AI Coding Agents in an Economics Team

Alex Guglielmone Nemi

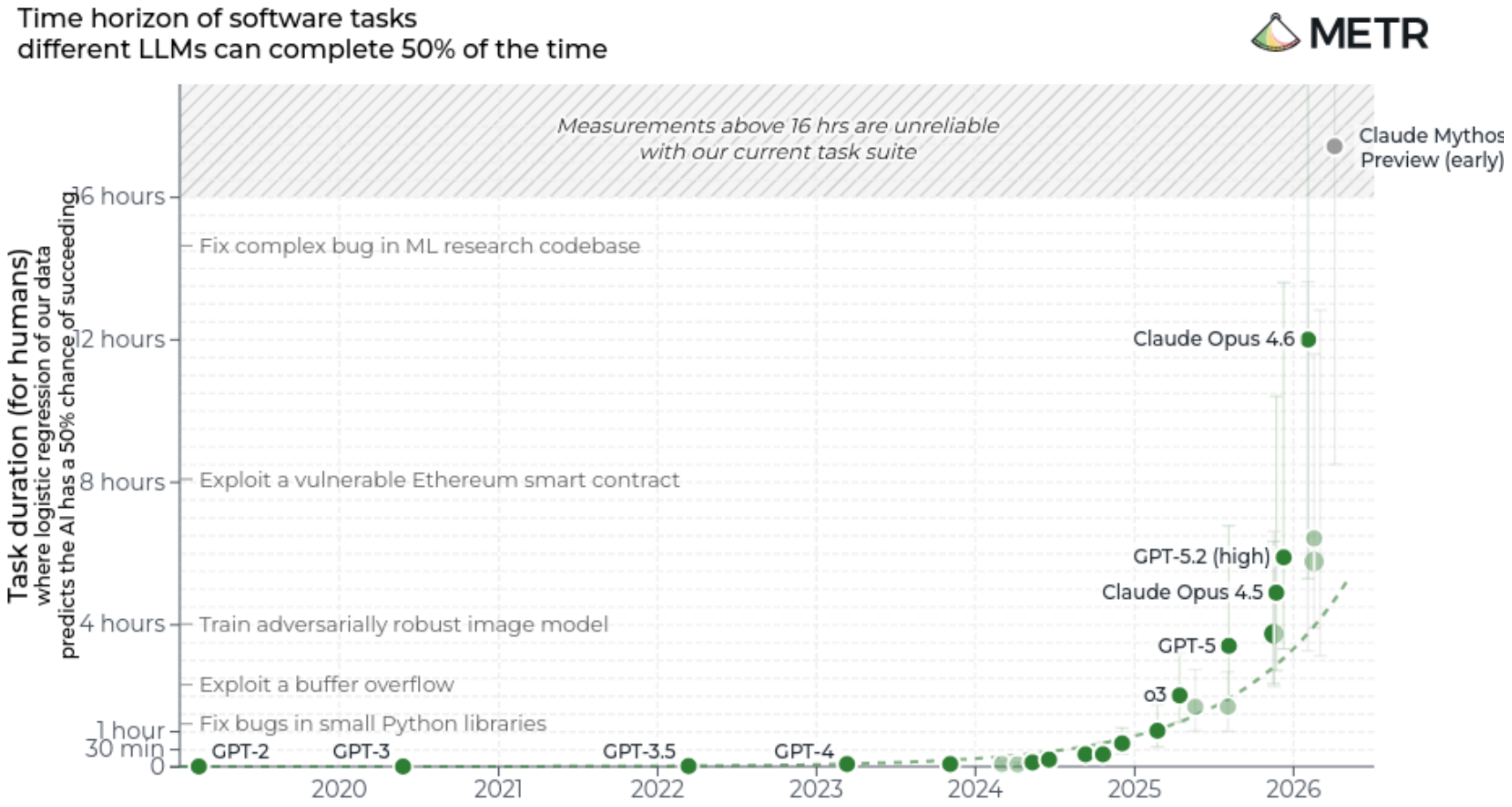
2026-05-29

How the team works

- Alex Guglielmone Nemi
- Economic Decision Science, Amazon — Engineering Lead
- Cross-functional team: Economists + Engineers

