OneDro1d

CASE STUDY

Three ways to test code you did not write

A long-lived scientific program that nobody on our team wrote or understood, and three questions asked without reading its source first. Each answer came from something else to compare with: the manual, an earlier release, or the original program.

Case study. An experiment by our team. Every run had a written prediction beforehand.

In short

- We took a long-lived scientific program that nobody on our team wrote or understood, and asked three questions without reading its source first.
- OneDroid Argus doesn't need to know what the code is meant to do. Each question got its answer from somewhere else: the manual, an earlier release, or the original program.
- **Against the manual:** places where the code and its manual disagree. Most mean the manual is out of date. The rest are wrong behaviour, and some of those end in a crash or a runtime error.
- **Against an earlier release:** no regression over these checks, and one old run that exited cleanly with a plausible file and a main number well off.
- **An AI rewrite against the original:** the checks from the manual passed nearly all of it. The differences, most of them wrong numbers with a clean exit, only showed up against the original.

1. Does the code do what its manual says?

Agents that read only the manual wrote requirements, each quoting the manual with a page number, and a script checked that every quote was on that page. Those became checks against the program as it ships.

A red check wasn't a finding yet. A person read each red against the source, and our own mistakes were fixed and not counted.

Result: places where the code and its manual disagree. Most mean the manual is out of date. The rest are wrong behaviour, and some of those end in a crash or a runtime error.

2. Did anything change between two releases?

The same checks ran against the current release and an older one, side by side.

Result: nothing went red on the new release only, so no regression over those checks. The checks that went red on the old release only had each been predicted from the project's own release notes and reference files.

One result stands out. On one input, the old release exited cleanly, reported no error and wrote a plausible file, and its main number was well off. A test that only looked at the exit code would have called that run good.

3. Does an AI rewrite do what the original does?

A rewrite of part of the program in another language got the same requests as the original.

Result: several kinds of difference, most of them wrong numbers with a clean exit. Some totals were many times too high, and some values came out negative where that makes no sense.

The checks written from the manual alone passed nearly all of the rewrite. The differences only showed up when the same request went to the original as well.

What to take from it

- You can test code you don't understand, if you have something else to compare it with: its manual, an earlier release, or the original.
- Exit code 0 is not a result. Check the numbers.
- A rewrite can pass the checks from the docs and still give wrong answers.
- A person still decides what a red means.

What this does not show

Not shown	Why
Anything the manual does not say	A check written from the manual only covers what the manual says.
The science	The checks say nothing about whether the program's science is right.
A sealed result	Every run had a written prediction beforehand and matched it. The runs weren't sealed.

The program, its authors and its owners are not named, and the details that would identify them are left out on purpose. OneDroid Argus is built by Providentia Worldwide. Questions: michal@onedroid.ai.