

Framing Is Human: Researcher–Brain–Executor Architecture for AI-Assisted Research*

Chenglong Fu

University of North Carolina at Charlotte
Charlotte, North Carolina, USA
cfu6@charlotte.edu

Abstract

AI systems have become capable collaborators for research, but capability is unevenly distributed across the research workflow. Current systems are strong at breadth, execution, and synthesis; they are measurably weaker at domain-specific framing and decision-making, the work of choosing what question to ask, what evidence counts, and what methodology is appropriate. Two dominant paradigms have emerged for AI-assisted research: autonomous research agents that attempt end-to-end research with minimal human framing (The AI Scientist, ARIS, Agent Laboratory), and coding-only assistants used for implementation (Cursor, plain Claude Code). The flagship autonomous system’s own authors enumerate the failure modes that follow (hallucinated results, methodology fabrication, frame-lock), concentrated at framing-adjacent stages. An independent 2025 survey identifies “collaborative approaches” as an articulated gap.

We present **RKA** (Research Knowledge Agent), an infrastructure that operationalizes the middle path. Three structural roles (human *Researcher*, strategic *Brain*, mechanical *Executor*) are connected only through a narrow mission–checkpoint–report interface. Between them sits a four-layer progressive-distillation knowledge base (journal entries, claims, evidence clusters, research themes) with enforced provenance, immutable raw observations, and supersede chains over the interpretation layers. Four operational control properties (traceability, reversibility, visible disagreement, human-ratified commitment) each map to a specific architectural mechanism and can be checked against the running system. We describe the architecture and eight design principles derived from eight months of self-use, and we evaluate through retrospective self-study, three case studies from the edge-cloud-agent project, and an independent-use replication. The architecture lets researchers scale the width and depth of AI-assisted work without surrendering the ability to audit, contest, or reverse any AI-produced interpretation.

Keywords

AI-assisted research; human–AI collaboration; research infrastructure; provenance; agent architecture; scientific workflow

1 Introduction

AI-assisted research has reached an inflection point. In 2025, a fully autonomous AI system generated a machine-learning research paper that passed peer review at a workshop of a major machine-learning conference (score 6.33 versus an average reviewer score

of 4.87). The system, The AI Scientist [25], now stands as the paradigm case of end-to-end research automation. In the same paper’s Limitations section, its authors catalogue seven characteristic failure modes of the system: implementation bugs, hallucinated experimental results, shortcut reliance, bug-as-insight reframing, methodology fabrication, frame-lock, and citation hallucinations. Each is a documented consequence of AI decisions made at stages where tacit domain judgment is load-bearing and AI judgment is weak. The paradigm is real; its ceiling is documented from within.

In parallel, a recent survey of agentic AI for scientific discovery [1] names a specific gap in the 2025 literature: “many existing frameworks prioritize fully autonomous workflows. This approach may limit usability for researchers who want to explore their unique ideas, underscoring the need for collaborative approaches that effectively integrate human expertise with AI capabilities.” Two paradigms currently dominate AI-assisted research: end-to-end autonomous systems that attempt full workflow automation, and coding-only assistants such as Cursor [6] and Claude Code [4] used by researchers for implementation work. Neither meets the need the survey describes. The first delegates the stages of research work where AI capability is weakest; the second leaves the stages where AI capability is strongest (breadth-heavy literature synthesis, cross-source pattern recognition) largely unassisted. The middle path, in which researcher framing authority is preserved while AI carries synthesis and execution load, is named as a need but lacks concrete infrastructure.

This paper argues that AI capability in 2026 is unevenly distributed across the stages of research work, and that this asymmetry should be structural in the tools researchers use rather than accidental. The argument is empirical, not philosophical. Frontier AI systems are strong at breadth and synthesis: reading across a literature, identifying patterns, rendering ideas into prose. They are weaker at framing: selecting a research question that is defensible in a specific subfield, committing to a methodology that is not contaminated or confounded, noticing when a tacit domain assumption is load-bearing. This is not a permanent state of affairs; future systems may narrow the gap. But in 2026, the gap is real, the failure modes it produces are documented, and a workflow architecture that treats the gap as structural can make use of AI strength where it exists while preserving researcher authority where it remains load-bearing. This paper’s central empirical claim is that structured provenance enforcement, validation-gated commitment, and a shared substrate for recording disagreement measurably change what an AI-assisted research record contains, not in aggregate outcomes, which require a formal study to assess, but in what is made auditable in the record itself, claim by claim.

*RKA is open-source; source code and documentation are available at <https://github.com/infinitywings/rka>. The system described in this paper was used to develop itself and to produce the present manuscript; see Acknowledgments for the specific division of labor.

We present RKA (Research Knowledge Agent)¹: a workflow infrastructure that operationalizes the middle path through four design commitments. First, a three-actor architecture distinguishes the Researcher (who ratifies framing), the Brain (an AI system that handles strategic synthesis), and the Executor (an AI system that handles mechanical implementation). The three actors are structurally non-peer; the architecture preserves the asymmetry rather than collapsing it. Second, a progressive distillation pipeline carries information from raw observations to typed claims to evidence clusters to research themes, with immutability at each earlier layer and reconstructable interpretation at each later layer. Third, an enforced provenance substrate requires every AI-produced claim to have a traceable chain back to a researcher-authored or cited-literature source; claims without provenance cannot enter the system. Fourth, four operational control properties (traceability, reversibility, visible disagreement, and human-ratified commitment) each map to a specific architectural mechanism and can be checked against the running implementation. Figure 1 previews both the architecture and one operational cycle; subsequent sections describe each component in detail.

The paper’s contributions are:

- **A decomposition of research-workflow AI assistance into a three-actor architecture** in which the Researcher owns framing, a strategic Brain owns synthesis, and a mechanical Executor owns implementation, with a provenance-enforced knowledge substrate mediating between the three.
- **A progressive-distillation pipeline with preserved immutability**, which produces a research record auditable at every stage from raw observation to research theme, with supersede chains rather than overwrites at each interpretive layer.
- **Four operational control properties that make concrete what researcher control means** in an AI-mediated record, each falsifiable against the running system.
- **Eight design principles distilled from eight months of use**, each paired with the architectural mechanism it implies (§5).
- **A four-part evaluation** comprising retrospective self-study, three documented case studies of the pattern the architecture is designed to support, independent-use replication, and a deferred-to-future-work formal multi-arm user study.

Section 2 documents the capability asymmetry; §3 places RKA in a four-position design space and names its control properties; §4 locates the work in five bodies of prior research; §5 presents the eight design principles; §6 describes the system’s architecture, entity model, validation gates, and implementation; §7 reports on the four forms of evidence; §8 draws out the implications and handles the priority of the collaborative-approach direction in the 2025 literature; §9 states the limitations; §10 concludes.

¹The work of building RKA and writing this paper were both carried out using the system described here; see Acknowledgments.

2 Motivation: Uneven AI Capability Across the Research Workflow

In 2026, AI capability is not uniformly distributed across research-workflow stages. At some stages (literature retrieval, synthesis across many sources, mechanical execution of well-defined tasks, paper-prose drafting) frontier AI systems perform at levels that exceed what a single researcher can produce in the same time. At other stages (framing a research question, selecting an evidence standard, committing to a methodology that will not be contaminated or confounded, noticing when a tacit domain assumption is load-bearing) the same systems are consistently weaker. An architecture that treats capability as uniform will either waste AI strength at the first kind of stage or commit AI weakness at the second. An architecture that treats the asymmetry as structural can avoid both. We establish the asymmetry from three sources of published evidence: a stage taxonomy, a catalogue of failure modes from a flagship autonomous system, and an architectural observation about coding-agent scaffolds.

2.1 Stages of research differ in cognitive character

A recent survey of AI-scientist systems [2] decomposes research into six stages: literature review and question formulation; hypothesis generation; experimental design; implementation and execution; result interpretation; and writing. The stages differ in cognitive character. Literature review is breadth-heavy: the work is finding, reading, and summarizing, and the dominant failure is missing something. Implementation is procedural: the work is translating a specification into running code, and the dominant failure is a mechanical error. Writing is pattern-and-language heavy: the work is rendering ideas into genre-conforming prose, and the dominant failure is inaccuracy or over-compression.

Hypothesis generation, experimental design, and result interpretation are different. Each depends on judgment about what counts as a good question in a specific field, what counts as a credible methodology, and what counts as a plausible explanation for a surprising finding. These are matters of tacit expert assessment [20]: recognition-primed judgment built from many accumulated cases, often unnamable until its criterion is violated. A researcher who works in a subfield knows, often without being able to articulate it, which benchmarks have contamination issues, which confounds are unavoidable, which apparent results are likely artifacts of the evaluation protocol.

The stage-by-stage difference in cognitive character is the first half of the motivation. The second half is evidence that today’s AI systems track this difference: capable where the work is breadth-heavy or procedural, weaker where it is judgment-heavy.

2.2 AI capability is unevenly distributed across stages

Two pieces of evidence make this concrete.

The first is the Limitations section of The AI Scientist [25], the first fully autonomous AI research system whose generated paper passed a peer-review round at a workshop of a major machine-learning conference (score 6.33 vs. reviewer average 4.87 at the ICLR

