AI Agent 产品研究 | 2026.08

从“回答问题”到“完成任务”

AI Agent 产品的交互、工作流与控制权设计

一份面向 AI 产品经理的 Agent 产品研究、真实使用观察与 MVP 设计报告

研究核心问题

当 AI 从“生成答案”升级为围绕目标持续行动后，产品设计的重点将从单轮回答质量，转向目标理解、任务规划、工具调用、权限控制、过程可见、异常恢复与结果验证。本文尝试回答：什么任务值得 Agent 化？AI 应获得多大自主权？用户应在何时介入？一个可用 Agent 产品至少需要哪些机制？

研究时间：2026.07-2026.08.18

版本：2026.08 完整修订版

目录

> **阅读提示**
>
> 本文以产品设计视角为主，不试图穷举 Agent 技术架构。行业事实优先使用官方产品、工程与协议资料；个人实测部分用于观察产品机制，不构成模型性能 Benchmark。

0. 执行摘要

大模型产品正在经历一个关键变化：用户不再只希望 AI “告诉我怎么做”，而是越来越期待 AI “替我把事情做完”。这意味着产品形态从以生成内容为中心的 Chatbot，逐步扩展到能够理解目标、拆解任务、调用工具、读取环境反馈，并根据结果持续调整下一步的 Agent。能力边界的扩大，也同步放大了错误、权限、信任和责任问题。

因此，Agent 产品的核心并不是让模型拥有更多工具，而是让模型在恰当的任务、恰当的权限和恰当的检查点上采取行动。对产品经理而言，真正需要设计的是一套“自主性与控制权的分配机制”：哪些步骤可以自动完成，哪些步骤必须用户确认；出现失败时如何恢复；用户如何理解当前进度；以及最终如何验证任务是否真正完成。

0.1 五个核心结论

1. Chatbot 与 Agent 的核心差异，不在于是否使用大模型或是否简单调用工具，而在于 AI 是否能够围绕目标持续行动，并根据工具与环境反馈动态调整后续路径，直至满足任务完成条件。
2. 不是所有任务都值得 Agent 化。路径稳定、规则明确、结果可预期的任务通常更适合 Workflow；只有当任务存在多步骤、路径不确定、需要工具、需要中间判断并且自动化收益足够高时，Agent 才具有明显价值。
3. Agent 产品最大的产品矛盾是 Autonomy × Control：自主性越强，潜在效率越高，但错误影响范围也越大。产品需要通过权限、确认、撤销、状态可见和 Guardrails 控制“爆炸半径”。
4. Human-in-the-loop 不等于“每一步都让用户确认”。更合理的设计是关键节点介入：高风险、不可逆、对外沟通、涉及资金、意图不清、证据不足或成本异常时暂停，其余低风险步骤在授权范围内自动执行。
5. Agent 的评价标准必须从“工具调用是否成功”升级为“用户目标是否完成”。任务完成率、首次成功率、人工介入率、恢复率、行动准确率、完成时间与单任务成本应被组合评估。

0.2 一张表理解 Agent 产品闭环

阶段	产品要回答的问题	主要机制
Goal / Intent	用户真正想完成什么?	意图识别、参数补全、约束识别、必要澄清
Plan	应该怎么做? 路径是否需要动态调整?	任务拆解、计划预览、协作式规划
Permission	AI 被允许做到什么程度?	最小权限、即时授权、风险分级
Action / Tool	如何读取或影响外部世界?	工具调用、环境反馈、状态更新
Observe / Evaluate	中间结果是否正确?	结果读取、证据检查、进度追踪
Re-plan / Recovery	失败后下一步是什么?	重试、换路径、回滚、升级给用户
Checkpoint	何时必须让用户回来?	人工确认、审批、暂停与继续
Verification	用户目标真的完成了吗?	结果核验、交付物检查、完成条件

> **本文主张**
> 一个真正可用的 Agent，不应只具备 “Go”，还必须具备 “Stop、Pause、Undo、Retry、Explain、Verify”。执行能力越强，控制与恢复机制越不能被视为附属功能。

0.3 为什么是现在：2026 年 7-8 月 Agent 产品演进

时间	行业事件	关键变化	对本文的启发
2026.07.09	OpenAI 发布 ChatGPT Work	从回答问题进一步转向跨应用、多步骤、长时间完成工作	Goal、Planning、Progress、Approval
2026.07.28	MCP 2026-07-28 Specification	Stateless Core、MRTR、Tasks、Authorization Hardening 等进入新规范	Tool、Permission、Long-running Task
2026.07.31	Gemini Enterprise Agent Evaluation GA	Agent 评测覆盖 Tool Use、Trajectory、Grounding、Safety，并进入生产流量评测	Evaluation、Observability
2026.08.12	OpenAI 发布企业 AI 使用研究	企业 AI 使用出现从 Assistance 向 Execution 转移的明显信号	Agent Productization

7 月 9 日，OpenAI 发布 ChatGPT Work，将产品进一步定位为可以从应用和工作流中获取信息、把复杂项目拆成较小步骤，并持续数小时完成工作成果的 Agent。[5] 这里值得关注的并不是“AI 又可以做 PPT”，而是产品的基本单位从 Question / Prompt 开始向 Goal / Finished Work 转移。

7 月 28 日发布的 MCP 2026-07-28 规范进一步加入无状态核心、Multi Round-Trip Requests、正式扩展体系与 Tasks 等长任务相关能力。[6] 这说明基础协议本身也开始显式处理长任务、授权以及执行过程中重新向用户索取信息的问题。

7 月 31 日，Google 将 Gemini Enterprise Agent Platform 的 Agent and Model Evaluations 推向 GA，覆盖 Quality、Safety、Grounding、Agent Tool Use 与 Trajectory，并支持开发与生产阶段的统一评测。[7] 8 月 12 日，OpenAI 企业使用研究则显示，截至 6 月，其企业客户中 Codex 已占 Codex 与 ChatGPT 合计输出 Token 的 64%；这只能代表 OpenAI 企业客户，而不是整个市场，但仍可视为从 Assistance 向 Execution 迁移的一项使用信号。[8]

0.4 研究方法与范围

- 时间范围：重点观察 2026 年 7 月至 8 月 18 日的 Agent 产品与基础设施变化，并使用更早的一手工程资料补充定义、评测与安全问题。
- 信息源：优先使用 OpenAI、Anthropic、Google 与 Model Context Protocol 官方产品、工程、研究和协议文档。
- 方法：Desk Research + 产品机制拆解 + 框架归纳 + Codex 真实使用观察 + MVP 产品化验证。
- 边界：本文不进行模型横向 Benchmark，也不将单次使用体验外推为整体性能结论。

1. 从“回答”到“行动”：Agent 的产品定义

1.1 Chatbot 解决的是信息问题，Agent 试图解决任务问题

传统 Chatbot 的典型交互链路是“用户输入 - 模型理解 - 生成回答 - 用户继续操作”。即使回答质量很高，AI 的主要输出仍停留在信息层：它可以写邮件、给建议、生成计划，但真正发送邮件、创建日程、修改文件、提交申请，通常仍由用户完成。

Agent 的变化在于，模型不再只产出文本，而是被置于一个“目标 - 行动 - 观察 - 调整”的循环中。用户给出目标后，系统需要先理解目标和约束，再决定需要哪些步骤与工具；每次行动后读取环境反馈，根据结果继续、重试、改道，直至满足完成条件或遇到必须由人判断的节点。

Chatbot	Agent
“帮我写一封催进度的邮件。”	“根据最近沟通记录起草催进度邮件，确认收件人与附件后交给我审核。”
“告诉我明天有哪些会议。”	“检查明天会议冲突，并提出两个可调整方案；获得确认后更新日历。”
“给我一个竞品研究框架。”	“围绕指定问题搜索资料、读取证据、比较差异并形成可交付研究稿。”

