????-Harness12????

??????????????× Harness 12 ?? ????

> 分析对象：`Agent/design/一种理想的智能体编排架构.md` 对照标准：`Agent/design/Agent-Harness解剖.md` 中归纳的 12 个生产级 Harness 组件 目的：把 "理想架构 "的设计意图，逐项映射到 Anthropic/OpenAI/LangChain 共识的工程组件上，找出 已覆盖 、等价但术语不同 ，缺失 三类位置。

???????

#	**Harness** 组件	覆盖度	对应文档位置（章节）	一句话判断
1	Orchestration Loop	□ ◫	§4.5 本能 /调度器、§4.5.4 循环闭环	"本能 "= dumb loop，主循环显式建模
2	Tools	□ ◫	§4.4.7 Frame 可见的工具、 §5.4	全局 `ToolRegistry` + 分层路径，比 PDF 更严格
3	Memory	□ ◫ + 增强	§2.2、§4.2、§4.3、§5.1	短 / 长时分离，关键词树组织，比 Claude Code 更结构化
4	Context Management	□ ◫ + 增强	§3.4、§4.4、§4.4.7、**§4.4.8**、§6.1、§6.3	三段式布局 + "删中段 "压缩，把语义难题降为长度算术问题
5	Prompt Construction	□ ◫	§5.3、§3.1、§4.5.1	强调多层级动态组织 + 任务类型差异化模板
6	Output Parsing	⚠ 部分	§4.4.3、§5.3	偏向 "LLM 输出格式化任务图本地执行 "，缺乏 native tool calling 的明确选型

#	**Harness** 组件	覆盖度	对应文档位置（章节）	一句话判断
7	State Management	□ ◫	§4.4.2、§4.4.6、§5.1 公共存储区	Frame 状态 + 事件总线（`SUBFRAME_DONE`），未提 git checkpoint
8	Error Handling	⚠ 部分	§4.4.3、§4.4.5（回退动作）	有"节点失败重设计"+"回退"，但缺四类错误的分级处理
9	Guardrails & Safety	⚠ 维度不同	§4.1、§6.6、§3.1（人格）	用"人格 + 自洽 + 价值观"做软约束；缺工程级 tripwire / 权限闸
10	Verification Loops	⚠ 部分	§4.4.4、§4.5.2（主动维护自洽）	"自洽判断"本质是 LLM-as-judge；缺 rules-based / visual 校验
11	Subagent Orchestration	□ ◫	§4.4.3、§4.4.6、§6.3 sub-agent	SubFrame nested-to-tree；缺 Fork/Worktree 的物理隔离讨论
12	Long-Horizon Execution	□ ◫ + 增强	§4.5.4、§6.4 自动更新与学习	"本能 + 永久记忆 + 学习闭环"= 比 Ralph Loop 更高阶的连续性

结论速览：12 项中 **7** 项完整覆盖、**3** 项部分覆盖、**2** 项采取了不同维度的解决思路（Output Parsing、Guardrails & Safety）。

????????

---

1. Orchestration Loop??????— ? ??

Harness 标准：实现 TAO（Thought-Action-Observation）循环 ——assemble prompt → call LLM → parse → execute tools → loop。Anthropic 提倡 "dumb loop"，把所有智能留给模型。

理想架构对应：

- **§4.5** 本能 / 调度器（主循环）：**"创建当前状态的下一步行动，是整个系统的主循环"**——直接对应 dumb loop。
- **§4.5.4** 循环闭环：**"本能任务不断产生记忆，存入临时记忆，作为下一次行动的依据**——临时记忆 ↔ 本能目标 形成循环"。

- **§2.4** 本能： "先思考，再行动 / 自动触发记忆检索 / 自动反思总结 "——把 TAO 三段内化为本能行为。

□□ □□ 完整，且做了 抽象升级 ——把循环本身命名为 "本能 "，赋予人格化语义。比 Anthropic 的 dumb loop 更厚一些 （包含主动反思 + 自动记忆检索），与 "harness 应越薄越好 "的原则有轻微张力。