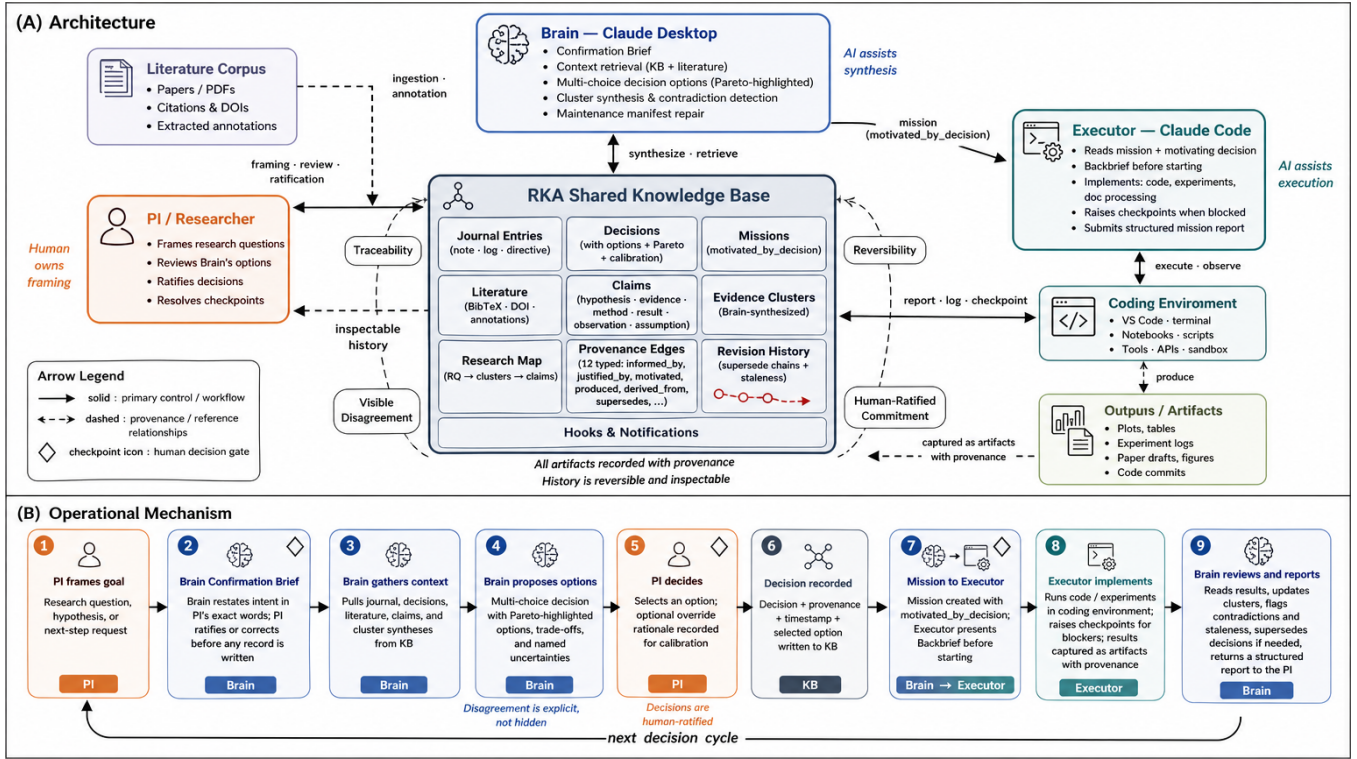


Figure 1: RKA at a glance. (A) Architecture. The three-actor architecture: the Researcher (left) frames problems and ratifies decisions; the Brain (Claude Desktop) synthesizes literature and proposes structured options with trade-offs; the Executor (Claude Code) implements bounded missions and reports back. All three share a typed, provenance-aware knowledge base whose tiles—journal entries, decisions, missions, literature, claims, evidence clusters, research map, hooks and notifications—are the persistent substrate. Four operational control properties (traceability, reversibility, visible disagreement, human-ratified commitment) name what the architecture is designed to provide. **(B) Operational mechanism.** One decision cycle: the Researcher frames a goal; the Brain restates intent (Confirmation Brief), retrieves context, and proposes structured options; the Researcher selects an option and optionally records an override rationale; a mission with the motivating decision is dispatched to the Executor; the Executor implements within bounded scope and submits a structured report; the Brain integrates results, flags contradictions and staleness, and opens the next decision cycle. Solid arrows mark primary control flow; dashed arrows mark provenance and reference relationships; checkpoint icons mark human-ratified decision points.

2025 workshop). The paper’s authors catalogue seven characteristic failure modes: implementation bugs, hallucinated experimental results, shortcut reliance, bug-as-insight reframing, methodology fabrication, frame-lock, and citation hallucinations. Table 1 maps each to the workflow stage where it typically originates.

The pattern in the table is the observation this section argues. The failure modes cluster at framing-adjacent stages (experimental design, methodology commitment, interpretation, the cross-stage judgment of when to revise versus commit) and not at literature retrieval, code writing, or paper-prose drafting. This is consistent with the general pattern: The AI Scientist operates well where the cognitive character is breadth-heavy or procedural, and fails specifically where it is judgment-heavy.

The second piece of evidence is from a more recent survey [1] that evaluates agentic AI systems across benchmark tasks. Agent Laboratory, a representative autonomous-research system, performs competitively on execution-heavy sub-tasks but shows a

significant performance drop on literature-review sub-tasks that require domain judgment about which sources are authoritative. The survey’s authors summarize directly: “many existing frameworks prioritize fully autonomous workflows. This approach may limit usability for researchers who want to explore their unique ideas, underscoring the need for collaborative approaches that effectively integrate human expertise with AI capabilities.” The collaborative-approach gap is named as an open problem in the 2025 literature; this paper provides concrete infrastructure for it.

2.3 The workload-scaffold mis-allocation

A different kind of evidence comes from coding-agent tools repurposed for research. A recent analysis of Claude Code’s internal architecture [24] finds that approximately 1.6% of the system’s architectural complexity handles the AI decision-making path (prompt construction, LLM invocation, result interpretation), while 98.4%

Table 1: The seven failure modes enumerated by the authors of The AI Scientist [25], mapped to the research-workflow stage where each typically occurs, and the RKA mechanism that structurally addresses it.

Failure mode (Lu et al. 2026)	Workflow stage	RKA mechanism that addresses it
Implementation bugs	Execution	Not addressed structurally; depends on Executor and conventional testing.
Hallucinated results	Execution / Synthesis	Enforced provenance; no claim without traceable source.
Shortcut reliance	Framing / Synthesis	Problem Framing gate; multi-option decision records.
Bug-as-insight reframing	Analysis	Immutable raw layer; supersede chain preserves earlier interpretation.
Methodology fabrication	Framing	Researcher owns framing; no method change without attributed decision.
Frame-lock	Framing	Structural disagreement surface; override rates tracked.
Citation hallucination	Synthesis / Writing	Literature records required; cites edges enforced.

handles the deterministic coding harness: file operations, shell invocations, permission checks, context compaction, diff management. This ratio is correct for coding, where the mechanical substrate legitimately dominates.

This ratio is not correct for research. A scaffold optimized for one workload is not automatically suited to a different one, even when the two involve overlapping primitives. The 98.4% of Claude Code’s complexity that handles mechanical coding operations provides limited leverage on the cognitive substrate research work requires: recording which evidence informed which decision, surfacing disagreement between AI interpretation and researcher judgment, tracking which AI-produced claims have been reviewed, preserving a supersede chain when a direction changes. These are research-bookkeeping problems, and the scaffold was not designed for them. We call this pattern the *workload-scaffold mis-allocation*. An architecture for AI-assisted research needs most of its complexity devoted to the research-bookkeeping substrate.

2.4 Researcher expertise remains load-bearing at framing

The final component of the motivation is a claim about researcher expertise: in 2026, no available AI system reliably substitutes for domain-specific tacit judgment at framing stages. We do not argue this must always be true; we argue that at this moment, for most research domains, it is, and that an architecture that treats this as a durable fact will remain useful even if future AI systems narrow the gap.

Consider benchmark selection in empirical machine-learning research: two benchmarks that purportedly measure the same capability give different answers, and which answer to trust depends on contamination status, evaluation-protocol nuances, and whether

the benchmark’s sampling artifacts align with the deployment distribution the paper is trying to speak to. These assessments are not propositional knowledge; they are pattern recognition accumulated from many prior cases. Section 7 documents a specific instance from one of the author’s own research projects where an AI-authored plan was plausible on paper but research-defensibly thin: the plan did not address contamination, saturation, and confound questions that the specific problem class is known to have. The researcher’s domain judgment was what made the plan defensible, through iterative reframing rather than rewriting.

3 Positioning: Four Positions on the Autonomy Axis

The previous section argued that AI capability is unevenly distributed across the stages of research work. This section argues that the tools currently available for AI-assisted research occupy only three of the four defensible positions in the resulting design space. The fourth position, where framing stays with the researcher, execution and synthesis are delegated to AI, and a substrate mediates between them, is the design point this paper addresses.

Figure 2 places four system categories along a single axis: the degree of AI autonomy over research-framing decisions. The axis is not agentic autonomy in general; it is autonomy over the framing work specifically: question selection, evidence standards, methodology commitment. §2 argued this is where AI capability is weakest.

3.1 Four positions, not two

The design space has four distinguishable positions.

High AI autonomy at framing. End-to-end autonomous research systems such as The AI Scientist [25], along with the Claude-Code-hosted wrapper family exemplified by ARIS [43] and EvoScientist [13], delegate framing decisions to AI. A researcher provides a research idea or a topic area; the system selects questions, designs experiments, runs them, interprets results, and writes the paper. This position is coherent and is being pursued seriously. Its characteristic failure mode is the one §2 documented: AI decisions that require domain judgment are made without it, producing the seven failure modes Lu et al. enumerate in their own limitations section.

Middle path. Systems that preserve researcher framing authority while delegating synthesis and execution to AI, mediated by a structured substrate that enforces provenance and surfaces disagreement. This is RKA’s position. The need for this position is named as an open problem in the 2025 literature [1], which observes that “many existing frameworks prioritize fully autonomous workflows” and calls for “collaborative approaches that effectively integrate human expertise with AI capabilities.” Identification of the gap is prior work; the contribution of this paper is operational infrastructure (an architecture, a running system, and an evaluation) for the direction the 2025 literature identifies.

Post-framing automation. PaperOrchestra [42] occupies a narrower and coherent position: automate the paper-writing pipeline after the human has done the research work. Its scope restriction is explicit. The researcher frames, executes, and interprets; a five-agent pipeline handles outline, plot generation, literature integration, section drafting, and refinement. This is not a failure position.

*Aghzal et al. (2025) names the collaborative-approach gap as an open problem.
This paper contributes concrete infrastructure for the middle-path position.*

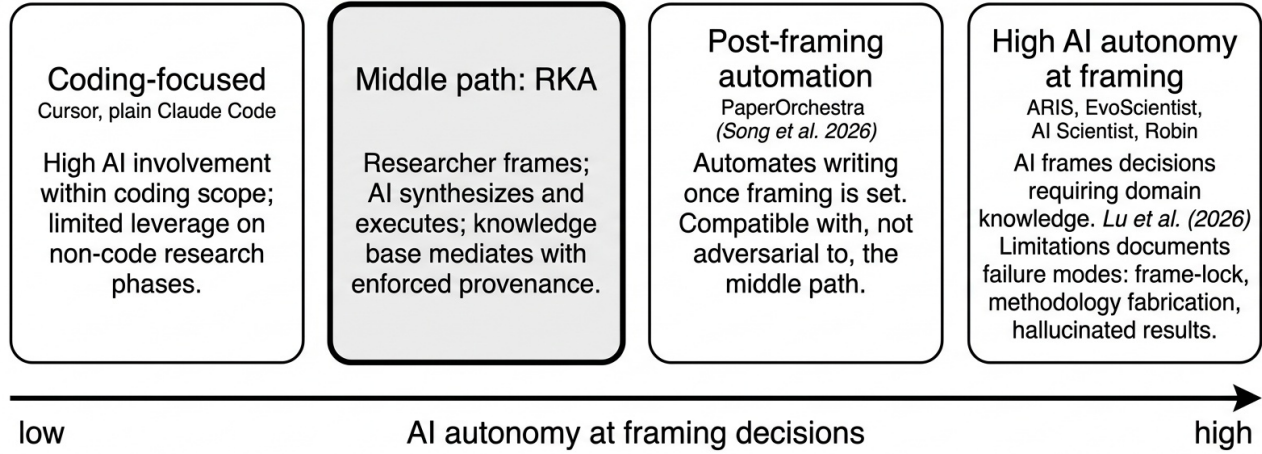


Figure 2: Four positions on the “AI autonomy at framing” axis for AI-assisted research tools. The middle path (RKA) is the only position that structurally separates framing work, where current AI struggles, from execution and synthesis work, where AI excels. PaperOrchestra occupies a distinct post-framing-automation position: it automates paper writing once the researcher has framed the work. Lu et al. 2026 documents failure modes at the high-autonomy position from within the paradigm; Aghzal et al. 2025 names the middle-path gap.

It is a valid design point for a narrower problem, and we name it here so that the four-position picture is more accurate than a two-position dichotomy would be. Post-framing automation is friendly to the thesis of this paper: it stops where the framing work begins, which is where §2 argues researcher authority should remain.

Coding-focused. Cursor [6] and Claude Code [4] occupy a fourth position: high AI involvement within coding scope, with no particular leverage on non-coding phases of research. A researcher using one of these tools for a research project receives meaningful AI assistance for the coding portion of the work and essentially no assistance for the literature-review, framing, synthesis, or interpretation portions. This is a sensible position for many software-engineering tasks. For research tasks, it leaves the breadth and synthesis capabilities §2 documented on the table.

3.2 The middle-path gap is articulated but unoperationalized

The four-position picture clarifies what this paper does and does not contribute. The collaborative-approach direction has been named as a need in the 2025 literature; what has not yet existed, as of 2026, is a concrete architecture that implements the direction at enough depth to support evaluation. The three-actor architecture, progressive distillation pipeline, and provenance-enforced substrate §6 describes are the contribution: not the insight that a middle path is needed, but the specification of what a middle path looks like at the level of entity models, control mechanisms, and validation gates.

