zorp: A Human-Checkpointed Research Agent with a Tamper-Evident Evidence Record

Aviskaar

August 2026

Abstract

Answers are cheap; evidence is not. A large language model will produce a fluent answer to a hard question in seconds, but it will not tell you whether to believe it, what evidence it weighed, or what it found that pointed the other way. Recent autonomous-research systems address the wrong half of this problem: they treat the finished paper as the artifact the system produces end to end, an assumption that does not survive contact with how research and technical decisions actually get published or adopted, since work with no human author of record is rejected outright at most venues and distrusted inside most organizations. We present zorp, a research agent built on the opposite premise. zorp turns a question into a pre-registered investigation, an evidence record, and a draft in which every claim resolves to a row in that record. It decomposes the work into four capabilities, validate, investigate, co-write, and deliver, each usable on its own and chained only through explicit human checkpoints, over a shared foundation that commits the hypothesis, metric, and falsification threshold to git before any evidence is gathered, and hash-verifies them on every load. We describe the architecture, the six-table evidence record, and the design decisions that follow from treating the human as the author of record rather than a downstream reviewer. zorp is implemented in Rust as a single multi-subcommand binary. This is a design and status report, not a benchmark study: all four capabilities are built and tested (538 passing tests over 24,965 lines as of August 2026), and we are explicit about what is not yet claimed, chiefly any comparative evaluation of output quality.

1 Introduction

An uncertain question worked through to a defensible answer has the same shape regardless of domain. Whether the question is "should we migrate off Kafka," a competitive teardown, an investment thesis, a due-diligence package, or an academic hypothesis, the work is the same: state the question, gather evidence, reason about evidence that conflicts, produce an answer or an artifact, and be able to show how you got there. What distinguishes a defensible answer from a merely fluent one is not the prose. It is the record behind it.

Large language models are very good at the prose and structurally indifferent to the record. They produce confident output whether or not the underlying evidence supports it [4], and the techniques that improved their reasoning, chain-of-thought prompting [14] and interleaved reasoning with tool use [16], improved the process without making its evidence inspectable afterward. An agent that reasons well and leaves no auditable trail has moved the problem rather than solved it.

The most ambitious response to this has been to automate the entire research loop. The AI Scientist [8] and its successor [15] generate hypotheses, run experiments, and author manuscripts end to end, with an automated review step appended afterward; the latter produced a manuscript that passed peer review at a workshop. This is a genuine result, and it also encodes an assumption worth separating from it: that the deliverable is a document the system produces, and that human involvement is a review that happens after.

That assumption does not survive contact with how this work is consumed. A paper an AI wrote end to end is not submittable at most venues. A technical recommendation nobody on the team can vouch for is not actionable, however sound the reasoning behind it was. In both cases the blocker is not output quality; it is that authorship and accountability cannot be retrofitted onto a finished artifact. Treating document generation as the finish line optimizes the step that was never the bottleneck.

zorp starts from the other end. The human is the author of record for whatever gets produced, and the system's job is to make that person's evidence trail complete, inspectable, and hard to revise after the fact. Three commitments follow, and this paper is largely about their consequences:

1. **The record is the product.** Every attempt is stored, including the ones that failed and the ones that contradicted each other, as typed rows rather than narrative logs, so a claim in a draft resolves to a measurement rather than to a paragraph the agent wrote about itself.
2. **Falsification is committed in advance.** The hypothesis, the metric, and a numeric threshold that would refute it are written to a git-committed file before any evidence is gathered, and hash-verified on every load, so the bar cannot be quietly lowered after results are seen.
3. **Capabilities are standalone and checkpointed.** The four capabilities each run alone and compose only through explicit human decision points, because most real use is partial rather than a full loop.