---

2. Tools????— ? ??

Harness 标准：工具以 Schema（name/description/parameters）注入； Tool 层负责注册、校验、入参提取、沙箱执行、结果格式化；典型如 Claude Code 的 6 大类工具。

理想架构对应 ：

- **§4.4.7 Frame** 可见的工具 ：
  - 全局 `ToolRegistry`， Frame 不持有本地实例
  - 分层路径组织： `memory/*`、`frame/*`、`file/*`、`skill/*`
  - `Frame.Next()` 在装配 prompt 时按当前状态 /动作集 拉取可见子集注入 **System** 段
- **§5.4** 提供给 **LLM** 的基础设施 ：记忆与存储、 Frame/Action 管理、逻辑思维管理三类。

□□ □□ 完整， 比 **PDF** 更严格 ——

- "全局 Registry + 分层路径 + 按状态拉取可见子集 " 直接命中 **Harness** 决策表的第 **6** 项（ **Tool Scoping**）： Vercel 删 80% 工具反而更好的经验，这里通过架构层面就避免了 " 工具一次性塞满上下文 "。
- 与 Claude Code 的 "渐进披露 / Skills"机制等价。

---

3. Memory????— ? ?? + ??

Harness 标准：短期（ session 内对话） + 长期（跨 session 持久化）； Anthropic 三层结构：轻量索引（~150 字符常驻） → 主题文件 → 原始 transcript（搜索访问）； 关键原则： **agent** 把自己的记忆当 **hint**，行动前必须验证真实状态 。

理想架构对应 ：

- **§2.2** 记忆： 明确分 临时记忆 与 长期记忆 ；强调 "强大的分级信息检索 "。
- **§4.2** 临时记忆 ：意识节点，提供当前背景；通过 Frame 包装接口访问。
- **§4.3** 长时记忆 ：基于 关键词 分类存储； 多层级动态组织 ；外挂知识接口。
- **§5.1** 存储与记忆基础设施 ：分布式隔离 ——每个任务管理自己的存储。

□□ □□ 完整，且 结构化程度高于 **Claude Code**：

- Claude Code 用 `CLAUDE.md` + `MEMORY.md` 平铺文件
- 理想架构用 关键词树 **+ Info** 节点 **+** 多对多关联 （见 §4.3 旁注与 `WeaveAgentDesign.md` §8）

- 这与 MindStore 现有实现（ Qdrant + 关系边）天然对齐

潜在差距 ： Anthropic 的 "agent 把记忆当 hint，行动前验证真实状态 "原则，文档没有显式表达；建议在 §4.3 加上一条 "读出的长时记忆需经过当前 Frame 的自洽校验 "。

---

4. Context Management???????— ? ?? + ??

Harness 标准： 应对 Context Rot；四种生产策略： Compaction / Observation Masking / Just-in-time Retrieval / Sub-agent Delegation。Anthropic 目标： "找到最小高信号 token 集合 "。

理想架构对应 ：

Harness 策略	理想架构对应
Compaction	**§4.4.8** 三段式布局 + "删中段 "压缩（无需 LLM 调用， O(1) 截断）
Observation Masking	§4.4.7 工具按状态拉取子集（隐式屏蔽）
Just-in-time Retrieval	§4.4.7 分层加载 + §6.3 "本地可理解的思维步骤动态组织上下文 "
Sub-agent Delegation	§4.4.3 SubFrame nested-to-tree + §6.3 sub-agent
额外： Frame 隔离	§4.4.6 "每个 Frame 对应一个独立的上下文， session 上下文与 Frame 状态分离 "

