ACCELERATE **CHICAGO**

Craft software with confidence in 2026

// speaker & sponsor

RIVET

Where I work

They pay my bills

rivet.work/careers



Phil Borel

Speaker

philborel.com

// detroit software developers

Detroit Software Developers

- → Monthly meeting
- → Software development focus
- → Technical Talks | Good conversation



detroitdevelopers.com

// intro

AI is an Amplifier

Harness Engineering for 3x Velocity

Phil Borel · GOTO Accelerate Chicago · June 2026

detroitdevelopers.com/talks/goto-accelerate-chicago

// act I — the goal

Act I · February 2026 · All-Hands

3x Product Development Velocity

BHAG - Big Hairy Audacious Goal

by Q1 2027

// how we got started

The BHAG gave us an excuse to take Agentic Dev seriously

- → **Learned** - read papers & blog posts; watched talks; listened to podcasts
- → **Convened** - started a weekly *AI Roundtable* where devs share what's working & what's not
- → **Experimented** - every engineer using Claude Code; open-door policy on tools & workflows; no caps on skills, CLAUDE.md updates, etc
- → **Planned** - created quarterly goals tied to outcomes to track progress against the BHAG

· *High-volume learning environment, not a slow-roll adoption process*

// what does 3x mean?

What does 3x even mean?

→ **Vibes** - execs, CS, sales, product feel like we shipped 3x the value

→ **Metrics** - DORA metrics (CFR, MTTR, DF), 3x PRs, bugs / time

→ **Outcomes** - 3x projects shipped; users get more value more quickly

- *All three matter. None are sufficient alone.*
- *Risk: 10x some metrics while seeing modest gains on outcomes*

→ **Increase velocity | reduce bugs** - Bugs / Release; PRs / Sprint; Projects / Quarter

// work in progress

A working case study.
Not a success story.
Yet.

Progress from Feb → June:

→ **~2x velocity increase** vs last year

much of the acceleration came from process changes, code refactors & quality initiatives - not from specific tools or Agentic Dev

→ **<1 incident per month** down from >1 per 2-week sprint

// act I take-away

Act I take-away:
Set a Goal

Vibes · Metrics · Outcomes

// act II

Act II

AI is an Amplifier

y = mx + b || Inputs → Outputs

// the research

AI is an amplifier.

→ **DORA 2025** (5,000 professionals): high-performing orgs see 50% fewer customer incidents. Struggling orgs: 2× more.

→ **Carnegie Mellon 2026** (807 open source repos): 281% increase in lines added month one. Month two: velocity gone, back to baseline. Complexity +41% permanently.

→ **In the wild** (Amazon retail): spike in outages → senior sign-off mandate.

· *"AI is moving organizations in different directions." - Laura Tacho, AWS*

// what AI amplifies

What does AI amplify?

- → Good architecture → maintainable code
- → Bad architecture → friction increases with every new feature
- → Good specs → accurate, testable implementations
- → Vague specs → confidently wrong code that passes tests you don't understand
- → Consistent patterns → idiomatic output on the first try
- → n divergent patterns → tech debt increases agentically

// the implication

Best practices matter *more* with AI, not less.

Agentic Dev has unlocked a renewed focus on quality

// keeping quality in check

4-Step Review Process

→ **1. Architectural review** - human, before code is written

→ **2. Automated review** - /review-pr skill, SonarCloud, AI code review (Copilot || Greptile), linting, tests (all in CI)

→ **3. Human code review + manual QA** - focused on the feature, based on test steps in the PR

→ **4. Full-app UAT/regression** - end of every sprint (working toward replacing with Playwright)