Contributions. We contribute (i) a decomposition of automated investigation into four independently usable capabilities joined by human checkpoints, rather than a single autonomous pipeline; (ii) a tamper-evident evidence record that applies the discipline of pre-registration [11] to agent runs, with git-backed commitment and hash verification as an enforced precondition rather than a convention; and (iii) an implementation of both in a single Rust binary, together with an honest account of what is built, what is tested, and what remains unevaluated.

We state the scope plainly. This is a systems and design paper. We report what the mechanisms do and that they are tested, and we do not report a comparative evaluation of investigation quality against the AI Scientist [15] or any other system. Section 9 says what such an evaluation would require and why we have not run one.

2 Related Work

Autonomous research systems. The AI Scientist [8, 15] is the closest point of comparison and the clearest statement of the position we argue against: the manuscript is an artifact the system produces, with review bolted on afterward. Aviskaar's internal lab-engine/Catalyst pipeline occupies similar ground for an internal audience. zorp borrows one piece of Catalyst's discipline directly, its `prereg.md` convention, in which a git-committed, human-readable pre-registration file gives a commit timestamp that cannot be moved after results are seen. It does not adopt Catalyst's hard experiment budget (150 lines of code, 10 minutes, no GPU), which is well-tuned to that system's narrow validation-experiment use case but arbitrary for questions zorp targets; zorp ships guidance rather than enforcement.

Pre-registration and research integrity. The commitments zorp enforces mechanically are borrowed from a body of work on why results fail to replicate. Undisclosed flexibility in analysis lets a researcher present almost anything as significant [13]; pre-registration and registered reports were introduced to constrain that flexibility by fixing the hypothesis and the analysis in advance [3, 11, 10]. That literature concerns human researchers and social incentives. Our observation is that an agentic system has the same degree of freedom and none of the social friction: nothing stops a run from adjusting what counted as success once results are visible, and no reviewer is watching. zorp therefore treats pre-registration as a precondition the system checks rather than a norm it encourages.

Agent harnesses and their evaluation. zorp's execution layer is a fork of quecto [6], a minimal vendor-neutral harness, extended rather than depended upon as a crate. Tool access is via the Model Context Protocol [1], which lets the same capability run against different evidence sources without changes. Benchmarks such as SWE-bench [5] have made agent evaluation concrete for software tasks by fixing a task set with a mechanical success criterion. No comparable fixture exists for open-ended investigation, where the interesting property is whether the evidence supports the conclusion, which is why the evaluation we owe (Section 9) is harder to construct than a pass rate.

Where zorp diverges. Across these systems the common thread is where the human sits. Prior autonomous-research work places the human after the fact, reviewing an output. zorp places the human inside the loop at explicit checkpoints, and as the author of record for both capabilities that produce human-facing artifacts.

3 Design Principles

Three principles determine most of the design, and each rules out an option that would otherwise be natural.

The record is the product, not the prose. Because a claim must resolve to a measurement, metrics are stored as typed key-value rows rather than as narrative logs. A narrative log is easy to write and useless as evidence: it cannot be checked mechanically, and it lets an agent describe its own results in terms it chose. Typed rows make the co-write step a lookup instead of a paraphrase.

Every attempt is retained. The record keeps failed and mutually contradictory attempts alongside successful ones. Keeping only the attempt that worked produces a highlight reel, which is exactly the selection effect pre-registration exists to prevent [13].

No silent defaults at decision points. A checkpoint reached with no human available and no explicit approval flag is an error, not a skipped step and not an implicit yes. Unlike a single tool call, a research checkpoint has no safe default: proceeding and halting are both wrong in ways the system cannot distinguish. Making it an error means unattended runs must opt in explicitly and visibly.

4 System Architecture

zorp is a single Rust [9] workspace of five crates. The core crate holds model transport and raw primitives. `zorp-agent` is the agent proper: tools, sandboxing, trust levels, verification, sessions, and MCP integration, compiled to one binary. `zorp-mcp` implements the Model Context Protocol client and server integration [1]. `zorp-eval` is a deterministic evaluation harness. `zorp-track`, described in Section 5, is the evidence foundation, and is the only crate written specifically for this work rather than inherited from the upstream harness.

