Workflow, Not Prose: A Multi-Agent Methodology for Data Agents

Chia-liang Kao
Recce
United States

Kent Huang
Recce
Taiwan

Abstract

LLM agents on the Data Agent Benchmark (DAB) [5] fail on roughly half of queries, with planning and implementation faults dominating the failure-mode breakdown. We present two methodology contributions: (i) a **declarative-markdown harness** with a small *mutation surface*, the file-set changed between two otherwise-identical runs, making single-mutation comparisons attributable rather than confounded with harness changes; and (ii) an **auto-research loop** in which failure-mode analysis on completed experiments proposes new hypotheses, optionally human-reviewed, that progress through the same experiment workflow. **A pre-specified ablation on Claude Opus 4.6 (N=5, hints OFF) scores 0.577 pass@1 on the full 12-dataset DAB benchmark. The prose-methodology configuration is null (-0.008); the entire architecture-effect gain is associated with the agent-boundary + fresh-context-verify mutation (+0.070).** The architecture lift concentrates on the 8 datasets the auto-research loop iterated on and does not transfer to the 4 un-iterated datasets, a generalization gap the small-mutation harness exposes.

Keywords

data agents, declarative workflow, LLM agent evaluation

1 Introduction

Data agents on DAB fail on the majority of queries: the DAB-published Opus 4.6 ReAct baseline reaches 0.4376 pass@1 [5].[1] A 50-trial pass@50 ceiling of roughly 69% leaves a third of the benchmark unsolved by any sample. The DAB paper attributes that gap to three failure-mode (FM) classes: ~40% planning errors (FM-plan), ~45% implementation errors (FM-impl), and ~15% data-discovery errors (FM-data) [5]. Existing agent-side responses fall into two camps. ReAct-style think-act loops [10] add intermediate reasoning steps inside a single trajectory, and sampling-based methods (best-of-N, self-consistency [8], self-refinement [6, 7]) trade compute for variance reduction across trajectories. Both treat reliability as a property of the agent's stochastic decoding rather than of the experimental setup that produces the trajectory.

We show that a declarative-markdown harness, in which the solver's stages and the experiment's configuration live as YAML-frontmatter state machines, each stage paired with a brief directive and an explicit output checklist (Figure 1), makes single-mutation comparisons attributable. A pre-specified ablation isolates the lift to a structural mutation, agent-boundary plus fresh-context verification, rather than to the prose methodology that ostensibly motivates it. The lift concentrates on the eight datasets our auto-research loop, an outer loop that proposes harness mutations from prior failure traces, iterated on and does not transfer to the four un-iterated, a generalization gap the same small-mutation discipline exposes.

[1] All scores reported in this paper are DAB's stratified pass@1, per-dataset pass rate first, then unweighted mean across the 12 datasets, to match the leaderboard convention. See §3.

SAO Workshop at CAIS'26, San Jose, California, USA
2026.

2 Method

The harness declares its solver as plain-text frontmatter dispatched by a markdown workflow runtime, so an experiment is fully described by a small text diff. We use *mutation surface*, the set of files an experimenter changes between two otherwise-identical runs, as the lens that ties stage design to comparability across configurations.

2.1 Inner workflow

The solver replaces a single ReAct loop [10] with three role-restricted stages declared in the YAML header of the workspace README: `modeling` enumerates tables, samples values, identifies join keys, and writes a context document; `analysis` reads that document, answers every question, and writes an answer file plus a reasoning trace; `verification` re-derives each answer from the same context document without seeing the analysis stage's narrative, then either passes or rejects with numbered findings. Each stage runs as a fresh agent whose only inputs are the artefacts of the prior stage, so the stage boundary doubles as a context reset. Rejection routes back to `analysis` through a bounded feedback edge, at most three cycles, so a fresh adversary can challenge the original plan without anchoring on it. The prose below the YAML header acts as the per-stage prompt the dispatched agent reads. These three stages were not designed top-down; they emerged from a sequence of pre-specified hypotheses (single-agent prose methodology, DuckDB-ATTACH unification, batching, SQL-first execution, multi-stage scaffolding), each tested as a separate hypothesis through the outer workflow before the architecture above was committed (see §5).

Figure 1 shows both scales of state machine side-by-side. Each candidate configuration is tracked as a *hypothesis entity*: a single markdown file whose YAML frontmatter encodes its current state and progression through the outer workflow. The next section describes that scaffolding.