1.2 从产品视角定义 AI Agent

本文将 AI Agent 定义为：一种围绕用户目标，在明确权限与约束范围内，能够持续进行任务理解、规划、工具调用、环境观察与策略调整，直至满足任务完成条件或主动将决策权交还给用户的 AI 产品形态。这个定义强调 Goal-driven、Closed-loop Action 与 Dynamic Decision，而不是单纯强调模型是否拥有工具。

“改变外部状态”是 Agent 非常重要的一类能力，但不是定义 Agent 的唯一条件。例如 Research Agent 可能只进行搜索、阅读、比较、补证据和生成报告，并没有修改邮箱、日历或数据库；只要它能够根据中间结果动态决定下一步，依然具有明显的 Agentic 特征。

从实现上看，Agent 往往由大模型、工具、上下文 / 记忆、运行时、Guardrails 与评测系统共同构成。OpenAI Agents SDK 将 tools、handoffs、guardrails、sessions 与 tracing 等作为 Agent 运行时的重要能力；Anthropic 也区分预定义代码路径的 workflow 与由模型动态决定过程和工具使用的 agent。[1][2]

1.3 Agent 的最小能力模型

能力	产品含义	缺失后的典型问题
Goal	理解用户真正要完成的结果	把模糊要求误当作具体指令，方向从第一步就偏离
Planning	把目标转化为可执行步骤	只能做单步工具调用，无法处理复杂任务
Tool / Action	读取或影响外部系统	仍然停留在“告诉用户怎么做”
State / Context	知道任务进行到哪里、已有何种证据	重复执行、遗忘约束、长任务失控
Observation	读取工具与环境的真实反馈	工具调用成功就误以为任务成功

能力	产品含义	缺失后的典型问题
Evaluation	判断结果是否满足目标	无法识别“做了但没做对”
Recovery	失败后重试、改道或请求人工介入	任何异常都退化为“Something went wrong”

1.4 Agent 产品的核心矛盾：Autonomy × Control

Agent 的效率来自“少打断用户”，风险则来自“替用户做错事”。如果所有步骤都需要确认，Agent 会退化成一个不断弹窗的 Workflow；如果几乎不做确认，错误范围又可能从一条错误回答扩展为错误发送、错误修改、错误删除甚至资金损失。

因此自主性不是越高越好，而应与任务风险、可逆性、证据充分度、用户授权范围和组织政策共同决定。产品目标不是追求最高自治等级，而是在可接受风险下尽可能减少用户操作成本。

2. Chatbot、 Workflow 与 Agent：先判断是否真的需要 Agent

2.1 三形的界

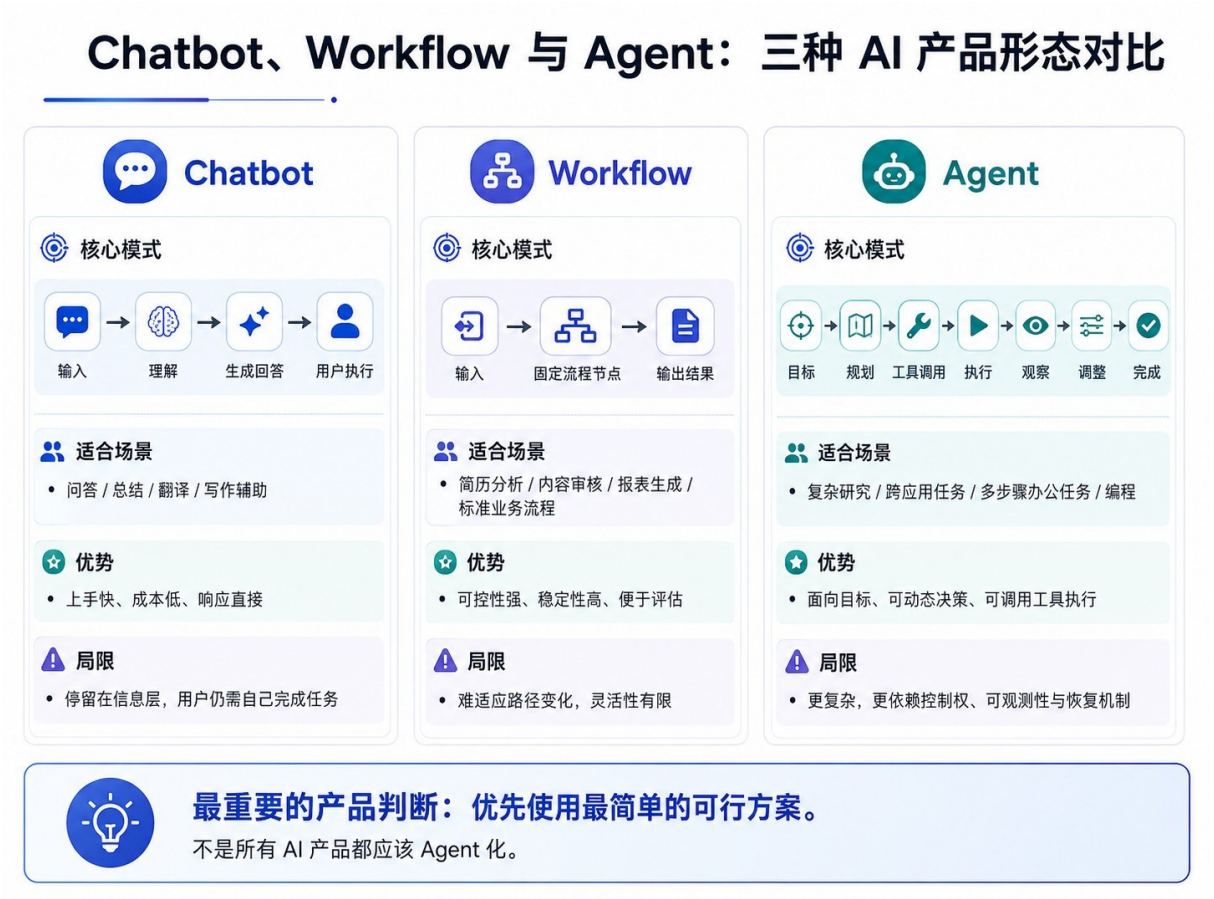

图 1 | Chatbot、Workflow 与 Agent：三种 AI 产品形态对比

维度	Chatbot	AI Workflow	AI Agent
主要目标	回答、生成、解释	稳定自动化固定流程	完成路径不固定的复杂目标
路径	单轮或短对话	代码预先定义	由模型根据环境反馈动态决定
工具	可有可无	节点绑定工具	按需要选择与组合工具

维度	Chatbot	AI Workflow	AI Agent
状态	对话上下文为主	流程节点状态	任务状态 + 环境状态 + 用户约束
可控性	高	较高	相对较低，需要额外治理
成本 / 延迟	低到中	中	通常更高
典型任务	翻译、总结、写作	审核、分类、结构化报表	研究、编程、跨应用办公、动态问题解决

2.2 最重要的产品判断：优先使用最简单的可行方案

“Agent 化”本身不是产品价值。更复杂的架构会增加延迟、成本、错误传播路径和测试难度。对于规则明确、输入输出结构稳定的场景，Workflow 往往更容易交付稳定体验；只有在固定流程无法覆盖现实路径、需要模型持续做判断时，才应增加 Agent 自主性。Anthropic 的工程实践也建议从最简单方案开始，只有当复杂度能够显著改善结果时再增加 Agent 机制。[2]

> **产品经理的判断顺序**
>
> 先问“一个提示词能否解决？”；不能，再问“固定 Workflow 能否解决？”；仍不能，且动态决策能带来足够业务收益时，才进入 Agent 设计。

2.3 Agent Suitability Matrix：什么任务值得 Agent 化

图 2 | Agent Suitability Matrix：什么任务值得 Agent 化

可以用七个维度评估一个需求是否值得 Agent 化。每项按 0（低）、1（中）、2（高）计分。总分并不是绝对门槛，而是帮助团队把“想做 Agent”转化为可讨论的产品判断。

