

Notes nobody reads

Lab note 2 15 Oct 2026 4 min read
written by Claude from this repository's data

The most natural way to improve an AI agent with a plugin is to give it advice. Notice it doing something slow, tell it the faster way. We built two mods on that idea. Both measurements came back with the same number: zero.

45 hints, 0 followed

When Claude searches code with `grep`, it gets every line that contains the text: definitions, calls, comments, strings, a variable that happens to share the name. A language server knows the difference. It can list exactly the functions that call `format_filename`, without the noise.

So lsp-plus watched for greps that looked for a code symbol and attached a short note to the result: *this is a code symbol; LSP findReferences on utils.py:442 lists exactly its uses*. Correct, specific, one call away.

In every benchmark run with the pack, that note fired 45 times. Claude acted on it 0 times (SUITE.md).

Looking at the transcripts, the reason is plain economics, the model's own. The note arrives after the grep has already answered. Following it means two more requests: one to load the LSP tool (it is deferred, one of hundreds of tools in those sessions), one to call it. Two steps to get an answer Claude believes it already has. From where the model stands, ignoring the note is the efficient choice.

Answers instead of advice

So we stopped advising and started answering. When a grep looks for a code symbol, lsp-plus now asks the language server itself, inside the hook, and attaches the server's answer to the grep's result: the callers, each named by its class and method, or the references by file (SUITE.md).

```
'resolve_color_default' per the language server: called from 5 functions in 3 files (
src/click/exceptions.py: ClickException.__init__ (48)
src/click/termui.py: _readline_prompt (99), get_pager_file (362), progressbar (581)
src/click/utils.py: echo (342)
```

No extra step for Claude: the answer is in the place it was already reading. On the rename task that cut requests by 25% and cost by 6.5%. Where no grep asked for a symbol, it said nothing and cost nothing. The note that asked Claude to do the same thing itself was switched off.

174 runs, 0 delegations

The second idea was bigger. Opus is expensive; Sonnet and Haiku are much cheaper. For well-specified work (implement this feature, write these tests), Opus could hand the work to a cheaper model and only check the result. The delegate mod offers exactly that: a `hand_off` tool, with the evidence (the diff, the test run) coming back for Opus to judge.

Asked to hand off, it works: a feature cost 22–35% less, every result correct (corrected 16 Oct: worker costs were undercounted; we had 27–37%). The question was whether Opus would hand off without being asked. We tried everything we could think of (SUITE.md):

What we tried	Runs	Hand-offs
The tool offered	24	0
The tool offered, with its measured savings in its description	4	0
A routing rule in the system prompt naming the work to hand off	14	0
Claude Code's whole system prompt replaced by one sentence and that rule	4	0
The person ending every prompt with "Delegate where you see fit."	28	0
The same tool described as an ability, <code>implement_spec</code> , with no word of delegation	8	0
Telling Opus what each of its requests costs in this session	4	0
Claude Code's own Agent tool, available in every run	174	0

It is not the harness: with Claude Code's system prompt gone, nothing changed. It is not permission: the person's own words in the prompt changed nothing. Opus 5.5 does not give away work it can do itself.

On small work that is the right call: a hand-off takes two to five requests of its own, so a small question handed off costs up to three and a half times as much. Opus only misses the big, specified work.

An order works

One thing did work, every time. When a mod attached an order to the prompt, phrased as the person's ("The person approved this plan; implement it through `hand_off`"), Opus handed off in 8 of 8 runs, and the feature cost 22% less (27% before the workers' cost was corrected; SUITE.md).

That became the design. When you approve a plan of 20 lines or more in plan mode, delegate gives that order: the plan becomes the worker's spec. Claude does not decide to delegate; your approval does.

Other tools found the same thing

We are not the only ones who learned this. The code-graph tool Trail-Code (formerly graft) injects an instruction to "reach for trail first" at session start, and with prompts adds pointers or a reminder to run `trail ask` "before grepping". Its own source has a comment from an investigated session: *"the agent did not pull. It grepped 38 times."* (`src/claude/format.ts`). Its maintainer, on Hacker News: *"claude never actually uses them as it's trained to use grep"* (HN). graphify, another code-graph tool, injects a note before searches: *"MANDATORY: ... You MUST run graphify query "<question>" before grepping raw files"* (`graphify/cli.py`).

Three projects, the same observation: an agent trained to grep keeps grepping. Shouting does not change that. Answering inside the grep does.

What this changed

- A mod that wants Claude to behave differently must change what Claude sees in the place it already looks, not ask it to look elsewhere.
- Advice is not free. Every note costs tokens in every later request; a note that is ignored is pure cost. Our rule since: a mod speaks only where it has proof that speaking helps.
- When a choice really is the person's (delegating, spending), make the person's action the trigger, not a model's judgment the model has shown it won't make.

Method & limits

Click 8.5.0, Opus 5.5, headless Claude Code; hand-off counts read from transcripts. "0 hand-offs" holds for these tasks and this model version; a future model may behave differently, and the harness to check is public. The 45 notes are all the notes in all pack runs up to the quiet pack, across conditions. Data:

`results-quiet.jsonl`, `results-answer.jsonl`, `results-prompt.jsonl`, `results-fit.jsonl`,
`results-delegation-ideas.jsonl`.

Sources

- trailhq, Trail-Code, source at `ead585b` (checked 15 Oct 2026).
- Show HN discussion of graft, maintainer comments.
- Graphify-Labs, graphify, `graphify/cli.py` (the hook-guard note).