How My AI-Assisted Coding Productivity Improved 48× - 70×

A 16-week experiment in AI-assisted engineering — measured, not guessed.

Author : Apurv Singh Baghel
Feb 15, 2026

Assumptions (Transparency)

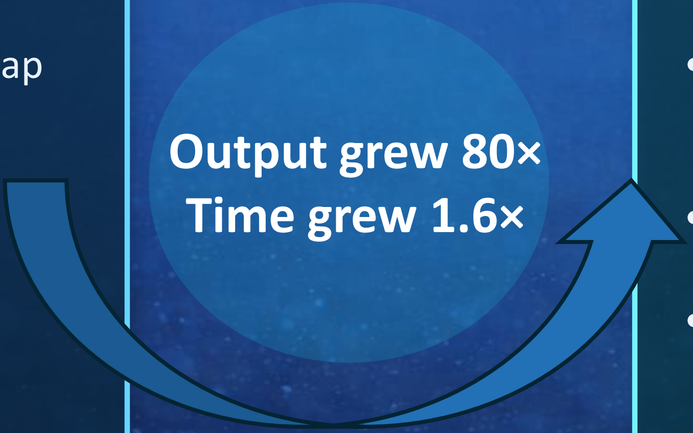
- KLOC (Kilo Lines of Code) = only human-verified, production-ready tested code
- Metrics exclude any auto generated scaffoldings for both POC and Full Application
- 1 message = average 100 tokens (some lower, some higher)
- Assuming 40 hours per week (variable distribution – 24 hours each weekend and 16 hours over each working week (night)
- Metrics measure AI-assisted productivity, not traditional engineering efficiency

The Productivity Jump: POC → Full App

The numbers alone don’t explain the jump — the workflow did

POC

- Python backend
- HTML/JS/CSS frontend
- 3rd Party tools integrated: OpenStreetMap + OpenRouteService
- Timeline - 6 weeks ~ 240 hours
- Output:
  - KLOC: 2.8
  - No. of files: 26
- Total Messages sent: 990
- Approx Tokens consumed: 99,000



