

Claude Code alone or with an independent tester: the same quality, and where the money went

One feature ticket, built three times by Claude Code testing its own work and three times by Claude Code with OneDroid Argus as the tester, then judged on checks neither setup had seen.

Case study. Runs of 16 October 2026. An experiment by our team, 3 judged runs per setup.

In short

- Our team gave Claude Code the same feature ticket in two setups: **alone**, testing its own work as it normally would, and **with a tester**, implementing only while OneDroid Argus ran checks it never saw.
- Both setups were then judged on 13 hidden checks that neither had seen, not even by name: all 6 runs passed all 13.
- With the tester, the coding agent ran its own tests twice in its three runs together; alone, it ran them 3 times in an average run. It used 69% fewer tokens and cost 49% less (\$1.01 against \$1.97).
- The tester's one-off session, spread over its three runs, brought the whole to \$2.78 a run: 41% more than Claude Code alone.
- One ticket, 3 judged runs per setup, and a ticket too easy to separate quality. The limits are listed below.

The question

A coding agent that writes its own tests spends tokens checking itself, and its tests share its own reading of the ticket. An independent tester costs money too. On the same codebase and the same ticket, does Claude Code with an independent tester cost less, and deliver more, than Claude Code testing itself?

The setup

Item	Value
The codebase	A small tasks app with a JSON API and a web page. Due dates, priority and search were already live and had to keep working.
The ticket	One feature: tags on tasks, with validation rules, adding and removing one tag, filtering by tag, a tag list with counts, and chips on the web page. The same text for both setups, word for word.
The model	Claude Code on Claude Opus 5.5, high effort. The same start commit, permission mode and "do not commit" for both setups.
Setup A: alone	Claude Code working as it normally would. It may write and run tests and use subagents.
Setup B: with a tester	Claude Code implementing only: no test suite, no review or test subagents. OneDroid Argus then ran the build checks, and the coding agent got back only the names of the checks that failed, for up to 4 rounds. The build checks ran together with 37 older checks of the same app, as regression checks.
Who wrote the checks	A separate tester session wrote the build checks and the hidden checks before any build. The coding agent never saw a check body or an expected value.
The judge	13 hidden checks, stored in OneDroid Argus only after the last run, so neither setup saw them, not even as names. The 12 build checks were also run on every setup at the end.
Runs	3 judged runs per setup, in random order, on 16 October 2026. A fourth run of setup A was not judged and is left out of every figure here.
Cost	List prices for Claude Opus 5.5: \$4 input and \$20 output per million tokens, \$0.20 cache read, \$5 cache write, \$8 for a one-hour cache write.

The result

Mean per run	A: alone	B: with a tester
Hidden checks passed	13 of 13 in every run	13 of 13 in every run
Build checks passed	12 of 12	12 of 12
Tester rounds	none	1 in every run: it passed first time
Times the coding agent ran its own tests	3	under 1: twice in the three runs together
The coding agent's tokens	2.37 million	0.72 million (-69%)
The coding agent's cost	\$1.97	\$1.01 (-49%)
The whole cost, the tester's session included	\$1.97	\$2.78 (+41%)
Wall time	not compared	not compared: the tester runs include waiting for human approvals

Claude Code alone vs Claude Code + OneDroid Argus.

Same codebase, same ticket. Opus 5.5, 3 runs per arm.
Quality: a tie. All 6 runs passed 13/13 hidden checks and 12/12 build checks.

The builder, per run

Claude Code alone

 **\$1.97**

Claude Code, implementing only

 **\$1.01 (-49%)**

The whole cost, with the tester's session

Claude Code alone

 **\$1.97**

Builder + tester

 **\$2.78 (+41%)**

The builder stopped testing itself: 0.7 test runs vs 3.

The tester is paid once per feature. Its checks stay as regression checks.

List prices. n = 3 per arm. The ticket was too easy to separate quality:
both arms hit the ceiling. Wall time not compared (human approvals in arm B).

OneDroid Argus · onedroid.ai/argus

The same numbers as a chart. Means of the 3 judged runs per setup, at list prices.

The tester's session wrote both sets of checks once and cost about \$5.30. It is split evenly over the three B runs, which adds about \$1.77 to each.

What the numbers say

- **Quality was a tie.** Both setups hit 13 of 13, so this ticket cannot show whether an independent tester improves quality. It was too easy for that.
- **The coding agent got cheaper because it stopped checking itself.** It ran its own tests twice in its three runs together, against 3 times in an average run alone, and used 69% fewer tokens. Every B run passed the tester in its first round.
- **On one feature, the tester's one-off cost outweighed that saving.** The tester is paid once per feature, and its checks stay as regression checks for the next change: setup B already ran 37 older checks of the same app that way. Whether that pays back over several features was not measured here.

What this experiment did not cover

Not covered	Why it matters
A ticket hard enough to separate quality	Both setups passed every hidden check, so the result says nothing about which one builds better.
More than 3 runs per setup	The figures are means of 3 runs, not a distribution.
Wall time	Times aren't comparable: the tester runs include waiting for human approvals.
Identical wrappers	Setup B ran inside an agent runner with its own fences. Setup A ran plain Claude Code with an allow-list: the honest default, but not the same.
Equal feedback	Setup B also got regression feedback from the 37 older checks. Setup A had only the repository's own tests.
Several features	The tester's cost was spread over one feature. Reuse of its checks across features was not measured.

What we take from it

- On an easy ticket, an independent tester buys no extra quality and costs more in total. Its value has to show on harder work, and over several features.
- Moving the testing out of the coding agent made the coding agent cheaper. Whether the whole gets cheaper depends on how often the tester's checks are reused.
- Next: the same comparison on a ticket hard enough that the first build fails some checks.

Claude and Claude Code are products of Anthropic, named only to identify what was used. OneDroid is not affiliated with Anthropic. OneDroid Argus is built by Providentia Worldwide. Questions: michal@onedroid.ai.