□□ □□ 完整覆盖， 且在 **Compaction** 策略上有原创性增强 ——

- §3.4 "降低对 LLM 上下文的需求 "是这一组件的 显式设计目标
- §6.1 "LLM 上下文有限，难以实现大范围高深度自洽 " = Context Rot 的中文表述
- "Frame 隔离 + 分层加载 "是文档独有的 架构级解法 ，对应 PDF 决策表第 3 项的 "sub-agent delegation + structured note-taking"组合
- **§4.4.8** 三段式布局 是本架构对 Anthropic Compaction 的降维替代方案 ：
  - 核心不变量 ： `TOP（System 段）+ BOT（有效段）`
    在任意时刻是可独立执行的最小完整上下文，长度静态可知且 `len(TOP)+len(BOT) << len(LLM_max_context)`
  - 压缩 = 直接删除 **MID** 中段： 不调 LLM、不做摘要、 O(1) 截断
  - 不破坏完整性的根据 ： MID 段本质是 "过程性证据 "，结论性事实由 §4.4.5 的 "笔记 /学习 /抽象 "动作主动落入 Frame 状态 / 临时记忆 / 长时记忆 ——**LLM**
    上下文是高速缓存，不是事实主存

维度	**Anthropic Claude Code**	本架构 **§4.4.8**
触发	接近窗口上限	接近预设阈值（如 80%）
方式	LLM 摘要历史	直接删 MID 中段
成本	每次压缩 1 次 LLM call	0 次 LLM call
不变量保障	依赖摘要质量	依赖 Frame 状态 /记忆作主存

维度	Anthropic Claude Code	本架构 §4.4.8
失败模式	摘要丢关键信息	MID 过大时 BOT 仍完整

评价：把上下文压缩从一个 "语义难题 "降级为 "长度算术问题 "，代价是要求 Frame 状态系统与记忆系统承担 " 事实主存 "职责 ——这与本架构 §4.5.2 "总结 /抽象 /记忆 "和 §4.3 长时记忆的设计天然对齐，并不增加额外负担。 原 **P2** 级 **"Compaction** 触发条件缺失 "已闭环。

---

5. Prompt Construction?Prompt ???— ? ??

Harness 标准：层级化拼装 ——System Prompt → Tool Definitions → Memory Files → Conversation History → Current User Message； OpenAI Codex 用严格优先级栈。

理想架构对应 ：

- **§5.3 Prompt** 工程：
  - 基础 prompt 维度: 人格、自洽、行动规划、学习总结、记忆
  - **Prompt** 的多层级动态组织
  - 不同任务类型用不同 system prompt
  - **Frame** 分类 对应不同 prompt 模板
- **§3.1** 自我意识 ： "精心设计的 prompt"——把人格从大模型内部移出来到编排器
- **§4.5.1** 根任务 ：维持 "自我意识 "的入口 ——对应 system prompt 最高层

□□ □□ 完整，且 多了 "人格层 "维度 ——把人格、价值观显式作为 prompt 的最高层结构，Anthropic/OpenAI 通常只暗示在 system prompt 里。

---

6. Output Parsing??????— ?? ??

Harness 标准：现代系统依赖 原生 **tool calling**——模型返回结构化 `tool_calls` 对象，无需解析自由文本； OpenAI/LangChain 用 Pydantic 做 schema 约束。

理想架构对应 ：

- **§4.4.3 SubFrame nested to tree**： "LLM 输出 格式化 的任务图（任务描述、功能、任务之间的关系），本地自动执行 "——是格式化输出，但偏向 自定义 **DSL**。
- **§5.3 LLM** 输入输出的抽象 ： "格式化 "———笔带过，没具体方案。

评估： ⚠ 概念到位，工程选型缺失 ——

- 文档承认 "格式化输出 "是必要的，但 没有明确选型 ：用原生 tool_calls？ JSON Schema？自定义 DSL？
- "LLM 输出格式化的任务图 "听起来像自定义结构化输出，会 损失原生 **tool calling** 的训练增强收益