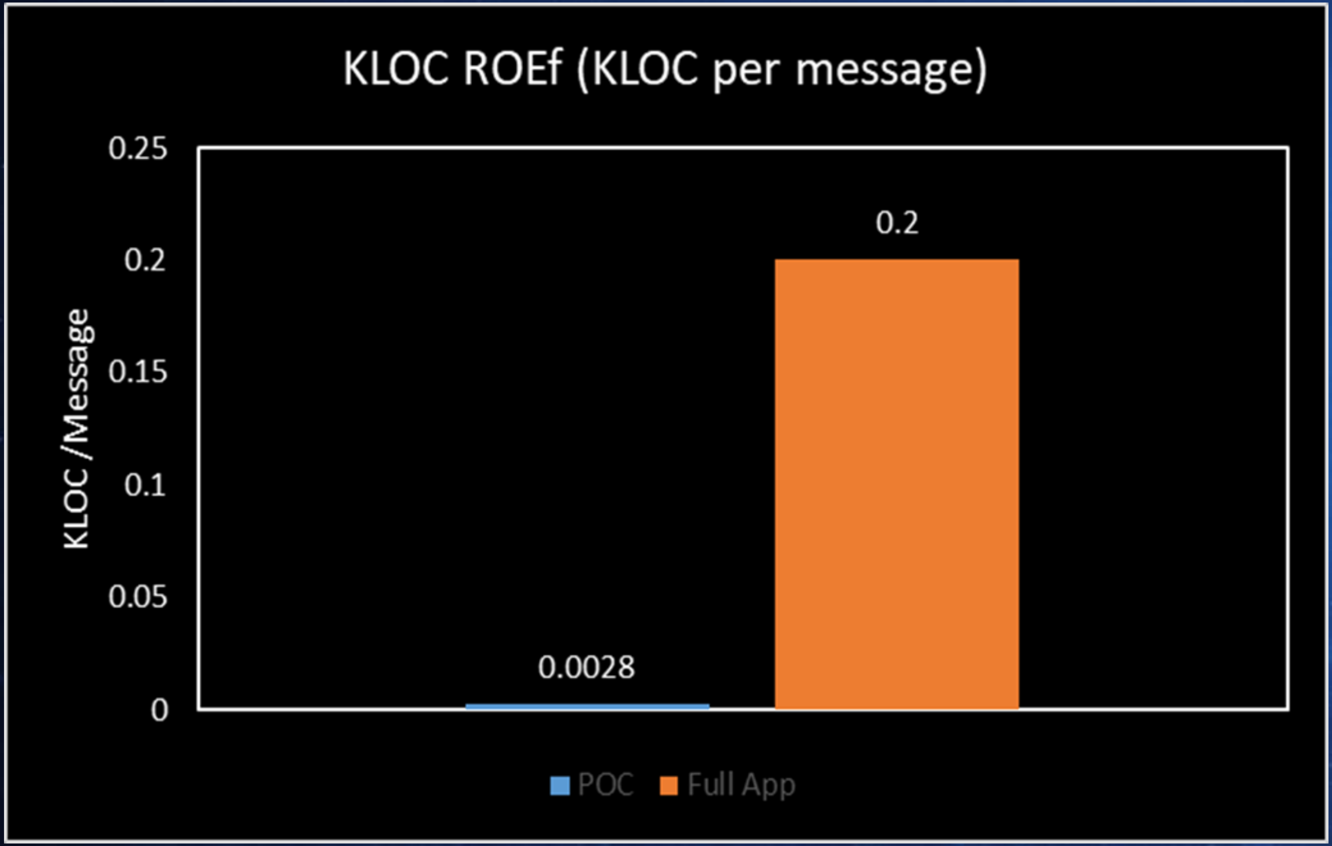
Output grew 80×
Time grew 1.6×

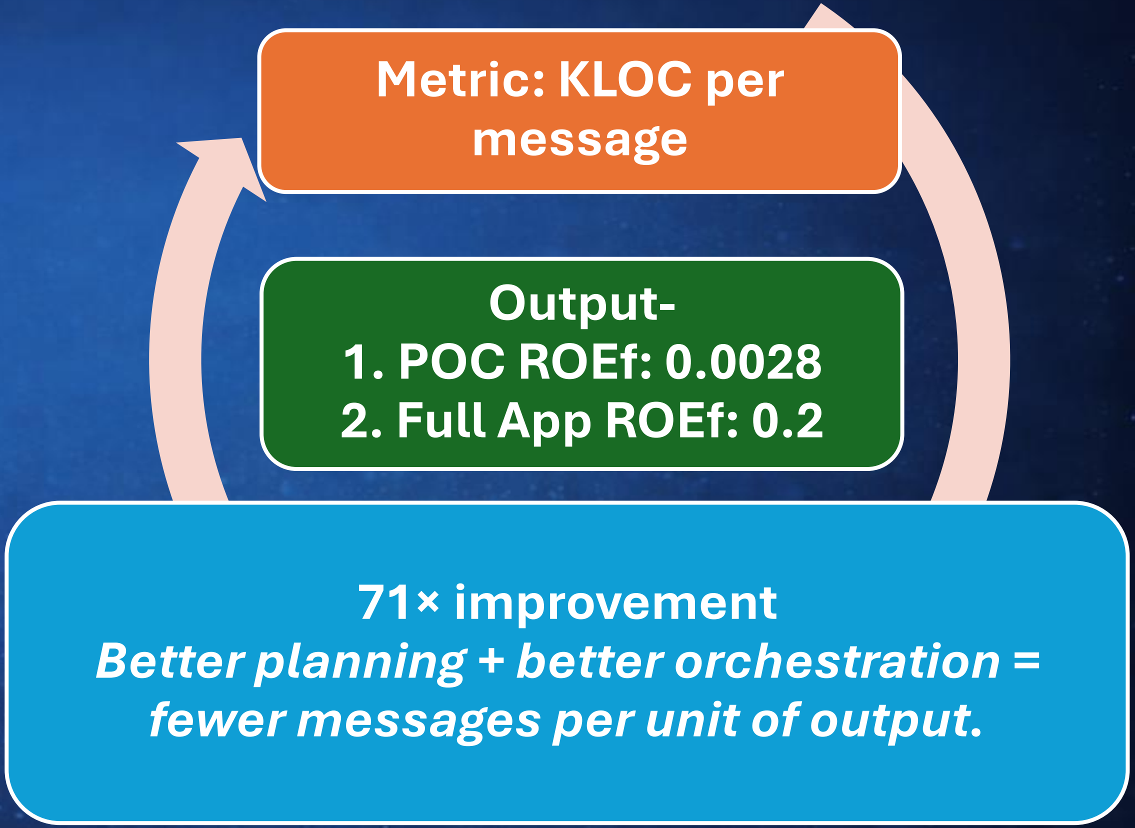
Full Application

- Flutter mobile front end (android)
- Python backend + API
- 3rd Party tools integrated: OpenStreetMap + OpenRouteService
- Timeline - 10 weeks ~ 400 hours
- Output:
  - KLOC: 225
  - No. of files: 993
- Total Messages sent: 1,125
- Approx Tokens consumed: 112,500