One binary, not one pipeline. All four capabilities are subcommands of `zorp-agent` rather than a separate research binary that shells out to it. Parallel investigation workers are additional copies of

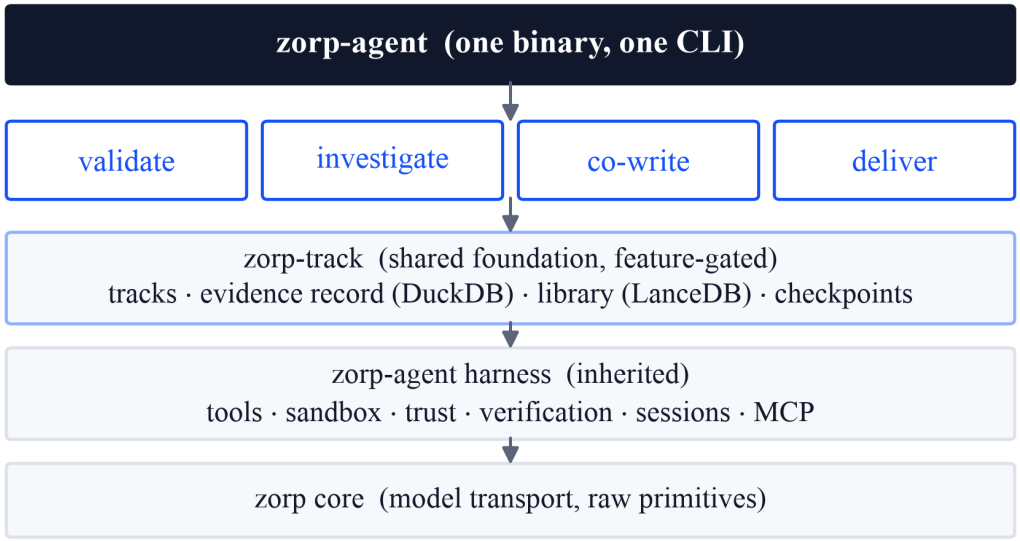

Figure 1: Layered architecture. The four capabilities are subcommands of one binary rather than separate programs, and sit on a shared evidence foundation that has no knowledge of any individual capability.

the same binary run as isolated subprocesses, not a second program with its own lifecycle. This keeps one artifact to install, learn, and support.

The foundation does not know its consumers. `zorp-track` owns the track model, the evidence record, pre-registration, and the checkpoint primitive, and knows nothing about validate, investigate, co-write, or deliver. They are built on it, not into it, which keeps the tamper-evidence machinery testable independently of any capability's logic.

Heavy dependencies are opt-in. `zorp-track` bundles DuckDB [12] and provisions LanceDB [7], which pulls in Arrow and DataFusion. Compiling these from a cold cache takes 20 to 30 minutes, so `zorp-agent` depends on `zorp-track` behind an optional `research` feature, matching the existing pattern for MCP and OpenTelemetry support. A user who wants only the base agent never pays that cost.

5 The Evidence Record

5.1 Two stores, split by question

DuckDB [12] holds the transactional and analytical record as six tables: `tracks` (one investigation, its question and status), `preregistrations` (the committed hypothesis, metric, and threshold, with the hash that makes them tamper-evident), `experiments` (every attempt, including those that conflicted and the one that ended the run), `metrics` (typed measurements), `checkpoints` (each human decision and when it was made), and `validations` (redundancy and feasibility scores, each with the citation that justified it).

A metric is a `(key, value_type, value)` tuple where `value_type` is `number`, `string`, or `bool` and the value occupies the matching typed column. This is the concrete form of the first design

principle: when co-write drafts a claim such as "p99 latency fell by 40 percent," that claim resolves to a row in `metrics`, not to a sentence the agent wrote about its own results. It is also what makes the finished artifact checkable by someone who was not present for the work.