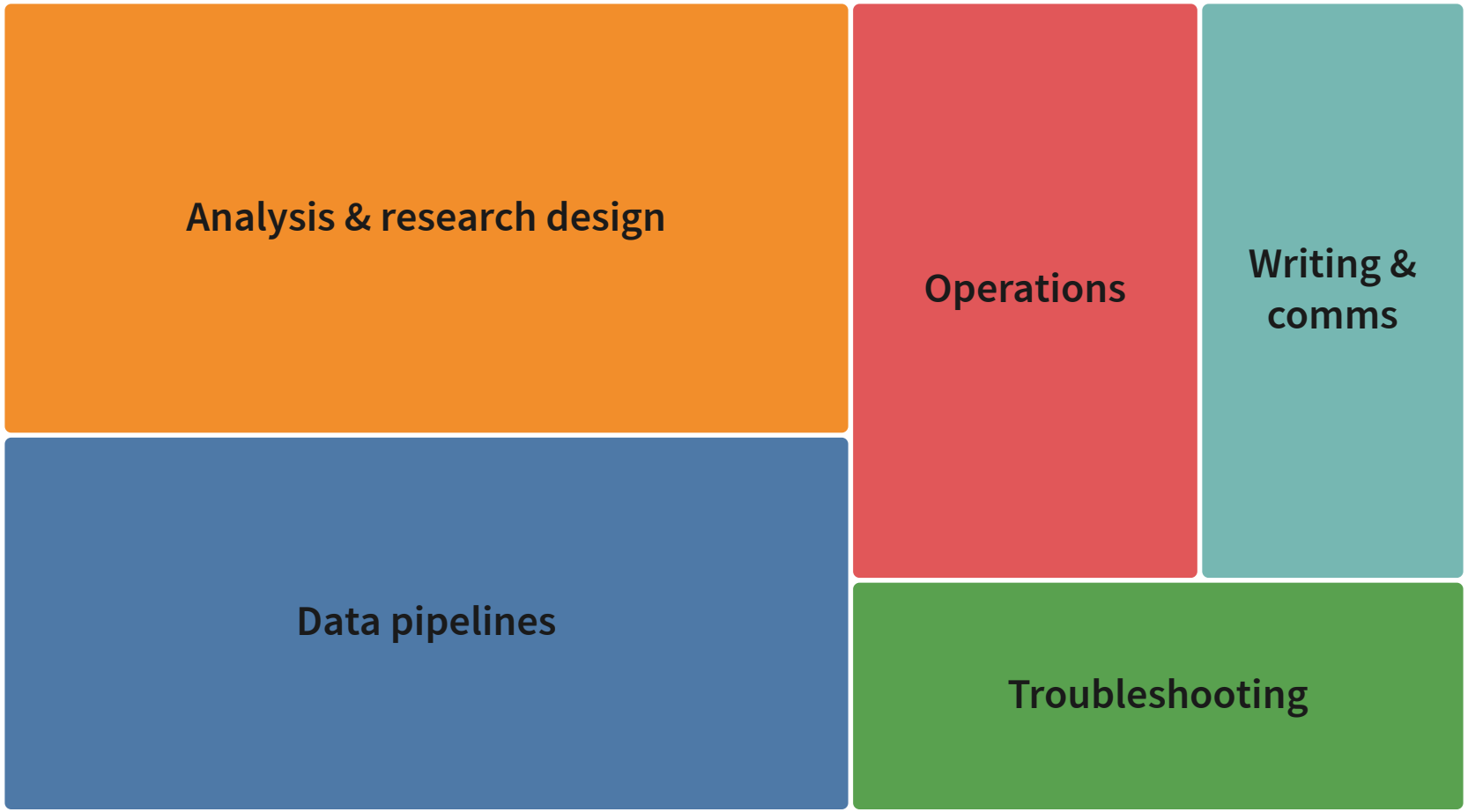
- Goal: economists deliver without engineering bottlenecks
- Fast iteration, minimal infrastructure
- No tradeoff between exploration and production