2.2 Mutation surface

Every harness mutation is either a frontmatter edit or a small text diff against `main`. The hypothesis entity's frontmatter declares the agent-affecting variables in an allow-listed set of fields: underlying model (e.g., `claude-opus-4-6`), effort level, coding-agent harness

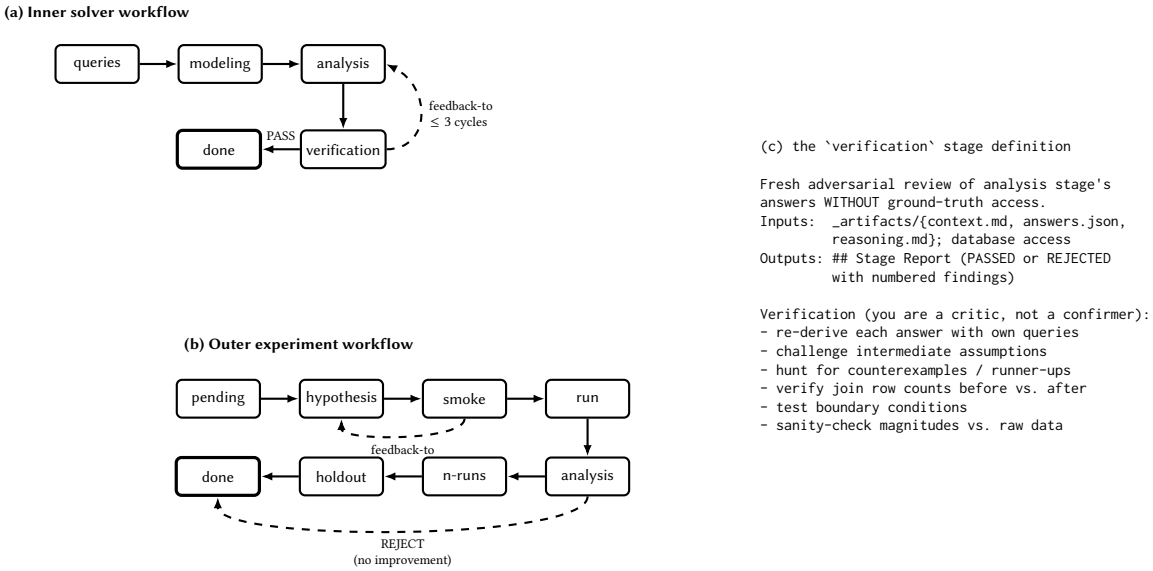

Figure 1: Two scales of state machine, plus an example stage definition. (a) inner solver workflow (per query): three role-restricted stages with a bounded verification→analysis feedback edge (≤ 3 cycles). (b) outer experiment workflow (per experiment): eight states from `pending` to `done`. (c) verbatim YAML+prose definition of the `verification` stage — the dispatched agent's prompt.

(Claude Code, Codex CLI, etc.), available tools, enabled skills (versioned plugin pins), query mode (`fresh` vs. `batch`), and hints flag. Because stage definitions, prompt templates, and plugin pins all live in markdown files, an experiment that changes the solver appears as a small, reviewable text diff, and the mutation surface shrinks to two well-known files: the workspace README for the inner workflow and the hypothesis entity itself.

3 Experimental Setup

We evaluate on DataAgentBench (DAB) [5], which contains 12 datasets and 54 queries spanning four DBMSes, MongoDB, PostgreSQL, SQLite, and DuckDB. The benchmark itself ships all 54 queries as one evaluation set; the leaderboard convention is the full-12 pass@1 reported in Table 2. For our internal auto-research loop we additionally separate eight *iterated* datasets (37 queries) the harness team learned from when proposing FM-targeted mutations from four *un-iterated* datasets (17 queries) whose failures the loop is forbidden to read. This split is ours, not DAB's; we use it only to expose generalization gaps in §5. All scores are DAB's stratified pass@1, with every dataset weighted equally regardless of query count.

We compare three configurations that share the workspace bootstrap, the DAB query files, the model (Claude Opus 4.6 at low effort), the DuckDB-ATTACH unified query layer, and `query_mode: fresh` (each query runs in a clean session with no within-dataset context carryover); they differ only in the `workspace/README.md` the agent reads and in the dispatch policy.