LanceDB [7] is provisioned for multimodal, semantically searchable content, keyed by track so a search can be scoped to one investigation. As of this writing the store exists and is reachable but has no producers or consumers; what goes into it is each capability's own concern and is not yet implemented.

5.2 The kill threshold

Before zorp gathers anything, it commits the hypothesis, the metric, and a **kill threshold**, a number supplied by a human stating in advance what would refute the hypothesis, to a human-readable `prereg.md` file in git. The agent never proposes this number; a human does, and only a human can change it.

The choice of a git-committed file rather than only a database row is the mechanism, not a stylistic preference. A commit timestamp cannot be moved quietly after results are seen, which is precisely the guarantee a mutable row cannot offer. The DuckDB and LanceDB stores are gitignored, regenerable indexes over these files rather than the source of truth: if either is lost or corrupted, `zorp-track` rebuilds it by re-reading every `prereg.md`.

On every track load, `zorp-track` re-hashes each `prereg.md` (SHA-256 over raw bytes) and compares against the hash recorded at commit time. A missing file, a missing row, or content that no longer matches its recorded hash is a hard error that refuses the run, not a warning. This is what makes tamper evidence real rather than decorative: a threshold edited after results are visible, without a new commit, is caught on the next load.

5.3 Checkpoints

A `Checkpoint` gates progress at track granularity, mirroring the existing per-tool-call approval mechanism at a coarser grain. Two modes exist: `Interactive`, the default, which blocks and prompts; and `AutoApprove`, an explicit opt-in for unattended runs. Per the third design principle there is no third mode. Each checkpoint records what the human was shown and what they decided, so a track's history captures not only what was tried but what was asked and how it was answered.

6 The Four Capabilities

Each capability is a standalone subcommand. A full loop is simply all four run in sequence with a checkpoint between each; it is not a separate mode. This follows from observed use: validating a single question, or running one investigation, without committing to the whole loop is at least as common as wanting all four, and routing everything through one pipeline would serve the rarer case at the expense of the common one.

validate searches for evidence through whatever MCP tools are configured and scores redundancy and feasibility, each score requiring a citation. If no search-capable tool is configured it fails immediately rather than scoring anything, on the principle that a feasibility judgment with no evidence behind it is worse than no judgment.

investigate runs one pre-registered attempt per invocation against the committed metric and threshold. Every attempt enters the record, and a checkpoint decides whether to continue or kill the track.

co-write drafts from the record alone. It is handed validate's verdict, if one exists, and every metric

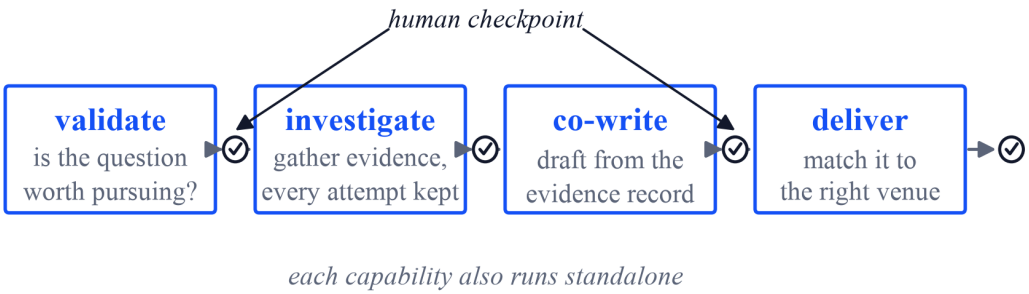

Figure 2: The four capabilities and the checkpoints between them. Each capability also runs alone. validate and deliver additionally require a tool to be configured before they will run at all.

investigate stored, as structured data, with instructions to cite only those figures, and it writes to `draft.md`. The human remains the author of record: co-write drafts and does not finalize. Rejecting its checkpoint does not kill the track, since a draft needing another pass is not a failed investigation.

deliver matches the finished draft against real venues. Scoped to academic venue-matching in this version, it requires a configured venue-database tool, gated exactly as validate's search requirement is, and writes a ranked shortlist for human review.