Late-2025 quality jump



Source: METR Task Horizons · metr.org/time-horizons

How we leverage Coding Agents



1. “Explain this model to me”

- “Explain this model to me”
- Agent explains methodology step by step
- Economist spots: “that’s counting the wrong window”
  - *(or: wrong benchmark scope, double-counting, extensive vs intensive confusion…)*
- “Fix it” → corrected pipeline in 15 minutes

2. “Help me analyze this policy”

- Economist has a policy question, no design yet
- Dialogue: what’s the right approach? assumptions? what could go wrong?
- Power analysis, clustering, spillovers. Methodology through conversation
- Complete research design

3. “Help me respond to this challenge”

- Leadership / stakeholders challenge methodology in a meeting
- Economist uses agent: pull the data, produce the chart, draft the response
- Prove it instantly instead of deferring it.
- Real data backing the argument without effort or time wasted

Quick clarification

- We also leverage AI-powered economics workflows (synthentic conjoints, model-driven scoring systems, text extraction and harmonization for assortment models)
- Coding agents are the focus: the everyday working interface
- Broad, open-ended, close to the production process
- What I’ll discuss next applies to both

What economists never delegate

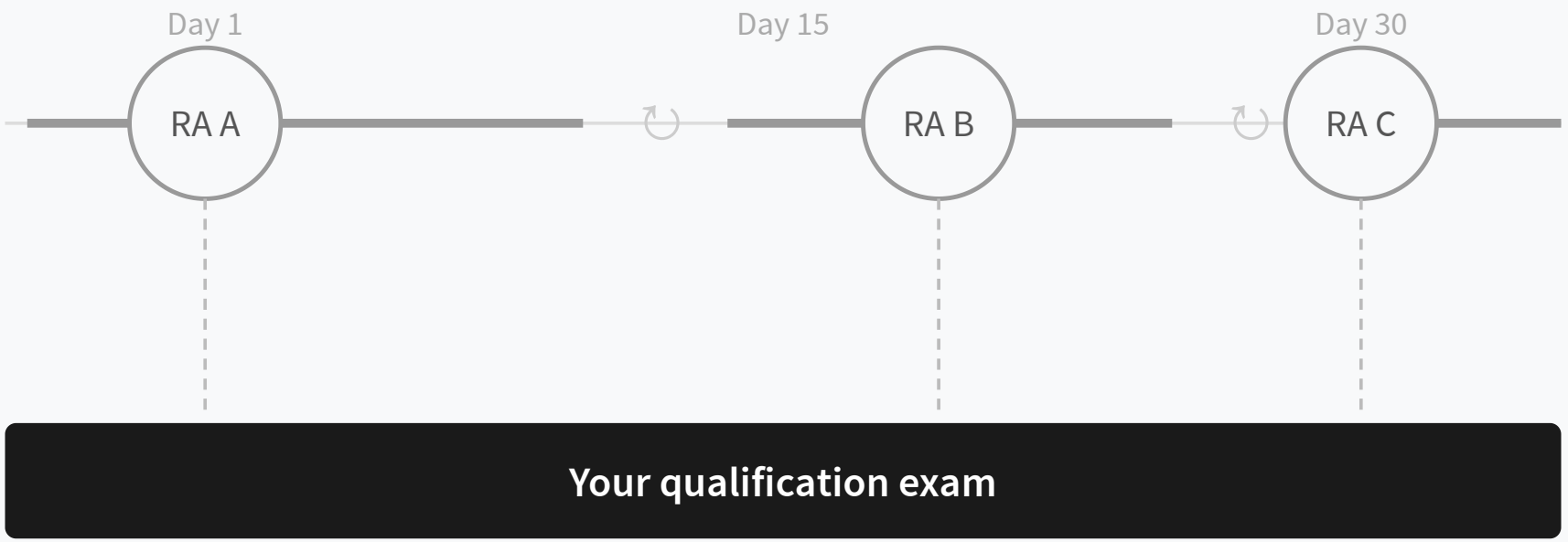
1. **Methodology selection** — *Which estimator. Which design.*
2. **Causal identification** — *How to isolate the effect.*
3. **Economic interpretation** — *Whether results make sense.*
4. **Stakeholder judgment** — *Tone, framing, politics.*
5. **Research taste** — *What questions matter.*