建议：对齐 Harness 共识 ——Frame 的动作集（ §4.4.5 扩展 /总结 /尝试 /回退 /收集 /抽象 /笔记 /学习）应该实现为原生 **tool** 列表，而不是文本指令。这样能直接受益于模型厂商对 tool calling 能力的训练优化。

---

7. State Management?????— ? ??

Harness 标准： LangGraph 用 typed dict 流过图节点 + 超步 checkpoint; OpenAI 四种互斥策略; Claude Code 用 git commit 当 checkpoint + progress 文件当结构化草稿本。

理想架构对应 ：

- **§4.4.2 Frame** 的状态 ：实际结果（失败 /成功）、执行的所有过程、历史信息和当前状态、思维逻辑的记录 、执行结果的总结。
- **§4.4.6 Frame** 接口：子 Frame 完成通知 经事件总线 （ `SUBFRAME_DONE` ） ——这是显式的状态变更协议。
- **§5.1** 公共存储区 ：当前基础信息 ——各任务的状态、本能状态、精神状态。

□□ □□ 完整 ——

- Frame 的 7 个状态字段（结果 /过程 /历史 /思维 /总结） 比 **LangGraph** 的 **typed dict** 更结构化
- 事件总线驱动的状态推进，避开了直接方法回调，有更好的解耦

潜在差距 ：缺 **checkpoint /** 时间旅行调试 ——LangGraph 在 super-step 边界打 checkpoint 支持回滚，文档里只有 §4.4.5 的 "回退 "动作但没说怎么持久化中间状态。可考虑借鉴 Claude Code 的 "git commit 当 checkpoint"模式，把每次 Frame 状态变更落到本地 git。

---

8. Error Handling?????— ?? ??

Harness 标准： 10 步 × 99% = 90.4% 端到端的复利效应; LangGraph 四类错误: Transient（ retry）、LLM-recoverable（回传 ToolMessage）、 User-fixable（ interrupt）、 Unexpected（ bubble up）; Stripe 重试上限 2 次。

理想架构对应 ：

- **§4.4.3**: "节点执行失败可触发重新设计节点的拆分、提示词，成功后更新节点（提升效率、节省 token） "——属于 LLM-recoverable 类。
- **§4.4.5**: 动作集中包含 回退： "当前方向是错的，回退到前面的第 5 步 "——这是模型层的自纠。

评估： ⚠ 思想到位，工程层缺失 ——

- 文档 只覆盖了 "模型可自纠 "那一类
- 缺: 瞬时错误的 retry/backoff、需要人介入的 interrupt 机制、意外错误的 bubble up
- 缺: 重试上限 （ Stripe 经验: 2 次）
- 缺: 错误复利的数学认识 ——长任务里 99% 单步成功率不够

建议：在 §4.5.2 补一节 "错误处理协议"，按 LangGraph 四分类显式约定 Frame 层与 Tool 层的错误传递方式。

---

9. Guardrails & Safety??????— ?? ????

Harness 标准：OpenAI 三层（input/output/tool guardrails）+ tripwire；Anthropic 把权限执行从模型推理中架构性分离——模型决定"想做什么"，工具系统决定"允许做什么"；Claude Code 分别上闸 ~40 个能力。

理想架构对应：

- **§4.1** 智能主体：人格与个性、价值观、灵魂——软性护栏。
- **§3.1** 自我意识：把人格从大模型内部移出来到编排器。
- **§4.4.4** 自洽要求："所有行动都要有合理的理由、逻辑链条、结果"——自我合理性检查。
- **§6.6** 价值观：利用"自我"提供终极目标指引。

评估：⚠ 维度不同，互补而非重合——

- 理想架构走的是"内化的人格 + 价值观自洽"路线，类似阿西莫夫机器人三定律
- Harness 标准走的是"外化的工程闸门"路线（input/output/tool guardrails + tripwire）
- 两者并不冲突：人格是"想不想做"，guardrail 是"能不能做"。两者都需要