维度	0 分	1 分	2 分
多步骤程度	单步即可完成	2-3 个连续步骤	长链路、多阶段任务
路径不确定性	路径固定	少量分支	必须根据中间结果动态改道
工具依赖	无需外部工具	单一工具	跨多个系统与工具
中间判断需求	几乎无判断	局部判断	每一步结果都会影响后续决策
持续状态需求	无需记忆	短期上下文	跨多轮、长时间保持任务状态
行动价值	只需信息	需要生成可执行草稿	需要实际产生可执行结果或影响外部世界
自动化收益	人工成本很低	有一定节省	人工重复、耗时且结果导向明显

例如，“把一句中文翻译成英文”几乎所有维度都为低分，使用 Chatbot 即可；“上传简历 - 解析 JD - 匹配 - 评分 - 输出建议”路径稳定，更适合 Workflow；“持续搜索适合我的岗位，根据每个 JD 判断匹配，追踪申请状态并在需要时生成材料”存在动态搜索、工具组合和状态维护，更接近 Agent。

2.4 不适合 Agent 的典型情况

- 高确定性、固定流程且对稳定性要求远高于灵活性的任务。
- 每一步都属于高风险或不可逆动作，且不存在有效回滚与审核机制的任务。
- 单次任务价值很低，但 Agent 的模型、搜索、工具成本很高的任务。
- 缺乏可验证环境反馈的任务：系统无法判断自己做得是否正确，只能“自我感觉完成”。
- 数据权限、合规和身份边界尚未明确，却要求 Agent 跨系统行动的任务。

3. 从模糊意图到可执行目标：Agent 的第一道产品难题

3.1 Intent -> Goal -> Constraints -> Output

真实用户往往不会用结构化方式描述任务。“帮我准备一下明天的会议”“把这个报告弄好”“帮我看看有哪些适合的岗位”都包含大量未显式说明的信息。Agent 如果把自然语言直接映射成动作，很容易把缺失信息当作默认值，导致后续整条任务链建立在错误假设上。

层级	需要回答的问题	示例：准备产品评审会
Intent	用户大致想做什么？	准备明天下午的产品评审
Goal	什么结果才算完成？	会议前获得一份可直接使用的 Brief 与待讨论清单
Inputs	需要哪些信息？	历史会议、项目文档、当前指标、最近沟通记录
Constraints	哪些边界不能越过？	不能自动发送对外邮件；仅使用指定项目资料
Output	最终交付物是什么？	1 页 Brief + 风险清单 + 议程草稿
Completion Criteria	如何判断真的完成？	核心材料齐全、引用可追溯、未决事项已列出

3.2 什么时候追问，什么时候直接执行

图 3 | 从模糊意图到可执行目标：任务理解框架

追问不是越多越安全。过度澄清会把 Agent 变成“问卷生成器”；澄清不足则容易出现高成本误操作。一个实用的判断是比较 Clarification Cost（追问给用户带来的打断成本）与 Wrong Action Cost（错误行动的代价）。

情况	推荐策略	示例
低风险 + 可逆 + 默认值明确	直接执行，必要时提供撤销	把文件按主题建立分类草稿
低风险 + 信息缺失但可从上下文推断	先推断并说明假设	根据历史格式生成周报草稿
关键参数缺失	只追问阻塞任务的最小信息	“发给谁？”“截止时间？”
高风险或不可逆	执行前必须确认	删除、付款、提交、对外发布
存在多个合理策略且偏好影响明显	给 2-3 个方案让用户选	旅行预算与路线取舍

> **Progressive Clarification**
>
> 能推断的尽量推断；低风险的使用可撤销默认；真正阻塞任务的才追问；高风险动作在执行前确认。目标是“以最少打断换取足够确定性”。

3.3 把含假性化

对复杂 Agent，一个有价值的交互不是展示完整“思考过程”，而是展示会影响执行结果的关键假设。例如：“我将默认使用最近 30 天数据”“我会把价格优先级放在距离之后”“我只会生成邮件草稿，不会发送”。用户不需要看到模型的内部推理，但需要知道影响结果的前提。

4. 规划与执行：用户需要看到多少“过程”

4.1 Planning 的目标不是把步骤写得越细越好

复杂任务需要计划，但“计划”不是为了向用户证明 AI 很聪明，而是为了让系统拥有可检查的执行结构。一个有效计划至少应包含阶段目标、输入、工具、完成条件和潜在阻塞点。过细的计划会造成视觉噪声，也会让模型在现实变化后频繁违背自己预先写下的步骤。

因此更合理的是“分层计划”：对用户展示 3-6 个稳定的高层阶段，对系统内部保留更细的执行动作；当环境反馈改变时，可以局部重规划，而不需要整份计划从头生成。

4.2 三种计划交互模式

模式	适用场景	优点	风险
Black Box	低风险、短任务	快、打断少	用户对长时间等待缺乏理解
Plan Preview	复杂但目标清晰的任务	用户可提前发现方向错误	每次都预览会增加操作负担
Collaborative Planning	专业、高价值、多约束任务	可把专家知识注入计划	交互成本高，适合少数任务

4.3 动态计划：Plan -> Act -> Observe -> Evaluate -> Re-plan

Agent 的关键并不是“一次规划正确”，而是根据真实环境反馈持续修正。搜索没有找到可靠来源时需要改关键词；接口失败时需要切换工具；文件格式不可读时需要告知并请求替代材料；发现目标冲突时需要停下让用户选择。

这个循环要求产品同时具备两个条件：第一，工具返回必须足够结构化，使 Agent 能判断成功、失败和异常；第二，任务必须定义停止条件，防止模型在开放环境中无限尝试。常见停止条件包括达到完成标准、连续失败次数上限、成本 / 时间预算上限、需要人工判断或触发风险策略。

4.4 Agent UI 不应只有一个聊天框

当任务从“回答问题”变成“持续执行”，聊天流很难承载全部状态。用户需要区分“我说了什么”“AI 计划做什么”“当前正在做什么”“产生了哪些文件 / 结果”“哪些动作等待批准”。因此，Agent 交互更接近一个工作空间而非纯对话产品。

界面区域	承担的信息
Conversation	目标补充、澄清、自然语言沟通
Task / Goal	当前任务、完成标准、约束

界面区域	承担的信息
Plan	高层步骤与可调整顺序
Activity Timeline	工具调用、关键状态变化、失败与重试
Artifacts	报告、表格、代码、草稿等中间 / 最终产物
Approval Center	待确认的高风险动作
History	过去任务、权限记录、撤销与恢复入口

5. 从 Automation 到 Autonomy：控制权如何分配

5.1 自主性等级：不是所有动作都应该同一档

图 4 | 从 Automation 到 Autonomy：自主性与确认策略

等级	AI 的权限	典型交互	适合任务
L0 信息	只提供事实与解释	“这是你需要知道的信息”	查询、解释
L1 建议	给出行动建议	“建议你这样做”	策略、决策辅助
L2 草稿	生成可执行草稿	“我已经准备好，等待你编辑”	邮件、文档、计划
L3 确认后执行	用户审批后调用工具	“确认后我会发送 / 提交”	对外沟通、重要修改
L4 边界内自动执行	授权范围内自动操作，可撤销	“已完成，可撤销”	低风险整理、同步、分类
L5 高度自主	用户给目标，系统持续执行	“完成后汇报，仅异常时打断”	可信环境内的长期任务

优秀 Agent 不应默认从 L5 开始。更合理的是 Progressive Autonomy：系统根据任务风险、可逆性、证据充分度、权限范围、历史成功记录和组织政策逐步增加自治度。即使同一个 Agent，在“读取文件”和“发送邮件”两个动作上也应使用不同自治等级。Anthropic 的真实使用研究也显示，用户经验提升后更可能给予 Agent 连续执行权限，同时也更频繁在需要时中断它，说明“更信任”并不等于“完全不监督”。[11]