// where we are focused

Where we are focused

→ **Code quality** - refactors

→ **DevOps** - infra updates focused on reduced dev process

→ **QA** - focus on Playwright & test automation

→ **All driven by metrics**

// act II take-away

Act II take-away:
Focus on the Fundamentals

Leverage AI to improve your codebase

// act III

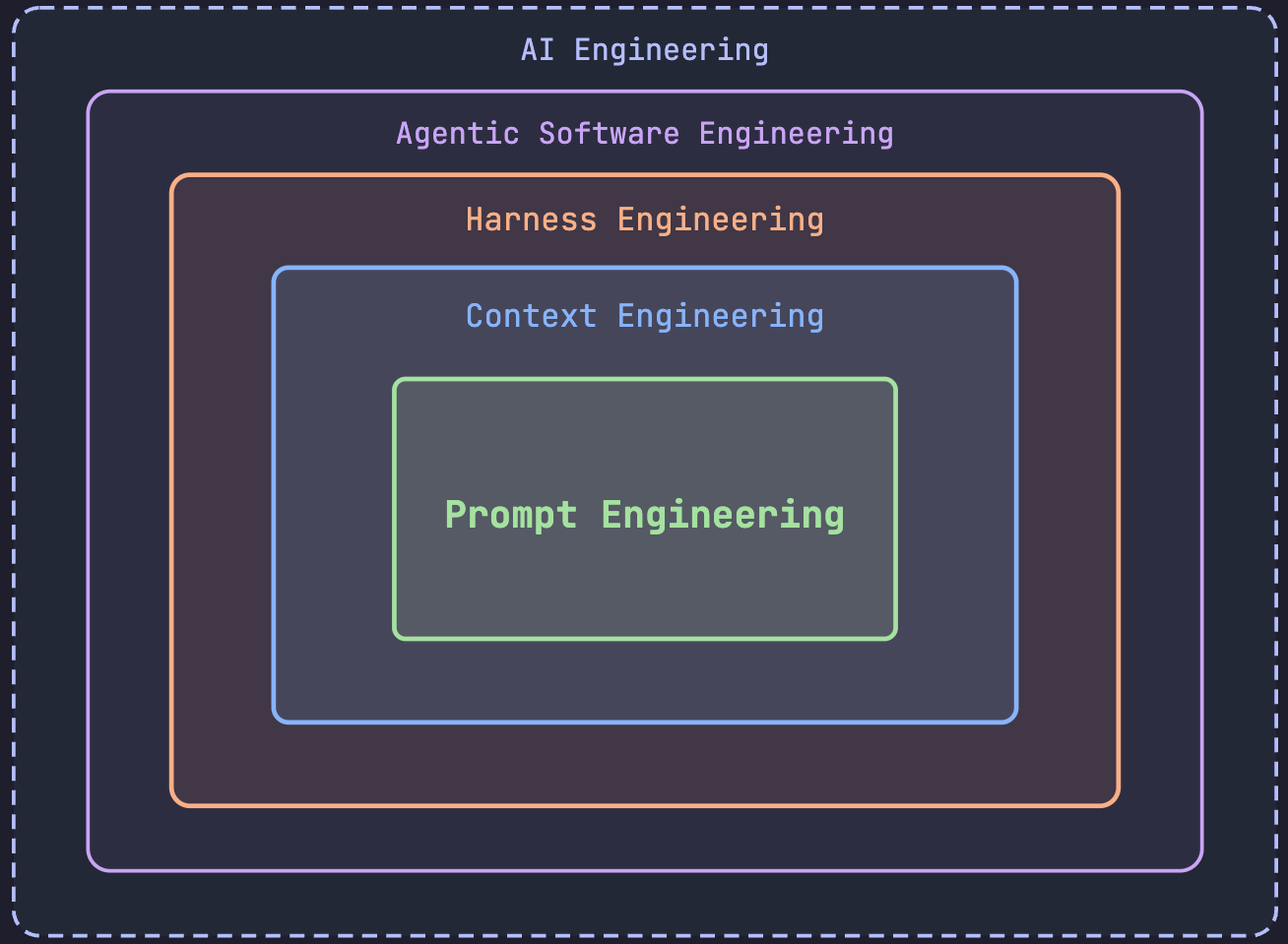
Act III

Agentic Dev Maturity

Prompt · Context · Harness

detroitdevelopers.com/blog/agentic-coding-maturity

// the three layers



Each outer layer *contains and depends on* the inner ones

// layer 1: prompt engineering

Prompt Engineering

- → Telling an agent what to do effectively
- → Written prompts, rules, skills
- → In practice: **almost entirely markdown optimization**
- → Tight feedback loop - edit a file, run a session, see what changed
- → Ceiling: ~+1x velocity

· *systems to measure & optimize prompt effectiveness (Chip Huyen)*

// layer 2: context engineering

Context Engineering

- → Managing what knowledge the agent has access to - and when
- → PILRs - Persistent Indexed Learning Repositories
- → **Type 1** (Ephemeral): per-feature planning docs, test plans
- → **Type 2** (Evergreen): architecture docs, system design, API contracts
- → **Type 3** (Cumulative): solved problems, incident patterns, institutional memory
- → Ceiling: ~+3x velocity

// context engineering: claude.md

CLAUDE.md: map, not encyclopedia

→ Large files degrade agent performance due to "the dumb zone" (context window)

→ When everything is "important", agents fall back to local pattern-matching

→ ~100 lines max - structured as a map with pointers to deeper sources of truth

→ Checked into the repo. Reviewed in PRs. Evolved alongside the codebase.

