Dr. Claw: An AI Scientist Workspace for Vibe Research

Dingjie Song[1], **Hanrong Zhang**[2], **Dawei Liu**[3], **Yixin Liu**[1], **Zongxia Li**[4], **Zhengqing Yuan**[5], **Siqi Zhang**[1], **Henry Peng Zou**[2], **Zhiling Yan**[1], **Yuxuan Zhang**[6], **Yanfang Ye**[5], **Philip S. Yu**[2], **Lichao Sun**[1]

[1]Lehigh University [2]University of Illinois Chicago [3]University of Pennsylvania
[4]University of Maryland [5]University of Notre Dame [6]University of British Columbia

Abstract

Command-line coding agents (e.g., Claude Code, Gemini CLI) can already read and write files and sustain long sessions, yet end-to-end research still fragments across chat tools, IDEs, terminals, and writing environments, and the decisions that make it auditable are rarely preserved. We present **Dr. Claw**, an open-source workspace that wraps existing coding-agent executors in a *controllable* and *auditable* human-in-the-loop workflow rather than introducing another autonomous agent. Persistent state objects, a reusable skill library, and multi-executor coordination link human decisions to AI execution, turning planning, execution, and writing into one traceable, recoverable loop. We demonstrate Dr. Claw through an interactive three-view scenario and a failure-recovery walkthrough, and evaluate it against a bare command-line agent sharing the same backend executor, so the comparison contrasts the whole orchestration layer (task graph, state objects, and skill library) with the agent it wraps. Holding the executor fixed, Dr. Claw scores higher on research completeness while persisting an auditable, recoverable process trail. Demo access: repository `https://github.com/OpenLAIR/dr-claw`, released under AGPL-3.0 with GPL-3.0 upstream components.

1 Introduction

Large foundation models and agentic tools have improved the five core AI research operations—literature review, idea generation, code implementation, results analysis, and drafting (OpenAI, 2023; Brown et al., 2020; Yao et al., 2022; Schick et al., 2023; Lu et al., 2024; Yu et al., 2025; Yang and Weng, 2025; Schmidgall et al., 2025; Baek et al., 2025). Command-line coding agents such as Claude Code and Gemini CLI push this further, living in the terminal, reading and writing project files, and sustaining context across long sessions (Chen et al., 2021; Barke et al., 2022; Dakhel et al., 2022; Jimenez et al., 2024; Yang et al., 2024). Yet these agents optimize execution, not *control*: the plan, intermediate decisions, and artifacts that make a research process reviewable are scattered or lost, and the human has few explicit takeover points.

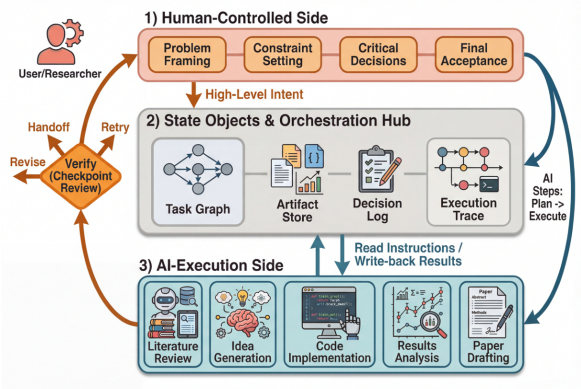


Figure 1: Control split in Dr. Claw: the Human-Controlled side sets goals, constraints, and acceptance decisions; the AI-Execution side runs the five core AI operations (Plan → Execute), connected through four state objects and checkpoint feedback (Verify/Revise/Retry/Handoff).

The bottleneck is now full-process orchestration rather than isolated capability: researchers still switch across tools for decomposition, scheduling, tracking, validation, and writing, which weakens reproducibility and delivery reliability. HCI evidence consistently shows that collaboration cost is dominated by verification and context maintenance, and that process visibility and interruptible control are critical (Gu et al., 2024; Kazemitabaar et al., 2024a; Xie et al., 2024; Kazemitabaar et al., 2024b; Flores-Saviaga et al., 2025); existing demos improve usability but leave cross-stage state continuity and artifact closed-loop management limited (Dibia et al., 2024; Cai et al., 2024).

We target a mode of work we call **Vibe Research**: a controllable, human-in-the-loop paradigm in which a researcher states high-level goals and constraints in natural language, AI com-

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 868–879
October 24-29, 2026 ©2026 Association for Computational Linguistics

piles them into an executable research loop and carries out the five core operations, and final acceptance rests on observable outputs, with humans governing direction and final decisions throughout. The paradigm is defined not by full autonomy but by an operational human–AI division of labor: AI handles high-throughput, parallelizable, templatable execution (retrieval, coding, running, summarizing, drafting), while humans own research direction, evaluation criteria, key trade-offs, and final acceptance. Unlike end-to-end autonomous research agents (Lu et al., 2024; Yamada et al., 2025; Tang et al., 2025; Schmidgall et al., 2025) or general multi-agent frameworks (Wu et al., 2023; Qian et al., 2024), we emphasize sustained human takeover and research-centric artifact management (Section 2). Crucially, Dr. Claw does not introduce yet another executor: it *wraps* an existing command-line coding agent, adding the state, control, and audit layer that such agents lack while reusing their execution capability.

