OneDro1d

CASE STUDY

Two AI models, 86 checks, and 4 reported differences neither tested

Two AI models, Claude and Codex, checked a decades-old Fortran library against its own user guide with OneDroid Argus. Run 26 to 27 October 2026.

Case study. An experiment by our team, run 26 to 27 October 2026. OneDroid Argus sealed and ran both sets.

At a glance

Held = the check passed. **Sealed** = locked and timestamped before the run. **Proven able to fail** = it turned red on a deliberately broken copy of the library.

86 of 86 checks held. 84 of them were proven able to fail. We deliberately broke a copy of the library and re-ran every check. A check that never turns red on a broken copy proves little. The broken copy was built after the sets were final, with edits chosen so the checks could see them: this shows the checks can react, not that they would find an unknown difference.

- **84** held, and proven able to fail on a deliberately broken copy.
- **2** held, but never shown able to fail, so they prove less.
- **0** failed: in all 86 checks the code did what the guide says (not a clean bill of health).
- **4** differences between code and guide, reported by an earlier study, that neither set tested (not reproduced here).

Why this can be trusted, and its limits

- **Sealed before the run.** OneDroid Argus sealed each set of checks, then ran it as a certified run. Both certificates were verified with the open-source verifier against a public chain (Base Sepolia). Only that anchor was read back by us, and the Bitcoin timestamp anchor was still pending on 27 October.
- **Same team.** We ran the study and the tool. The seal proves the checks were fixed before the run, not that they are independent.
- **The running build was not measured.** Both certificates say the build's identity is only what the tester declared. Our referee separately checked it on the test service before the runs.

In short

We gave two different AI models the same job: read a library's user guide, write checks that quote it, and prove them against the real code. They worked blind, reviewed each other's drafts, and OneDroid Argus sealed and ran the final sets. Every check held. The more useful result is how much those passes are worth: 84 were shown able to fail, 2 were not, and 4 differences reported by an earlier study sit outside anything either model asked.

The question

A team that lets an AI write its tests wants to know how far a green result can be trusted, and whether a second, different model makes it more trustworthy. A sealed, certified run answers whether the code does what its guide says. A deliberately broken copy answers whether the checks can notice when it does not.

The method

Item	Value
Code under test	A decades-old Fortran library, pinned by checksum. The authors and maintainers are not named here.
The contract	Its user guide. Every expected value had to come from it (one Claude check is the tester's own property, noted below). The testers also had the code and a running build to try it on.
Testers	Claude (`claude-opus-5-5`, low reasoning effort; the first drafting step on `claude-opus-4-8`) and Codex (`gpt-6.1-sol`, settings not exposed). They wrote their first drafts blind: no sight of each other's work or of the earlier study.
Cross-review	Each tester challenged the other's draft. Our referee, itself an AI session, ruled on every challenge before anything was sealed.
Checks	28 from Claude, 58 from Codex: 86 in all. Sealed by OneDroid Argus, then one certified run each.
The broken copy	A copy of the library with 49 deliberate edits, with every check run on both builds (the normal-build results came from earlier runs of byte-identical sets). Each tester wrote down, before the runs, which of its checks would fail.
Readers	A writer from the other model family read each tester's results and wrote a report. The referee compiled one final report.
Environment	A sandboxed test service in a Kubernetes cluster, GNU Fortran 13.5.0 with `-O2`, and a development deployment of OneDroid Argus. Only development and test systems were used.
Dates	Drafts 26 October 2026. Final sets, seal and certified runs 27 October 2026.
Who wrote and ran the checks	Our team. Two AI sessions wrote the checks. An AI referee session ran the study, with team members authorising the seal and the rebuild of the broken copy. OneDroid Argus sealed the sets and ran the certified runs. The library under test was not modified; the broken copy is a separate copy.

The result

Tester	Checks	Held	Red on the broken copy	Predictions matched
Claude	28	28	27	56 of 56
Codex	58	58	57	116 of 116
Both	86	86	84	172 of 172

Every set held on the normal build in its certified run: 28 of 28 and 58 of 58. None failed in this run, so this run produced no finding against the code. It says nothing about the 4 reported differences (see below).

Read with care: one run per model, one library, and the same team wrote the checks and ran them.

The 2 checks that stayed green on the broken copy are a Claude check on the machine constants and a Codex check on one internal quantity. They held, but we cannot say they would have caught a problem.

Cost: about 1.6 million new input and 0.34 million output tokens across the four AI agents, plus about 244 million cached reads. The referee's own use is not counted, and the figure includes restarts, so treat it as an upper bound. The study ran over two calendar days (26 to 27 October), after several days of preparation. Compiling and checking the report took a further two days.

