见的依据。架构图、设计文档和面向产

品团队的实现说明可以辅助形成共同理解，但不能替代对关键代码、测试、运行状态和真实验收结果的检查。

研发 Agent 在读取代码之外，还应获得与当前任务直接相关的 Decision Context 和 Failure Context，包括架构决策、历史事故、失败迁移、客户兼容性、安全要求和未验证假设。合并后需要写回能力、约束、风险或技术债的变化，并说明哪些文档、测试和产品承诺需要同步更新。

异常路径尤其需要结构化保存。Preview 通过但生产失败、依赖人工操作恢复、特定客户环境需要兼容处理等情况，不应只停留在聊天记录。它们应带有来源、版本、适用范围和失效条件，供下一次执行读取。

验证设计需要覆盖“流程看似成功、业务状态已经错误”的情形。例如，命令返回成功但应用并未更新、旧任务延迟返回后覆盖新结果、源码正确但线上仍提供旧产物。此类静默错误要求测试读取外部状态，而不是只检查模型回复、命令退出码或单个 CI 结果。测试数量和测试代码比例可以描述投入，不能单独证明验证有效。

缺陷修复还应形成一条学习链：修复当前问题，增加能够使旧版本失败的回归测试，检查同类路径，把原因和边界写入规则，再将高频检查固化为 Skill 或 Gate。短纠错边完成当前修复，长学习边降低同类问题再次发生的概率。规则与检查也需要版本化并定期移除失效逻辑，避免验证系统与业务代码分离演进。

适合首批托管的研发工作包括 PR 首轮审查、CI 失败归因、候选修复、E2E 与视觉验证、固定 SHA 证据包和发布前检查。自动修改生产数据、扩大权限、处理不可逆迁移或直接发布外部承诺，不宜作为早期自动化目标。

独立 Review Agent 可以在与实现 Agent 不同的运行上下文中读取仓库约束、PR 目标、代码差异和测试结果，用于发现共享盲区。它不构成客观裁判，也不能替代最终审查。具名负责人仍需判断每项发现是否成立、补充必要证据，并将接受、拒绝、豁免或升级的理由写入可追溯记录。

任务图需要成为可审查的工程契约

受治理的研发工作流可以表示为：

Input State → Split → Independent Workers → Code Reducer → Evidence Gate → Accepted State

这条链需要同时满足五项条件：

- State Schema：输入、输出、证据、版本、来源、失败与降级状态具有稳定字段；
- Edge Data：每条依赖能够明确说明跨节点传递的字段，仅有执行顺序而没有数据消费的关系不应被视为真实依赖；
- Deterministic Reducer：去重、排序、分组、格式校验和计数等确定性转换优先由代码完成，再将压缩后的高信号状态交给模型判断；
- Failure Policy：只有当输入输出契约稳定、共享状态未被污染、失败域可证明隔离且全局不变量仍成立时，失败单元才可独立重试；否则回退到最近一致状态并重新验证受影响范围。有副作用的节点需要幂等或具备补偿机制；
- Budget and Stop：最大并行宽度、重试次数、运行时间、费用和人工升级条件在执行前确定，并在运行时生效。

并行只适合可以独立拆分的工作。其价值在于缩短关键路径或增加独立有效覆盖；重复调查、上下文膨胀、汇聚和验证成本可能抵消并行收益。因此，Agent 数量不构成先进性指标，任务图需要以真实依赖、关键路径和单位被接受成果成本评价。

七、内容团队需要管理可追溯的内容资产状态

内容团队的组织能力不应以生成数量或文件数量衡量。内容进入真实产品、课程或传播场景之前，需要同时保留来源、权利、适用对象、专业审查、版本关系和使用结果。相应的状态链可以表示为：

来源与授权 → 内容解释 → 结构化对象 → 候选生成 → 专业审查 → 批准发布 → 采用与修订 → 结果写回

这条状态链用于减少四类风险。第一，来源或授权不明的材料进入长期资产；第二，生成内容缺少适用对象、学科范围或使用条件；第三，新版本覆盖旧版本，却无法说明修改原因和影响范围；第四，内容发布后只统计数量，没有记录采用、修改、拒绝、纠错和实际效果。