We propose **Dr. Claw**, a one-stop workspace that unifies planning, execution, and writing into one controllable, traceable, recoverable, and auditable research loop (Figure 1). In each cycle, users provide goals, constraints, and acceptance criteria; the system decomposes tasks, executes actions, writes back artifacts, and supports revise/retry/handoff without losing process state. We evaluate this loop in Section 5, holding the backend executor fixed so that the comparison contrasts the orchestration layer as a whole with the bare executor, and complement it with a retrospective human study on efficiency, output quality, and integrated experience (Appendix A). Our contributions are as follows:

- We formalize *Vibe Research*, a controllable, human-in-the-loop research-orchestration paradigm that clarifies the boundary between AI execution and human decision responsibilities, distinguishing it from end-to-end autonomous approaches.
- We implement this paradigm in Dr. Claw, with a task-graph-centric orchestration, a chat-driven planner, a modular skill library (58 stage-mapped skills across five research stages, 171 in the deployed catalogue), and a multi-agent execution layer compatible with mainstream coding-agents.
- We provide a controlled pilot evaluation and a scenario-based demonstration. Holding the executor fixed, Dr. Claw scores higher than the bare agent on completeness by closing its research-hygiene gaps (consistent across tasks, though not statistically powered at one run per task), while uniquely persisting an auditable, recoverable process trail; a retrospective study additionally associates the integrated workflow with gains in efficiency, quality, and usability over non-integrated ones.

2 Related Work

2.1 Research Agents and End-to-End Automation

End-to-end systems automate the path from ideas to papers with minimal human intervention (Lu et al., 2024; Yamada et al., 2025; Schmidgall et al., 2025; Baek et al., 2025), but rarely prioritize controllability in sustained collaboration within real local engineering environments. A parallel line lowers the barrier to building or steering agents: no-/low-code authoring (Dibia et al., 2024; Cai et al., 2024; FlowiseAI, 2026) and general orchestration runtimes such as LangGraph (LangChain, 2026), which give developers durable checkpointing, human-in-the-loop interrupts, and replay over a state schema they declare themselves; TinyScientist (Yu et al., 2025), ResearStudio (Yang and Weng, 2025), and IRIS (Garikaparthi et al., 2025) add intervenable agents. Dr. Claw differs along three axes taken together: it (i) *wraps* an existing command-line coding agent rather than introducing a new executor; (ii) makes the research process a first-class object through four persistent state objects (Task Graph, Artifact Store, Decision Log, Execution Trace); and (iii) sustains human takeover across the full ideation→experiment→publication arc (Table 1). Each axis has precedent taken alone—TinyScientist also delegates to an external coding agent, AI Scientist-v2 persists a structured experiment tree, and ResearStudio allows a pause at any moment rather than at checkpoints, where it is stronger than Dr. Claw—but prior work optimizes end-to-end autonomy, agent *construction*, or steering within a single stage, whereas Dr. Claw optimizes *controllability and auditability of an existing agent* over a long-horizon workflow.

2.2 Human–AI Collaboration and Context-Switching Costs

Recent HCI research has shifted from code-generation utility to process controllability and ver-

System	*Execution*		*Orchestration*		*Interaction*		
	Wraps CLI agent	**Research skills**	**Built-in research state**	**In-place recovery**	**Mid-run takeover**	**End-user workspace**	**Unattended end-to-end**
End-to-end autonomous research systems							
AI Scientist v1/v2	◐	◐	●	◐	○	○	●
Agent Laboratory	○	◐	◐	◐	◐	○	●
ResearchAgent	○	○	◐	–	○	○	◐
Agent authoring tools and orchestration runtimes							
AutoGen Studio	○	○	○	○	◐	◐	●
Flowise (archived)	○	◐	○	●	◐	●	●
LangGraph	○	○	○	●	◐	○	●
Intervenable research agents							
TinyScientist	●	◐	◐	◐	○	●	●
ResearStudio	○	◐	◐	●	●	●	●
IRIS	○	○	◐	◐	◐	●	◐
Dr. Claw (ours)	●	●	●	●	●	●	●

Table 1: **Design-space comparison.** ● supported, ◐ partial, ○ absent, – out of scope by design. *Built-in research state*: research-artifact objects (task graph, artifact store, decision log, execution trace) shipped with the system rather than a schema the developer declares. *Mid-run takeover*: intervention at arbitrary points during a run, not only at stage boundaries or developer-placed pause nodes. Marks are assessed from published papers, official documentation, and public repositories as of August 2026, not from runs of these systems. Each dimension is realized, fully or in part, by prior systems; Dr. Claw differs by combining them.

ification burden. In data analysis and knowledge work, the key bottlenecks are cross-step interpretation, validation, and correction rather than one-off generation quality, and interactive decomposition and process visualization improve monitoring and intervention (Gu et al., 2024; Kazemitabaar et al., 2024a; Xie et al., 2024); in programming settings, CHI evidence shows the added reading, confirming, and revising of assistant outputs harms fluency, cognitive load, and confidence (Kazemitabaar et al., 2024b; Flores-Saviaga et al., 2025). These converge on a design requirement—clear supervision affordances, interpretable intermediate states, and timely takeover—that aligns with Dr. Claw's goal of reducing context-switching costs. Accordingly, we evaluate the systemic effect of workflow integration on efficiency, quality, and usability rather than intrinsic model gains.

3 System Overview

Dr. Claw is designed around one central question: how to let researchers complete problem definition, experiment progression, and paper production in a continuous workflow rather than switching among isolated tools. It models research as an explicitly traceable workflow that organizes human decisions with AI execution into a stable collaboration structure. Additional implementation, failure-recovery, and reproducibility details are in Appendix B.2, Appendix B.2, and Appendix B.2.