· *Team artifact, not personal config*

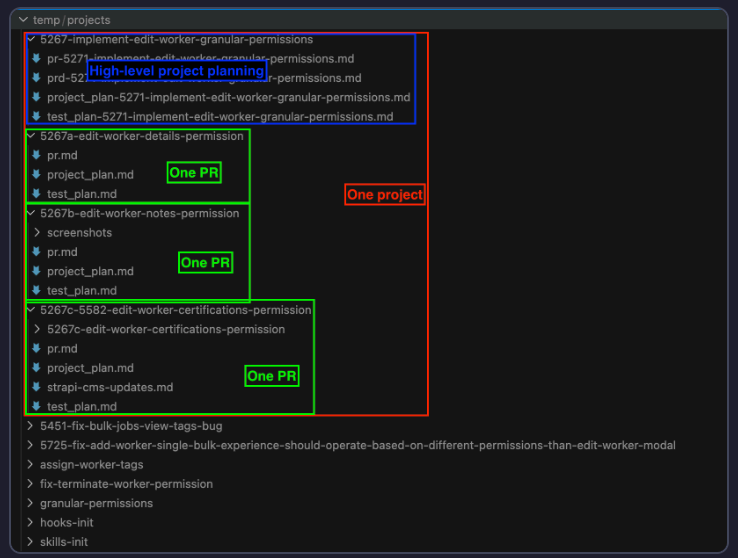
// context engineering: pilrs

PILRs: knowledge bases that learn

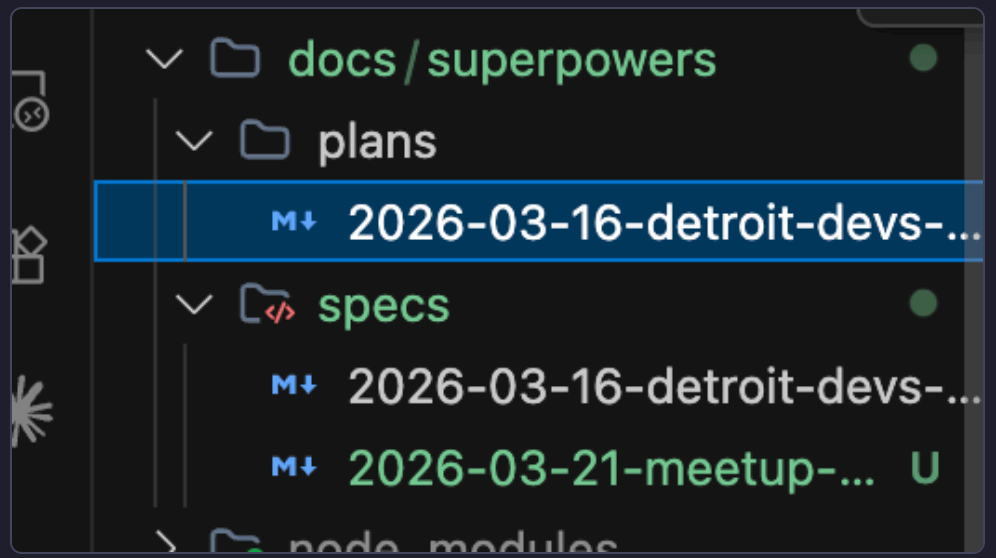
- → Structured markdown files + an index.json that tells the agent where to look
- → The agent loads only what it needs - reducing context window bloat
- → Note: PILRs *continue to evolve* as the agent does the work
- → Every fix, every new pattern, every bit of domain knowledge the agent picks up → feed back to the PILR

· *Unlike a wiki that goes stale, a PILR is maintained by the thing that reads it*

// context engineering: pilrs in practice



PILRs at RIVET



PILRs in Superpowers

github.com/obra/superpowers

// layer 3: harness engineering

Harness Engineering

- → Programming *around* the model - systems that run agents *for you*
- → Multi-step autonomous workflows with human review of outcomes
- → Deterministic steps where reasoning isn't needed
- → **Expands what work gets done *at all*** - not just speed. Collapses a full ticket - research, implementation, PR - into a parallelizable workflow
- → Ceiling: ~+10x velocity (S-curve)