Averages over time

KLOC ROEf (Return on Effort)





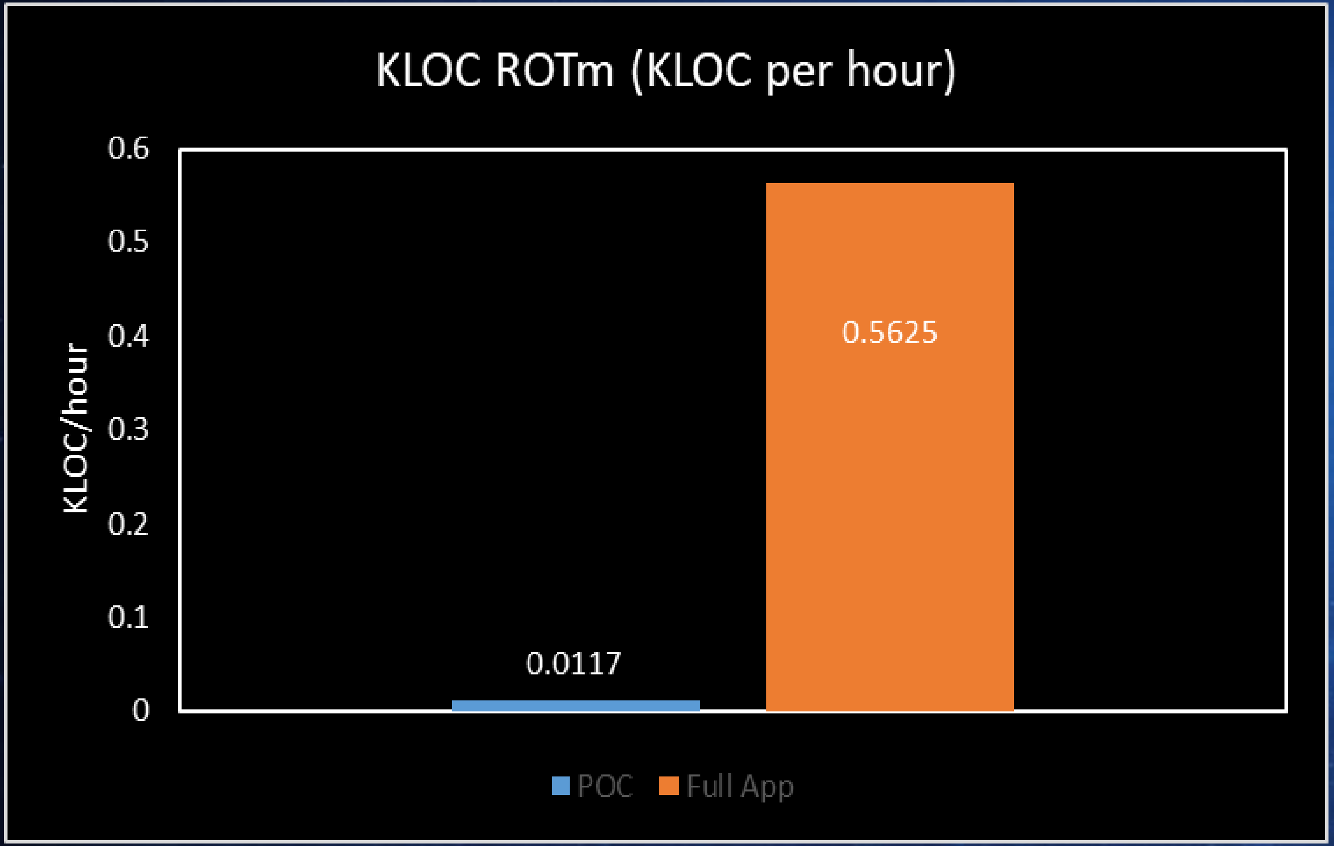
Metric: KLOC per message

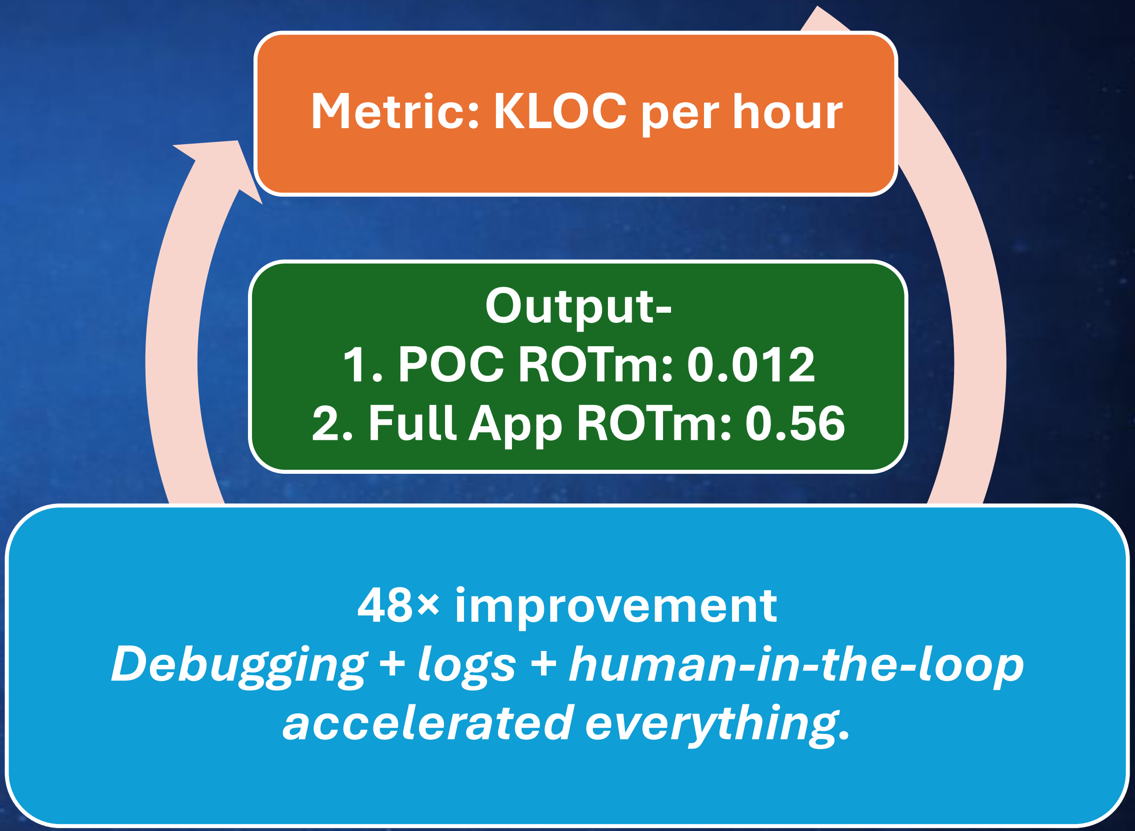
Output-
1. POC ROEf: 0.0028
2. Full App ROEf: 0.2

71× improvement
Better planning + better orchestration = fewer messages per unit of output.

Averages over time

KLOC ROTm (Return over Time)