3.1 Design Goals and System Abstraction

Dr. Claw orchestrates four state objects: `Task Graph`, `Artifact Store`, `Decision Log`, and `Execution Trace`. Together they convert interaction history into a reviewable process supporting iterative orchestration rather than one-shot generation: users provide high-level goals/constraints, and the system maps them to executable tasks with continuous inspection and takeover. One task interaction is a state transition:

$$\text{State}_{t+1} = f(\text{State}_t, \text{Action}_t, \text{Obs}_t), \quad (1)$$

where State_t denotes the workflow state at time t (jointly formed by Task Graph, Artifact Store, Decision Log, and Execution Trace), Action_t is a system- or user-triggered action, and Obs_t is the observed feedback. This formulation highlights that Dr. Claw optimizes iterative state updates rather than single responses.

3.2 Three-Layer System Architecture

Dr. Claw uses three collaborative layers:

- **Interaction**: unified workspace for chat, task views, files, and version operations.
- **Orchestration**: state and lifecycle management from high-level intent to stage tasks.

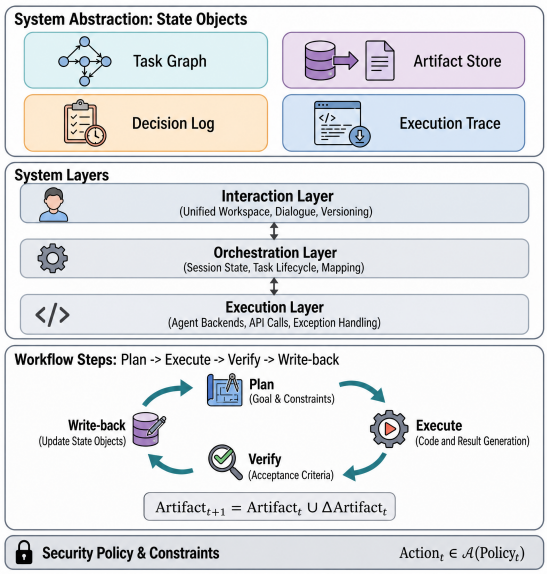


Figure 2: **Overall system view of Dr. Claw. Top**: four state objects (Task Graph, Artifact Store, Decision Log, Execution Trace). **Middle**: three system layers (Interaction, Orchestration, Execution). **Bottom**: iterative workflow steps (Plan–Execute–Verify–Write-back). **Lowest**: safety policy constraint ($\text{Action}_t \in \mathcal{A}(\text{Policy}_t)$).

- **Execution**: backend invocation, result return, and exception handling across heterogeneous executors.

This design enables backend switching without changing workflow semantics while preserving unified state and audit views.

3.3 Workflow-Centric Interaction Loop

Given a research idea, the system generates a structured brief and dependency-aware task plan, then runs workflow steps of Plan–Execute–Verify–Write-back. Execution outputs are written into `Artifact Store`, task states are updated in `Task Graph`, and all interventions are retained in `Decision Log`/`Execution Trace`. This workflow supports long-horizon iteration with explicit human checkpoints.

3.4 Skill-Based Capabilities and Multi-Executor Coordination

Dr. Claw provides a reusable skill library (58 stage-mapped skills across five research stages—survey, ideation, experiment, publication, promotion—and 171 skills in the deployed catalogue) covering ideation, literature processing, experimentation, analysis, and writing. Each skill is a directory with a `SKILL.md` manifest whose YAML frontmatter declares a name and description, and commonly a version, license, allowed tools, and stage/domain tags. Skills are versioned and schema-checked before activation, and reach a task by three routes: a stage-skill map resolves the task's stage and type to a set of suggested skills, which are attached to the task node and injected into its next-action prompt; keyword detection over user instructions and task text can auto-load a skill; and users may invoke any catalogue skill manually from the Skills dashboard. Authoring, validation, and transfer are detailed in Appendix B.2. Dr. Claw also coordinates multiple executors in one project context, so users switch execution strategy by task type, and on failure or constraint violation can retry, revise, or take over without breaking global state. Formalization of artifact updates is in Appendix B.

3.5 Safety and Controllability Mechanisms

Given risks from external calls and code execution, Dr. Claw treats permission management as a first-class mechanism, supporting fine-grained tool/command policies that distinguish secure defaults from trusted extended settings. Actions are allowed only if they belong to the action set defined by the current policy:

$$\text{Action}_t \in \mathcal{A}(\text{Policy}_t), \tag{2}$$

where Policy_t is the permission configuration at time t and $\mathcal{A}(\text{Policy}_t)$ is the executable action space, guaranteeing consistency between execution capability and safety boundaries.

4 Demo Scenario

We use a research task to illustrate Dr. Claw under human-in-the-loop conditions—whether high-level research intent can be stably transformed into executable workflows while preserving controllability, recoverability, and auditability—focusing on workflow orchestration quality rather than one-shot model output.

4.1 Main Interface Demonstration

The center interface is the primary orchestration view. It uses one unified research prompt with explicit goals, constraints, and acceptance criteria. The user acts as research lead (goal confirmation and key decisions), while Dr. Claw handles decomposition, dispatch, and state feedback. We monitor four state objects throughout the process: `Task Graph`, `Artifact Store`, `Decision Log`, and `Execution Trace`.