建议：在 §5 工程实现里显式补一节"权限与护栏"——

- Tool 调用前的权限检查（继承 Anthropic 的 ~40 个能力闸门思路）
- 高风险操作（写入永久记忆、删除、外部 API 调用）需要显式确认或 tripwire
- 与"价值观/自洽"形成双层防护

---

10. Verification Loops??????— ?? ??

Harness 标准：三种 verification——Rules-based（测试/linter/类型检查）、Visual（Playwright 截图）、LLM-as-judge（subagent 评估）。Boris Cherny："给模型验证自己工作的方式，质量提升 2-3×"。

理想架构对应：

- **§4.4.4** 自洽要求：所有行动都要有合理理由、逻辑链条、结果——**LLM-as-judge** 的雏形。
- **§4.5.2** 下一步行动的决策："主动维护自洽：通过任务上下文、子任务总结等进行自洽判断"。
- **§3.2** 大范围、高深度的信息自洽："主动的检查和主动的行动逻辑判断"。

评估：⚠ 只覆盖了 **LLM-as-judge** 一种——

- "自洽判断"本质就是用 LLM 评估自己输出的合理性

- 缺 **rules-based**：单元测试、linter、类型检查这类确定性 **ground truth**
- 缺 **visual**：UI / 图片输出的截图比对

建议：把 §4.4.4 自洽要求扩成"三层校验"——

1. 规则校验：本地代码可判定的（schema 校验、断言、类型检查）
2. 执行校验：跑测试 / 编译 / 静态分析
3. 自洽校验：LLM 判官（已有）

参考 Martin Fowler/Thoughtworks 的 **guides**（**feedforward**）**vs sensors**（**feedback**）框架。

---

11. Subagent Orchestration?? agent ???— ? ??

Harness 标准：Claude Code 三种执行模型——Fork（字节级副本）、Teammate（独立终端 + 文件邮箱）、Worktree（独立 git 分支）；OpenAI 用 agents-as-tools + handoffs；LangGraph 用嵌套 state graph。

理想架构对应：

- **§4.4.3** 组织形式：分层树状结构：`"SubFrame List nested to tree" = LangGraph 嵌套 state graph 的同构表达。`
- **§4.4.6 Frame** 接口：子 Frame 完成通知经事件总线（`SUBFRAME_DONE`）——比 LangGraph 的 reducer 更解耦。
- **§6.3** 上下文调度管理器：sub-agent 执行特定子任务、并行多个上下文支持嵌套。

□□ □□ 完整 ——

- "嵌套 Frame + 事件总线"= LangGraph 的"嵌套状态图"
- **Frame ID +** 独立存储空间 **+ Session/Frame** 状态分离（§4.4.6）= Anthropic 的 Fork 思想

潜在差距：未讨论物理隔离层级（同进程/跨进程/跨机器）——Claude Code 的 Worktree 模式是文件系统级隔离，理想架构如果要支持长程并行子任务，需要决定隔离粒度。

---

12. Long-Horizon Execution????? / Ralph Loop ?— ? ?? + ??

Harness 标准：Anthropic 两阶段 Ralph Loop——Initializer Agent 搭环境（init script + progress 文件 + feature 列表 + 初始 commit），Coding Agent 每个会话读 git log + progress 文件自我定位，挑高优 feature → 干活 → commit → 写摘要。文件系统提供跨上下文窗口的连续性。

理想架构对应：

- **§4.5.4** 循环闭环：本能任务不断产生记忆，存入临时记忆；临时记忆 ↔ 本能目标 形成循环 = Ralph Loop 的本质。
- **§3.3** 自动进化、自动学习：“灵活的记忆接口 + 预设的学习本能任务”。
- **§6.4** 自动更新与学习：
  - 创造和更新 skills
  - 制造和整理 tools、脚本
  - 学习的方法学：软件工程开发流程
  - 复杂任务可经过非常多次尝试后解决，并永久学习
- **§4.5.2** 总结/抽象/记忆：主动提取和组织框架性的知识，存储到基础知识框架。

