WHITE PAPER · CASE STUDY

AI agent needs a process to build a commercial application.

What happened when two AI teams reviewed the same 374,000-line commercial codebase: one working alone, one working inside NoMoreHappyPath.

> **The short version.** I gave the same job, in the same words, to Claude and to NoMoreHappyPath (www.nomorehappypath.com) : read a large, real application and find the problems that could stop it from being used commercially. Both needs to use reviewers before submitting the findings. An impartial judge from Open AI checked every claim against the original code.
>
> Graded only on **real, verified problems found**, with presentation ignored, NoMoreHappyPath found **33** and Claude found **10**. The two overlapped on only 6. Both judging runs preferred NoMoreHappyPath.
>
> The first judge gave Claude a little more credit on the presentation, and that was the real happy path that Claude took. Good presentation, weaker content.

1. The problem

An AI that writes code and then checks its own work, or even calls a different AI to review it, will always try to direct the reviewer towards what it knows. It reports success on the path it walked and says little about the paths it did not. For a review that decides whether software is safe to put in front of paying customers, that is the wrong failure to have. Even with strong directives and skills, the AI will drift and take you to a happy path.

NoMoreHappyPath (NMHP) operates on a single principle: nothing is marked complete without rigorous, independent verification. One AI builds the code, a reviewer AI from a completely different vendor tests it, and a third AI acting as the CTO steps in only when human-like judgment is required. By enforcing a strict adversarial protocol governing how these agents interact, NMHP ensures that "DONE" actually means “DONE”.

This paper tests that idea on a hard case.

2. The test

The subject: Acremuse

Acremuse is a Texas property risk analysis service for small investors and law offices. It is a cloud application. Many customers and their staff can use it at once, and each runs many analyses in the background. Claude built it with Codex as an ordinary AI-assisted project. **It was not built with NoMoreHappyPath**, and neither team had seen it before.

≈374,000	1,070	≈11,400	3,067
lines of program code	source files (Python and TypeScript)	functions and methods	automated tests

Part	Technology	Size
Back end: licensing, billing, analysis engine, job queue, AI gateway	Python, FastAPI	≈175,000 lines; about 210 endpoints
Back-end tests and migrations	Python, Alembic	≈107,000 lines; 57 migrations
Customer web app	TypeScript, React, Vite, Tailwind, TanStack Query, Zustand, MapLibre	≈59,600 lines
Admin and simulation web apps	TypeScript, React, Vite	≈11,000 lines
Database	PostgreSQL via SQLModel / SQLAlchemy	62 tables
Other services	Redis, Stripe, OpenAI and Anthropic SDKs, LangGraph	—

The count is the project's saved snapshot (844 documentation files and about 332,000 lines of JSON test data are not counted as code). TypeScript function counts are by pattern and approximate.

The two teams

Team	Setup
Claude	One Claude session working alone on its own copy of the code. It also received the role statement and the report fields that NoMoreHappyPath supplies to its own agents, so neither side had an advantage in format.
NoMoreHappyPath	A Claude builder (Sonnet 5, medium), a Codex reviewer from another company, and a Claude CTO, on its own copy of the code.

Both were given the same project description and the same rules, in the owner's own words. Each worked on a separate, identical copy of the code and never saw the other's output.

The judging

An impartial judge from a different company (OpenAI Codex, high reasoning effort) received the two reports blind, renamed with neutral letters in random order, plus a clean copy of the original code. It was told to verify claims against the code, to trace each one far enough to see whether a guard already prevents it, and not to reward length or confidence. It ran twice, with different rules:

- **Judging run 1** weighed everything: findings, severity, evidence, and presentability.
- **Judging run 2** ignored format, presentability, and length completely. It counted only problems that were true in the code and mattered to the application, using this scale: **Level 1** (the app or portion of the app cannot work, customer data is exposed, a security leak, or money is wrongly taken), **Level 2** (an important function fails or gives wrong results, no way around it), **Level 3** (an annoyance with a simple workaround). Ideas that are not bugs scored nothing, and false claims counted against the report.

3. Results

Judging run 2: real problems found



Verified real problems (the judge's own levels)	NoMoreHappyPath	Claude
Level 1: cannot work, data exposed, a security leak, or money wrongly taken	6	5
Level 2: an important function fails or gives wrong results, no way around it	22	4
Level 3: an annoyance with a simple workaround	5	1
Total real problems	**33**	**10**
False claims (contradicted by the code)	7	2

Six problems were found by both. NoMoreHappyPath found 27 that Claude did not; Claude found 4 that NoMoreHappyPath did not. On Levels 1 and 2 alone, the count is 28 to 9. The judge's own words: “I am highly confident” that NoMoreHappyPath found more.

Judging run 1: everything weighed

The first judge, which considered presentation, chose NoMoreHappyPath “by a moderate margin, after correcting its priorities and removing the unsupported claims”.

Time and cost

From the owner's go-ahead to the reviewer's pass, NoMoreHappyPath took about 74 minutes, including time lost while it found and worked around a defect in the product itself. The Python version NMHP was working on was 3.9, the OS version, and it didn't have pytest installed, which was needed to verify some claims. After 15 minutes of blockage, the CTO stepped in and told the agent to create a venv with Python 3.13, install the corresponding pytest, and move the project along. NMHP CTO reacts when 15 minutes pass without movement from agents.

To be fair to Claude, Claude did create a venv from the get-go.

NMHP record shows about **$24** of API-equivalent spend, mostly by the builder and the reviewer. Claude finished in a few minutes; its cost was not instrumented. These figures are a yardstick, not a bill.

NMHP takes over an hour and costs twenty-four bucks. But for a 374k-line codebase, finding 23 more real issues overall that vanilla Claude missed makes $24 a massive win compared to human audit rates.

4. This is a case study, not a benchmark

> One codebase, one run each, one judge model run twice. The conclusion is a strong signal about this case, not a measured rate across all software.
>
> It confirms my idea: **an AI agent without a real process is not yet capable of building a commercially ready application, even with a frontier model and reviews that are initiated by the same agent.**

Method summary: same words and rules for both teams; separate identical copies; blind, randomised labels; an impartial judge from a different company, run twice with different grading rules; every figure above is quoted from the judges' written reports, which are kept with this paper.