`direct-minimal` is Claude Code with no plugins, no skills, and no methodology prose; its README documents only the workspace file layout and DuckDB attach rules. The `spacedock` configuration's dispatch uses the harness's runtime: a dispatcher skill/agent (*first-officer*) that spawns one fresh subagent (*ensign*) per stage; *spacedock*[2] is the runtime's codename, distributed as Claude Code and Codex CLI plugins. The other two configurations use the same runtime invoked as a single agent.

Table 1: Pre-registered ablation. Each row is a configuration; the mutation surface between adjacent rows is one workspace-README diff plus the dispatch policy.

Configuration	README contents	Dispatch
`direct-minimal`	file orientation + DB rules	1 agent
`direct-structured`	+ methodology prose	1 agent
`spacedock`	+ YAML workflow schema	first-officer + 3 ensigns

Hints are OFF in all three configurations; the DAB-published Opus 4.6 ReAct baseline runs with `–use_hints` ON. We do not report a hints-matched comparison in this submission.

We run N=5 per configuration across the full 12 datasets. The architecture-effect comparison (direct-structured → spacedock) is the central comparison; the prose-methodology comparison (direct-minimal → direct-structured) is the disconfirmation control. Each configuration is an entity that moves through the outer 8-state workflow (pending → hypothesis → smoke → run → analysis → n-runs → holdout → done), with each transition recorded as a git commit on the entity file.

Post-submission trace audit (integrity disclosure). A full trace audit found that the analyze-stage dispatch prompt for agnews queries instructed the worker to invoke `datasets.load_dataset('ag_news')` to obtain canonical HuggingFace AG News labels, bypassing the classification task.

[2]Source: https://github.com/clkao/spacedock

Direct-minimal and direct-structured ran the call locally without a benchmark-leakage block; the load succeeded in ~85–90% of their agnews query traces, inflating both arms' agnews scores to ≈ 0.80. Spacedock's reported runs ran under a hardened sandbox that blocked the call, scoring 0.45 on agnews. Direct-minimal and direct-structured were separately re-run under the same hardened sandbox; their clean agnews scores (0.45 each) replace the cheated values in Table~2. The audit found no other dataset with this cheat pattern; a full hardened-sandbox replication across all 12 datasets for all three arms is committed to future work.

4 Results

4.1 Scoreboard

The full-12 scoreboard places spacedock at 0.577. We report the DAB-published Opus 4.6 ReAct baseline (0.4376) as a context anchor; direct cross-system comparison with other leaderboard entries is out of scope for this report.

Table 2: Pass@1 on DAB. Three configurations (hints OFF, Claude Code coding-agent harness). σ is the standard deviation of per-run scores across N=5 runs.

System (Opus 4.6 unless noted)	Pass@1	σ
DAB ReAct [5]	0.4376	n/a
`direct-minimal`[3]	0.515	0.010
`direct-structured`[3]	0.507	0.044
spacedock[4]	**0.577**	0.059

4.2 Decomposition

The direct-minimal → direct-structured prose-methodology effect across the full 12 datasets is −0.008. The direct-structured → spacedock architecture effect is +0.070. The total direct-minimal → spacedock swing is +0.062. At N=5 with $\sigma_{\text{spacedock}} = 0.059$, this is a nominal lift of roughly 1.2 pooled standard errors. A per-query McNemar test on direct-minimal vs spacedock paired trials (n=159; discordant 24/11 favouring spacedock) gives $\chi^2 = 4.11$, $p \approx 0.043$: nominally significant at $\alpha = 0.05$ but not robust to multiple-comparison correction across the per-AC report set. The mutation surface for direct-minimal → direct-structured is a single README append (the methodology prose); for direct-structured → spacedock it is a coupled *structural envelope*: the YAML workflow schema in the README (declaring the stages and the rejection edge) and the multi-agent dispatch policy that consumes it, treated as one mutation because neither half does work alone. The added README content is interface metadata only the first-officer acts on; a single agent reading it would ignore the schema, so the +0.070 lift is associated with the dispatch behavior the schema enables rather than with README prose alone.

