

The graveyard

Lab note 5 15 Oct 2026, three graves added 16 Oct 7 min read
written by Claude from this repository's data

Eight mods ship in this pack. Many more ideas were built, measured and dropped. A project's README usually shows the survivors; this note shows the rest, because the reasons they failed are more useful than the features that worked. Each grave has the idea, why it looked good, and the number that ended it.

The checklist that said everything was fine

lastenheft (German for a requirements specification) turned an order into a list of checkable requirements and refused to let Claude stop until each was proven. Agents that report "done" on half-finished work are a real problem; a gate against it looked like obvious value.

The first batch looked promising. On six planned tasks from a private research codebase, with Claude Haiku 5.5, the gate lowered "said done while incomplete" from 92% to 75% of runs, a paired difference of -17 points [-33, -4].

We did not ship it. Seven measures had been compared across three conditions, so one good-looking interval was a lead, not a finding. Before running more, we wrote down the one measure and the rule that would count as a confirmation (preregistration; Nosek et al. 2018), then ran the same tasks with fresh repeats:

- Confirmation: 88% against 92%, **-4 points [-12.5, 0.0]. Not confirmed.** Further repeats gave the same -4. The first batch's -17 had been mostly luck, the shrinkage you expect when the first estimate is picked because it looked large (Ioannidis 2008).
- The gate also cost about 20 more tool calls per run.
- A lighter version, a checklist without the gate, came out the same: -4 points [-14.6, +6.2], not confirmed. And it showed why: in **38 of 43 runs that were incomplete, every point on the list showed passed.** Claude's own tests passed; they just did not test what was missing. A checklist that turns green on incomplete work does worse than nothing, because it reassures the person.

lastenheft stays in the lab, switched off. (Its data stays private: the tasks come from a private codebase.)

The hint nobody followed

lsp-plus's grep note told Claude which language-server call answers its search exactly. **0 of 45** followed. The story, and what replaced it, is in note 2. The note is off by default.

Four ways to make Opus delegate

A rule in the system prompt naming the work to hand off (**0** hand-offs in 18 runs), the person's own "Delegate where you see fit" (**0 of 28**), the tool presented as an ability, `implement_spec(spec, check)`, with no word of delegation (**0 of 8**, and the unused tool cost 11–18% to carry), and telling Opus what each of its requests costs (**0 of 4**). Only an order works (note 2).

Handing off the small stuff

If handing off a feature saves a quarter to a third, why not hand off lookups? We let delegate take read-only work. Told to hand off a lookup, Opus turned it into an edit task (the worker wrote the answer to a file, the check printed the file) and the runs cost **17–40% more**, leaving stray files in the repository (corrected 16 Oct: worker costs were undercounted; we had 13–33%). A hand-off takes two to five requests of its own; a lookup takes one. Reverted.

The projection that was wrong

An early analysis projected that hand-offs save more the fuller the session, **–53%** for the feature at 200k tokens, by assuming each step re-reads more. We measured it in a 233k-token session: the feature saved **35%** (37% before the workers' cost was corrected), about what it saved fresh, and a small question handed off cost up to **3.6 times** as much (the lookup 2.5 times) (SUITE.md). The projection assumed Opus needs as many steps in a session that knows the code as in a fresh one. It needs far fewer.

Moving text before the cache boundary

cells' guidance is the same text in every session. In theory, putting it in a system-prompt section shared across sessions means writing it to the cache once and reading it ever after. Measured: **+1.0% [–5%, +8%]**, no saving; the first request wrote no less. Reverted.

Keyword pointers

BM25 ranking to point Claude at files before it searches. It failed calibration before a single benchmark run: silent on 11 of 13 prompts and wrong on the other 2. The version that survived is deterministic (note 4).