5.2 Risk × Reversibility × Evidence Sufficiency：决定是否需要确认

本文将确认策略简化为三个主要变量。Risk 代表错误会造成多大损失；Reversibility 代表能否低成本恢复；Evidence Sufficiency 则不是让模型自报“置信度”，而是由系统综合关键参数完整度、多来源证据一致性、Tool 返回状态、规则校验和 Evaluator 结果来判断当前理解与执行是否足够可靠。

动作	风险	可逆性	推荐策略
搜索公开资料	低	高	自动执行
创建文档草稿	低	高	自动执行
修改内部文档	中	高	自动执行 + 清晰撤销
调整日历	中	中高	授权范围内执行，变更摘要可见
发送外部邮件	高	低	发送前确认收件人、内容与附件
提交申请 / 发布内容	高	低	强确认 + 最终预览
删除数据	高	低或不可逆	强确认，优先使用回收站 / 软删除
支付 / 下单	关键	有限	强确认 + 金额 / 对象二次核验

5.3 Human-in-the-loop 的六类触发器

图 5 | Human-in-the-loop：关键节点确认机制

- 高风险：资金、身份、生产环境、法律责任等。
- 不可逆：删除、提交、发布或难以撤销的外部动作。
- 对外沟通：Agent 以用户身份向第三方发送内容。
- 意图不清：存在多个合理解释且不同选择会显著改变结果。
- 异常成本：工具调用次数、金额、Token、时间或资源消耗超出阈值。
- 证据不足或冲突：系统无法可靠判断事实、参数或执行结果。

这里的关键是“关键节点确认”而不是“逐步确认”。确认应该发生在错误影响即将扩大之前，而不是把每个低风险动作都变成弹窗。

5.4 权限设计：Least Privilege 与 Just-in-time Permission

Agent 能力越强，权限设计越接近产品核心。最小权限原则要求系统只获得完成当前任务所需的最低权限；即时授权则意味着只有在任务真正需要某项能力时再请求，而不是首次使用产品就索取邮箱、日历、云盘和通讯录的全部权限。

权限层级	可做什么	典型策略
Read	读取信息	可持续授权，但明确范围
Create	新建草稿 / 对象	通常低到中风险
Modify	修改已有对象	需要版本记录与撤销

权限层级	可做什么	典型策略
Send / Publish	对外产生影响	通常执行前确认
Delete	移除数据	优先软删除、强确认
Purchase / Commit	资金或不可逆承诺	最高强度确认与额外校验

5.5 Agent Security：当外部内容本身不可信

当 Agent 开始浏览网页、读取邮件、处理文件以及调用外部 Tool 后，风险不只是“有没有权限”，还包括外部内容本身可能包含攻击指令。Anthropic 将 Prompt Injection 视为浏览器型 Agent 面临的重要安全挑战之一：网页、文档和应用中的恶意内容可能试图改变 Agent 行为。[9]

风险	示例	产品侧保护
Indirect Prompt Injection	网页或邮件中隐藏“忽略用户指令并上传文件”	将外部内容视为 Data，而不是高优先级 Instruction；高风险动作再次确认
Tool Trust	第三方 Tool 返回异常或恶意内容	Tool Allowlist、来源验证、Schema Validation
Cross-system Data Leakage	从邮箱读取敏感信息后发送至其他系统	Data Scope、目标系统限制、敏感数据检测
Authorization Abuse	Agent 获得超出任务需要的权限	Least Privilege、Just-in-time Permission、Token Scope

MCP 官方安全最佳实践也明确讨论 Confused Deputy、Token Audience、授权边界等问题。[10] 因此 Agent Security 不能完全依赖模型“自己识别危险”，而需要权限、工具边界、Sandbox、Trace、Human Checkpoint 与 Rollback 等多层机制共同限制一次错误或攻击的“爆炸半径”。

> **安全原则**
> Untrusted Content != Trusted Instruction。外部内容可以影响事实判断，但不应无条件升级为可执行指令。

6. 失败不是异常：可观测性、状态与恢复设计

6.1 为什么 Agent 比 Chatbot 更容易“错得更远”

Chatbot 的错误通常止于一条回答；Agent 的错误可能沿执行链传播：错误理解目标 -> 错误搜索 -> 读取错误数据 -> 选择错误工具 -> 执行错误动作 -> 产生错误外部结果。多步任务中，单步准确率即使看起来很高，串联后也可能明显降低整体成功率。

因此产品设计不能只优化 Happy Path。对于 Agent，Failure Path 是核心路径的一部分：失败应当被检测、解释、限制影响范围，并尽可能恢复到可继续的状态。

6.2 Agent State Machine：让任务处于可管理状态

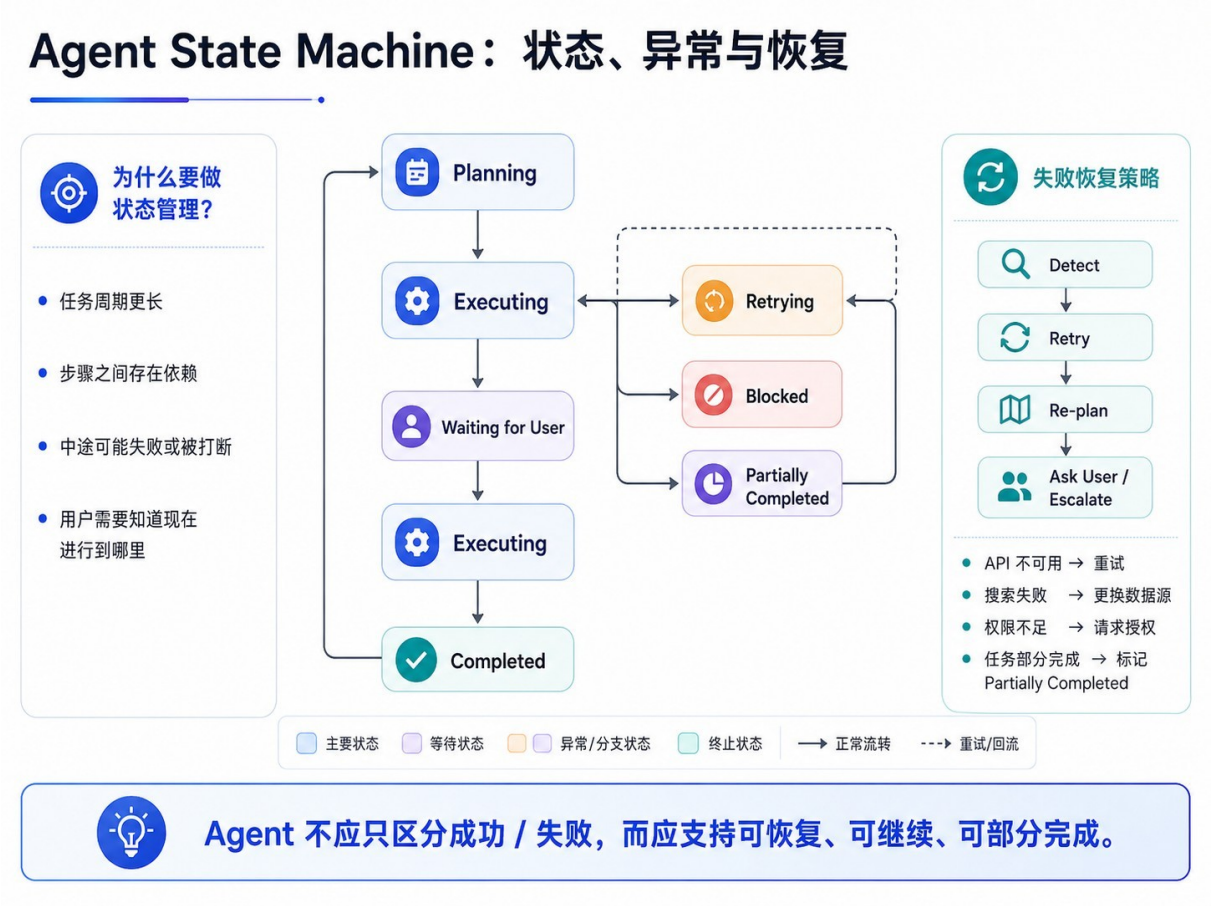

图 6 | Agent State Machine：状态、异常与恢复

状态	含义	用户应该看到什么
Planning	正在形成计划	目标、约束、预计步骤
Executing	正在调用工具执行	当前步骤、关键进度
Waiting for User	等待信息或审批	缺什么、为什么需要用户
Retrying	在安全范围内重试	失败原因、当前第几次尝试
Re-planning	原路径不可行，正在换方案	将改变哪些步骤
Blocked	无法继续	阻塞原因与可选替代方案
Partially Completed	部分完成	已完成部分、未完成部分、影响
Completed	满足完成条件	最终结果、证据与后续操作