Two of the four, validate and deliver, have a hard external dependency on a configured tool. investigate and co-write do not, since they work entirely from the track's own record.

7 Implementation and Status

All four capabilities and the foundation beneath them are built and tested. Table 1 and Figure 3 report per-crate test counts at default features, which sum to the 538 that `cargo test --workspace` reports directly, alongside the research-feature run that exercises the four capabilities including integration tests that stand up a stub MCP server over stdio to verify tool-gating and round-trip behavior.

Table 1: Test and line counts at commit `fd07e81`, 2026-08-13. *A separate invocation covering the four capabilities and their integration tests. It is not additive with the 538 above, since the two runs share most of `zorp-agent`'s default suite.

Crate / feature set	Passing tests	Lines of Rust
`zorp` (core)	24	460
`zorp-mcp`	23	887
`zorp-eval`	41	2,095
`zorp-track`	69	2,623
`zorp-agent` (default)	381	18,602
top-level integration tests	n/a	298
Total (`cargo test --workspace`)	**538**	**24,965**
`zorp-agent` (`--features research`)	445*	n/a

The tests cover the failure modes the design turns on, not only the success paths: a `prereg.md` whose hash no longer matches, a checkpoint reached with no terminal and no approval flag, a capability

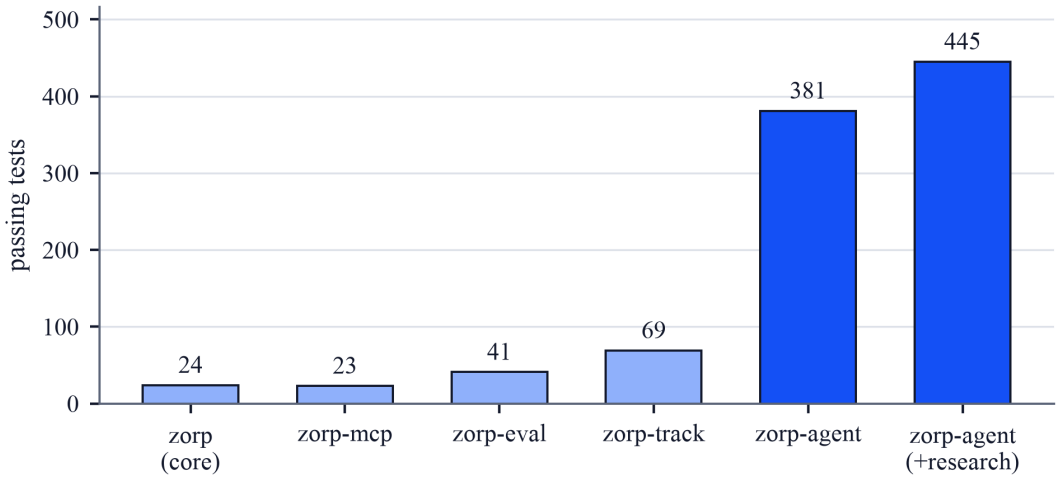

Figure 3: Passing tests by crate and feature set, from a `cargo test` run at the commit described in Table 1.

invoked with its required tool absent, and an index rebuilt from pre-registration files after the database is deleted.

The repository holds 98 commits since its first on 2026-08-08. The entire fork, foundation, and four capabilities were built in under a week, which is a reason to read this as a design and status report rather than a mature-system retrospective.

8 Discussion

What the tests establish, and what they do not. A passing suite demonstrates that the mechanisms behave as specified: that tampering is detected, that gates refuse rather than degrade, that the record survives losing its index. It says nothing about whether the resulting investigations are good. These are different claims, and conflating them is the specific failure this paper is trying not to commit.