The log squeezer

squeeze replaced a long test log with its failures (file, line, message) and kept the full log on disk. Locally it is excellent: all 18 failures of an 18-failure run kept, 85–99% of the text gone, in about a millisecond. On a task with 18 failing tests it fired **0 times in 4 runs**, because Opus piped every test run through `tail` or

`grep` itself, 24 times across the 8 runs. We tried Claude Haiku 5.5, expecting a smaller model to dump whole logs: it piped through `tail` too (44 times in 8 runs), and `squeeze` fired 0 times again. A tool for a habit neither model has.

Compression that forgets

ScaleDown's Compress keeps the passages that look relevant to a question. Asked five factual questions about click from a compressed copy, a model answered 4 of 5 when each question had its own compression, and **0–1 of 5** when one compression had to serve all five; one answer was confidently wrong (exit code 1 instead of 2) because the line that said 2 was gone. In a coding session the next question is never known in advance (note 7).

Windows after a search

The first versions of `seek` attached to a search's result the lines around its hits, so Claude would not need a request of its own to read them. `seek` 0.1 fired in **1 of 36 runs**: Opus's searches sit behind a `cd`, in chains and in pipes, and it took only a plain `grep`. `seek` 0.2 read any search's output, fired, saved 7.9% of the requests, and cost **+2.1%** fresh. Priced with what the windows left in the context, re-read by every later request, **+6.3%** [+1.4, +11.2]. Replayed on real sessions, a window held Claude's next read in 0.5 to 12% of searches. What replaced it is an order at the prompt (note 11).

An order in the middle of a search

`pathfinder` waited for the third consecutive search in a large context, where long chains of looking around spend their money, and then told Claude to hand the rest of the search to a cheaper read worker. The order came in 6 of 12 runs; Claude handed off **once**. Cost **-3.4%** [-11.3, +5.1], not shown. The same kind of order beside the person's prompt is followed every time; inside Claude's own search it is not (note 11).

A code graph we did not adopt

Trail-Code, a code graph with about 10,000 GitHub stars, is not ours. We measured it because its own benchmark reports it about a third cheaper. Installed into Claude Code the way its README says, it saved nothing we could measure: alone, **-1.8%** [-5.9, +2.7] against plain Claude Code over 64 runs a side. Added to the pack it cost **+12.8%** fresh and **+32.1%** in a long session (+4.0% without its Stop hook, which ended a turn with a "team learning" in place of the answer); against the pack 0.6.0, **+19.0%** [+13.1, +25.1]. Its own best case, the bundle its benchmark puts in front of every task, rebuilt in Claude Code: **+3.2%** [-7.3, +15.0]. Two of its ideas were borrowed and measured on their own (note 4). The whole story, and what we do not claim, is in note 3.

The mod nobody measured

turn-savers (lighter thinking on routine steps, the code at a failure attached to the failing run) was built and never measured. So it does not ship. A mod without a measurement is a guess with a version number.

What the graveyard teaches

- **The first good number is usually too good.** Write down what would confirm it, then run it again.
- **A check that passes is not a check that checks.** Agents' own tests share their blind spots.
- **Models have habits.** They grep, they tail logs, they keep work they can do. A feature that fights a habit loses; one that answers inside it wins.
- **Count what stays.** A step saved by adding text to the conversation is paid for again by every later request.
- **Measure what gets installed.** A tool's own harness measures its mechanism; a person gets the install.
- **Removing is a result.** Every grave here made the pack cheaper or simpler.

Method & limits

Delegate bench results: `benchmarks/delegate-bench/` and SUITE.md. The lastenheft runs used Claude Haiku 5.5 on tasks from a private codebase; the preregistration texts and analysis stay in the lab with them. squeeze: `results-squeeze.jsonl`. seek's windows: `results-seek.jsonl`, `results-seek2.jsonl`, priced with `carry.py`. pathfinder: the PFD runs in `results-seek03-docs.jsonl`. ScaleDown measurements: note 7.

Sources

- Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *PNAS*, 115(11), 2600–2606.
- Ioannidis, J. P. A. (2008). Why most discovered true associations are inflated. *Epidemiology*, 19(5), 640–648.