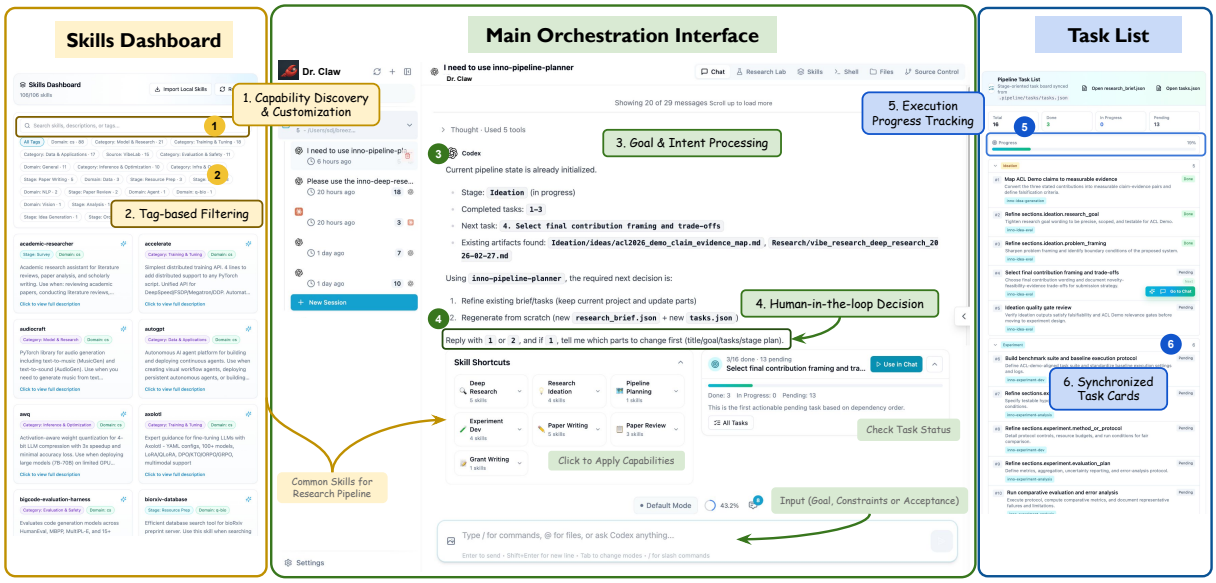


Figure 3: **Three-view scenario diagram**. **Left**: Skills Dashboard for capability discovery and filtering. **Center**: main orchestration interface for planning, execution feedback, and human approval. **Right**: Task List for progress tracking and next-step selection.

Figure 3 summarizes the resulting cross-view loop, in which discovered capabilities flow into goal/intent processing and explicit human decisions and are finally reflected as execution progress and synchronized task cards.

4.2 Skills Interface Demonstration

The left panel validates capability management during workflow execution, corresponding to annotations (1) and (2) in Figure 3. This sub-scenario contains three interactions:

- **Skills board browsing**: users browse available skills by research stage to quickly locate suitable capabilities.
- **Tag-based filtering**: users filter skills by theme (e.g., Ideation, Experiment, Publication) to shorten retrieval paths.
- **Manual skill addition**: users add new skills into the current project so they can be explicitly invoked in subsequent tasks.

This sub-scenario tests not skill count but whether users can perform *Capability Discovery & Customization* and *Tag-based Filtering* in context, then convert selected skills into executable steps.

4.3 Task List Interface Demonstration

The right panel (`Task List`) is the execution-control view, corresponding to annotations (5) and (6) in Figure 3. It presents synchronized task status—overall counts (`Total`, `Done`, `In Progress`, `Pending`), a progress bar, and stage-grouped *Synchronized Task Cards*—and supports three operations during execution: **(1) progress inspection** (assess stage completion and backlog); **(2) task-level traceability** (each card exposes task ID, objective, and linked skill tags); and **(3) direct action entry** (trigger the next step from pending items).

5 Evaluation

We evaluate Dr. Claw against the *bare command-line coding agent it wraps*. The question is not whether the wrapper runs faster, since an orchestration layer that records state necessarily does more work, but whether, at a bounded time cost, it leaves the delivered output no less complete while turning a flat pile of files into an auditable, recoverable trail. The operator-facing context-switch reduction the paper claims is measured by the retrospective three-condition human study (Appendix A), not by this automated comparison.

5.1 Research Completeness Under Open-Ended Goals

A fully enumerated instruction leaves little room for orchestration to add value: when every requirement is spelled out in the prompt, a capable backend executor simply reads them off. We therefore evaluate Dr. Claw against the bare command-line agent it wraps in the regime where a research assistant should matter—an *open-ended* goal, where best practices must be supplied rather than tran-

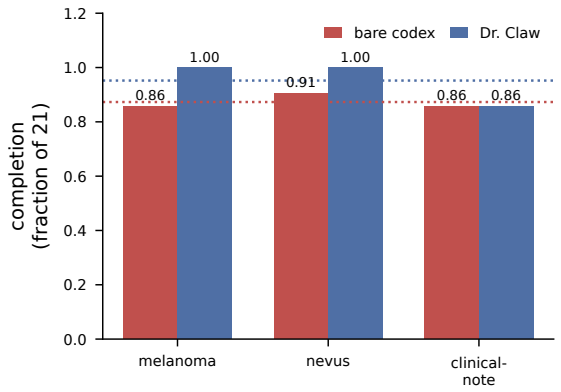


Figure 4: Open-ended completion per task (fraction of 21 elements); dotted lines are pooled means.