□□ □□ 完整 + 比 **Ralph Loop** 更高阶 ——

- Ralph Loop 是“用文件系统续命”
- 理想架构是“用 长时记忆 + 自动方法学进化 续命”——把跨会话连续性从 文件层 升到了 知识层
- **§4.4.5** 学习动作（“把指定的信息或笔记写入永久记忆”）= Ralph Loop 的“写摘要”，但语义化更强

潜在差距：Ralph Loop 用 **git commit** 作为天然 **checkpoint**，理想架构没有显式接入 git——可作为工程上的最小补充：每次 Frame 关键状态变更落一次 commit。

???????

3.1 ????

维度	评分	说明
概念完备性	**A**	12 项中 10 项有显式或等价表达
工程具体度	**B**	大量“目标级”描述，落地细节相对薄弱
创新点	**A**	“本能 + 人格 + Frame + 关键词树记忆”是有特色的组合
与 Harness 共识对齐度	**B+**	主要差异在 Output Parsing、Guardrails、Verification 三项的工程化

3.2 ???????????PDF ??????

1. 人格作为编排器一等公民（§3.1、§4.1、§6.6）——把价值观/自洽显式提到 prompt 最高层
2. **Frame** 作为统一抽象——同时是 task / context / 思维节点 / handle，比 LangGraph 的“node”语义更通用
3. 关键词树长时记忆——比 Claude Code 的平铺 `MEMORY.md` 结构化得多
4. 本能/调度器把“先思考再行动”内化为系统行为——而不是依赖模型每次自己决定

3.3 ????????????

优先级	缺口	对应 **Harness** 组件	最小补丁
P0	错误的分级处理	#8 Error Handling	§4.5.2 增加四类错误协议（瞬时 /可纠 / 可修 /意外） + retry 上限
P0	工程级护栏	#9 Guardrails	§5 增加 "权限与 Tripwire" 小节，与人格自洽形成双层防护
P1	校验回路三分类	#10 Verification	§4.4.4 自洽要求扩成 "规则 + 执行 + 自洽 "三层
P1	Output 选型	#6 Output Parsing	明确 Frame 动作集用原生 tool calling 实现
P2	Checkpoint 机制	#7 State / #12 Long-Horizon	借鉴 Claude Code，把 Frame 关键状态变更落 git commit
~~P2~~	~~Compaction 触发条件——~~	~~#4 Context Mgmt~~	□ □□□ — §4.4.8 三段式布局 + 删 MID 中段策略

3.4 ?????????

已对齐	待对齐
关键词树长时记忆 ↔ MindStore（ Qdrant + 关系边）	git checkpoint → 接入 Auto/ 流水线
Frame 状态 ↔ WeaveAgent Chronicle（ "原因 + 结果 "）	工程级 guardrail → WeaveAgent ToolRegistry 加权限闸
Sub-agent ↔ WeaveAgent SubFrame	错误四分类 → 当前 Chronicle 未区分
渐进披露工具 ↔ §4.4.7 分层路径 ToolRegistry	三层 verification → 接入 Qwen3-Embedding 训练数据校验

???????

> 理想架构在 "概念完整性 "和 "哲学高度 "上超过了 **Harness 12** 组件共识；但在 " 工程化细节 "——尤其是错误处理、权限护栏、校验三分类 ——还需要把 **Anthropic / OpenAI / LangChain** 已经踩过的坑显式吸收进来。
>
> 下一步最经济的演进路径：保留全部独有抽象（人格 / 本能 / Frame / 关键词树），把 P0 的两个缺口（错误协议 + 工程级护栏）作为下一版 `WeaveAgentDesign.md` 的增量补丁。

Revision #1
Created 2026-05-01 03:59:58 UTC by Colin
Updated 2026-05-01 03:59:58 UTC by Colin