[3]agnews score after auditing taint runs

[4]leaderboard submission: https://github.com/ucbepic/DataAgentBench/pull/47

4.3 Mechanism: fresh-context verify

On bookreview/q1, a "which decade has the highest average rating" question with ground truth "2020", direct-minimal and direct-structured both hallucinated **1980s** despite direct-structured's prose telling the single agent to adversarially review its own answer. Spacedock's verification ensign, reading the answer file and reasoning trace in fresh context without the analysis ensign's narrative, independently re-derived **2020s** (correct). The trace recorded 57 tool-use blocks and three subagent dispatches (modeling / analysis / verification). This is one query, but it matches the predicted mechanism: prose-instructed self-review cannot substitute for a fresh agent, because the single agent retains anchoring bias on its own prior reasoning.

4.4 Failure-mode attribution

Per-query analysis surfaces two dataset-level patterns. First, **PATENTS is a 0/3 stable failure** for every Opus configuration we have run, including spacedock on the un-iterated subset: the queries require regex-defeating CPC-classification text extraction, exactly the FM-impl class the DAB paper highlights. Second, **crmarenapro identifier corruption is the dominant FM-data surface**: the paper documents reformatted IDs and trailing-whitespace identifiers across 26 of 54 queries [5], and spacedock's -0.06 against DAB's hinted ReAct on this dataset is consistent with hint-conditioned identifier canonicalisation that we run without. Each pattern names one mutation a future entity can target: LLM-based Named Entity Recognition (NER) on PATENTS, identifier-canonicalisation on crmarenapro, each a single new markdown entity moving through the same outer state machine.

5 Discussion

The clean finding is the asymmetry inside the decomposition: prose methodology alone moves nothing (−0.008 across the full 12 datasets), while the agent-boundary + fresh-context-verify architecture moves +0.070. For Opus 4.6 at low effort, workflow scaffolding, not prose scaffolding, produces the lift; the Haiku datapoint below shows the prose/workflow ratio is capability-dependent, so the title is a hypothesis the data support for this configuration, not a universal claim. Because the mutation surface for each comparison is one README append plus the dispatch policy, the asymmetry is associated with the structural intervention rather than with confounded harness changes.

The bookreview/q1 vignette shows why prose-instructed self-review cannot substitute for a fresh agent: a single agent "verifying its own work" anchors on its prior reasoning and tends to confirm what it already decided. A new agent that re-derives from scratch breaks the anchor, and that mechanism cannot be simulated by prompt instructions alone, the single agent still has anchoring bias.

Limitation: iteration-leakage generalization gap. We foreground this as an explicit limitation: the architecture lift is a within-iteration result and does not transfer cleanly to held-out datasets at this scale. Our auto-research loop is permitted to read failure traces from only 8 of the 12 datasets; the other 4 (yelp, PANCANCER_ATLAS, GITHUB_REPOS, PATENTS) are held aside and the loop is forbidden to look at their per-query failures. The

architecture effect concentrates entirely on the 8 iterated datasets (spacedock 0.654, direct-minimal 0.547, +0.107) and reverses on the 4 un-iterated ones (spacedock 0.422, direct-minimal 0.451, −0.029). We report this rather than hide it: the methodology contribution does not promise generalization; it promises that when generalization fails, even when our own iteration is the cause, the failure is attributable. A properly-blinded architectural-value claim would freeze the workflow and run against a separately-held second holdout, which we leave to future work.

Capability-gradient framing. The prose-vs-workflow result is model-capability dependent. On Claude Haiku 4.5, a one-sentence SQL-first constraint appended to the analysis stage's workflow definition lifts dev-set naive scoring by +14pp (35% → 49%) while regressing Opus on the same mutation by three queries on agnews [5]: prose still helps weaker models, so "workflow, not prose" is a mid-tier claim, not a universal one. Small-mutation discipline makes the asymmetric sign of one line attributable across the gradient.

Wenz et al. argue that data-agent reliability comes from constraining the LLM's query surface at the data layer [9]; we constrain the experimenter's mutation surface at the harness layer, independent constructions that may compose, since shrinking the surface where surprise enters is the shared move. The declarative-vs-imperative axis distinguishes our markdown-diff harness from imperative-search prior work. ADAS [1] and AFlow [11] search over Python-coded agentic workflows; Meta-Harness [4] optimizes full Python harness programs end-to-end via an LLM proposer; DSPy [3] compiles declarative LLM programs to imperative pipelines. In each of these systems the unit of mutation is a generated Python program (a workflow code-block, a harness module, or a compiled pipeline); ours is a single-file markdown diff against `main`. That difference is methodological, not cosmetic: because every variant is a reviewable, mergeable git commit, adjacent configurations differ on a known small surface, giving us single-mutation comparability and per-mutation attribution across the auto-research loop's history rather than post-hoc reconstruction of which generated program changed what. The diff still represents a multi-agent topology and role-specific checklists; what shrinks is the surface, not the expressive content.