Table 2: The four operational control properties that RKA’s architecture is designed to provide. Each property is named, traced to a specific mechanism, and paired with a falsifiable check a reviewer can run against the implementation.

Property	RKA mechanism	Falsifiable check
Traceability	Every claim has a provenance chain to journal or literature.	Can a reviewer answer <i>why does the system believe X?</i> by traversing the graph?
Reversibility	Immutable raw layer + supersede on interpretation.	If a decision is overturned, can earlier interpretations be reconstructed from the record?
Visible disagreement	Structured decision UX with override rationale.	Do override rates and near-miss patterns surface in aggregate data?
Human-ratified commitment	Problem Framing gate; no direction change without attributed decision.	Can an auditor find a lasting project-direction change without a corresponding researcher-attributed decision?

3.3 Four operational control properties

The middle path raises an obvious question: what does “control” mean, operationally, when most of the work is done by AI systems and the researcher is expected to retain framing authority? The word is imprecise on its own. This paper commits to a specific meaning via four falsifiable control properties, each of which maps to an architectural mechanism in §6. Table 2 summarizes them.

The four properties are *traceability*, *reversibility*, *visible disagreement*, and *human-ratified commitment*. Each is a property of the system that can be checked against the implementation, against self-study data, or against case-study narratives. Traceability asks whether every claim has a provenance chain back to a researcher-authored or cited-literature source; a reviewer can answer this by traversing the graph. Reversibility asks whether immutability plus supersede mechanics allow earlier interpretations to be reconstructed after a revision; a reviewer can answer this by examining the supersede chain on any revised decision. Visible disagreement asks whether AI proposals and researcher decisions are recorded as structured, queryable data such that override patterns surface; a reviewer can answer this by querying the structured-decision log. Human-ratified commitment asks whether AI-proposed directions become project direction only after a researcher-visible decision; a reviewer can answer this by examining the validation-gate records.

The four properties are collectively meant to cash out what “researcher control over an AI-mediated research record” means as an engineered claim rather than a slogan. §6.5 shows the architectural mechanism for each.

4 Related Work

RKA draws on and distinguishes itself from five bodies of prior work: agent memory substrates, multi-agent orchestration frameworks, AI-native knowledge tools, scientific research agents and Claude-Code wrappers, and human-AI decision support.

4.1 Agent memory and retrieval substrates

A growing body of work addresses persistent memory for AI agents across sessions. Zep and its successor Graphiti [38, 39] build temporal knowledge graphs with time-valid edges; Letta [23], the productized MemGPT successor, introduces OS-inspired tiered memory; Mem0 [28] provides managed memory with extraction, deduplication, and retrieval APIs; Cognee [9] combines knowledge graphs with vector embeddings; LangMem [22] offers semantic, episodic, and procedural memory tiers for LangChain; LightRAG [16] extends RAG with graph-structured indices. The category’s priority is agent performance: making the agent remember more, forget less, and retrieve faster. Designs optimize for agent-facing retrieval latency.

RKA’s primary consumer is the researcher, not the agent, which reframes the design. Storage is optimized for human auditability rather than agent retrieval latency: provenance chains a researcher can walk, typed edges supporting queries about what the system believes and why. Writes are constrained by attribution requirements enforced at write time. Agent-memory systems could in principle be adapted to this purpose, but none is designed for it. Agent memory sits between an agent and its task; RKA sits between a researcher and the record of AI-assisted work they remain accountable for.

4.2 Multi-agent orchestration frameworks

A second body of work coordinates multiple AI agents. MetaGPT [18] organizes agents around software-engineering roles with structured artifacts passing between them; AG2 [29], continuing Microsoft’s AutoGen, provides configurable agent-to-agent protocols and event-driven orchestration; CrewAI [11] models coordination

as role-specialized agent teams; LangGraph [21] builds systems as explicit state graphs; OpenClaw [35] provides a lightweight multi-agent gateway with markdown agent-identity files. These frameworks solve a real coordination problem: a single frontier-model agent on a long multi-phase task tends to lose context or mix registers, and structured communication between multiple agents outperforms free-form communication.

RKA shares that insight but sits at a different layer. Multi-agent frameworks orchestrate *agents*: how many, what roles, what protocol, what tools. RKA provides the *record* that a three-actor arrangement produces; it does not orchestrate the actors, which are orchestrated by their own runtimes. A multi-agent framework could wrap RKA and use its knowledge base as a shared tool; the inverse is less natural because RKA’s commitments (immutability, attribution enforcement, validation gates) are properties of the record rather than the runtime. Orchestration below, record above. Additionally, multi-agent frameworks default to peer-agent coordination, while RKA’s three actors are structurally non-peer: the Researcher ratifies framing, and the Brain and Executor do asymmetric strategic-versus-mechanical work that the architecture preserves rather than collapses.

4.3 AI-native knowledge tools

A third body builds user-facing tools around retrieval and summarization over heterogeneous document collections. Onyx [34] (formerly Danswer) provides open-source enterprise search across forty-plus sources; Quivr [37] positions as a personal second-brain with opinionated RAG; Khoj [41] indexes personal documents; Notion AI [32] layers AI over collaborative workspace documents; Elicit [12] supports literature search and semi-automated evidence extraction; Semantic Scholar Academic Graph [3] operates at scale with AI-generated TLDRs and citation-context extraction. The category’s priority is access: making large collections queryable through natural language.

RKA has a different primary object. Access tools treat the document collection as primary and the query as a transient event; the collection persists, the query is answered. RKA treats the research project’s decisions, claims, and reasoning chain as primary and documents as provenance-anchored references. The access question is “given this corpus, how do I find what is relevant?”; the RKA question is “given this research project, how do I preserve and audit what I and the AI have concluded so far?” A researcher can sensibly use Elicit or Semantic Scholar to discover literature *and* use RKA to record what they made of that literature.

4.4 Scientific research agents and Claude-Code wrappers

The fourth body is closest to RKA in stated ambition: end-to-end pipelines automating scientific research from literature review through paper generation. The AI Scientist [25] is the flagship recent system, with the failure-mode catalogue §2 discusses. SciAgents [30] predates Lu et al. and implements autonomous scientific discovery through bio-inspired multi-agent graph reasoning. A sub-category has emerged around Claude Code as host runtime: ARIS [43], EvoScientist [13], Orchestra Research [36], Academic Research Skills [19], and PaperOrchestra [42] each package research

workflows as skills, prompts, or orchestration layers invoked from Claude Code. A recent survey [2] catalogs the broader space; an adjacent survey [1] on agentic AI for scientific discovery names collaborative human-AI approaches as an open problem.

Table 3 compares the Claude-Code wrapper sub-category across architectural pattern, scope claim, and failure-mode awareness.

The category’s priority is end-to-end automation. The underlying claim is that with enough scaffolding (role specialization, pipeline orchestration, skill libraries, checklists) research stages can be handled by agents with minimal human intervention. Each component capability has demonstrably improved; composing them with the right scaffolding is a natural engineering goal.

RKA takes the opposite position on where the researcher sits. The wrapper family places the researcher at the endpoints: at the start, prompting the system with a research idea, and at the end, reviewing output. Between those points, the system operates on the researcher’s behalf. RKA places the researcher throughout: framing decisions are ratified at validation gates, not delegated; synthesis is reviewed at evidence-and-theme-elevation checkpoints, not accepted; the record is auditable at every intermediate stage. Paper-Orchestra’s deliberate scope restriction to post-framing writing is another coherent answer. Full end-to-end automation is the posture that makes the Lu et al. failure modes hardest to address structurally. §8 develops the contrast between post-hoc detection (the wrapper family’s evolving approach) and structural prevention (RKA’s bet).

4.5 Human-AI decision support and over/underreliance

A fifth body of work, drawn from HCI, provides the empirical grounding for how humans engage with AI recommendations. Bućinca et al. [7] show that cognitive-forcing functions (requiring user commitment before seeing the AI’s recommendation) reduce overreliance even when the AI is right on average. Ma et al. [27] extend this to trust calibration; a follow-up [26] finds that user self-confidence calibration shifts during AI-assisted work in ways that can mislead users in both directions. Sharma et al. [40] document systematic sycophancy in RLHF-trained models, compounding overreliance whenever the user is unsure. Klein’s [20] naturalistic decision-making establishes that expert judgment operates through pattern recognition rather than exhaustive option evaluation, and Clark and Brennan [8] show that shared understanding is established through structured grounding exchanges rather than assumed. Appropriate reliance on AI — not too much, not too little — is not the default, and effective interventions are either cognitive-forcing or structural; passive improvements do not reliably produce it.

RKA’s design commitments align with the structural-intervention side of this literature. The structured decision UX (§6.5) forces the Brain to present options with tradeoffs and forces the researcher to select and record an override rationale, a cognitive-forcing pattern implemented architecturally rather than as a per-interaction prompt. Validation gates interrupt workflow at transitions, asking for explicit ratification rather than allowing drift through unstructured dialogue. Immutability-plus-supersede makes it possible to audit whether trust was appropriately calibrated after the fact. §7

reports on specific predictions from this literature tested against RKA use.

5 Design Principles

Eight design principles emerged over eight months of building and using RKA across four research projects. Each was arrived at by observing a recurring failure when the rule was not followed. The principles are not universal laws; they are claims about what works for the specific case of human-led research with AI synthesis and implementation support. Table 4 summarizes the eight principles with the consequence we observed when each rule was violated. We discuss each in turn, beginning with the most structural and moving toward the more operational.

5.1 The knowledge base is a shared substrate, not a logging sink

The most fundamental principle names what the knowledge base *is for*. RKA’s substrate is not where the Brain deposits outputs for later retrieval, not where the researcher extracts summaries of AI-completed work, not a trace log read rarely when something breaks. The substrate is the shared memory all three actors read from and write to, continuously, as their primary coordination mechanism.

The distinction determines what gets built. Treating the substrate as a logging sink prioritizes write-side features (low-friction capture, indexing). Treating it as genuine shared memory prioritizes differently: how each actor’s reads and writes fit together, how the record stays navigable, how one actor’s output becomes structured input for the next. Read access is symmetric across actors; write authority is differentiated by role (the researcher ratifies decisions and authors journal entries; the Brain synthesizes claims and drafts missions; the Executor files reports and trace entries). This read-symmetry / write-authority-asymmetry pattern, visible in Table 5, is the structural shape of coordination, not an access-control policy grafted on after the fact.

Violating this principle produces two characteristic failures. The record becomes a write-only log: entries accumulate but don’t inform the next move by another actor. Or one actor’s writes become invisible to the others: the Brain’s synthesis doesn’t surface in the Executor’s mission context; the Executor’s reports don’t reach the researcher’s next audit. In both, the substrate exists but does not function. Building the substrate is not enough; the actors must be designed to use it as their coordination medium.

5.2 Raw data is immutable; interpretation is reconstructable

The second principle names a structural asymmetry: raw data (journal entries, dated observations, researcher inputs, literature records) is immutable once written; interpretation (claims, evidence clusters, themes, ratified decisions) is revisable, but revisions preserve the chain rather than overwriting it.

The pattern comes from a practice older than AI research. Lab notebooks are not erased when a result is later revised; methods sections do not retroactively rewrite the observations that preceded them. What is written is written. RKA applies the same discipline to the AI-authored layer. A claim the Brain wrote on Monday cannot be silently updated on Friday; if it is later found wrong or imprecise,

Table 3: Cluster 4 wrapper-family comparison. Scientific-research-agent systems and Claude-Code wrappers arranged by architectural pattern, scope claim, and failure-mode awareness. PaperOrchestra is listed as a reference point: a post-framing automator rather than an end-to-end system.