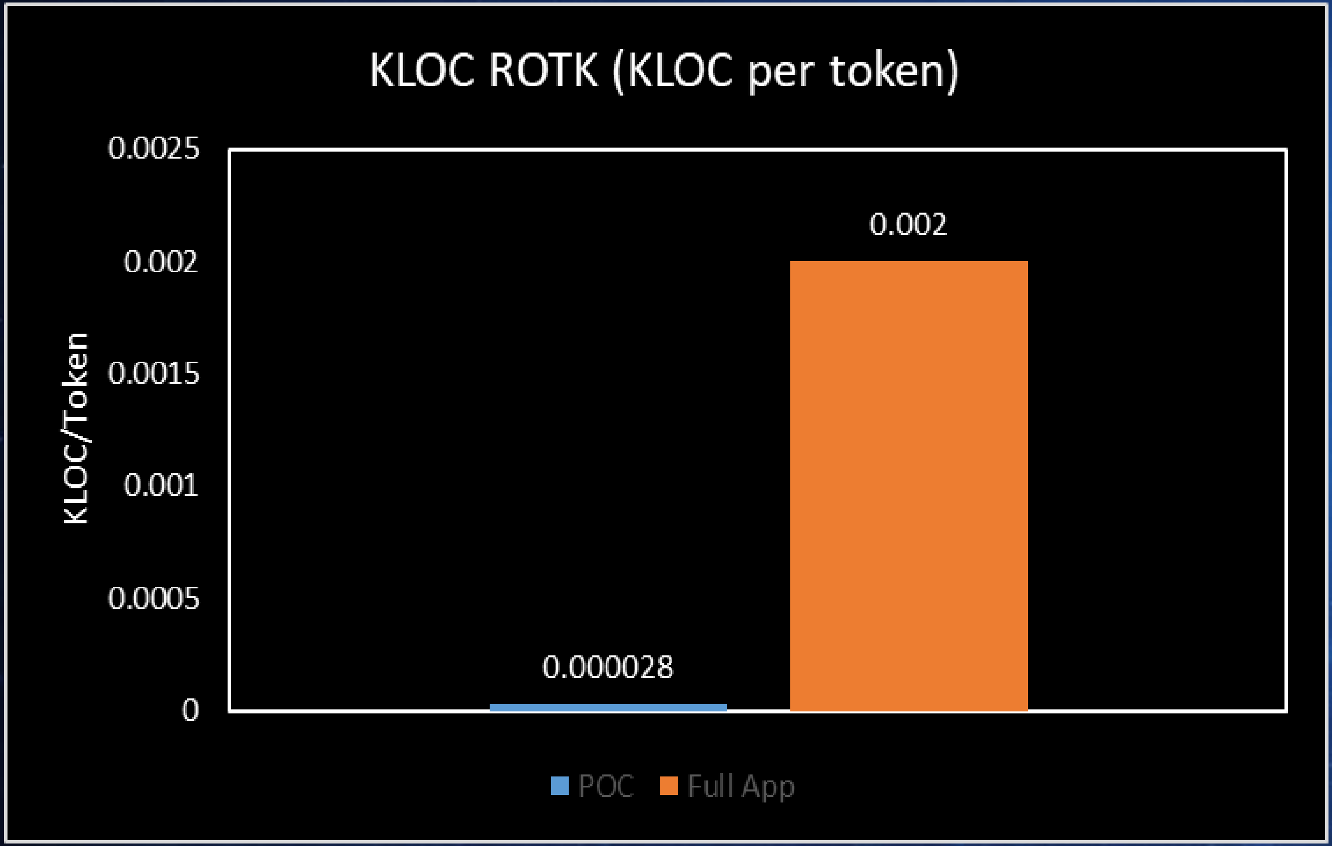
Metric: KLOC per hour

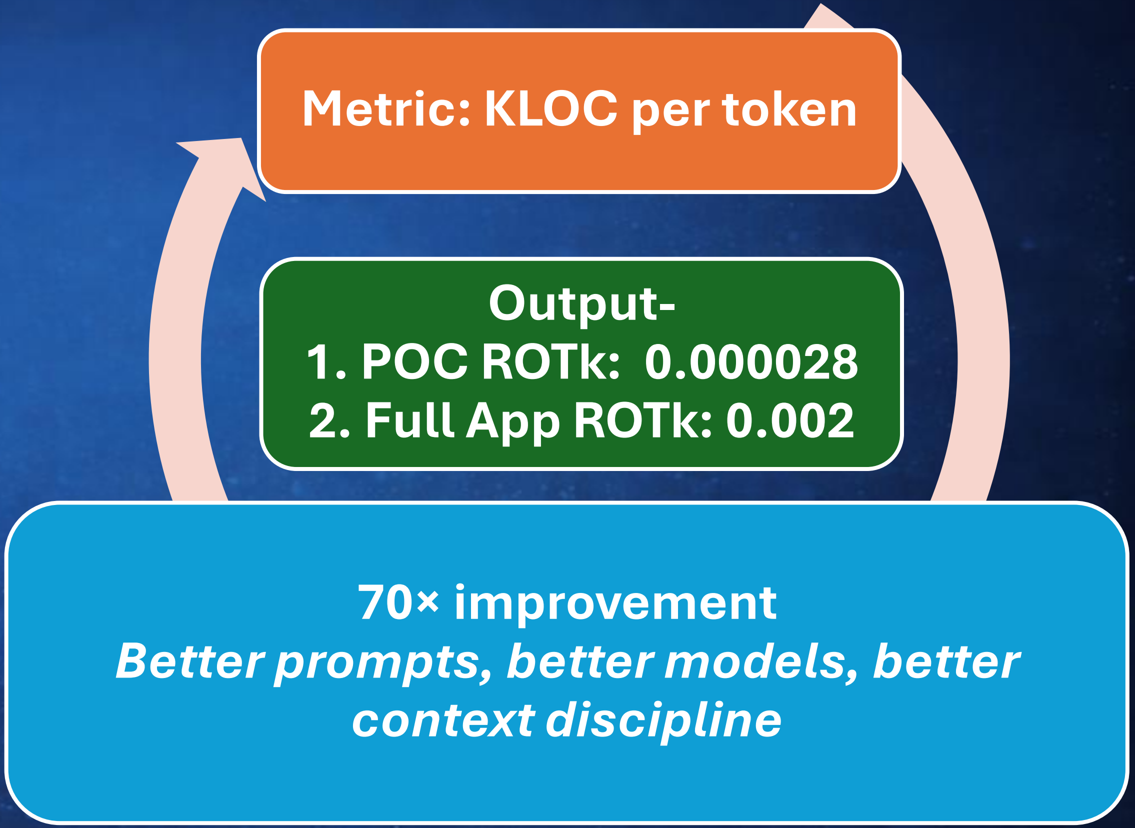
Output-
1. POC ROTm: 0.012
2. Full App ROTm: 0.56

48× improvement
Debugging + logs + human-in-the-loop accelerated everything.

Averages over time

KLOC ROTk (Return on Tokens)





Metric: KLOC per token

Output-
1. POC ROTk: 0.000028
2. Full App ROTk: 0.002

70× improvement
Better prompts, better models, better context discipline

What Changed

1. Models evolved
GPT-5.2 + Claude Sonnet 4.5 changed the game.

2. The ecosystem evolved
Agent-native IDEs (like Antigravity) reduced friction.

3. I learned to plan better
Clear BRDs, PRDs, and architecture → clear code.

4. I changed the workflow (including more human intervention)
Gemini (plan) → GPT (sanity check) → Claude (Code) → GPT (sanity check) → Me (test/debug/feedback)