内容团队的改造重点包括：

- 为素材、概念、活动、案例和评价标准记录来源、授权、适用对象、版本与有效期；
- 将高频内容拆分为可检索、可组合、可替换的结构化对象，而不是只保存最终文档；
- 为事实、教学、语言、版权和安全建立专业审查规则，并保留具名批准；
- 记录资产之间的引用、派生和适用关系，使上游修正能够定位受影响版本；
- 将采用、人工修改、拒绝、纠错、撤回、复用和真实使用证据写回内容状态。

AI 系统可以协助检索、拆分、标注、生成候选内容和定位冲突，但不能自行确认来源权利、专业适用性或对外发布。涉及学生数据、课堂记录、未公开客户材料或受版权保护的原始内容时，应遵守授权、最小化和访问控制要求。

八、产品、内容与研发需要共享工作契约

产品判断、内容资产和研发执行之间需要一个共同工作契约。该契约至少包含：

- 目标结果与验收标准；
- 用户或业务证据及其适用边界；
- 尚未验证的假设、已知异议和非目标；
- 内容来源、权利、专业标准和适用范围；
- 技术、数据、安全和兼容性约束；
- Outcome Owner、执行负责人和 Approver；
- 验证环境、发布条件和回退机制；
- 结果观察窗口与写回责任。

该契约的作用不是增加文档负担，而是减少产品、内容、设计、研发、测试和管理之间反复重建上下文。Agent 可以帮助检查契约完整性、定位矛盾和生成执行计划，但不能自行补写未经确认的业务目标、内容授权、专业标准或发布承诺。

九、团队成员的工作重点转向六类责任

AI Native 团队并不取消专业分工。随着执行成本下降，团队成员的工作重点需要从重复生产转向六类不可外包的责任：

- **目标定义责任**：选择值得解决的问题，明确成功、失败、优先级、非目标和停止条件；
- **证据判断责任**：判断来源、样本、反例、专业标准和证据是否足以支持决定；
- **委托设计责任**：定义运行状态、节点边界、真实依赖、工具、权限、预算、验证门和停止条件；
- **例外处理责任**：处理低置信、冲突、高风险、权限不足、失败恢复、人工接管和升级；
- **结果责任**：理解并说明关键选择，完成验收、批准发布、观察实际结果，并决定继续、修订、回滚或撤回；
- **学习与演进责任**：判断哪些失败、修正和业务结果应写回 Context、评估集、Skill、Workflow 或治理规则，并批准相应版本变化。

这六类是责任类型，不是六个固定岗位。目标定义与证据判断主要约束 Context，委托设计与例外处理主要约束 Harness，结果责任与学习演进主要约束 Loop。Outcome Owner、Workflow Steward、Agent Platform Owner、Evaluation Owner 或专业审查者都可以承接其中一项或多项责任。高风险、跨团队和长期工作流应完整记录责任链；低风险短任务可以由同一人合并多项责任并简化记录，但不能取消结果责任与必要证据。同一个 Agent 不应同时生成、验收并批准自己的结果；同一位人类负责人也不应在缺乏独立证据的情况下，将形式上的流程完成视为结果成立。

Agent 系统可以承担执行，具名负责人仍承担对关键选择的理解、解释和最终责任。人类判断也不能被缩减为流程的第一步和最后一步；运行中的冲突、低置信、阻塞、权限不足、异常恢复和高后果行动都可能触发具名介入。系统是否完成任务，应由外部状态、测试、用户验收或业务结果证明，不能依据模型的完成声明判定。Joway 所称“尽职编程”在本框架中对应六类责任的综合履行，不构成第七类责任，也不构成新的成熟度等级。

委托设计与例外处理需要明确区分。委托设计发生在执行之前，规定工作如何被拆分、允许调用什么工具、何种证据足以通过验证以及何时停止；例外处理发生在运行期间，处置冲突、低置信、高风险、权限不足和失败恢复。结果责任面向当前产出，学习与演进面向未来版本。六类责任构成“意图、依据、委托、介入、承担、学习”的责任链。

十、衡量方式覆盖结果、监督负担与风险

Agent 数量、调用次数、Token 总量和 AI 参与率可以反映使用情况，但不能单独证明组织价值。衡量体系不宜压缩为一个北极星比率，而应联合观察四组不可相互抵消的指标：结果与真实价值、监督与等待负担、单位成本、风险与恢复。

“通过验收”需要进一步区分四个层次：

1. **合同验收**：约定的交付物、范围和格式是否满足；
2. **系统验收**：技术状态、数据状态、版本和运行环境是否成立；
3. **一线采用**：成员是否在真实工作中持续使用并获得可观察收益；
4. **业务或学习结果**：预期的用户、业务或学习变化是否发生。