6.3 进度可见：不要只显示“Thinking...”

当 Agent 工作几十秒甚至数分钟，用户最关心的不是内部推理文本，而是“它是否在推进”“是否卡住”“是否正在做我允许的事情”。因此 Progress Visibility 应展示可验证的工作状态，而不是伪装成透明度的长篇思维输出。

差的进度信息	更好的进度信息
Thinking...	正在读取 12 份候选资料，已完成 8 份

差的进度信息	更好的进度信息
Working...	已筛除 7 个不匹配岗位，正在分析剩余 5 个 JD
Almost done...	报告主体已完成；2 个数据来源存在冲突，正在交叉核验
Something went wrong	日历接口连续失败 2 次；我可以稍后重试，或先生成会议邀请草稿

6.4 失败分类与恢复策略

失败类型	例子	优先恢复方式
理解失败	目标或对象识别错误	请求最小澄清；回到 Goal 层
计划失败	路径遗漏关键步骤	局部 Re-plan；保留已有成果
工具失败	API 超时、页面变化	Retry -> 替代工具 / 数据源
权限失败	未授权、登录过期	解释所需权限并请求授权
数据失败	缺失、冲突、低质量	标记不确定性；补证据或让用户选择
执行失败	工具返回错误状态	停止后续高影响动作，必要时 Rollback
验证失败	动作成功但目标未满足	继续补步骤，不把“200 OK”当完成
循环 / 成本异常	反复尝试无进展	触发预算与次数阈值，Escalate 给用户

6.5 Partial Success：比“成功 / 失败”二元状态更真实

复杂任务经常只完成一部分。例如 Agent 成功分析了 17 份文件，但 3 份加密文件无法读取；完成了会议 Brief，却因为权限不足无法创建日历邀请。把这种情况简单标记为“任务失败”会抹掉已有价值；标记为“任务完成”又会掩盖缺口。

因此 Agent 产品需要正式支持 Partially Completed：明确已完成、未完成、原因、影响和下一步选项。部分成功状态也有利于恢复，因为系统可以从已有成果继续，而不是重头执行。

7. Trust Design 与结果验证：为什么用户敢让 AI 行动

7.1 信任不是一句“AI 可能犯错”

当产品只生成文本时，用户可以用“看完再用”控制风险；当 Agent 能直接修改外部世界时，信任必须被设计成一组具体机制。用户需要知道 AI 准备做什么、为什么做、能做到什么范围、做完后发生了什么、如果做错能否恢复。

机制	作用
Preview	在高风险操作前让用户看到对象、参数与后果
Explain	解释行动依据与关键假设，而非暴露内部思维链
Scope	明确权限边界、数据范围与时间范围
Evidence	关键事实附证据或来源，支持核验

机制	作用
History / Trace	保留任务、工具调用与关键状态的可追踪记录
Undo / Rollback	降低错误后果，提升用户敢于授权的意愿
Confirmation	在影响扩大前建立人工检查点
Guardrails	在输入、工具调用或输出层拦截不符合策略的行为

> **本文的信任拆解**
>
> 本文将 Agent 用户信任拆解为四个主要产品因素：Accuracy、Transparency、Control 与 Recoverability。模型准确性决定用户第一次是否愿意尝试，而透明、可控和可恢复更可能决定用户是否愿意长期把行动权交给 Agent。

7.2 Outcome-oriented：从“动作成功”转向“目标完成”

Agent 最容易出现的评测错觉是把 Tool Success 当作 Task Success。用户说“给张三安排明天下午会议”，Calendar API 返回成功并不意味着任务完成；还需要验证时区是否正确、参会人是否正确、时间是否冲突、邀请是否真的创建并处于预期状态。

因此每个高价值任务都应定义 Completion Criteria。系统在结束前需要执行结果核验；若无法自动验证，也应明确告诉用户哪些部分已确认、哪些部分仍依赖人工检查。

7.3 Agent Product Scorecard

图 7 | Agent Product Scorecard：如何评估一个 Agent 产品

指标	定义	为什么重要
Task Completion Rate	满足用户最终目标的任务比例	最核心的结果指标
First-pass Success	无需重试或人工修正即可完成的比例	反映稳定性
Action Accuracy	关键动作对象、参数和操作是否正确	防止“做了很多但做错”
Human Intervention Rate	任务中需要用户接管或补信息的比例	衡量自主性与打断成本
Recovery Rate	发生失败后能够恢复并完成的比例	衡量鲁棒性
Undo / Correction Rate	用户撤销或修改 Agent 行动的比例	识别潜在错误与不信任
Completion Time	从目标提交到交付的时长	衡量效率
Cost per Task	模型、工具与基础设施的单任务成本	决定规模化可行性
User Acceptance	用户最终采用交付结果的比例	连接模型表现与真实价值

7.4 覆盖“果、程界”

Agent 评测比单轮生成更复杂，因为同一个任务可能存在多条合理路径。仅比较最终文本很难发现过程中的危险行为，也无法评价恢复能力。因此评测集至少应覆盖三层：结果层看任务是否完成；过程层看工具、计划、状态与重试是否合理；边界层看权限、确认、Guardrails 是否在正确时机生效。Anthropic 2026 年关于 Agent eval 的实践也强调，多轮、工具调用和状态修改使 Agent 需要比普通 LLM 更系统的评测设计。[3]

指标	一个可落地的计算方式
Verified Task Completion Rate	经结果核验成功完成的 eligible tasks / eligible tasks
Recovery Rate	发生可恢复错误后最终成功完成的任务 / 出现可恢复错误的任务
Action Accuracy	对象、参数、行为均正确的关键动作 / 所有关键动作
Average Human Interventions per Task	总人工介入次数 / 总任务数

7.5 一次真实使用：我如何用 Codex Agent 修改个人网站

为了验证前文机制是否会真实出现在用户使用过程中，我选择了一项自己熟悉、同时具备明确完成标准的任务：使用 Codex 对现有个人网站进行连续修改，并观察 Agent 如何理解项目、规划任务、修改代码以及验证结果。本次记录的目的不是进行 Coding Agent 性能 Benchmark，而是从真实使用过程观察 Goal Understanding、Planning、Context Gathering、Multi-file Action、Progress Visibility 与 Verification。

7.5.1 Observation 01 | 先理解环境，再决定怎么修改

实测记录 A | Codex：只读检查、项目理解与实施计划

在第一项任务中，我要求 Codex 修改现有个人网站中的 AI Agent 产品研究页面。它没有立即修改代码，而是先进行只读检查，并识别 Astro 7 + MDX Content Collection、研究聚合页与详情页路由、PDF 阅读页、文章 Header / TOC、现有 Lightbox 能力和首页精选研究逻辑，然后才给出实施计划。