System	Year	Architectural pattern	Scope claim	Failure-mode awareness
ARIS (Auto-Research-In-Sleep)	2025	Markdown skills; cross-model review via Codex MCP	End-to-end: idea → paper	Recent /research-wiki addition signals convergence toward KB.
EvoScientist	2024	Self-evolving AI scientist	End-to-end: “Chat an Idea. Get a Paper.”	Minimal explicit handling.
Orchestra Research AI-Research-SKILLs	2026	87 skills across 22 categories; two-loop Autoresearch	End-to-end across coding agents	Prompt Guard (Meta) for input filtering.
Academic Research Skills (Imbad0202)	2026	Claude Code pipeline: research → write → review → revise	End-to-end with integrity gates	7-mode failure-mode checklist; Reviewer Calibration.
PaperOrchestra (reference point)	2026	5-agent pipeline (Out-line/Plot/Lit/Section/Refine)	Post-framing writing only	Scope limitation is itself the failure-mode response.

Table 4: Eight design principles derived from eight months of self-use. Each principle is stated as a rule and paired with the consequence observed when the rule was violated.

Principle	One-sentence statement	Consequence if violated
Knowledge base is a shared substrate	All three actors read from and write to the structured record, each authorized to write only their role’s work.	Write-only log nobody reads, or one actor’s writes invisible to the others.
Raw data is immutable	Journal entries are never overwritten, only superseded by interpretation.	AI-driven erasure of inconvenient observations.
Intelligence lives in the Brain, not the tools	The knowledge base is structured but does not reason; the Brain does.	Feature creep into the substrate; opaque reasoning.
Session-driven, not sync-driven	Enrichment happens when a session does it, not in the background.	Claims without framing context; clusters that aggregate surface keywords rather than thematic arguments.
Provenance as enforced discipline	Every claim has an attributed source; the system refuses claims without one.	Unattributed AI content masquerading as verified knowledge.
Structural surface for disagreement	AI proposals and researcher decisions are structured data, not conversational.	Overrides lost in chat transcripts; patterns invisible.
Defense in depth, not perfect filters	Multiple layers catch different failure subsets; no single check catches all.	Single-point-of-failure illusion; brittle to novel failure modes.
Usage drives features	Features are retired or elevated based on usage; calibration data is an artifact.	Feature bloat; mismatch between design intent and actual use.

it is superseded by a new claim with a supersedes edge back to the old. The old claim remains queryable, and the chain records why the change was made. This asymmetry is what makes reversibility structural rather than procedural: if the raw layer were mutable, no provenance chain could be trusted; if the interpretation layer were immutable, research would be impossible.

Violating this principle produces two mirror-image failures. If raw data is not immutable, an inconvenient observation can quietly disappear when it becomes inconvenient, and the record loses its capacity to contradict the AI’s current story. If the interpretation layer is not revisable (or if revisions overwrite rather than supersede), mistakes calcify and the project cannot learn from its own errors.

5.3 Intelligence lives in the Brain, not in the tools

The third principle constrains the substrate itself: the knowledge base is structured and indexed, but it does not reason. Reasoning happens in the Brain. The substrate stores, retrieves, and enforces constraints; it does not synthesize, evaluate evidence, or make judgments. This is why RKA is not itself an agent framework.

The temptation to put intelligence into the substrate is strong. Modern knowledge-base and retrieval systems routinely include features that look reasonable on their surface: automatic summarization, on-read generative rewriting, “smart” background clustering, inline claim scoring against confidence thresholds. Each collapses the separation between substrate and reasoner. Once the substrate silently rewrites or re-summarizes what it stores, the record becomes a function of when it was read, not of what was written. The traceability property of §6.5 then fails. RKA’s substrate stores what was written, retrieves it without rewriting, and enforces the attribution and edge-type constraints that keep the graph queryable. The Brain reads, reasons, and writes new entities back; summaries are produced at read time rather than cached by the substrate.

Violating this principle produces feature creep into the substrate and opaque reasoning. A reviewer asking why the system believes X cannot get a clean answer when the answer involves silent substrate transformations. Keeping intelligence out of the substrate is the mechanism that keeps the record a record.

5.4 Session-driven, not sync-driven

The fourth principle governs when knowledge work happens. Enrichment (extracting claims from journal entries, assembling clusters, synthesizing themes) is done by the Brain during a session, in response to researcher prompts or as part of a scoped task. It is

not done by a background process silently transforming the record between sessions.

The alternative, a sync process that watches and enriches new entries as they arrive, is obvious-seeming. In practice it creates two problems. It re-introduces the “intelligence in the substrate” failure mode: the background process is effectively a third reasoner whose judgments enter without attribution to the three named actors. And it operates without framing context: claims extracted by a background process can be generic, stripped of the specific question the researcher was pursuing when the entry was written; clusters can aggregate claims sharing surface keywords but addressing distinct research questions. The record fills up, but with material that does not serve the next move.

Session-driven enrichment keeps the Brain’s framing context attached to each act of synthesis. Claims written in a session that has the current research question and recent literature loaded carry that context as provenance. Violating the principle produces a record that fills up faster than it composes: claims without sharp framing, clusters around surface similarities rather than thematic arguments, a research map that is a taxonomy of what the substrate noticed rather than a scaffold of what the researcher is asking.

5.5 Provenance is enforced discipline, not voluntary hygiene

The fifth principle governs how provenance is maintained. Every claim has an attributed source (a journal entry or cited literature). Every decision has an attributed decider. Every mission has a motivating decision or cluster. These are not best-practice encouragements; they are enforced at write time. A write that lacks the required attribution is rejected.

The difference between enforcement and encouragement is a difference in what survives ordinary work. Under encouragement, attribution happens when the actor thinks of it, when the workflow isn’t rushed, when nothing interrupts. Under enforcement, attribution happens because the write cannot succeed without it. There is no “quick note” mode that elides the source field, no claim that enters the graph as a free-floating assertion to be linked up later. The cost is real: some writes are slower than they would otherwise be. The cost buys the property that every entity has a traceable origin. MLflow [44] and the W3C PROV model [15] both developed around this observation: provenance only works as a property of the system if the system will not let you write without it.

Violating this principle produces unattributed content masquerading as verified knowledge. A claim enters the record without a journal entry behind it; another claim later cites it; a theme synthesizes it into a narrative. The narrative looks grounded, and superficially is, but its foundation cannot be reconstructed. Enforcement at write time is what prevents this failure, not because actors are careless, but because under pressure everyone eventually is.

5.6 Disagreement has a structural surface

The sixth principle governs how the researcher and the Brain communicate about choices where they might differ. Rather than exchanging positions in chat and leaving the resolution as conversational residue, RKA records proposals, selections, overrides, and near-misses as structured data. The Brain invokes `rka_present_decision`

which creates a decision entity with proposed options and tradeoffs. The researcher’s selection is recorded as `pi_selected_option_id`; when it diverges from the Brain’s recommendation, a `pi_override_rationale` records why.

The structural form matters because chat does not compose. A hundred conversational exchanges leave behind a hundred utterances in a hundred transcripts: legible one at a time, unsearchable as a pattern. With structured decisions, the same hundred choices leave behind a hundred decision entities in one queryable graph. A reviewer can ask how often the researcher overrode the Brain’s methodology recommendations, on what grounds, whether patterns changed as the project matured. This draws on grounding-in-communication research [8]: shared understanding accumulates through structured exchange. When structured around explicit proposals and selections, grounding becomes visible and repairable. Over time, the record of near-misses and overrides becomes a research artifact itself: a calibration dataset for how the two actors’ judgments diverge.

Violating this principle produces overrides lost in chat transcripts and patterns that remain invisible. The researcher may know they frequently disagree with the Brain on methodology choices but cannot cite the pattern. The Brain cannot learn from near-misses because they were never recorded as such.

5.7 Defense in depth, not perfect filters

The seventh principle governs how RKA responds to the fact that no single check catches every failure. Rather than one filter for all forms of AI drift, RKA relies on multiple partial mechanisms, each catching a different subset: validation gates catch what happens at transitions; attribution enforcement catches what happens at writes; the supersede chain catches what happens when interpretations change; the structured decision UX catches what happens when the researcher disagrees. Coverage comes from composition, not from any one mechanism.

The alternative, one comprehensive filter for all failures, is appealing because it simplifies the story. It also does not work. Failure modes in AI-assisted research are not drawn from a closed distribution that a single classifier can be tuned against. Hallucinated results, methodology fabrications, citation hallucinations, and bug-as-insight reframings are distinct failures with distinct causes. A filter trained on one category will miss the others; a filter broad enough for all will over-fire on innocent work or under-fire on its targets. Several autonomous-research systems have adopted post-hoc failure-mode checklists at late pipeline stages; the pattern those systems display is that the checklist catches what it was built to anticipate and misses what emerged after it was written (§8).

Violating this principle produces the single-point-of-failure illusion: confidence that one comprehensive check catches the failures, while novel failure modes slip past. The principle keeps the architecture honest about what it can and cannot prevent. Each mechanism does one thing well; no mechanism is the last line of defense.

5.8 Usage drives features, not the reverse

The eighth principle governs how RKA evolves. Features are added when use exposes a gap; features are retired when use reveals nobody touches them. The calibration data is RKA’s own record: how

often each tool is invoked, which queries the researcher actually runs, which fields carry information and which stay empty. This is a natural consequence of the principles above: a substrate whose writes are all attributed, timestamped, and typed is also a substrate that makes its own usage legible.

The alternative is feature growth decoupled from use. RKA’s early versions included features added because they seemed principled but were rarely touched in practice: a tiered confidence-decay model for claims, a branch-and-merge UI for clusters, several varieties of automatic enrichment. Each was defended at the time it was built. When usage data was examined, each was used once or twice, never revisited, and the cognitive load of their presence exceeded their benefit.

Violating this principle produces feature bloat: capabilities the researcher works around because removal costs more than ignoring. The principle keeps the system’s complexity bounded by its utility rather than by its history of ambition.

6 System Design

RKA is not a monolithic system. It is a set of structural commitments about what happens at which role, with infrastructure that enforces those commitments at the points where they would otherwise slip. This section names the commitments and describes the mechanisms that instantiate each. Figure 3 shows the three-actor architecture; Figure 4 shows the progressive-distillation pipeline; Tables 5, 6, and 2 name the structured types, the validation gates, and the control properties the architecture is designed to provide.

6.1 The three actors and the narrow interface

RKA distinguishes three structural roles. The *Researcher* owns framing decisions: what research question to pursue, what counts as evidence, what methodology is appropriate. The *Brain* owns long-horizon synthesis and strategic execution: reading the literature, proposing plans, drafting claims, sustaining the argument across sessions. The *Executor* owns mechanical implementation: reading and writing files, running code, invoking shell commands, executing the plans the Brain specifies.²

The distinction is architectural rather than nominal. In current practice the Brain is instantiated by a frontier-model chat assistant with deliberative affordances (broad tool access, extended reasoning, session-scale memory), and the Executor is instantiated by a coding-agent runtime optimized for file-level action at speed. The difference in behavior is not a personality choice. It is produced by differences in system prompts, tool sets, and alignment optimization targets: the Brain is shaped to reason before acting and to sustain strategic context; the Executor is shaped to act decisively within scoped tasks. RKA does not attempt to flatten this gap. It treats the gap as a resource: Brain-like reasoning and Executor-like implementation are each useful, and the architecture is what keeps each actor operating in the register where it performs well.

All communication between the three actors flows through exactly three message types. A *mission* is a directive from one actor to another with an objective, a task list, and links to its motivating

²The RKA codebase uses the short identifier *pi* (principal investigator) for the Researcher role; we use “researcher” in paper prose for readability, but schema identifiers in database fields and tool names (decided_by: *pi*, *pi_override_rationale*, *pi_selected_option_id*) retain the *pi* spelling.