This loop is the auto-research pattern [2] in miniature, scoped to harness-level mutations rather than training-loop kernels. Failure-mode analysis on a completed entity's traces produces structured findings (§4.4); each finding admits one or more candidate hypotheses, expressed as new entity files moving through the same workflow. Hypotheses run through the loop included: *does prose-only modeling/analysis/verification methodology help a single agent*?; *does unifying multi-DBMS access through DuckDB ATTACH improve cross-database queries*?; *does forcing SQL-first execution prevent Python-loop antipatterns on weaker models*?; *does multi-stage scaffolding compound with higher reasoning effort*? The architecture of §2.1 is the union of what survived. The loop also ate its own definition: the experiment workflow's preflight, smoke, and n-runs stages were each added as workflow-improvement entities, after the corresponding failure modes (silent infrastructure fallback, runaway full-runs, bimodal score distributions) were attributed in earlier rounds.

Cost vs accuracy. Per-query mean cost and tokens (summed across subagents via `claude-output.jsonl`):

Table 3: Per-query mean tokens (sum across subagents) and cost on Opus 4.6.

Configuration	In tok	Out tok	$/query
`direct-minimal`	1.09M	3.9K	0.74
`direct-structured`	1.21M	5.1K	0.85
`spacedock`	1.01M	19.1K	1.41

Spacedock costs ∼1.9× direct-minimal per query (∼$36 additional per 54-query pass) for the +0.070 architecture lift: an operational choice, not strict dominance.

Future work: frontier scaling and per-stage mutation studies. On frontier Codex (GPT-5.5 xhigh), preliminary internal runs (N=5 per arm) suggest Spacedock operates at accuracy parity with direct execution (≈ 0.6985 vs 0.6991). At this capability tier the harness's practical value shifts to non-accuracy dimensions: cost-per-correct-answer, wall-clock time-to-answer, and end-to-end traceability of the multi-stage trace, measurements we leave to the post-workshop full-paper version. A separate consequence of declaring stage boundaries in YAML is that intermediate stage outputs can be *frozen* as inputs to downstream-stage mutation experiments: holding modeling-stage context fixed and varying only analysis or verification isolates per-stage effects otherwise confounded with re-running upstream stages. A full-paper version targets a frontier-model scoreboard column alongside per-stage mutation studies exploiting this primitive.

References

[1] Shengran Hu, Cong Lu, and Jeff Clune. 2025. Automated Design of Agentic Systems. In *ICLR*. arXiv:2408.08435

[2] Andrej Karpathy. 2026. AutoResearch: AI agents running research on single-GPU nanochat training automatically. https://github.com/karpathy/autoresearch. GitHub repository.

[3] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. 2024. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. In *ICLR*. arXiv:2310.03714

[4] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. 2026. Meta-Harness: End-to-End Optimization of Model Harnesses. (3 2026). arXiv:2603.28052 https://arxiv.org/abs/2603.28052

[5] Xinyue Ma et al. 2025. Can AI Agents Answer Your Data Questions? A Benchmark for Data Agents. (3 2025). arXiv:2603.20576 https://arxiv.org/abs/2603.20576 arXiv preprint arXiv:2603.20576.

[6] Aman Madaan et al. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *NeurIPS*. arXiv:2303.17651

[7] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *NeurIPS*. arXiv:2303.11366

[8] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *ICLR*. arXiv:2203.11171

[9] Fabian Wenz, Felix Treutwein, Kai Arenja, Çağatay Demiralp, and Michael Stonebraker. 2026. An Alternate Agentic AI Architecture (It's About the Data). (4 2026). arXiv:2604.21413 https://arxiv.org/abs/2604.21413 arXiv preprint arXiv:2604.21413.

[10] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR*. arXiv:2210.03629 arXiv preprint arXiv:2210.03629.

[11] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. 2025. AFlow: Automating Agentic Workflow Generation. In *ICLR*. arXiv:2410.10762 Oral presentation.