四个层次不能互相替代。合同完成不等于系统状态正确，系统通过不等于一线采用，一线使用也不等于业务或学习结果成立。单位成果成本可表示为：

> **模型调用、工具运行、人工复核、等待与返工总成本 ÷ 指定验收层次的结果单位。**

分母必须明确采用哪一层结果口径。监督时间是需要优化的成本，但高风险工作中必要的专业判断、批准和接管能力不能被无条件压低。

产品团队可观察证据链接率、从证据到批准决定的周期、开发开始后的需求变更、无负责人承诺数和发布后结果写回率。研发团队可观察一次验收率、review minutes / merged PR、返工、revert、生产缺陷、cost / accepted PR、固定证据完整率和恢复成功率。内容团队可观察来源与授权完整率、专业审查通过率、人工修订量、有效复用率、纠错与撤回时延，以及使用结果写回率。

研发 Review 的质量需要与结果证据联结。建议同时观察有效问题闭环率、闸门后缺陷逃逸、无证据覆盖、重复出现的审查问题、修复时延和无充分依据的人工覆盖。评论数量、Review 轮次和停留时间属于过程描述，不能单独证明审查充分，也不宜作为鼓励增加审查负担的目标。独立 Review 应按影响范围、可逆性和失败成本决定是否启动，并设置最大轮次、时间和证据停止条件；递归要求模型继续寻找固定数量的问题，可能持续产生低价值或虚构发现。

测试数量、测试代码比例和绿色通过率同样不是质量本身。研发侧还需要观察静默错误是否被识别、闸门后缺陷逃逸、同类问题复发、不同交付状态是否一致，以及真实用户是否收到被验收的产物。AI 系统可以执行测试和候选修复，人仍需决定什么证据足以通过、哪些动作必须保持不可达、何时停止或升级，并对最终结果承担责任。

跨团队还需要关注上下文重建时间、相同问题重复调查次数、冲突状态解决时间、人工审批队列、无 Owner 行动、越权行动、升级时延和回退成功率。多 Agent 并发后，监督负担应进一步表示为人工注意力队列：按影响范围、阻塞关系、待批准状态与截止时限排序，并观察高优先级等待时长、积压数量、升级响应时间和接管后的恢复时间。任务图的运行健康可观察失败单元率、超过停止上限的比例、错误逃逸、有效阻断、人工抽检差异、人工介入位置、轨迹完整率和关键路径延迟。Verifier 阻断率、压缩率或并行数量不能单独解释：阻断率为零既可能表示上游质量较高，也可能表示检查无效；压缩率较高也可能伴随证据损失。过程指标需要与验收质量和人工抽检共同解释。

随着 Agent 行动数量增长，即使单次错误率下降，错误绝对数量和人工复核负担仍可能增加。因此，提升自动化权限之前，需要同时确认结果证据、可恢复性、人类批准与接管能力三项条件成立。它们构成不可相互补偿的联合证据门：任何一项缺失，都不应由另外两项的高分抵消。具体行动再依据影响范围与可逆性分流；模型自报置信度只作为次级信号。

指标需要先建立基线，再在同一工作流内比较版本变化。产品、研发和内容的结果单位不同，不宜用统一分数进行横向排名；可比较的是每条工作流在质量、监督时间、风险和实际结果上的变化。Agent 数量、Token 总量和 AI 参与率属于采用指标，只能解释系统使用程度，不能单独支持权限晋级或资源投入决定。

十一、可持续转型需要积累五类可迁移资产

十二个月路线图规定时间顺序，但可持续能力取决于每次真实任务留下什么组织资产。建议持续积累以下五类资产：

- **上下文资产**：来源、版本、关系、权限、约束、决策原因和历史状态；
- **评估资产**：真实任务集、反例、失败样本、验收门槛、基线和人工判断记录；

- **工作流资产**：Skill、Workflow、Graph Spec、State Schema、节点输入输出、工具契约、版本、运行轨迹和适用范围；
- **治理资产**：身份、权限、审批、预算、Checkpoint、审计、停止、回退和人工接管记录；
- **结果资产**：Accepted State、显式未完成或降级信息、采用、质量、成本、失败、修正和业务影响，并能够更新下一轮 Context 与评估集。

五类资产可以由一条最小运行记录连接：

task → attempt → Agent/Skill/Model → tools/permissions → change → tests/evidence → human intervention → result → cost → approval