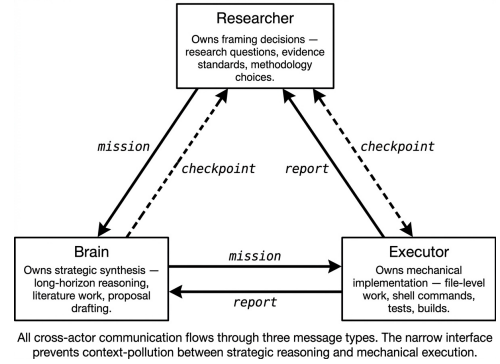


Figure 3: The RKA three-actor architecture. The Researcher owns framing; the Brain owns long-horizon synthesis and strategic execution; the Executor owns mechanical implementation. All communication flows through exactly three types of message (mission, checkpoint, report), which prevents context-pollution between strategic and mechanical work. The knowledge base (not shown here; see Figure 4) sits between the actors as the shared cumulative substrate.

context. A *report* returns findings, artifacts produced, and a status record. A *checkpoint* escalates an unresolved question that the receiving actor cannot or should not decide alone. These are not merely labels on arrows: missions, reports, and checkpoints are persisted first-class entity types in the knowledge base, each with its own lifecycle, provenance metadata, and audit trail. A mission has a motivating decision; a report has produced entities; a checkpoint has a resolver and a recorded resolution.

The narrowness is a feature, not a limitation. Wide interfaces invite the Brain to begin editing files directly (and thereby inherit the Executor’s action-biased register) or the Executor to begin making framing decisions (and thereby inherit the Brain’s latitude without its context). The narrow mission-checkpoint-report interface makes both directions structurally awkward. When the Brain needs file-level work done, it must frame a mission and hand it to the Executor; when the Executor encounters ambiguity in the framing, it cannot paper over it; it must raise a checkpoint. The friction of the interface is not incidental; it is the mechanism by which each actor stays in its role.

One consequence of this design is that the Brain and Executor do not share a conversation. Each session for each actor is independent. What they share is the knowledge base: every mission, report, and checkpoint is a queryable record that either actor can read at session start. We describe this shared substrate in the following subsections; for now, the point is that the architecture’s three-actor separation is enforced at the interface level, not at the memory level. The Brain cannot reach into the Executor’s session and edit an implementation in flight; the Executor cannot reach into the Brain’s session and propose a framing revision mid-task. Each has to go through the record. The record is the synchronization primitive.

6.2 Entity model: seven types with typed provenance edges

The knowledge base that mediates between the three actors is a typed, structured record, not a free-text log. Seven entity types cover what RKA needs to represent: *journal entries* for raw dated observations, *claims* for atomic assertions extracted from journal entries, *clusters* for aggregated claims organized around a theme, *decisions* for recorded choices with alternatives and rationale, *literature* for external references, *missions* for Executor task specifications, and *checkpoints* for researcher-visible unresolved questions. Table 5 names each type together with its purpose, the actor who typically authors it, the actors who typically consume it, the attribution metadata enforced at write time, and whether the entity is immutable or belongs to the supersedable interpretation layer.

Entities relate to each other through typed edges rather than generic links. The seven edge types (*derived_from*, *member_of*, *cites*, *produced*, *references*, *supersedes*, *justified_by*) each carry semantic weight. A claim’s *derived_from* edge back to a journal entry is not the same relation as a claim’s *member_of* edge to an evidence cluster, and neither is interchangeable with a decision’s *justified_by* edge to a cluster. The typing allows the graph to be queried meaningfully: a reviewer can ask for every claim derived from a given journal entry, every decision justified by a given cluster, every theme citing a given paper, without the graph collapsing into an undifferentiated web of connections. This is the mechanism behind the traceability property introduced in §6.5: provenance is not a narrative told after the fact but a navigable structure present at the moment each entity is written.

Attribution is enforced at write time, not recovered at read time. Every journal entry carries a *source* field (brain, researcher, executor, or llm) and, when the source is the researcher, an optional *verbatim_input* field that preserves the researcher’s words exactly. Every claim carries a *decided_by* field recording which actor committed it. Every decision carries a *pi_selected_option_id* field when the researcher has ratified the choice, with a separate *pi_override_rationale* field when the researcher’s selection diverged from the Brain’s recommendation. The schema does not allow these fields to be blank: a write that lacks attribution is rejected. Unattributed content cannot quietly enter the record and later be treated as verified.

The entity model is the shared substrate in the precise sense named in §5: all three actors read from it, and each writes only the entities their role authorizes. The Researcher ratifies decisions and authors journal entries; the Brain synthesizes claims, assembles clusters, and drafts missions; the Executor files mission reports and records implementation-trace entries. Read access is symmetric: every actor can query any entity type to reconstruct the context it needs. Write authority, however, is differentiated by role. Table 5’s *authored by* and *consumed by* columns make this read-symmetry / write-authority-asymmetry pattern visible as structured data. The consequence is that the record functions as a genuine meeting point: when the Brain drafts a mission, the Executor picks it up with full access to the motivating decision and related evidence; when the Executor files a report, the Researcher auditing the work later does not have to reconstruct what was known at the time. Each actor’s reads and writes are bounded by what the substrate enforces, and

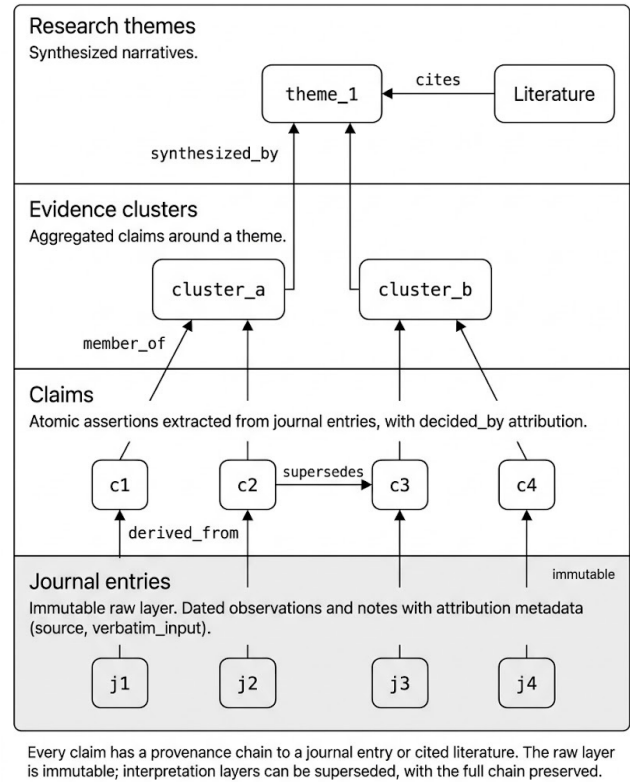


Figure 4: Progressive distillation with enforced provenance. Raw journal entries (bottom layer, immutable) are distilled into atomic claims; claims aggregate into evidence clusters; clusters synthesize into research themes. Each higher layer cites the lower via typed provenance edges. Attribution metadata (*source*, *decided_by*, *verbatim_input*) is enforced at write time; no claim lives in the graph without traceable origin.

those bounds are what allow the three sessions to cohere into a single cumulative record.

6.3 Progressive distillation pipeline

The knowledge base is organized as a four-layer distillation pipeline, illustrated in Figure 4. At the bottom are journal entries, the raw dated observations and notes that each actor writes as work happens. Journal entries feed claims: atomic assertions extracted from one or more journal entries, each with a *derived_from* edge back to its source. Claims feed evidence clusters: aggregations of claims organized around a common theme, connected by *member_of* edges. Clusters in turn feed research themes: synthesized narratives that tell the cumulative story of a line of inquiry, connected to their constituent clusters by *synthesized_by* edges and to external literature by *cites* edges. Each layer compresses and abstracts the layer beneath it; none of the layers erases what came before.

The pipeline shape is not a storage choice. Raw journal entries, taken alone, accumulate but do not compose: a research project with two thousand journal entries and no further structure is a diary, not

Table 5: RKA’s entity model. Seven entity types with their purpose, the actor who typically authors each, the actors who typically consume each, the attribution metadata enforced at write time, and whether the entity is immutable or belongs to the supersedable interpretation layer. The *authored by* and *consumed by* columns make visible the read-symmetry / write-authority-asymmetry pattern that realizes the shared-substrate principle introduced in §5.

Entity type	Purpose	Typically authored by	Typically consumed by	Attribution required	Immutable?
Journal	Raw dated observation or note	All three (Researcher / Brain / Executor); verbatim_input preserves researcher words	All three (context for next move)	source (brain/researcher/executor/llm) + optional verbatim_input	Yes
Claim	Atomic assertion extracted from journal entries	Brain (with researcher review)	Researcher (audit); Brain (synthesis)	derived_from edges + decided_by	Yes once reviewed
Cluster	Aggregated claims around a theme	Brain	Researcher (audit); Brain (synthesis)	synthesized_by + member_of edges	Interpretation layer, supersedable
Decision	A recorded choice with alternatives and rationale	Researcher ratifies; Brain proposes	All three (anchors mission motivation)	decided_by + pi_selected_option_id (when researcher-decided)	Interpretation layer, supersedable
Literature	External reference (paper, article, etc.)	Brain (typically); researcher adds key works	Brain (synthesis); Researcher (audit)	Authors + DOI/URL	Yes
Mission	Executor task specification	Brain	Executor (primary); Researcher (audit)	Initiator + parent decision or cluster	Yes
Checkpoint	Researcher-visible unresolved question	Brain or Executor	Researcher (primary resolver)	Raiser + target resolution	Open until resolved

a knowledge base. Claims give the raw material a form that can be queried, counted, and cross-referenced: *here is an atomic assertion, here is what it was derived from, here is who committed it*. Clusters give claims a thematic organization that supports argument: *these eight claims collectively establish that the chosen benchmark is subject to contamination rather than saturation*. Themes give clusters a narrative arc that supports publication: *this is the story the evidence tells, with these citations to the outside literature*. The distillation pipeline turns the record from a pile of observations into a scaffold for cumulative reasoning.

The asymmetry between the layers is load-bearing. The bottom layer, journal entries, is immutable once written. An observation recorded on a Tuesday cannot be quietly revised on a Thursday because the researcher no longer likes what it says. The interpretation layers above it (claims, clusters, themes) are not immutable. A claim can be superseded by a better-formulated claim; a cluster can be split, merged, or re-synthesized as evidence accumulates; a theme can be rewritten as the argument sharpens. When this happens, the earlier entity is not deleted: a supersedes edge records that the new entity replaces the old, and the old entity remains queryable in the graph. Figure 4 shows one such supersede edge between two claims. The effect is that the interpretation layer is revisable, as it must be because research interpretation changes with evidence, while the raw data that interpretations were built on stays fixed. The full history of how a theme was arrived at, including the interpretations that were replaced, remains navigable.

This asymmetry is what the reversibility property of §6.5 rests on. A researcher who suspects that an AI-generated claim has subtly drifted from its evidence can walk back down the chain (theme to cluster to claim to journal entry) and check. If the claim is wrong, it can be superseded without removing the record of its having been made. If the claim is right, the chain documents why. Neither outcome requires the researcher to trust an AI-produced interpretation on faith. The distillation pipeline is the mechanism

that makes AI-assisted iteration auditable as a matter of structure rather than as a matter of discipline.

6.4 Validation gates

The narrow mission-checkpoint-report interface handles the ongoing flow of work between actors, but some transitions in a research project are too consequential to be resolved as ordinary messages. Choosing a research question, committing to an experimental plan, accepting that a body of evidence supports a claim, elevating a set of claims to a research theme: each of these is a point at which a project’s trajectory can be locked in, and each is a point at which structural pause is warranted. RKA formalizes these as *validation gates*: structured pause points with explicit entry criteria, a named decider, and a recorded outcome.