5. I debugged like an engineer (having a coding background helped) : Logs, breakpoints, targeted fixes — not "guess from output log".

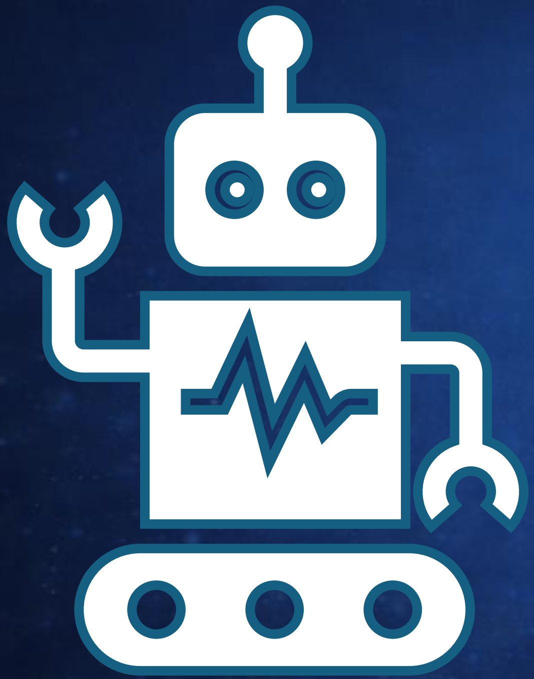
Why the metrics seem low?

Data points reflect time-averaged estimates and may vary by phase.

Metrics like ROTk, ROEf and ROTm may appear low because tokens are spent not only on code generation, but also on planning, debugging, testing, and iterative refinement. These numbers are indicative of real engineering workflows rather than synthetic benchmarks.

Think of tokens, messages & time like calories - Not all calories go into muscle - most go into the work your body does behind the scenes.

Similarly, most tokens/messages & time go into reasoning, debugging, and planning - not just code output!



AI doesn’t replace product managers/ engineers.

It augments teams and helps you plan, design, experiment and orchestrate better!