该记录用于把人工反馈关联到具体任务、模型、工具权限、变更、证据和批准，而不是形成无法定位原因的笼统评价。重大架构决定可用简短 ADR 保存当时 Context、决定、原因、替代方案和后果；后续决定变化时新增取代记录并保留历史，不直接改写旧决定。

Review 过程产生的异议、负责人回应、测试证据和最终决定需要进入上述资产，而不是停留在对话界面。被否决方案及理由进入上下文资产，反例与逃逸缺陷进入评估资产，Reviewer 配置与验证门进入工作流资产，批准、豁免和回退进入治理资产，返工、缺陷与用户影响进入结果资产。

一次真实失败应尽量同时留下五类记录：失败样本和回归测试进入评估资产；原因、边界和被否决路径进入上下文资产；可复用的 Skill 或 Gate 进入工作流资产；批准、恢复和回退进入治理资产；用户影响与修正结果进入结果资产。外部 FDE 或其他现场交付也应采用同一回流方式，否则缺陷修复与客户项目都只能形成一次性交付，不能成为组织后续可复用的能力。

五类资产之间存在两种反馈。失败域可证明隔离时，短纠错边将失败单元送回当前 Workflow；否则回退到最近一致状态并重新验证受影响范围。长学习边把经过确认的失败模式、人工修正和结果送回 Context、评估集与 Workflow 版本，目的在于改变后续运行。未经确认的模型总结不应直接成为长期组织规则。

可迁移性是判断能力是否归组织所有的重要标准。更换模型、供应商或 Agent Harness 后，核心 Context、评估集、Workflow 版本、批准记录和运行结果应能够导出并继续使用；替换后的真实任务评测不应出现不可接受的退化。文件导出只是最低条件，内部 Owner 还需能够独立解释工作契约、运行工作流、识别失败、接管与恢复、修改规则、重新评估并在必要时停用。若能力只存在于某位成员的会话、某个模型的隐藏状态、某个无法导出的平台配置或外部供应商的现场经验中，则尚未形成可持续组织资产。资产积累仍应遵守目的限定、数据最小化、授权、有效期和删除要求；“可追溯”不构成无限期保存全部输入、对话或个人数据的理由。

十二、建议的改造顺序

M0：共同状态语言与基线（第1-2周）

产品、内容与研发共同定义 Evidence、Assumption、Dissent、Decision、Commitment、Content Asset、Trigger、Action、Outcome 和 Owner，选择两条工作流测量上下文重建时间、返工和人工复核基线。

M1：只读状态链（第3-4周）

选择一个真实功能，建立从用户证据、产品判断、开发任务到固定 SHA 验收和发布结果的可追溯链。Agent 仅用于检索和生成候选状态，不自动写入公司事实。

M2：人工确认后的状态写回（第2月）

会议、PR 和发布后生成候选状态变化，由具名 Owner 确认后写入。系统保留来源、版本、异议、权限和失效条件。

M3：Trigger 影子运行（第3月）

对承诺没有 Owner、假设被新证据挑战、历史失败模式重现、发布条件满足或失效等情况进行只读告警。独立 Review Agent 同样以影子方式运行，不阻断合并；通过历史回放和人工复核测量有效发现、误报、漏报、重复问题和人工处理成本。

M4：单条受控 Action Loop（第4-6月）

选择一条低风险链路，从已批准需求进入开发、验证、人工发布批准和结果写回。工作流必须具备 wait、abstain、escalate 和 rollback，并通过连续运行和回退演练后再扩大权限。只有在影子运行中证明有效的高风险规则才进入强制 Evidence Gate，其他 Review 发现继续由具名负责人判断。M4 还需要满足以下验收条件：Graph Spec 与 State Schema 已评审；每条 edge 有明确的跨节点数据；失败域可证明隔离时仅重试失败单元，否则回退到最近一致状态并重新验证受影响范围；最大并行宽度、重试次数、时间、费用和人工升级条件已经实际生效。当同一仓库存在并发修改时，不同 Agent 使用隔离 checkout 或 worktree 与独立分支，由单一 merge/publish Owner 依次完成合并、固定 SHA 验收、CI、部署和健康检查。并发宽度只有在冲突、回退和发布锁演练通过后才提高；小项目或单 Agent 工作不要求为形式完整而强制建立 worktree。