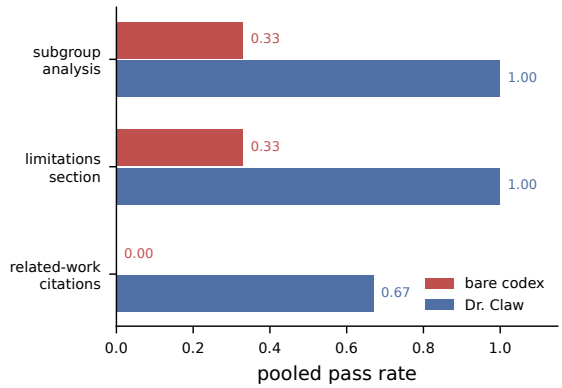


Figure 5: Pooled pass rate on the three research-hygiene elements where the conditions diverge.

scribed. We hold the backend executor fixed (the `codex` provider with model `gpt-5.4` under a matched `danger-full-access`/approval-never profile), so bare `codex` and `drclaw` differ only by Dr. Claw's task graph, artifact store, decision log, execution trace, and skill library. These arrive as one bundle: the comparison measures what the orchestration layer adds *as a whole*, and cannot attribute the difference to any single component of it. Each task gives an identical, unenumerated instruction ("conduct a rigorous, publication-quality study…") across three medical problems: melanoma and nevus classification on Derm7pt (Kawahara et al., 2019), and a clinical-note risk baseline. Neither prompt is told the rubric. *Completeness* is the fraction of **21** research best-practice elements spontaneously included—code and reproducibility, multiple models, cross-validation, calibration, ablation, statistical rigor, figures, a write-up citing its own numbers, plus research-hygiene elements (a limitations section, subgroup analysis, real related-work citations)—scored deterministically against the produced files, with no model-in-the-loop judgment. In the Dr. Claw condition the task graph was verified active (mean 17 tracked tasks; the executor read ∼12 skill files per run).

What the open-ended test shows. Dr. Claw wins two of three tasks and ties the third, pooling 0.952 against the bare agent's 0.873 (Figure 4). The advantage is *not* in modeling: both conditions train three-plus models with cross-validation, calibration, ablation, and statistical rigor—those elements pass at 1.00 on both sides. It lives entirely in research hygiene (Figure 5): Dr. Claw's reference-audit, analysis, and paper-writing skills reliably add a limitations section (0.33→1.00 pooled pass rate), subgroup analysis (0.33→1.00), and real literature citations (0.00→0.67, the bare agent producing *zero* across the three tasks). On the one tie (the clinical-note task), Dr. Claw's reference audit did not fire, so it too missed citations—the mechanism, when engaged, is exactly what closes the gap.

Triggering reliability. Skill *selection* is deterministic given a task's stage and type, but skill *invocation* is not enforced: the resolver can only place a suggestion in the task prompt. Across the three runs the executor read ∼12 skill files per run, yet the one non-firing reference audit above accounts for the single task on which Dr. Claw failed to beat the bare agent. Suggestion is guaranteed; invocation is best-effort, and closing that gap—by verifying skill execution rather than recommending it—is the clearest reliability improvement the current design admits.

Auditability is an architectural affordance, not a score. The conditions also differ in whether a completed run can be *re-traced* (Figure 6). Every Dr. Claw run persists a **queryable task graph** (mean 14 nodes), a timestamped execution trace (mean 14 transitions), a decision-log brief (mean 10 entries), and named research stages with explicit claim→evidence maps; **the bare agent persists none**. We read this as a **design affordance for human oversight, not a performance score**: these objects are Dr. Claw's own file format, so "bare = absent" holds by construction. On *format-neutral* traceability **we claim no superiority**—write-up file references resolve at 100% versus 62%, but this is volume-confounded (48 vs 8 references) and non-decisive at this n. Both results are an exploratory pilot: with one replicate per task the pooled completion $\Delta = +0.079$ has a 95% bootstrap CI of [-0.00, +0.14] that includes zero, so the direction is consistent (Dr. Claw ≥ bare on all three tasks) but

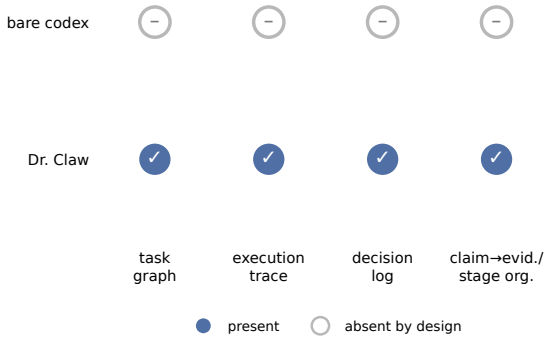


Figure 6: Whether a run persists a queryable process model (present/absent by design, not scored).

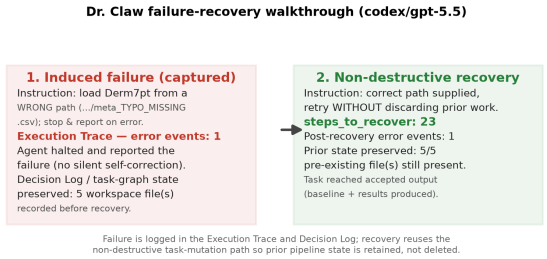


Figure 7: Captured failure and non-destructive recovery inside a project. Left: the induced wrong-path error, captured in the execution trace. Right: the in-place fix and recovered result, with all prior files retained.

not yet significant, and Dr. Claw remains slower—a bounded overhead the orchestration layer incurs by recording state.

5.2 Non-Destructive Failure Recovery Under Audit

We run one coherent Derm7pt mini-project through Dr. Claw and recover it from an induced failure, on the same backend executor (`codex/gpt-5.4`). The advantage on display is not speed or accuracy but the auditable, structured artifact trail the orchestration layer maintains: when a step fails, the captured execution trace and preserved prior state let the project recover in place rather than restart.