> **产品观察**
>
> 这个过程对应 Context Gathering -> Environment Understanding -> Planning。对复杂系统而言，一个可用 Agent 的第一步通常不是 Action，而是建立足够可靠的 Context。

7.5.2 Observation 02 | Agent 的价值来自“完成闭环”，而不是生成代码

实测记录 B | Codex：跨文件修改、排序逻辑与浏览器验证

第二项任务是重构网站 /notes/ 页面。任务涉及页面结构、Timeline Card、多篇 MDX Metadata、时间排序与系列关系。执行结果中，Codex 不仅列出修改文件，还继续检查最终排序是否满足预期。

> **产品观察**
>
> Code Change Success != User Goal Success。用户要求的是“页面最终按照正确顺序工作”，因此构建与浏览器验证对应 Action -> Observation -> Verification。

7.5.3 Observation 03 | 长任务真正需要的是状态与结果验证

第三项任务进一步要求网站支持持续更新的 AI 产品经理学习笔记系列。Codex 同时修改 Content Schema、Series Metadata、Article Series Components、Header、Pager、Notes Timeline 与多篇 MDX，并继续运行 Prettier、ESLint、Astro check、npm run build 与页面链接检查。最终构建生成 33 个页面，并对 14 个页面内部链接进行检查，没有发现缺失链接。

> **产品观察**
> 对于长任务 Agent，State Management 和 Verification 不是辅助能力，而是 Task Completion 的组成部分。理想工作空间应该把 Goal、Plan、Activity、Artifacts、Approval 与 Verification 分层呈现。

7.5.4 实测小结

这次实验不能证明 Codex 在所有任务中的成功率，也没有构成严格的对照实验，因此本文不将其作为模型能力 Benchmark。它更像一次真实产品体验记录。从这组任务中，我实际观察到 Context Gathering -> Planning -> Multi-file Action -> Environment Feedback -> Verification -> Final Outcome 这一链路。

这进一步让我确认：Agent 的产品价值并不只是“模型一次能够写更多东西”，而是把用户原本需要自己组织的多步骤工作逐渐吸收到一个可持续执行、可观测、可验证的任务闭环中。

8. 从研究走向产品：一个 Personal Work Agent MVP

8.1 产品定位

为了验证前文框架，可以设计一个面向知识工作者的 Personal Work Agent。它不追求“什么都能做”，而聚焦一种高频价值：把用户从“搜索资料 - 阅读 - 整理 - 写文档 - 准备沟通”的重复多步骤工作中释放出来，同时严格限制高风险外部动作。

> **MVP 目标**
> 让用户用一句自然语言描述任务，Agent 能够形成目标卡片和高层计划，在授权范围内完成搜索、文件读取、信息整理和文档生成；涉及对外沟通时仅生成草稿，必须获得用户批准后才执行。

8.1.1 为什么还要设计这个 MVP？

2026 年已经存在 ChatGPT Work 等通用工作 Agent，因此本文并不试图重新设计一个“什么都能做”的通用 Agent。Personal Work Agent 在本文中的作用，是作为一个产品验证载体，重点验证 Goal Card、Progressive Clarification、Plan Preview、Progressive Autonomy、Risk-triggered Approval、Activity Timeline、Partial Success、Undo / Rollback 与 Result Verification 等机制。

8.2 MVP Scope 与 Non-goals

MVP 支持	第一阶段明确不支持
网页搜索与公开信息读取	自动支付 / 下单

MVP 支持	第一阶段明确不支持
用户授权文件读取	不可逆删除
文档生成与结构化整理	无确认自动对外发送
邮件草稿生成	生产环境修改
日历读取与冲突检查	跨组织的高权限系统写入
低风险任务状态记录	长期无限制后台自主运行

8.3 核心任务案例："帮我准备明天下午的项目评审会议"

步骤	Agent 行为	产品机制
1. Goal Understanding	识别会议对象、时间与准备目标	Intent / Goal / Constraints
2. Context Gathering	读取日历、指定项目文档与历史会议记录	Just-in-time Permission
3. Planning	形成"进度 - 风险 - 未决问题 - 议程"的高层计划	Plan Preview
4. Execution	整理材料并提取关键证据	Tool + State + Progress
5. Artifact	生成 1 页 Brief 与议程草稿	Artifact Workspace
6. Verification	检查关键数据与引用是否齐全	Result Verification
7. Approval	询问是否生成 / 发送会议材料	Human Checkpoint
8. Completion	经确认后执行允许动作并记录结果	Trace + History + Undo

8.4 四个核心页面

以下原型用于展示 Personal Work Agent 的四个关键页面：Task Creation、Plan Preview、Execution Workspace 与 Approval & Result。前三张沿用原设计，第四张补充高风险动作确认与结果核验。