M5：可迁移与规模化（第7-12月）

扩展第二条跨团队工作流，并进行模型或 Agent Harness 替换。核心 Context、Benchmark、Skill 版本、批准和运行记录应能够完整导出；更换模型后，真实任务评测不能出现不可接受的退化。M5 还必须通过供应商退出测试：在外部实施者不参与的情况下，内部 Owner 能独立解释工作契约、运行工作流、识别失败、接管与恢复、修改规则、重新评估并在必要时停用。仅完成文件导出或模型替换，不能证明能力已经归组织所有。

十三、当前最值得实施的纵向切口

第一阶段不宜建设全公司知识图谱，也不宜追求全面自动化。建议选择以下真实链路：

用户或教师证据 → 产品判断 → 内容资产或 PRD → 开发任务 → PR → 固定 SHA 验收 → 人工发布批准 → 使用结果 → 状态写回

该切口覆盖产品、内容与研发的主要交接点，并且能够检验以下条件：事实与判断是否区分，来源与授权是否完整，决策原因是否可追溯，Agent 是否获得必要上下文，执行是否具备权限和停止条件，批准是否具名并有证据依据，发布结果是否反向更新产品判断、内容资产和工程经验。

首个试点需要同时完成三项证明。第一，一线成员在真实任务中获得可观察收益，而非只满足管理层的采购意图；第二，内部业务 Owner 与工程 Owner 能独立理解、运行、恢复和改进工作流；第三，固定 SHA、构建或签名产物、线上文件和用户实际收到的版本保持一致。管理层采购、Demo 完成或 CI 变绿都不足以单独证明组织采用和真实交付成立。

首期成果不应以新增 Agent 数量表述，而应以一条端到端状态链是否可检查、可纠正、可恢复和可复现来判断。

结语

AI Native 转型可以被理解为五级能力演进：个人工具使用、工作流复用、Agent 托管执行、组织状态共享、基于结果的持续进化。五级演进说明当前能力位置；五类可迁移资产说明长期需要积累什么；十二个月里程碑说明实施顺序，三者不能相互替代。

组织改造同时涉及任务执行速度与状态治理方式：事实、判断、内容资产、行动和结果开始进入同一个可治理系统。人仍然承担目标定义、证据判断、委托设计、例外处理、结果责任、学习与演进，Agent 则在明确 Context 和 Harness 的条件下执行可验证工作。每次运行留下的证据和修正进入下一次任务，使工作不再依赖关键成员临场重建全部上下文。

判断转型是否成立，应观察组织能否在关键成员缺席时重复完成同类工作，是否能够说明某项结果为何可信，是否能够在失败后恢复，以及新证据出现时能否定位受到影响的决策和承诺。这些条件比模型使用率或自动化比例更接近 AI Native 组织的实质。

主要资料

1. Uber Engineering. *Running a Software Factory Efficiently at Uber Scale*. 2026-08-27. https://www.uber.com/us/en/blog/efficient-software-factory/
2. Uber Engineering. *uReview: Scalable, Trustworthy GenAI for Code Review at Uber*. 2025-08-12. https://www.uber.com/us/en/blog/ureview/
3. Uber Engineering. *Lessons from Building a First-Pass AI PRD Reviewer at Uber*. 2026-05-12. https://www.uber.com/us/en/blog/first-pass-prd/
4. Ashwin Gopinath. *Company Brain: Why Most Companies Have Data But No Memory*. 2026-05-04. https://nanothoughts.substack.com/p/company-brain-why-most-companies
5. Ashwin Gopinath. *Company Brain, Part 3: Interaction Memory*. 2026-05-06. https://nanothoughts.substack.com/p/company-brain-part-3-interaction
6. Ashwin Gopinath. *Company Brain, Part 4: Action Memory*. 2026-05-07. https://nanothoughts.substack.com/p/company-brain-part-4-action-memory
7. Ashwin Gopinath. *Memory Is State, Not a Service*. 2026-05-08. https://nanothoughts.substack.com/p/memory-is-state-not-a-service
8. Ashwin Gopinath. *Cheap Intelligence Makes Accountability Expensive*. 2026-05-14. https://nanothoughts.substack.com/p/cheap-intelligence-makes-accountability
9. Ashwin Gopinath. *Rent the Intelligence. Own the Context.* 2026-05-18. https://nanothoughts.substack.com/p/rent-the-intelligence-own-the-context
10. Mahax. *Agents, Loops, Graphs. Everything You Need to Know in One Place.* 2026-08-05. https://x.com/i/article/2084673534643597312
11. Hanako. *Loops and Graphs: how to stop babysitting agents and only approve the last step.* 2026-08-23. https://x.com/hanakoxbt/status/2091515787366306154
12. Lunar. *Graph Engineering: The Complete Guide to Building Multi-Agent AI Systems.* 2026-08-08. https://x.com/i/article/2086047414532587520
13. Anthropic. *Building effective agents.* 2024-12-19. https://www.anthropic.com/engineering/building-effective-agents
14. Anthropic. *How we built our multi-agent research system.* 2025-06-13. https://www.anthropic.com/engineering/multi-agent-research-system
15. Anthropic. *Effective context engineering for AI agents.* 2025-09-29. https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents
16. OpenAI. *A practical guide to building agents.* https://openai.com/business/guides-and-resources/a-practical-guide-to-building-ai-agents/
17. LangChain. *LangGraph Graph API overview.* https://docs.langchain.com/oss/python/langgraph/graph-api
18. LangChain. *LangGraph Persistence and Interrupts.* https://docs.langchain.com/oss/python/langgraph/persistence
19. Joway. *尽职编程：AI Coding 时代的个体产出差异的来源*. 2026-08-29. https://x.com/i/article/2093682363314155520
20. OpenAI. *Building an AI-native engineering team*. https://cdn.openai.com/business-guides-and-resources/building-an-ai-native-engineering-team.pdf
21. OpenAI. *Introducing upgrades to Codex*. 2025-09-15. https://openai.com/index/introducing-upgrades-to-codex/
22. NIST. *DevSecOps Practices and AI-generated Code*. https://pages.nist.gov/nccoe-devsecops/introduction.html