Failure and recovery. The same project also shows what happens when a step fails (Figure 7). Halting on an induced wrong-path error rather than silently self-correcting, then recovering to a real result (accuracy 0.892) with all 5 pre-existing files retained (23 tool events), the workspace *adds* the corrected artifacts rather than wiping the failed attempt—the prior pipeline state survives the fix. We report this as a single-condition design demonstration, without a matched bare-agent recovery run.

5.3 Human Study

To measure the operator-facing effect that the automated comparison cannot, we retain a retrospective study of seven AI PhD researchers across three research stages (Ideation, Experiment, Publication), comparing Dr. Claw against working with no AI tools and with general-purpose web/desktop AI assistants (e.g., ChatGPT, Gemini, Claude) on completion time, blind-rated output quality, tool-switching count, and self-reported experience. Under this hybrid, exploratory protocol, Dr. Claw is associated with **shorter completion-time bands, the highest output-quality ratings, and fewer tool switches with higher experience scores**; the effect is **strongest and fully pairwise-significant for experience**. Full setup, figures, and stage-wise statistics are in Appendix A.

6 Conclusion

We presented Dr. Claw, an integrated system for end-to-end AI research that unifies state-object management and skill-based execution in one workspace to reduce cross-tool orchestration costs and improve workflow continuity. Evaluated with the same backend executor run inside versus outside Dr. Claw, which compares the whole orchestration layer against the agent it wraps rather than ablating its parts, the layer preserves the measured completeness of the output (a count of which research components are present, not a correctness check) while producing a more auditable, better-structured artifact trail, as shown through a persisted-process-model analysis and a non-destructive failure-recovery walkthrough; a retrospective human study over three stages provides complementary evidence on time, output quality, and integrated experience.

Limitations

Our evaluation is a small-scale, exploratory demonstration rather than a powered comparative study: the pilot uses a limited number of tasks and participants, so the reported differences are directional evidence about workflow orchestration, not causal or statistically powered effects. Three limits deserve to be stated plainly. First, holding the backend executor fixed is not an ablation: Dr. Claw adds a task graph, persistent state objects, a skill library, and workflow instructions as one bundle, so the observed gap cannot be attributed to any single component, and a skill-only versus orchestration-

only ablation remains the natural next experiment. Second, completeness counts how many of 21 expected research components a run produces; it is a coverage measure, not a correctness check, and establishing scientific soundness would require expert per-artifact review. Third, our comparison target is the bare executor that Dr. Claw wraps: the matched control for what the wrapper adds, but not a state-of-the-art orchestration framework. Other confounds remain, such as prior familiarity with either interface, and the operator-facing context-switch and intervention reductions are measured separately in the human study (Appendix A). Our tasks also come from a single domain (medical), so transfer of the skill library and the structured Task Graph abstraction to other research areas, particularly open-ended work where rigid structure may add friction, remains to be shown. We do not claim model-level innovation; our contribution is workflow integration, and observed gains depend on configuration choices (backend model, permission settings, and skill coverage).

Acknowledgments

This work was partially supported by the National Science Foundation Grants CRII-2246067, ATD-2427915, NSF POSE-2346158, and NSF POSE-2449280.

Ethics Statement

Dr. Claw is designed to *assist* research under sustained human control, not to autonomize it. A recurring concern with AI research systems is that they may flood the literature with unverified or low-quality output. Our design responds to this concern directly rather than amplifying it: every stage passes through explicit human checkpoints, final acceptance rests with the researcher, and the Decision Log and Execution Trace keep a complete, auditable record of what was generated, approved, revised, or rejected. We view this human-in-the-loop, fully-traceable structure as a safeguard for verifiability, not a shortcut around it. Critical content (citations, experimental conclusions, and manuscript claims) requires human verification before use. When sensitive data are involved, users should follow least-privilege permission settings and retain operation traces; in our own study, the medical datasets remain on the authors' server and are not redistributed, and no patient-level data are released. For high-risk domains such as healthcare, system outputs must not be used directly for real-world clinical decisions. Human participation in the user study was voluntary and based on informed consent. We used AI-based coding assistants as part of the system under study and for writing assistance, consistent with venue policy.

References

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. 2025. ResearchAgent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6709–6738, Albuquerque, New Mexico. Association for Computational Linguistics.

Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2022. Grounded copilot: How programmers interact with code-generating models. *arXiv preprint arXiv:2206.15000*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Yuzhe Cai, Shaoguang Mao, Wenshan Wu, Zehua Wang, Yaobo Liang, Tao Ge, Chenfei Wu, Wang You, Ting Song, Yan Xia, Nan Duan, and Furu Wei. 2024. Low-code LLM: Graphical user interface over large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pages 12–25, Mexico City, Mexico. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 29 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Arghavan Moradi Dakhel, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, Michel C. Desmarais, and Zhen Ming Jiang. 2022. GitHub copilot ai pair programmer: Asset or liability? *arXiv preprint arXiv:2206.15331*.

Victor Dibia, Jingya Chen, Gagan Bansal, Suff Syed, Adam Fourney, Erkang Zhu, Chi Wang, and Saleema

Amershi. 2024. AUTOGEN STUDIO: A no-code developer tool for building and debugging multi-agent systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 72–79.

Claudia Flores-Saviaga, Benjamin V. Hanrahan, Kashif Imteyaz, Steven Clarke, and Saiph Savage. 2025. The impact of generative AI coding assistants on developers who are visually impaired. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, New York, NY, USA. Association for Computing Machinery.

FlowiseAI. 2026. Flowise: Build AI agents, visually. `https://github.com/FlowiseAI/Flowise`. Open-source project; repository archived 13 August 2026.