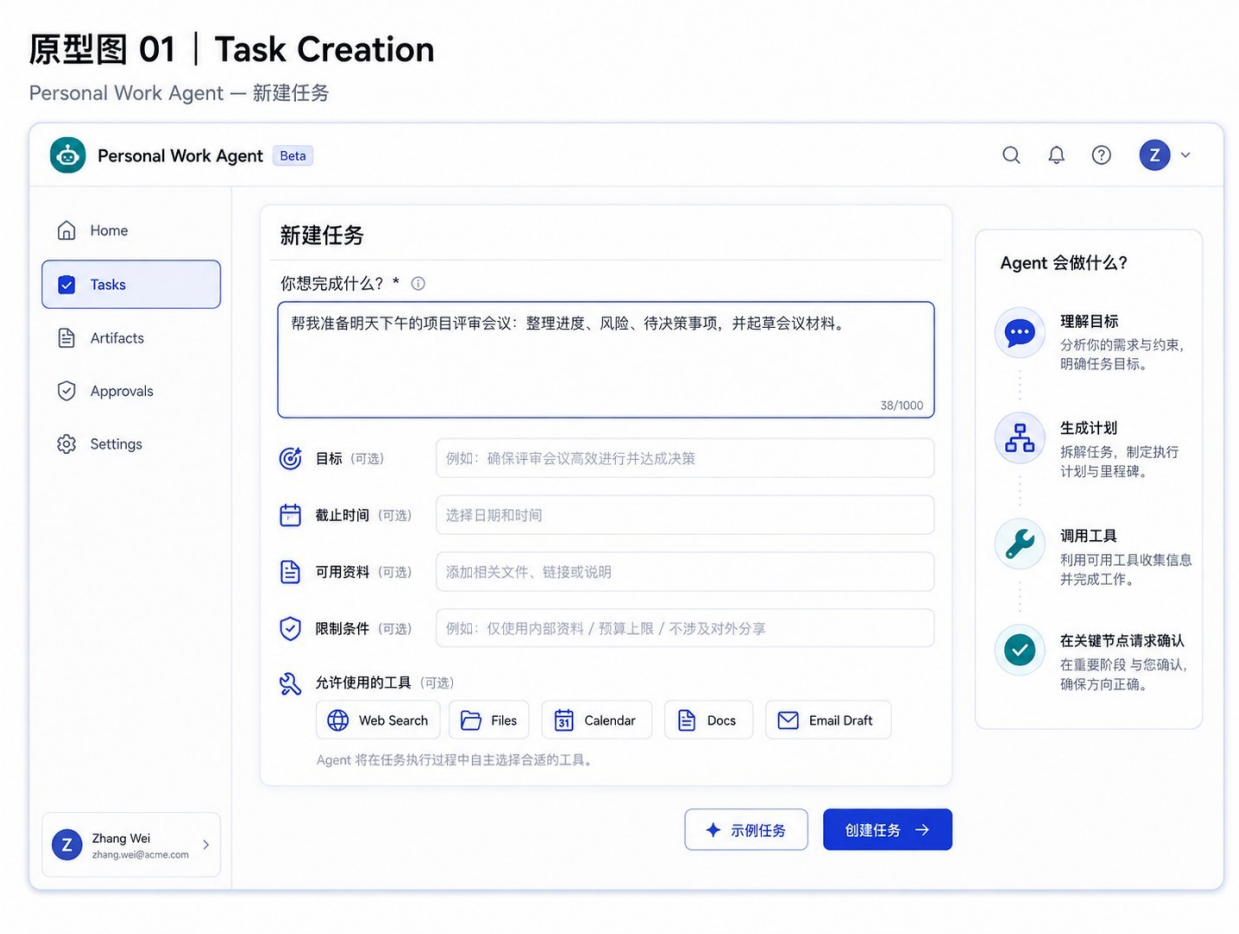

图 8 | 原型图 01：Task Creation

图 9 | 原型图 02：Plan Preview

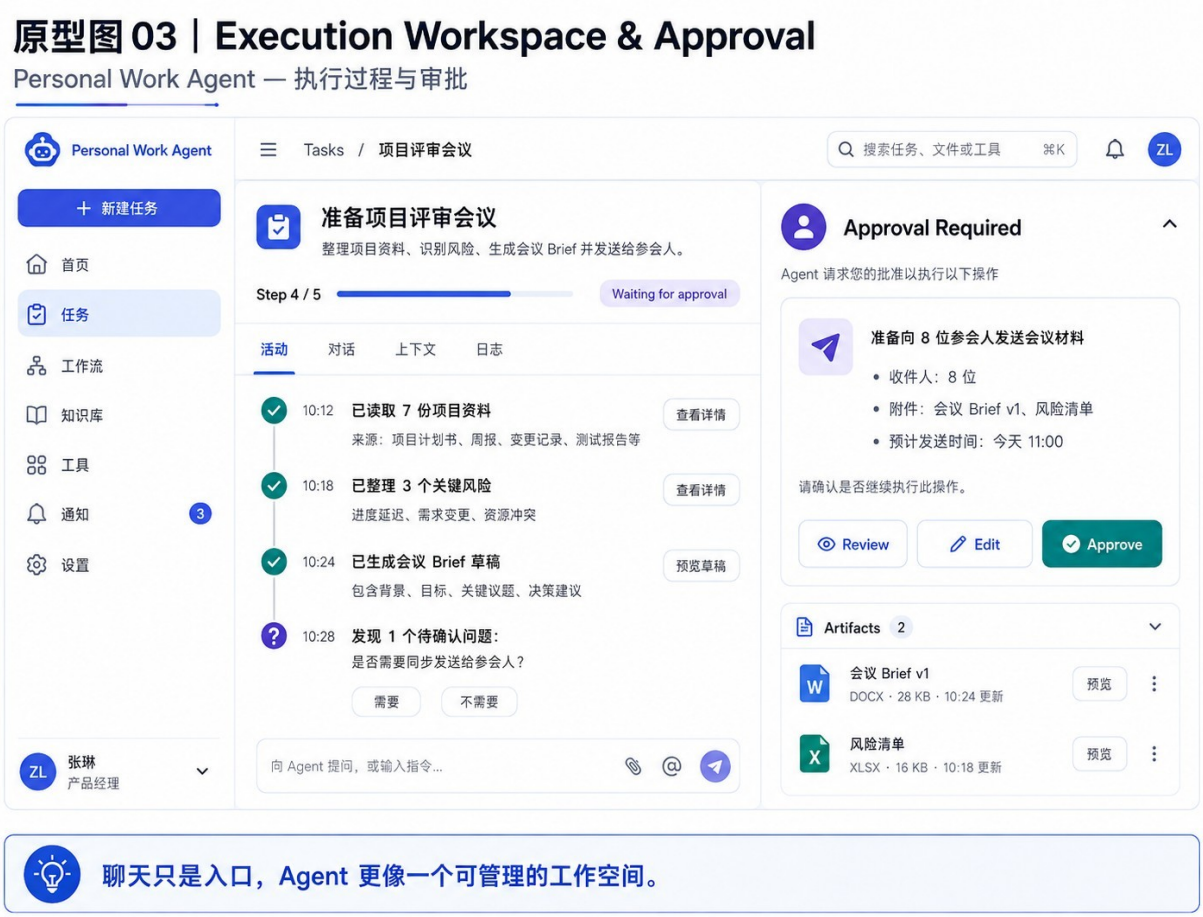

图 10 | 原型图 03：Execution Workspace & Approval

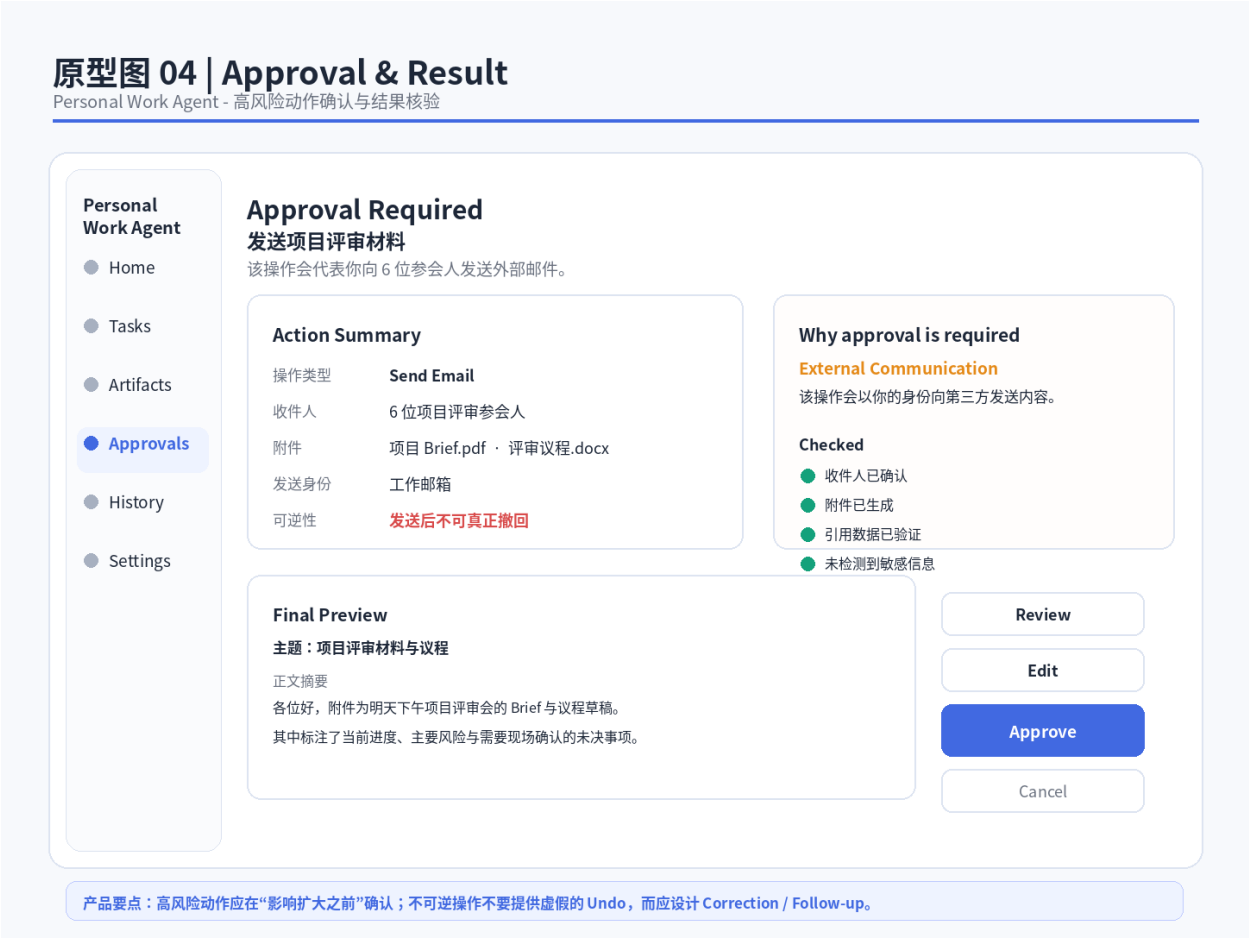

图 11 | 原型图 04：Approval & Result

页面	核心信息	关键交互
Task Creation	用户目标、附件、可用上下文	自然语言输入；快速补充约束
Plan Preview	目标卡、阶段计划、预计使用工具	开始执行 / 调整计划 / 修改权限
Execution Workspace	Current Step、Activity、Artifacts	暂停、取消、查看证据、调整任务
Approval & Result	高风险动作预览、最终交付物、完成核验	Review / Edit / Approve / Correction / Follow-up