23. Tw93. *在代码全部由 AI 生成的时代，如何保证产品代码持续迭代、可维护且不腐化*. 2026-08-30. https://x.com/HiTw93/status/2094015568101921254
24. Tw93. *Mole public repository*. https://github.com/tw93/Mole
25. Khazix. *那些被管理层请进公司，却不被一线员工接受的 FDE 们*. 2026-08-31. https://x.com/i/article/2094248428331311104
26. OpenAI. *Forward Deployed Engineer (FDE)*. https://openai.com/careers/forward-deployed-engineer-%28fde%29-sf-san-francisco/
27. Alex Lieberman. *30 Features of an AI-Native Company*. 2026-08-31. https://alexlieberman.com/ai-native-company/
28. David Ondrej. *Agentic Engineering Setup (after 2,000+ hours)*. 2026-08-31. https://x.com/i/article/2094424686499160065
29. David Ondrej. *Personal Agent Skills repository*. https://github.com/davidondrej/skills
30. Vectal Labs. *Agentic Productivity*. https://github.com/vectal-labs/agentic-productivity
31. OpenAI. *Running Codex safely at OpenAI*. 2026-05-08. https://openai.com/index/running-codex-safely/
32. Arvind Jain. *Becoming AI-first is not just a technology strategy. It’s an organizational design problem.* 2026-09-01. https://x.com/jainarvind/status/2094901374714597630

资料边界

Uber 的规模、质量和成本数据属于企业自述，公开资料未提供完整口径或外部审计。Ashwin Gopinath 同时是 Sentra 创业者，其 Company Brain 架构具有明确的产品立场。Arvind Jain 是 Glean CEO，其文章属于对自家客户活动的总结，本文不采用其中缺少完整计算口径的效果数字，也不将 CAIO 视为必要组织形式。Mahax、Hanako、Lunar、Joway 与 Tw93 的文章没有提供独立效果评测，因此仅作为实践观点源；Tw93 披露的 Swift 代码量、测试数量和提交数量无法通过公开 CLI 仓库独立复核。Khazix 的 FDE 文章基于少量匿名访谈，未披露样本量和访谈方法，只用于识别现场风险，不用于推断市场普遍情况。Alex Lieberman 的 30 项清单属于经营者提出的规范性检查表；David Ondrej 的文章属于个人工具栈和工作习惯记录，其 2,000 小时、访谈经历、效率判断、工具选择和模型偏好没有独立验证。公开仓库只能证明相关 Skills 与度量工具存在，不能证明个人或组织效果。本文没有采用其权限绕过、full root、凭据自动查找、订阅价格和工具排行等做法。本文提出的五级演进、三类团队要求、六类责任、指标体系、可迁移资产和实施顺序属于基于上述材料及现阶段实践作出的分析与建议，不构成已验证的普遍效果结论。