Pain points

- Trust: “How do we know it’s right?”
- Reliability: “It worked yesterday, not today”
- Blast radius: “What happens if it goes rogue?”
- Observability: “Can’t explain what happened”
- Tool churn: “Everything keeps changing”
- Cost: “Fast iteration without runaway spend”
- Speed: “Too slow to explore effectively”

Control

The RA analogy



persists across all replacements

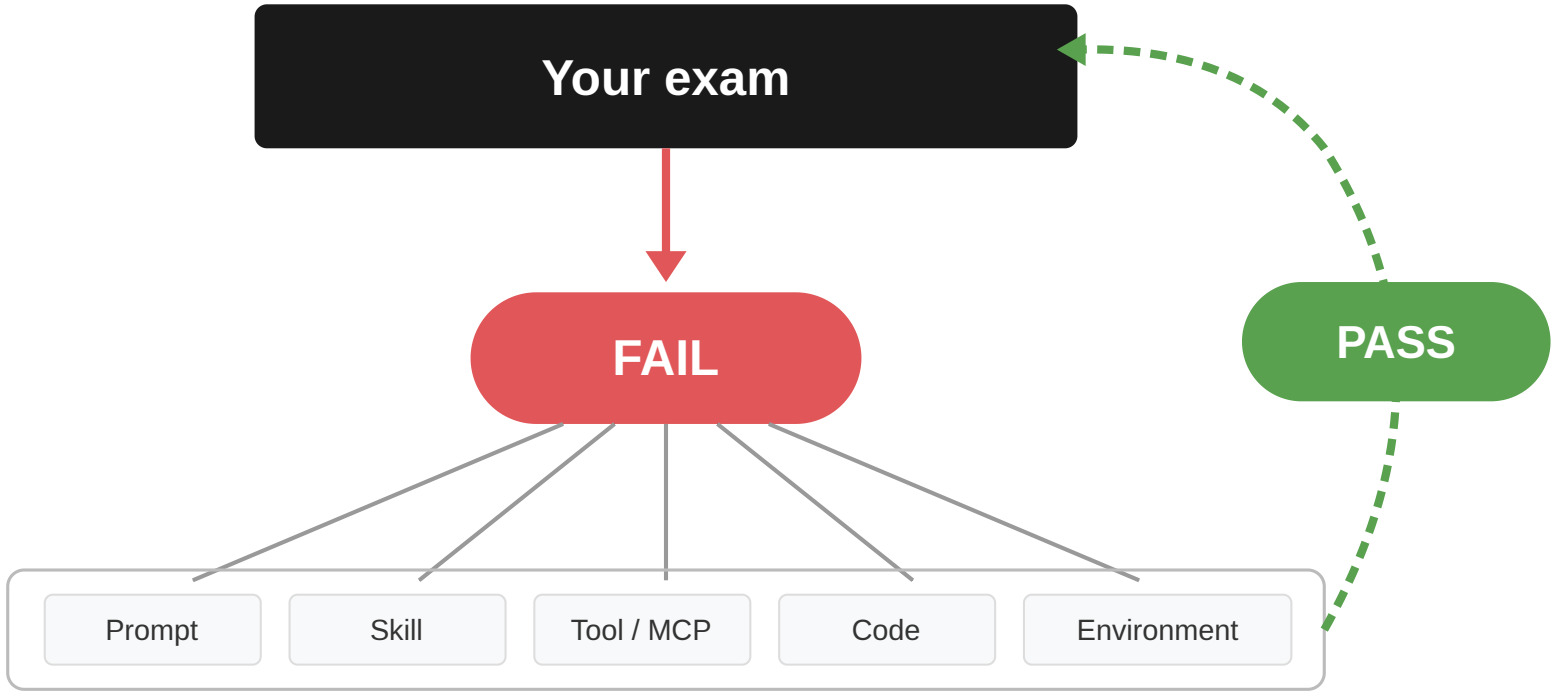
Same qualification. Different agent.