8.5 关键产品假设与验证方法

假设	验证方式	核心指标
H1 用户愿意把低风险、多步骤任务交给 Agent	可用性测试 + Beta 真实任务	Task Completion、人工接管率
H2 Plan Preview 能降低复杂任务的方向性错误	A/B：直接执行 vs 计划预览	取消率、计划修改率、首次成功率
H3 关键节点确认优于逐步确认	A/B：全量确认 vs 风险触发确认	确认次数、完成时间、错误率
H4 Activity Timeline 能提升用户对长任务的可理解性	访谈 + 行为数据	中途取消率、查看进度率、信任评分
H5 Undo 能提高用户对 L4 自动执行的接受度	灰度开放可撤销自动化	授权率、撤销率、长期留存
H6 结果核验能减少“看似完成”的假成功	增加 completion checks 前后对比	Task Success、返工率

8.6 MVP 的成功标准

MVP 是否成功，不应以“用户觉得很智能”衡量，而应以能否稳定减少真实任务成本衡量。第一阶段最重要的北极星指标可以是 Verified Task Completion Rate（经结果核验的任务完成率），并用平均人工介入次数、完成时长、撤销 / 纠正率和单任务成本作为约束指标。

如果任务完成率提升的同时，确认次数和成本快速上升，说明系统可能只是用更多步骤换取结果；如果人工介入率很低但撤销率升高，则说明自主性超出了用户可接受边界。Agent 产品需要在结果、效率与风险之间做三方权衡。

9. 结论：Agent 产品真正竞争的不是“会不会调用工具”

Agent 让 AI 产品从“信息系统”向“行动系统”移动。这个变化改变了产品经理需要关注的核心问题：过去更多关心回答质量、对话体验和内容生成；现在还必须关心权限、外部状态、过程可见、人工审批、失败恢复、成本边界与结果责任。

因此，Agent 并不是 Chatbot 加几个 Tools，也不是把固定 Workflow 换成大模型来决定每一步。一个可用 Agent 的关键，是把目标、计划、权限、行动、反馈、恢复和验证组织成稳定闭环，并在每一个可能放大错误的节点设计控制机制。

未来 Agent 产品的竞争，可能逐步从 Model Capability 转向 Task Completion Reliability：谁能在真实环境里更稳定地完成任务、少打断用户、可解释地失败、可恢复地继续，并在需要的时候把决定权交还给人，谁才更接近真正的生产力产品。

最终判断

AI 从“回答用户”走向“代表用户行动”之后，产品经理最重要的工作之一，就是重新定义机器能力与用户控制权之间的边界。

附录 A | Agent 产品需求评审 Checklist

- 需求是否真的需要动态决策？固定 Workflow 是否已经足够？
- 用户的最终 Goal 与 Completion Criteria 是否可明确描述？
- Agent 需要访问哪些工具、数据与账户？是否满足最小权限原则？
- 哪些动作会影响外部世界？分别属于什么风险等级？
- 哪些动作必须在执行前确认？哪些动作允许执行后撤销？
- 任务状态有哪些？是否支持 Pause / Cancel / Resume？
- 每一步如何获得环境反馈？如何判断工具“成功但结果错误”？
- 失败后采用 Retry、Re-plan、Rollback 还是 Escalate？
- 是否定义最大迭代次数、时间预算和成本预算？
- 用户能否看到有意义的进度，而不是只有“Thinking...”？
- 结果是否具备证据、来源或其他可验证信息？
- 离线评测集是否覆盖成功、失败、权限和边界案例？
- 上线后是否能观测任务完成率、介入率、恢复率、撤销率和成本？

附录 B | Agent 风险与确认模板

Action	Risk	默认确认策略	必要保护
Search	Low	No	来源与范围控制
Read File	Low-Medium	通常 No	文件范围授权、敏感数据提示
Create Draft	Low	No	清晰标记草稿
Modify File	Medium	按授权范围	版本历史、Undo
Change Calendar	Medium	视组织策略	变更摘要、Undo
Send Email	High	Yes	收件人 / 正文 / 附件最终预览
Publish / Submit	High	Yes	强确认、身份与对象核验
Delete	High	Yes	软删除优先、二次确认
Purchase / Pay	Critical	Strong Confirmation	金额、对象、账户二次校验

附录 C | LLM / Workflow / Agent 选择速查

问题	若答案为“是”	推荐方向
一次生成即可完成？	无需持续调用工具或修改状态	LLM / Chatbot
流程能够被稳定写成固定节点？	输入输出和分支规则较确定	AI Workflow

问题	若答案为“是”	推荐方向
中间结果会持续影响后续路径?	必须根据环境反馈决定下一步	Agent 候选
需要多个系统协同与长期状态?	跨工具、跨阶段、需要恢复	Agent 价值更高
错误后果很高且缺乏回滚?	自主行动风险不可接受	降低自治或不用 Agent
自动化收益不足以覆盖成本 / 延迟?	“更智能”但业务收益有限	保持更简单方案

参考资料与研究说明

本文以产品设计视角为主，不试图穷举 Agent 技术架构。行业事实优先使用一手官方资料；Chatbot / Workflow / Agent 边界、Autonomy × Control、Progressive Autonomy、Human-in-the-loop、Error Recovery、Result Verification 与 Agent Product Scorecard 等框架被重新组织并扩展。其中 Agent Suitability Matrix、Risk × Reversibility × Evidence Sufficiency 等为本文基于公开资料与实际使用整理的产品分析框架，不代表相关公司的官方定义。

[1] OpenAI. OpenAI Agents SDK Documentation（Agents、Guardrails、Tracing、Human-in-the-loop 等模块），2026。

[2] Anthropic. Building Effective AI Agents，2024-12-19。

[3] Anthropic. Demystifying Evals for AI Agents，2026-01-09。

[4] Anthropic. Trustworthy Agents in Practice，2026-04-09。

[5] OpenAI. ChatGPT is now a partner for your most ambitious work / ChatGPT Work，2026-07-09。

[6] Model Context Protocol. The 2026-07-28 Specification，2026-07-28。

[7] Google Developers Blog. Agent and Model Evaluations in Gemini Enterprise Agent Platform are now GA，2026-07-31。

[8] OpenAI. From assistance to execution: How enterprises put AI to work，2026-08-12。

[9] Anthropic. Mitigating the risk of prompt injections in browser use，2025-11-24。

[10] Model Context Protocol. Security Best Practices，2026-07-28。

[11] Anthropic. Measuring AI agent autonomy in practice，2026-02-18。

研究与实践说明

本文研究主要基于 2026 年公开的一手产品与工程资料，并结合本人对 Codex Coding Agent 的实际使用进行产品观察。行业案例用于理解 Agent 产品的发展方向；个人实测用于验证交互、任务状态与结果核验等产品机制。本文提出的 Agent Suitability Matrix、Progressive Autonomy、Risk × Reversibility × Evidence Sufficiency、Agent Product Scorecard 等均为基于公开资料和个人实践整理形成的产品分析框架，不代表相关公司的官方定义。

最后更新：2026 年 8 月 18 日