Aniketh Garikaparthi, Manasi Patwardhan, Lovekesh Vig, and Arman Cohan. 2025. IRIS: Interactive research ideation system for accelerating scientific discovery. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 592–603.

Ken Gu, Ruoxi Shang, Tim Althoff, Chenglong Wang, and Steven Mark Drucker. 2024. How do analysts understand and verify ai-assisted data analyses? In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, New York, NY, USA. Association for Computing Machinery.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*.

Jeremy Kawahara, Sara Daneshvar, Giuseppe Argenziano, and Ghassan Hamarneh. 2019. Seven-point checklist and skin lesion classification using multitask multimodal neural nets. *IEEE Journal of Biomedical and Health Informatics*, 23(2):538–546.

Majeed Kazemitabaar, Jack Williams, Ian Drosos, Tovi Grossman, Austin Henley, Carina Negreanu, and Advait Sarkar. 2024a. Improving steering and verification in ai-assisted data analysis with interactive task decomposition. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA. Association for Computing Machinery.

Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Z. Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024b. CodeAid: Evaluating a classroom deployment of an LLM-based programming assistant that balances student and educator needs. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, New York, NY, USA. Association for Computing Machinery.

LangChain. 2026. LangGraph: A low-level orchestration framework for stateful agents. `https://docs.langchain.com/oss/python/langgraph/overview`. Software documentation; accessed 30 August 2026.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5977–6043, Suzhou, China. Association for Computational Linguistics.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. 2025. AI-researcher: Autonomous scientific innovation. *arXiv preprint arXiv:2505.18705*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv preprint*. ArXiv:2308.08155 [cs].

Liwenhan Xie, Chengbo Zheng, Haijun Xia, Huamin Qu, and Chen Zhu-Tian. 2024. Waitgpt: Monitoring and steering conversational llm agent in data analysis with on-the-fly code visualization. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA. Association for Computing Machinery.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*.

Linyi Yang and Yixuan Weng. 2025. Researstudio: A human-intervenable framework for building controllable deep-research agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 896–905.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Haofei Yu, Keyang Xuan, Fenghai Li, Kunlun Zhu, Zijie Lei, Jiaxun Zhang, Ziheng Qi, Kyle Richardson, and Jiaxuan You. 2025. TINYSCIENTIST: An interactive, extensible, and controllable framework for building research agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 558–590.

A Human-Study Evaluation (Retrospective Three-Condition Study)

As complementary evidence to the automated evaluation in Section 5, we retain the retrospective three-condition human study from the prior submission. It compares no AI tools (**No-AI**), general-purpose web/desktop AI assistants (e.g., ChatGPT, Gemini, Claude; **Web/Desktop-AI**), and **Dr. Claw** over three research stages (Ideation, Experiment, Publication), on four metrics: completion-time bands, stage output score (blind rating, 1–5), switching-count bands, and experience score. The study uses a hybrid design—live logs for Dr. Claw and retrospective reports for the two controls—so all statistics are exploratory rather than causal.

Results. Across all three stages, Dr. Claw is associated with shorter completion-time bands (Figure 8a; mainly <1h and 1h–1d), the highest stage output scores (Figure 8c; Dr. Claw > Web/Desktop-AI > No-AI), and lower switching bands with higher experience scores (Figure 8b). The omnibus signal is strongest and fully pairwise-significant for Experience, with Time Band also significant in Stages 2–3.

A.1 Participants and Data Collection

We recruited seven AI PhD participants (subfields: high-performance AI, medical AI, and large language models), all with stable research and paper-writing experience and informed consent. To reduce unfamiliarity bias, each was given a three-week free-use period for Dr. Claw before formal comparison. Under the hybrid design, C3 (Dr. Claw) participants performed tasks under a unified task framework with live logs, while C1 (No-AI) and C2 (Web/Desktop-AI) were reported retrospectively (time, switching-count, and experience ranges) using the same stage definitions. Stage outputs in all conditions were scored by blind raters with the same stage-specific rubrics; to reduce recall bias, we required recallable tasks within the most recent month and a unified timing protocol.

A.2 Stage Definitions and Rubrics

We divide the workflow into three research stages: **Stage 1 (Ideation)** idea generation and problem framing; **Stage 2 (Experiment)** experiment setup, data processing, and analysis (**excluding model runtime**); and **Stage 3 (Publication)** drafting and final polishing. Stage output is rated on a 1–5 scale by blind raters using stage-specific rubrics: Stage 1 on novelty, feasibility, literature coverage, and clarity of problem definition; Stage 2 on reasonableness of setup, correctness of analysis, and quality of interpretation; Stage 3 on structural completeness, technical accuracy, readability, and reproducibility information.

Logging and statistics. Logs are organized at participant–system–stage granularity: Dr. Claw live logs record start/end time, stage duration, and switching count, while retrospective questionnaire values for the controls are normalized to the same metric definitions. Because retrospective data are included, statistics are exploratory: we apply Friedman tests for overall comparisons and Holm-corrected pairwise Wilcoxon tests post-hoc.

A.3 Detailed Statistical Results

Stage-wise Friedman tests use $n = 7$ matched participants; each entry reports $\chi^2(2), p$ per stage (S1/S2/S3), followed by Holm-corrected pairwise p versus Dr. Claw for No-AI and Web/Desktop-AI (each as {S1,S2,S3}).