Four gates are defined. The **Problem Framing** gate triggers before any substantial work on a new research question; the researcher decides whether the question is well-posed and what will count as success. The **Plan Validation** gate triggers after the Brain has produced an experimental or analytic plan; the researcher decides whether the approach fits the question and what the confounders are. The **Evidence Review** gate triggers after an evidence cluster has reached moderate-to-strong confidence; the Brain and researcher together decide whether the claims actually support the synthesis the cluster proposes. The **Synthesis Validation** gate triggers before a cluster is elevated to a research theme; the researcher decides whether the narrative faithfully represents its constituent claims. Table 6 names each gate with its trigger, decider, and typical content.

The gate concept is borrowed, not invented. Cochrane systematic review methodology [17] treats the protocol registration step as a structural commitment: the review’s scope, inclusion criteria, and analytic plan are fixed before evidence collection begins, and

Table 6: RKA’s four validation gates. Each gate is a structured pause point; the researcher decides whether to continue, revise, or abandon.

Gate	When it triggers	Who decides	Typical content
Problem Framing (Gate 0)	Before any substantial work on a new research question	Researcher	Is the question well-posed? What counts as success?
Plan Validation	After Brain produces an experimental or analytic plan	Researcher	Does the approach fit the question? What are the confounders?
Evidence Review	After a cluster reaches moderate-to-strong confidence	Brain + Researcher	Do the claims actually support the synthesis?
Synthesis Validation	Before a cluster is elevated to a research theme	Researcher	Does the narrative faithfully represent the claims?

subsequent deviations must be explicitly recorded. Cooper’s Stage-Gate model [10] treats product-development decisions as structured transitions between phases, each requiring a go/no-go verdict grounded in defined criteria. Both traditions recognize that unstructured decision-making at high-consequence transitions is where the hidden work of a project accumulates, and where frame-lock, methodology fabrication, and citation hallucination (among the failure modes named in Table 1) find the cracks they grow through. The four gates map to these failure modes directly: Problem Framing is where frame-lock and shortcut reliance are caught; Plan Validation is where methodology fabrication is caught; Evidence Review is where hallucinated results and citation hallucinations are caught; Synthesis Validation is where bug-as-insight reframing is caught.

A gate is distinct from a checkpoint. A checkpoint is raised when the Brain or Executor encounters an unresolved question it cannot or should not decide alone; a gate is a scheduled structural transition that is always required at the named point, regardless of whether an actor has raised a question. The distinction matters because it prevents the research process from skipping a consequential transition just because no actor happened to flag it. Gates are the mechanism behind the human-ratified commitment property of §6.5: no lasting change of project direction occurs without a researcher-attributed decision at the appropriate gate.

6.5 The four control properties, mechanically

Section 3 introduced four operational control properties that RKA’s architecture is designed to provide: traceability, reversibility, visible disagreement, and human-ratified commitment. Table 2 lists them alongside the falsifiable check a reviewer can run against each. Having described the entity model, the distillation pipeline, and the validation gates, we can now name the specific mechanism behind each property.

Traceability. Every claim in the knowledge base has a provenance chain back to a journal entry it was derived from, or to a cited piece of literature, or to both. The `derived_from` edges between claims and journal entries, the `member_of` edges between claims and clusters, the `synthesized_by` edges between clusters

and themes, and the `cites` edges to literature records together form a graph that can be walked in either direction. A reviewer asking *why does the system believe X?* traverses downward from the theme to the clusters that synthesized it, to the claims that constitute those clusters, to the journal entries those claims were derived from. No claim lives in the graph without this chain; a write that lacks the required attribution is rejected at the schema layer, as described in §6.2. Traceability is not a policy RKA asks the actors to follow. It is an invariant the substrate maintains.

Reversibility. The immutable-raw-layer / supersedable-interpretation-layer asymmetry described in §6.3 is the mechanism. Journal entries, once written, cannot be altered. Claims, clusters, and themes can be revised, but the revision produces a supersedes edge rather than an overwrite: the earlier entity remains queryable, linked to its replacement. A decision that later turns out to have been premature can be superseded by a better decision without erasing the record of the earlier choice. A cluster whose synthesis is found to overstate its claims can be re-synthesized, with the earlier synthesis preserved as context for how the understanding evolved. Reversibility is the guarantee that no interpretation, neither the AI’s nor the researcher’s, is permanent in a way that prevents later reconsideration.

Visible disagreement. When the Brain has a decision to propose, it does not state a conclusion in chat. It invokes `rka_present_decision`, which records the decision as a structured entity with multiple option entries, each carrying the Brain’s analysis of the option’s trade-offs. The researcher selects one option (recorded as `pi_selected_option_id`) and, when the selection diverges from the Brain’s recommendation, supplies a `pi_override_rationale` that explains why. The structured form matters: overrides, near-misses, and default-accepts become queryable data rather than conversational residue. Aggregate patterns surface. A reviewer can ask how often the researcher overrode the Brain on methodology choices, and on what grounds; a research lab can ask whether one researcher’s override patterns differ systematically from another’s. Disagreement is made visible not by documenting it after the fact but by giving it a structural surface at the moment it occurs.

Human-ratified commitment. The validation gates described in §6.4 are the mechanism. No new research question begins work before the Problem Framing gate records the researcher’s judgment that the question is worth pursuing; no experimental plan is executed before the Plan Validation gate records the researcher’s acceptance of the approach; no cluster is elevated to a research theme before the Synthesis Validation gate records the researcher’s agreement that the narrative is faithful. An auditor examining the project’s decision records can reconstruct, for any lasting direction change, the researcher-attributed decision that ratified it. The commitment property is an architectural claim, falsifiable against the record: if a lasting change exists without a corresponding gate-resolved decision, the claim fails. We do not make the claim lightly.

6.6 Implementation stack

RKA is built on commodity infrastructure (Figure 5). The backend is written in Python against a SQLite database, with the knowledge-base schema enforcing attribution, edge typing, and immutability at

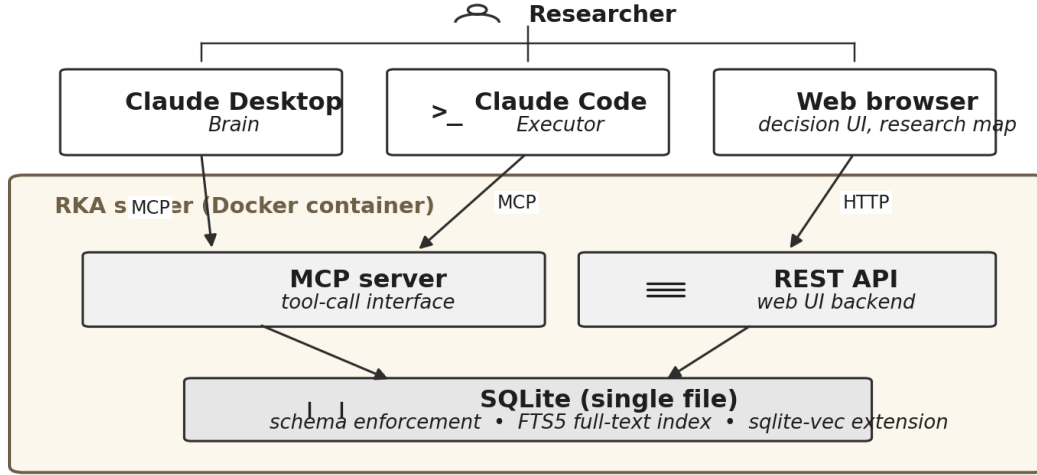


Figure 5: RKA’s implementation stack. The three actor runtimes (Claude Desktop for the Brain, Claude Code for the Executor, a web browser for researcher-side decision review) connect to a single RKA server through two typed interfaces: the Model Context Protocol for the two AI runtimes, and an HTTP REST API backing the React web UI. All three paths converge on a single SQLite file whose FTS5 index provides full-text search and whose sqlite-vec extension provides semantic search. The architecture is deliberately commodity: no distributed store, no message bus, no custom protocol.

the storage layer; the web frontend is a React application for browsing the research map, decisions, and supersede chains; integration with the actor runtimes is provided through a Model Context Protocol [31] server that exposes the knowledge-base operations as tool calls. The Brain is instantiated as Claude Desktop with the RKA MCP server connected; the Executor is instantiated as Claude Code with the same MCP server connected, together with role-specific Agent Skills [5] that shape the session to the Executor’s mechanical-implementation register. None of these components is exotic. The substrate is an ordinary embedded database; the actor runtimes are off-the-shelf chat and coding-agent products; the integration protocol is an open standard. What the architecture contributes is the schema and the discipline it enforces, not novel infrastructure.

7 Evaluation

This section presents four forms of evidence for RKA’s utility in practice: a retrospective self-study of the system’s use on its own development over eight months; three case studies from a separate research project that document the specific pattern RKA’s architecture is designed to support; three independent-use cases by researchers other than the author (one described in detail below, two in progress with artifacts being captured); and an honest deferral of a formal multi-arm user study to future work. Each form of evidence establishes a specific thing and is limited in a specific way; together they establish the system’s existence, utility, and failure-mode resilience as a matter of record rather than speculation.

The evaluation this section presents is qualitative and descriptive, consistent with the evaluation norms for systems contributions in

the HCI literature where the architecture’s claims concern what structures the system makes possible rather than what frequency distributions it produces. Three research questions organize the evidence. **RQ1:** Does the three-actor architecture produce an auditable record of AI-assisted research work, preserving provenance across extended use? **RQ2:** Does the structured-decision interface surface disagreement between researcher judgment and AI proposals in a form that supports later review? **RQ3:** Is the system usable by researchers other than the author on research problems other than the author’s? Self-study (§7.1) addresses RQ1 and RQ2 across eight months of use. Case studies (§7.2) document the specific reframing pattern the architecture is designed to support. Independent-use replication (§7.3) addresses RQ3 through the three N=3 cases. The formal multi-arm user study (§7.4) would address prevalence questions outside the scope of this paper.

7.1 Retrospective self-study of rka_development

The RKA project has itself been an RKA-tracked project for eight months, and the present paper was written using the same three-actor workflow the architecture formalizes. The knowledge base for this project, referred to internally as `rka_development`, contains every architectural decision, every implementation mission, every review gate, and every superseded direction from the system’s early design through the submission of this paper. This makes the RKA project a self-study candidate: a record of AI-assisted research work exists for the period during which the system was being built, and the question of whether the system supported the researcher’s work can be asked of the record.

We report three metrics over the eight-month window. The first is *provenance coverage*: the percentage of claims in the knowledge base with a complete provenance chain back to a researcher-authored or cited-literature source. Olah and Carter [33] argue that research debt, the accumulated burden of poorly documented inferences, is a first-order problem in how research fields accumulate or fail to accumulate knowledge. Provenance coverage is a direct measure of the system’s resistance to research debt in its own record. The second is *research-debt trajectory*: the rate at which claims enter the knowledge base in an uncovered state versus the rate at which covered claims are produced. A system that accumulates research debt faster than it retires debt has a trajectory problem that scales poorly over time. The third is *mission-cycle metrics*: the distribution of mission duration, checkpoint frequency, and mission-to-report cycle times. These bear on whether the structured mission workflow described in §6 adds enough friction to be ignored or little enough to be used. Concrete values for these three metrics are being accumulated longitudinally as part of the formal multi-arm user study described in §7.4; this section presents the methodology and the instruments, with the specific measurement values reported at study completion.

The self-study is explicit about its limitation: N=1 user, namely the author. Self-study shows that the system is usable by at least one researcher over extended time; it does not show that the system is usable by an arbitrary researcher, and it does not rule out the possibility that the author’s familiarity with the system biases the metrics upward. What it establishes is existence rather than prevalence, and a longitudinal trajectory rather than a single snapshot.