Two models, compared

Wider coverage. Of 19 groups of claims that our referee aligned across the two sets, 9 were tested only by Codex and 3 only by Claude. One of those 3 is a property the tester chose itself, not a claim of the guide. Codex covered 16 of the 19 groups and Claude 10, and Codex wrote twice as many checks, so the gain from two models looks small and is unproven until a second run repeats it.

Mistakes caught before they counted. 17 checks asserted more than the guide promises (11 in Claude's draft, 6 in Codex's), caught by the other model's review. All 17 challenges were upheld and fixed before the seal; none was rejected. They came from only 5 distinct mistakes, because most were the same mistake repeated across checks.

A second reader spots what the first missed. The Codex writer noticed that one Claude check counts a value exactly on a threshold as passing, where the guide covers only values above it. The guide's own example has no value near it, so it most likely did not change the result.

This is one run per model on one library. A model difference is an anecdote until a second run repeats it.

The 4 reported differences neither set tested

An earlier case study of the same library, in which one AI model wrote the checks and a second reviewed them (onedroid.ai/case-study-code-vs-user-guide), reported 4 differences between the code and its guide (listed as three groups on that page). We did not reproduce them here, and neither set tested any of them. It does not look like a question of what was checked: the earlier

study was also a check against the guide, and it reported them. What differed appears to be how deep the checks went.

- **The solution after a user stop:** every stop check in both sets reads only the routine's status flag, never the solution or the call counts afterwards.
- **Call counts on the stop path:** Codex's five count checks ran a normal solve and compared the counts. Claude had no count check.
- **Counts together with printing:** the count checks switched printing off.
- **Exact machine constants:** Claude's check only tested that the constants are in a sensible order and that 1 plus the machine precision exceeds 1; it did not test their exact values. Codex had no check on them.

This is a reading of what the sets contain. We did not test whether a longer brief would have found them. Next time we would tell testers to check each promise on every path it covers.

What held

- The guide's ten solver worked examples reproduced in each set, within a stated numerical tolerance (Claude's eleventh example, for a checking tool, is checked by category counts).
- Improper input returned the documented status values in both sets: 12 cases in Claude's, 10 checks in Codex's.
- Evaluation-limit exits and a user stop returned the documented status values.
- A basic helper returned the right value, to seven digits, for a small test vector.

The test boundary: what this result does not prove

Not proven	Why
That the library has no differences from its guide	Four differences reported by an earlier study were tested by neither set.
That one model beats the other	One run each, on one library, with Claude at low reasoning effort and Codex's settings not exposed.
That the checks would catch an unknown difference	The broken copy was designed after both sets were final, and its reach was measured against them before use. It shows the checks can fail, not that they would find something new.
That the 2 checks that stayed green can fail	Nor which edit each check actually saw.
Other machines, compilers, the running build	Not measured.
Independence	The same team wrote the checks, ran them and wrote this. The referee that ruled on the cross-review disputes was itself an AI session of the Claude family.

How it was checked

The sealed sets and certificates are held by OneDroid and are not published here, so an outside check starts with asking us for them. Each certificate was verified with the open-source verifier that ships with OneDroid Argus, against a public chain. Every number on this page comes from the study's records: the run records, the rulings, the agents' session logs and our findings log. Nine fresh AI reviews recounted the study report from those records, and further fresh AI reviewers read this page against the same records.

What we take from it

- **Make sure the checks can fail.** Our first broken copy had 3 edits, and no check turned red on it. We rebuilt it with 49 edits and a reach probe showed it was visible to 28 of 28 and 57 of 58 checks (in the runs, 27 of 28 and 57 of 58 turned red). Measure that reach before the testers deliver.
- **A pass is only as strong as the case tested.** For the four promises above, the checks that touched them covered only the simplest case, and on three of them one model had no check at all (Claude on the two count promises, Codex on the machine constants). Ask for every path a promise covers, not just the first.
- **A second model is a useful reviewer.** Each model found over-strict expectations in the other's draft that its author did not see.
- **Count causes, not rows.** "17 false alarms" sounded like a big gap between the models. It was 5 mistakes. We now label repeated rulings with one cause so the count is computed, not guessed.
- **Our own process broke in useful ways.** Our notes on the study kit run to 29 entries. Two examples: our referee program sealed both sets before the final broken-copy runs, which was the wrong order (the sealed sets were unaffected, and the report says so), and two writer agents shut themselves down after six idle hours and never saw their task. Both are addressed in the kit.
- **A report needs readers who did not write it.** Nine fresh AI reviews of this study's report, each followed by a recount from the raw records, fixed wrong dates, miscounts and overstatements. The last-but-one reading still found a statement that was wrong for 8 of 10 checks. Re-check after every fix, with a fresh reader.

Third-party names belong to their owners. Claude is a product of Anthropic and Codex of OpenAI. OneDroid is not affiliated with either, or with the authors of the library tested. OneDroid Argus is built by Providentia Worldwide. Questions: michal@onedroid.ai. How it works: onedroid.ai/argus.