ACCELERATE **CHICAGO**

Craft software with confidence in 2026

GOTO ACCELERATE CHICAGO 2026

The Validation-First Loop

How to ship production code with AI coding assistants

Cole Medin / Dynamous

THE PROBLEM

We accept the agent's first solution. It looks done, so we push it aside.



THE SHIFT AS AI IMPROVES

Agents stopped writing bad code. They write good code that misses the point.

Misalignment happens at intent time, not code time.

WHY CHECKING HARDER ISN'T THE FIX

None of these fix the root of the problem

Review

Runs after the agent already handed you the code

Tests

Passes the cases you or the agent imagined, not the intent you meant

Guardrails

Constrains how it codes, never defines what you wanted

THE PRODUCTIVITY MIRAGE

You feel faster but you're actually slower

19%

slower on real tasks, even while feeling faster (METR)

+66%

more throughput per team (DORA)

+242%

more incidents per PR (DORA)

PART 2

Work backwards

THE HIGHEST LEVERAGE YOU HAVE

Define done before you build

Classic TDD

Write the failing test first

Red, then green, then refactor

Defining done forces you to understand the task

TDD for agents

You + the agent write success criteria first

Then define the validation in natural language

Only then does the agent implement



A validation-first plan eliminates the nasty assumptions your coding agent makes.

Your number one job up front is to reduce its assumptions

BEFORE THE AGENT WRITES A LINE

What a validation-first plan pins down

Success criteria

The exact behaviors that mean done. Measurable, not vibes.

Assumptions to kill

Name what must never happen, so the agent cannot guess wrong.

The validation

The check that proves each criterion, written before the code.

PART 3 - THE LOOP ON A REAL APP

The hardest thing to validate is a game

A small browser tower-defense game: real-time, 60fps, visual, stateful.

We add one feature two ways: a frost tower that slows enemies.

A game is hard for an agent to test. Validate this, and you can validate anything.

Web apps, mobile, anything really is easy by comparison.



WHAT A “THIN TICKET” LEADS TO

Same ticket, a different tower every time

TypeScript

```
// One ticket, nine runs, nine different towers:
//   price:    $40  vs  $50
//   controls: keys 1/2 / C-F / S-F / buttons
//   state:    none / slowed / baseSpeed / ...

// Two runs wrote a speed mutation but never reset it
// Frozen enemies never “recovered” even after they got out of range
for (const t of towers)
  if (t.type === 'frost' && inRange(e, t))
    e.speed *= 0.5;   // never restored
```

The agent guesses everything you did not say, often the guesses do not match your intent, and sometimes they appear simply broken.

VALIDATION IS LAYERED

Four layers from static to user simulated testing

Static

Lint and type-check catch the obvious before anything runs

Unit

The contract's rules, run automatically on every change

End to end

The agent plays the game through the browser harness, like a user

IT IS NOT ABOUT THE TOOL

The contract is universal.
The tool is interchangeable.

Claude Code, Codex, Cursor, Gemini CLI. Any agent, any stack.

PART 4

Then it compounds

WHAT YOU DO WITH A CAUGHT MISS

Patch the bug, or fix the system

What to avoid

Patch the one bug

Move on to the next ticket

The same type of mistake comes back next week

Validation-first

Add a rule, a context doc, a tighter check

The “contract” gets stricter

That whole class of mistakes becomes much less common over time

WHY THIS KEEPS YOU IN CONTROL

You own the plan.
The agent owns the typing.

That is what staying in the driver's seat looks like at scale.

THE LOOP IN FOUR MOVES

Recap

Define

Make the agent define done first

Validate

It writes the check before the code

Build

Implement against the contract

Evolve

A miss becomes a rule, not just a fix

NO NEW TOOLING REQUIRED

Three things to try now

Make your agent write success criteria before any code

Make it write the test or check before the implementation

When something slips, write a rule. Do not just fix the bug

Build the contract, not better prompts

- YouTube and Dynamous: the full system, end to end
- Dynamous community plus the Agentic Engineering course
- Screenshot this, find me after, let's talk

Cole Medin / https://dynamous.ai / cole@dynamous.ai

QUESTIONS?

ASK ME IN THE APP

Download on the App Store

GET IT ON Google Play

PLEASE RATE MY SESSION