7.2 Three case studies from the edge-cloud-agent project

Three case studies come from a separate research project, a study of edge-cloud LLM-agent deployment referred to as edge-cloud-agent, that used RKA for its framing, synthesis, and decision-recording work. Each case study documents the same pattern: an AI-authored research plan that was plausible on paper but research-defensibly thin; researcher pushback from domain knowledge specific to the problem class; iterative reframing through structured decisions recorded in the RKA substrate; and a final plan that was research-defensible. The pattern is general; the specifics are ML-systems empirical work.

Each case is an *archival* artifact. The before-and-after states, the researcher’s verbatim pushback language, the AI’s responses, and the decisions ratified at each iteration are preserved in the RKA record as typed entities with provenance chains. The case-study section is not a reconstruction for the paper; it is a walk through evidence that was already in the record.

Case 1: Benchmark variant selection. The problem: the edge-cloud-agent project needed to choose between SWE-Bench Verified, SWE-Bench Pro, and full SWE-Bench as its evaluation benchmark. The AI-authored initial plan recommended one variant with a brief justification; the researcher’s pushback, recorded verbatim in the decision log, raised concerns about data contamination risk in the full benchmark (some of the resolved issues are in the training data of the models being evaluated) and ceiling effects on the Verified

subset (the smaller, hand-verified set saturates too quickly to distinguish meaningful improvements from the recent state of the art). The iterative resolution produced a decision with an explicit tradeoff analysis and a final variant choice with reasoned justification for each discarded alternative. Figure 6 traces the full decision chain for this case, including the supersede link from the initial AI plan to the revised plan and the evidence entries that justified each revision.

Case 2: Scaffold selection. The problem: the project needed to select an agent scaffold (SWE-agent fork-and-modify, Agentless, or a from-scratch reimplement) with enough comparability to published baselines that results would be interpretable. The AI-authored initial plan proposed fork-and-modify of SWE-agent for development speed. The researcher’s pushback identified a confound: modifications to a baseline scaffold produce results that cannot be cleanly compared to the published baseline, because the modifications themselves are a treatment in the experimental sense. The final decision selected Agentless as the scaffold, with the rationale that its simpler design produces a cleaner comparison surface even at the cost of somewhat slower development. The decision is recorded with both options represented and the choice rationale captured, which makes it auditable by a future reviewer or by the researcher themselves six months later.

Case 3: Parameter specification. The problem: the project required committing to values for twenty-four specific design parameters (temperature, output length caps, prompt form, token accounting for retry logic, rescue-rate protocols, and many others) before experiments could run. The AI-authored initial plan specified defaults for all twenty-four. The researcher’s iterative pushback, across seven rounds over three sessions, surfaced inconsistencies (two parameters set to incompatible values), missing justifications (a choice between two equally-reasonable values with no recorded reason), and underspecified handling of edge cases (retry protocol for sequential-API failures not distinguished from retry protocol for single-call failures). The final specification documented twenty-four parameter choices with reasoned justifications, including explicit flags on the three parameters for which the research team committed to a choice while acknowledging that the choice was under-motivated and would be revisited if results were sensitive to it.

Each case is a documented occurrence of the pattern RKA’s architecture is designed to support: structural prevention of frame-lock through researcher-visible decision surfaces, auditable revision through supersede chains, and preserved reasoning through typed provenance edges. The three cases collectively establish that the pattern is documentable. They do not establish that the pattern is the majority case across AI-assisted research broadly; that is a prevalence question and requires a larger study.

7.3 Independent-use replication

The closest evidence in this paper to external validation is a set of three independent-use cases: researchers other than the author using RKA on their own research problems. One of the three, a study of CVE-TTP mapping for vulnerability-intelligence research, is described in detail below based on the author’s observations

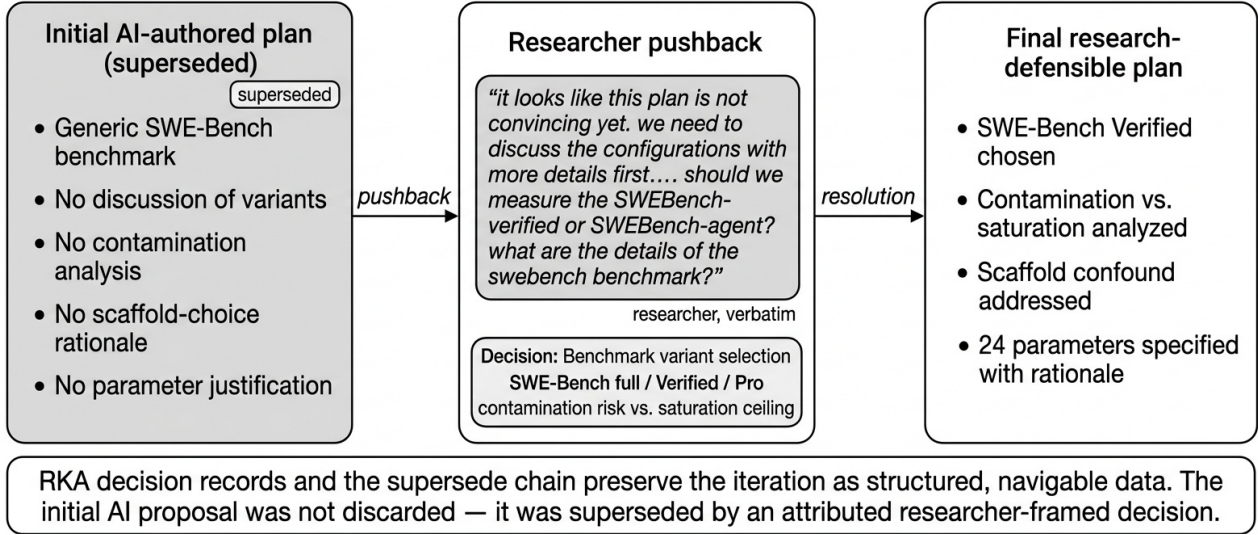


Figure 6: A case study from the edge-cloud-agent project. The initial AI-authored research plan (left) was plausible in form but thin on domain-specific justification. Researcher pushback from domain knowledge about benchmark contamination and saturation (verbatim in the middle panel) reshaped the plan through RKA’s decision-and-supersede mechanics. The final plan (right) is research-defensible because the domain knowledge shaped every non-trivial choice. The supersede chain and decision records are preserved in RKA’s journal; the iteration is navigable, not buried in chat transcripts.

of approximately 150 tool calls in a separate Claude instance; the other two uses are in progress and will be reported with full artifacts in the formal user study. The other two are currently being documented for the formal user study; this section describes the CVE-TTP case as the artifact-backed exemplar of the pattern all three cases exhibit. The researcher who used the system in the CVE-TTP case is not the author of this paper. The use pattern was not prompted by the author; it emerged from the researcher reading the RKA documentation and deciding that the tool fit the research workflow they were already doing.

The field report distinguishes three categories. First, features used without prompting: the mission-and-report cycle, decision records on research questions, and literature-ingest-and-link. Second, features ignored: the full distillation pipeline to evidence clusters was not exercised; the user treated the system primarily as a structured journal. Third, features revised in response to friction: the validation-gate workflow was experienced as too heavyweight for a solo researcher working in short bursts, and the user adapted by applying gates only at explicitly committed transitions rather than at every decision. This is the closest thing to a usability signal this paper produces: a different user using the same system on a different problem, self-reported in a field note as the artifact-backed exemplar of a pattern observed across all three independent users.

The limitations of independent-use replication as evidence are real. Three users is not yet representative of the population of researchers who might use the system. A 150-tool-call record is shorter than the author’s eight-month self-study. The field report is not a controlled study. What it establishes is that the system is usable by researchers other than the author, on research problems

other than the author’s, through periods of use long enough to expose real friction.

7.4 Formal user study: future work

A proper evaluation of whether RKA improves appropriate reliance on AI, reduces the failure modes in Table 1, or produces research-defensible work more reliably than unstructured AI-assisted work would require a multi-arm user study: Claude.ai conversational use in one arm, plain Claude Code coding-agent use in a second arm, and RKA-mediated use in a third arm, across matched research tasks and across multiple researchers. The design for such a study, including task specification, calibration-dataset metrics, and a pre-registered analysis plan, is drafted but not yet run. The blocking constraints are IRB logistics, onboarding-materials readiness for non-author users, and longitudinal data accumulation that is still in progress at submission (N=3 independent users as of this writing, with formal artifact capture underway for the two in-progress cases and the detailed CVE-TTP case described in §7.3 above).

The formal study would strengthen the prevalence claims that this paper does not make. The existence claims (that the failure modes are real, that they are documentable, that RKA’s architecture changes what happens when they occur) already stand on the evaluation this section has presented.

8 Discussion

8.1 Implications of the asymmetric-labor thesis

Three implications follow from the asymmetric-labor argument. First, autonomous-research-agent projects face a domain-framing ceiling that more autonomy cannot remove: adding autonomy expands AI authority over exactly the stages where AI judgment is

Table 7: Post-hoc detection (the wrapper family’s approach) vs. structural prevention (RKA’s approach) for five of the failure modes catalogued in [25].

Failure mode	Wrapper-family approach (post-hoc)	RKA approach (structural)
Frame-lock	7-mode checklist at Stage 2.5/4.5 blocks pipeline	Researcher owns framing; structural disagreement surface.
Methodology fabrication	Reviewer calibration mode (FNR/FPR on gold set)	No method change without attributed decision + Plan Validation gate.
Citation hallucination	Semantic Scholar API verification (Tier 0)	Literature records required at claim creation; cites edges enforced.
Hallucinated results	5× ensembling + cross-model review	Provenance chain required; result without source is rejected at write time.
Bug-as-insight	Anti-leakage protocol	Immutable raw layer; supersede preserves evidence even when interpretation changes.

weakest, compounding the failure modes of Lu et al. [25] rather than reducing them. Second, coding-only-assistant usage leaves AI breadth-capability unused: a researcher using Cursor or plain Claude Code receives AI assistance for one procedural stage out of six, while the breadth-heavy literature-and-synthesis work that frontier AI systems do well remains manual. Third, infrastructure that lets framing stay with the researcher while AI carries synthesis and execution load is a distinct, under-explored design point; RKA’s architecture is one implementation of it, with the three non-peer roles and provenance-enforced substrate as the specific decomposition.

8.2 Post-hoc detection versus structural prevention

The Claude-Code wrapper family described in §4.4 has converged on a shared pattern for addressing the failure modes Lu et al. catalogue: ship a failure-mode checklist that the pipeline runs its outputs through at designated stages. This is post-hoc detection. The pattern is reasonable: known failure modes admit detectors, and detectors run at the right stage catch failures before they propagate.

RKA takes the opposite approach to the same failure catalogue: make the failures difficult to commit in the first place rather than detect them after. Table 7 maps each of Lu et al.’s seven failure modes to the RKA mechanism intended to prevent it structurally. Frame-lock is blocked by validation-gated framing; bug-as-insight reframing is blocked by immutable records and auditable supersede chains; methodology fabrication is surfaced by the structured-decision interface; citation hallucinations are blocked by provenance enforcement at write time.

Post-hoc detection and structural prevention are not mutually exclusive. Where they differ is in what classes of failure they handle well: post-hoc detection needs failures to produce recognizable

output signals, while structural prevention handles failures that produce no distinguishable signal (typical of framing-adjacent failures, where the output looks reasonable and the failure is a domain-judgment problem the text does not surface).

8.3 Priority of the collaborative-approach direction

The middle-path direction that RKA occupies is not a novel identification. Aghzal et al. [1] name the need for collaborative approaches in the 2025 literature; the observation that fully autonomous workflows may limit usability for researchers exploring their own ideas is theirs. This paper’s contribution is operationalization of the direction at enough depth to support evaluation: a concrete three-actor architecture, a running system, eight months of self-study, three documented case studies, and independent-use replication. Identification of the direction is prior work; infrastructure for it is where this paper enters.