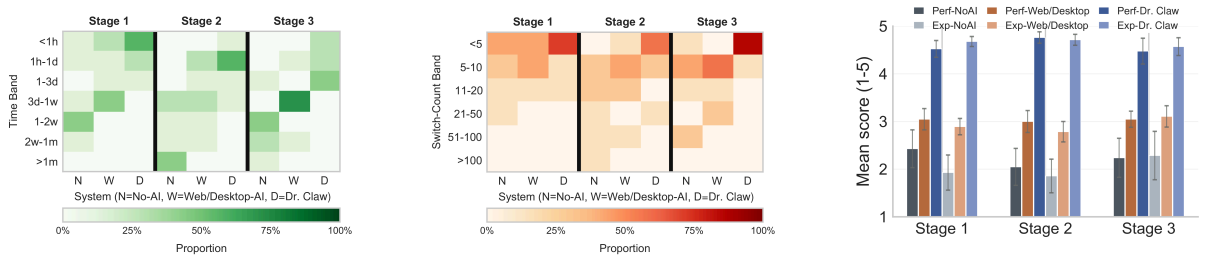
- **Time Band**: $8.96, 0.0114$ / $13.56, 0.0011$ / $14.00, <0.001$. Pairwise: No-AI {0.1250,0.0469,0.0469}, Web/Desktop-AI {0.1250,0.0469,0.0469}.
- **Switching Band**: $1.50, 0.4724$ / $11.57, 0.0031$ / $10.75, 0.0046$. Pairwise: No-AI {1.0000,0.0938,0.0625}, Web/Desktop-AI {1.0000,0.0938,0.0469}.
- **Performance**: $9.25, 0.0098$ / $11.57, 0.0031$ / $11.31, 0.0035$. Pairwise: No-AI {0.0938,0.0938,0.0469}, Web/Desktop-AI {0.1250,0.0938,0.0625}.
- **Experience**: $12.29, 0.0021$ / $13.56, 0.0011$ / $12.07, 0.0024$. Pairwise: both controls {0.0469,0.0469,0.0469}.

Key validity threats: limited sample size, system-familiarity differences, recall bias in retrospective controls, exclusion of model runtime in Stage 2, and subjectivity in experience scores.

B System Overview Details

B.1 Formal Model

The write-back step is $\text{Artifact}_{t+1} = \text{Artifact}_t \cup \Delta\text{Artifact}_t$, where $\Delta\text{Artifact}_t$ is the set of newly



(a) Time-band distributions (h/d/w/m: hour/day/week/month).

(b) Switching-count band distributions.

(c) Mean performance (Perf) and experience (Exp); error bars: SE.

Figure 8: Human-study results across three systems and three stages. Heatmap intensity denotes percentage within each system-stage cell.

added or revised artifacts in one iteration; task-node statuses update along dependencies (pending → running → done) with full histories retained in `Execution Trace`. Dr. Claw's target follows: lowering orchestration overhead while preserving output quality and controllability.

B.2 Implementation, Recovery, and Reproducibility

Implementation. Project initialization creates a fixed stage-folder layout for Ideation, Experiment, and Publication, plus a persistent pipeline-state store holding configuration, the research brief, and the task list. Task nodes store normalized fields (ID, status, priority, dependencies, stage, type, required inputs, suggested skills, next-action prompt); the server resolves status aliases and selects the next task by dependency completion, reading stage-specific skill recommendations from a stage-skill map. **Skill lifecycle.** The catalogue holds 171 skills, 87 of them top-level entries with their own `SKILL.md`; three in-house families supply most (`aris-*`, 44; `inno-*`, 16; `ds-*`, 14), the rest imported from public collections. *Authoring* is file-based: a skill is a directory whose frontmatter carries `name` and `description`, commonly `version`, `license`, `allowed-tools`, and `argument-hint`, plus optional `stage`/`domain` keys feeding the dashboard tag index. *Validation* parses the frontmatter, rejects a `SKILL.md` lacking a `name`, applies pre-flight schema and dependency checks, and version-stamps activated skills for replay and audit. *Selection* runs by stage-map resolution, keyword auto-load, or manual invocation: the resolver unions a stage's base skills with those for the task's type and writes them to the task node, reaching 58 skills across five stages (survey 11, ideation 14, experiment 18, publication 22, promotion 3); 30 top-level skills are not yet stage-mapped. *Transfer* to a new domain edits one JSON map rather than code. Dr. Claw uses backend adapters for the Claude and Codex SDKs (with Cursor hooks) and enforces action constraints via explicit policy settings (allowed/disallowed tools, permission mode, sandbox/approval). Pipeline mutations are written to persistent state first, then broadcast over WebSocket, keeping the interface synchronized to the same source of truth.

Failure handling and recovery operate at three levels: *pipeline/file* (missing paths, unreadable files, and JSON parse errors return explicit 4xx/5xx responses; initialization recreates pipeline-state defaults), *permission* (allow/deny checks with explicit denial reasons and bounded approval timeouts), and *session* (abort-supported execution with structured error events and consistent lifecycle states). Recovery relies on non-destructive task mutation APIs—update status, revise content, append tasks, continue from pending/in-progress nodes—supporting *revise/retry/handoff* recovery without deleting prior states.

Reproducibility. Setup requires Node.js LTS (v22 recommended), a standard install-and-run command sequence, and environment configuration. Each run should archive the instance metadata and pipeline-state files together with stage artifacts from Ideation, Experiment, and Publication. A minimum replication checklist: fix the same repository revision, lockfile, and runtime versions (Node, backend SDK/model); keep the same permission profile, three-stage task definitions with Stage-2 model-runtime exclusion, and time-/switching-band discretization; preserve blinded rubrics and rater instructions; and export raw pipeline state and run-time logs as supplementary material.