Cost of the position. Requiring a human at each checkpoint bounds throughput in a way a fully autonomous pipeline is not bounded. We consider this the correct trade for the target use, since an artifact nobody will attach their name to has no value regardless of how quickly it was produced. It is nevertheless a real cost, and it makes zorp a poor fit for work where volume matters more than defensibility.

Deliberate restraint. Several capabilities are deferred rather than dropped: no hard experiment budget, since a cap tuned to one system's narrow experiments would be arbitrary here; one active track per session, with concurrent execution left for later; and no cross-track memory, so memory stays local to each track's stores.

9 Limitations and Future Work

The paper's central omission is a comparative evaluation, and we would rather name its requirements than approximate it. Such a study needs a shared task set of questions with knowable answers, a protocol run identically through zorp's full loop and through at least the AI Scientist [15], and judgment on two axes that must be scored separately: process, meaning whether the evidence trail is complete and tamper-evident, and outcome, meaning whether the artifact is any good. The second

axis is where this is hard, because the obvious proxies reward confident prose, which is the failure mode under examination. Building that fixture honestly, and reporting it including where zorp loses, is the next piece of work on this paper.

Two further limits are worth stating. The tamper-evidence guarantee is scoped to the pre-registration file and its hash; it establishes that the committed threshold was not altered after the fact, not that the evidence gathered under it was collected competently. And the guarantee inherits git's trust model: a user who rewrites history can defeat it, which makes this a defense against quiet drift rather than against a determined adversary.

On the system itself, the LanceDB store is provisioned but unused, concurrent multi-track execution is unimplemented, and deliver's matching is scoped to academic venues although nothing in the design requires that.

10 Conclusion

zorp is built on a bet: that decomposing investigation into standalone, human-checkpointed capabilities over a tamper-evident record serves how research and technical decisions actually get made better than one autonomous pipeline that produces a finished document and asks a human to review it afterward. The bet is not yet validated empirically, and this paper does not pretend otherwise. What it reports is a working system, honestly bounded: a record that keeps every attempt, a falsification threshold that cannot be quietly moved, decision points with no silent defaults, and four capabilities that a person can pick up one at a time without committing to the whole loop.

11 Availability

zorp is open source under the MIT license. The repository, including this paper's source and the scripts that generate its figures from repository state, is at github.com/aviskaar/zorp [2].

References

[1] Anthropic. Model context protocol. https://modelcontextprotocol.io, 2024.

[2] Aviskaar. zorp: A research agent for evidence-based investigation. https://github.com/aviskaar/zorp, 2026.

[3] Christopher D. Chambers. Registered reports: A new publishing initiative at Cortex. *Cortex*, 49 (3):609–610, 2013.

[4] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

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

[6] Aditya Karnam. quecto: A minimal, vendor-neutral harness for LLM agents. https://github.com/adityak74/quecto, 2025.

[7] LanceDB. LanceDB: A developer-friendly, serverless vector database. https://github.com/lancedb/lancedb, 2024.

[8] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

[9] Nicholas D. Matsakis and Felix S. Klock. The Rust language. *ACM SIGAda Ada Letters*, 34(3): 103–104, 2014.

[10] Marcus R. Munafò, Brian A. Nosek, Dorothy V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis. A manifesto for reproducible science. *Nature Human Behaviour*, 1(1): 0021, 2017.

[11] Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606, 2018.

[12] Mark Raasveldt and Hannes Mühleisen. DuckDB: an embeddable analytical database. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD)*, pages 1981–1984, 2019.

[13] Joseph P. Simmons, Leif D. Nelson, and Uri Simonsohn. False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, 22(11):1359–1366, 2011.

[14] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[15] Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025. URL https://arxiv.org/abs/2504.08066.

[16] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.