9 Limitations

This paper is explicit about what its evaluation establishes and what it does not. Six limitations bound the claims.

First, the behavioral claims about framing authority are empirical 2026 observations, not philosophical commitments. Future AI systems may narrow the framing capability gap, and if they do, the specific distribution of work between Researcher, Brain, and Executor described in §6 could be revisited. The architectural argument of this paper (that structural provenance, reversibility, visible disagreement, and ratified commitment are useful properties for an AI-mediated research record) does not depend on the current framing gap being permanent. What depends on the current gap is the specific division of labor the paper proposes; the architectural commitments stand independently.

Second, the evaluation is author-use plus three independent-use replications (one described in detail, two in progress) plus three case studies. These are existence proofs, not prevalence claims. The paper does not quantify how often the failure modes in Table 1 occur in the wild, and it does not quantify how much RKA’s architectural mechanisms reduce their frequency. The claims this paper makes about failure modes are that they are real, that they are documentable, and that RKA’s structure changes what happens when they occur. The claims the evaluation does not support are distributional: the prevalence of the failures across AI-assisted research generally, the magnitude of RKA’s preventive effect, and the population of researchers for whom RKA would produce similar outcomes.

Third, the calibration layer is the paper’s primary evaluation instrument, and at submission, longitudinal calibration data is still accumulating (N=3 independent users). The metrics the calibration layer produces (selection patterns across structured decisions, override-rate distributions, provenance-coverage trajectories) require more data before the empirical claims they support become robust. The design of the calibration layer is defensible at this point; the empirical claims that would rest on it are not yet.

Fourth, case-study generalizability is bounded. The three case studies come from one research project, one researcher, and one problem class (ML-systems empirical work). The pattern the case

studies document (initial AI plan, domain-knowledge pushback, iterative reframing, research-defensible final plan) is plausibly general, because framing-iteration of this kind is common across research fields. Whether it is general is an empirical question this paper does not answer.

Fifth, the paper’s analysis of coding-agent scaffolds is narrowly scoped. The workload-scaffold-mis-allocation argument in §2.3 rests on architectural data for Claude Code specifically [24]. Equivalent architectural data for Cursor, Codex-based tools, or other coding-agent systems does not yet exist in the public literature, and the paper does not attempt to reconstruct it. A reader who wants to generalize the argument to other coding-agent scaffolds will find the pattern suggestive but not demonstrated.

Sixth, the paper’s own production involved the system it describes. The author’s use of RKA to develop the RKA system and to draft this paper creates a recursive-validation concern: the evaluation might be describing a workflow that is specifically calibrated to its author rather than a generalizable pattern. The independent-use replication in §7.3 partially addresses this concern by documenting non-author use, but the concern is real and the formal user study described in §7.4 is the appropriate instrument for distinguishing system-effect from author-effect. We name this limitation explicitly rather than leaving it as a background assumption.

10 Conclusion

AI capability in 2026 is unevenly distributed across research workflow stages: strong at breadth, execution, and synthesis; weaker at framing and tacit domain judgment. RKA treats this asymmetry as structural. Three non-peer actors (Researcher, Brain, Executor) collaborate through a provenance-enforced knowledge substrate with a progressive distillation pipeline and four operational control properties, letting researchers scale AI-assisted work without surrendering the ability to audit, contest, or reverse any AI-produced interpretation. The architectural claims stand on eight months of self-study and documented case studies; a formal user study is in preparation.

Acknowledgments

The RKA system described in this paper was itself developed with RKA assistance, and the writing of this paper was carried out using the same infrastructure [14]. The project’s knowledge base contains the decision history, mission reports, design-review checkpoints, and literature-ingestion records that produced the system over eight months, and the paper’s outline, draft, and revisions were produced through the same three-actor architecture the paper describes. Specifically, the Brain (Claude Desktop) drafted section text and proposed revisions; the Executor (Claude Code) handled \LaTeX typesetting, figure generation, build verification, and find-and-replace edits; the Researcher (the author) ratified framing, structural decisions, and content choices at validation gates. The paper’s argument, empirical claims, and architectural commitments are the author’s; the scaffolding that produced the manuscript is a documented, reproducible application of the system the paper describes.

References

- [1] Mohamed Aghzal, Erion Plaku, Gregory J. Stein, and Ziyu Yao. 2025. Agentic AI for Scientific Discovery: A Survey of Progress, Challenges, and Future Directions.

- arXiv preprint arXiv:2503.08979* (2025). doi:10.48550/arXiv.2503.08979
- [2] AI Scientists Survey Authors. 2025. A Survey of AI Scientists. *arXiv preprint* (2025).
- [3] Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, et al. 2022. The Semantic Scholar Academic Graph (S2AG). *Semantic Scholar* (2022).
- [4] Anthropic. 2024. Claude Code: Agentic coding assistant. <https://www.anthropic.com/claude-code>.
- [5] Anthropic. 2025. Equipping agents for the real world with Agent Skills. <https://www.anthropic.com/news/agent-skills>.
- [6] Anysphere. 2023. Cursor: AI-Native Code Editor. <https://www.cursor.com>.
- [7] Zana Bućinca, Maja Barbara Malaya, and Krzysztof Z. Gajos. 2021. To Trust or to Think: Cognitive Forcing Functions Can Reduce Overreliance on AI in AI-assisted Decision-making. *Proceedings of the ACM on Human-Computer Interaction (CSCW)* (2021). doi:10.1145/3449287
- [8] Herbert H. Clark and Susan E. Brennan. 1991. Grounding in Communication. *Perspectives on Socially Shared Cognition* (1991).
- [9] Cognee Team. 2025. Cognee: Self-improving memory engine for AI agents. <https://github.com/topoteretes/cognee>.
- [10] Robert G. Cooper. 1990. Stage-Gate Systems: A New Tool for Managing New Products. *Business Horizons* 33, 3 (1990), 44–54. doi:10.1016/0007-6813(90)90040-1
- [11] CrewAI Team. 2024. CrewAI: Framework for orchestrating role-playing AI agents. <https://github.com/crewAIInc/crewAI>.
- [12] Elicit Team and Ought. 2024. Elicit: Language Models as Research Assistants. <https://elicit.com>.
- [13] EvoScientist Contributors. 2025. EvoScientist: Harness Vibe Research with Self-evolving AI Scientists. <https://github.com/EvoScientist/EvoScientist>.
- [14] Chenglong Fu. 2026. RKA: Research Knowledge Agent. <https://github.com/infinitywings/rka>. Open-source research-orchestration infrastructure..
- [15] Paul Groth and Luc Moreau. 2013. W3C PROV: An Overview of the PROV Family of Documents. <https://www.w3.org/TR/prov-overview/>.
- [16] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. *arXiv preprint* (2024).
- [17] Julian P. T. Higgins, James Thomas, Jacqueline Chandler, Miranda Cumpston, Tianjing Li, Matthew J. Page, and Vivian A. Welch (Eds.). 2024. *Cochrane Handbook for Systematic Reviews of Interventions* (version 6.5 ed.). Cochrane. <https://training.cochrane.org/handbook> Updated August 2024.
- [18] Sirui Hong, Mingchen Zhuge, Jonathan Chen, et al. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *International Conference on Learning Representations (ICLR)*.
- [19] Imbad0202. 2026. Academic Research Skills for Claude Code: research → write → review → revise → finalize. <https://github.com/Imbad0202/academic-research-skills>.
- [20] Gary A. Klein. 1998. *Sources of Power: How People Make Decisions*. MIT Press.
- [21] LangChain Team. 2024. LangGraph: Build resilient language agents as graphs. <https://github.com/langchain-ai/langgraph>.
- [22] LangChain Team. 2025. LangMem SDK: Agent long-term memory for LangGraph. <https://github.com/langchain-ai/langmem>.
- [23] Letta Team. 2024. Letta (formerly MemGPT): Stateful AI agents with tiered memory. <https://github.com/letta-ai/letta>.
- [24] Jiacheng Liu, Xiaohan Zhao, Xinyi Shang, and Zhiqiang Shen. 2026. Dive into Claude Code: The Design Space of Today’s and Future AI Agent Systems. *arXiv preprint cs.SE* (2026). <https://arxiv.org/abs/2604.14228>
- [25] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2026. Towards end-to-end automation of AI research. *Nature* 651 (2026), 914–919. doi:10.1038/s41586-026-10265-5
- [26] Shuai Ma, Ying Lei, Xinru Wang, Chengbo Zheng, et al. 2024. Are You Really Sure? Understanding the Effects of Human Self-Confidence Calibration in AI-Assisted Decision Making. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*.
- [27] Shuai Ma, Ying Lei, Xinru Wang, Chengbo Zheng, Chuhan Shi, et al. 2023. Who Should I Trust: AI or Myself? Leveraging Human and AI Correctness Likelihood to Promote Appropriate Trust in AI-Assisted Decision-Making. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*.
- [28] Mem0 Team. 2025. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. <https://github.com/mem0ai/mem0>.
- [29] Microsoft Research and AG2 Contributors. 2024. AG2 (formerly AutoGen): The Open-Source AgentOS. <https://github.com/ag2ai/ag2>.
- [30] MIT. 2025. SciAgents: Automating Scientific Discovery Through Multi-Agent Intelligent Graph Reasoning. *arXiv preprint* (2025).
- [31] Model Context Protocol Working Group. 2025. MCP Specification: Elicitation. <https://modelcontextprotocol.io>.
- [32] Notion Labs. 2024. Notion AI: Enterprise Search and Workspace AI Assistant. <https://www.notion.com/product/ai>.
- [33] Chris Olah and Shan Carter. 2017. Research Debt. *Distill* (2017). doi:10.23915/distill.00005
- [34] Onyx Contributors. 2024. Onyx (formerly Danswer): Open Source AI Platform for Enterprise Search. <https://github.com/onyx-dot-app/onyx>.

- [35] OpenClaw Contributors. 2025. OpenClaw: Multi-Agent Gateway Runtime with Markdown-Based Agent Identity. <https://github.com/openclaw>.
- [36] Orchestra Research Contributors. 2026. Orchestra Research AI-Research-SKILLS: Open-source library of AI research and engineering skills. <https://github.com/orchestra-research/AI-Research-SKILLS>.
- [37] QuivrHQ. 2024. Quivr: Opinionated RAG for Integrating GenAI in Applications. <https://github.com/QuivrHQ/quivr>.
- [38] Preston Rasmussen, Pavlo Kovalov, and Daniel Paripati. 2025. Zep: A Temporal Knowledge Graph Architecture for Agent Memory. *arXiv preprint* (2025).
- [39] Preston Rasmussen and Zep Team. 2025. Zep/Graphiti: Temporal Knowledge Graph Architecture for Agent Memory. <https://github.com/getzep/graphiti>.
- [40] Mrinank Sharma, Meg Tong, Tomasz Korbak, et al. 2024. Towards Understanding Sycophancy in Language Models. In *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/2310.13548>
- [41] Debanjum Singh Solanky, Saba Imran, and Khoj Contributors. 2024. Khoj: Open-source personal AI assistant for your digital brain. <https://github.com/khoj-ai/khoj>.
- [42] Yiwen Song, Yale Song, Tomas Pfister, and Jinsung Yoon. 2026. PaperOrchestra: A Multi-Agent Framework for Automated AI Research Paper Writing. *arXiv preprint arXiv:2604.05018* (2026). doi:10.48550/arXiv.2604.05018
- [43] Ruofeng (wanshuiyin) Yang. 2025. ARIS (Auto-Research-In-Sleep): Lightweight Markdown-only skills for autonomous ML research. <https://github.com/wanshuiyin/ARIS>.
- [44] Matei Zaharia, Andrew Chen, Aaron Davidson, et al. 2018. MLflow: A Platform for ML Lifecycle Management. In *IEEE Data Engineering Bulletin*.