What an exam looks like

```yaml
name: econ-model-runner
questions:
  - name: runs-scenario
    input: Run scenario for client ABC, target €100k. Scope: EU5
    intention: Agent picks the right skill and executes with correct params
    assert:
      - type: tool-called
        value: run-model --client ABC --ask 100000 --scope EU5

  - name: plans-batch-safely
    input: Run all clients under €300k, all markets, 20% above last cycle
    intention: Agent should plan before executing a large ambiguous batch
    assert:
      - type: plan-contains
        judge:
          - clarifies 'last cycle' time range
          - identifies source table
          - smoke tests one case before full run
          - estimates total effort
```

The feedback loop

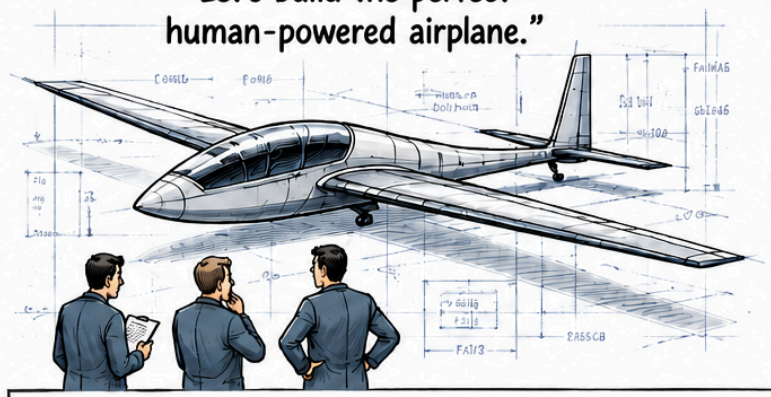


Human or agent iterates against the same exam

HE DIDN'T SOLVE THE AIRPLANE PROBLEM FIRST.

TRADITIONAL TEAM

"Let's build the perfect human-powered airplane."



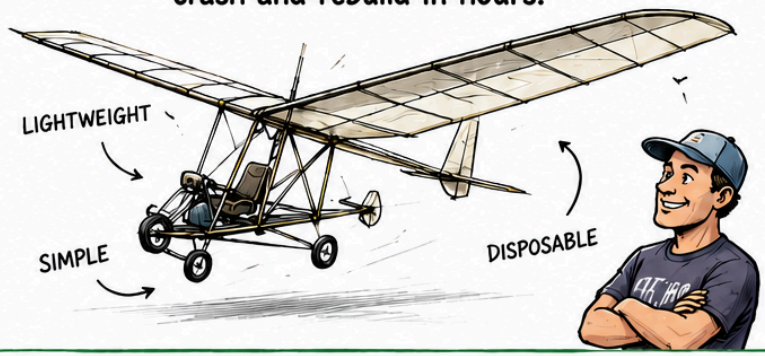
BUILD 12 MONTHS → TEST 30 SECONDS → CRASH → REPAIR 12 MONTHS → RETRY ...

1 LEARNING CYCLE PER YEAR

VS.

Paul MacCready

"Let's build a plane we can crash and rebuild in hours."



BUILD 2 HOURS → TEST IMMEDIATELY → CRASH → REBUILD 2 HOURS → RETRY ...

20+ LEARNING CYCLES PER DAY

HUMAN-POWERED FLIGHT ACHIEVED!

THE BREAKTHROUGH WAS SHORTENING THE LEARNING LOOP.

Operational realities

- Slow tools kill fast iteration
- Context management is still a challenge
- Security boundaries must live outside the LLM
- Be a good boss: delegate, but own outcomes
- The agent needs to work where the economist works

Takeaways

1. **Sketch the workflow early**. Try the UX.
2. **Break long tasks** into **testable blocks**
3. Use **code for stable steps**; use **LLMs for reasoning** and tool calling
4. Use **Exam First Iteration** to keep **control**

Questions?

https://linktr.ee/alex.guglielmone.nemi

Writings · Github · LinkedIn