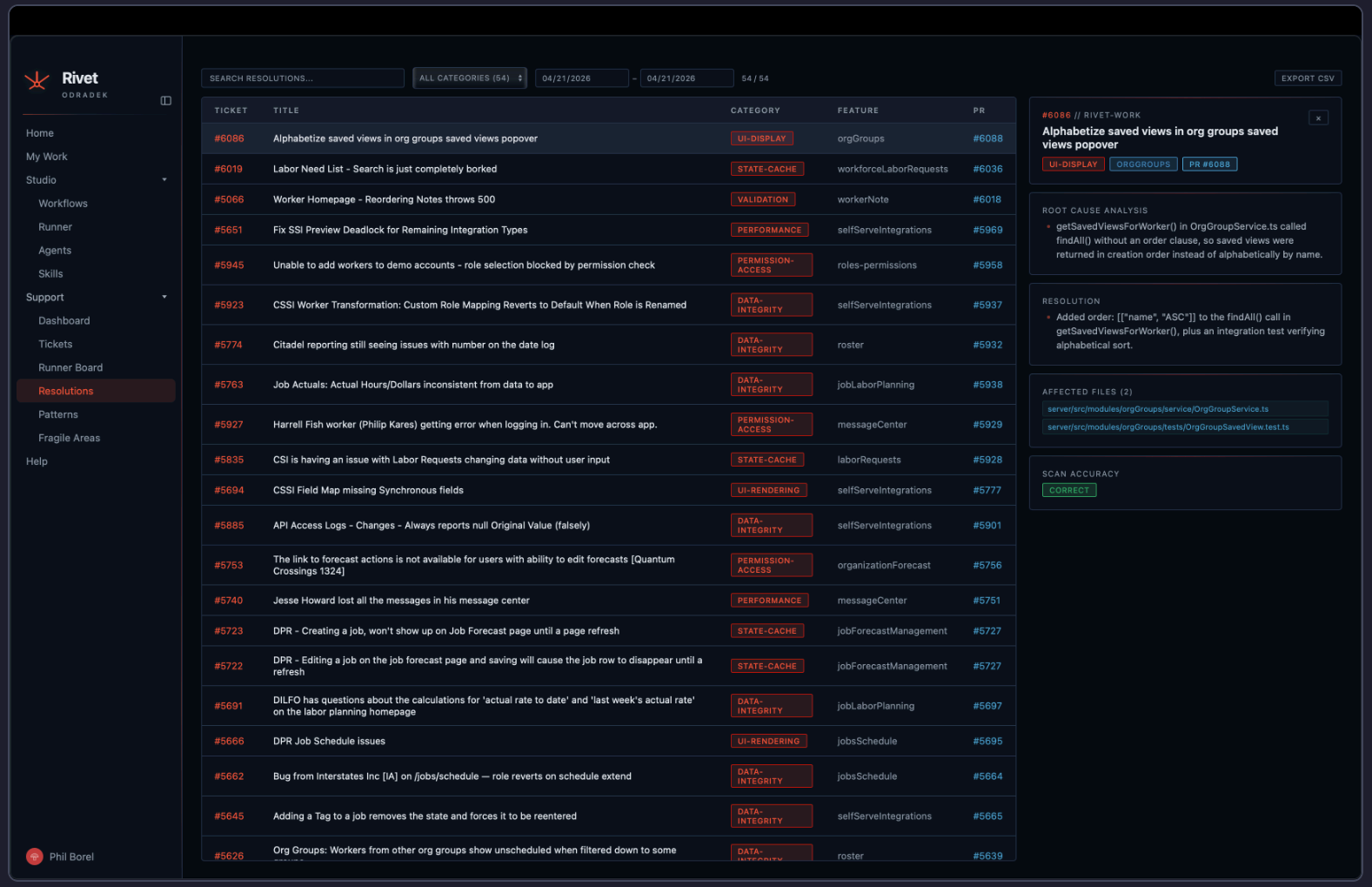
· *Every team has a long tail of valuable work that never makes the top of the backlog*

// case study: odradek

Odradek - Bug Resolution Agent

- → Built on the Claude Code SDK - Mac desktop app with cloud-hosted database for multiplayer support
- → **Investigates** - reads source, checks git, consults PILR knowledge base, pulls context from Notion (MCP) & GitHub (CLI)
- → **Fixes** - scoped, surgical edits to the relevant files
- → **Verifies** - regression checks, test additions & CI runs
- → **Opens a PR** - agentic-review phase + human-review phase
- → One-shot resolution rate: **80%**

// odradek: resolutions view



Odradek Resolutions - every fix with cause, resolution & affected files

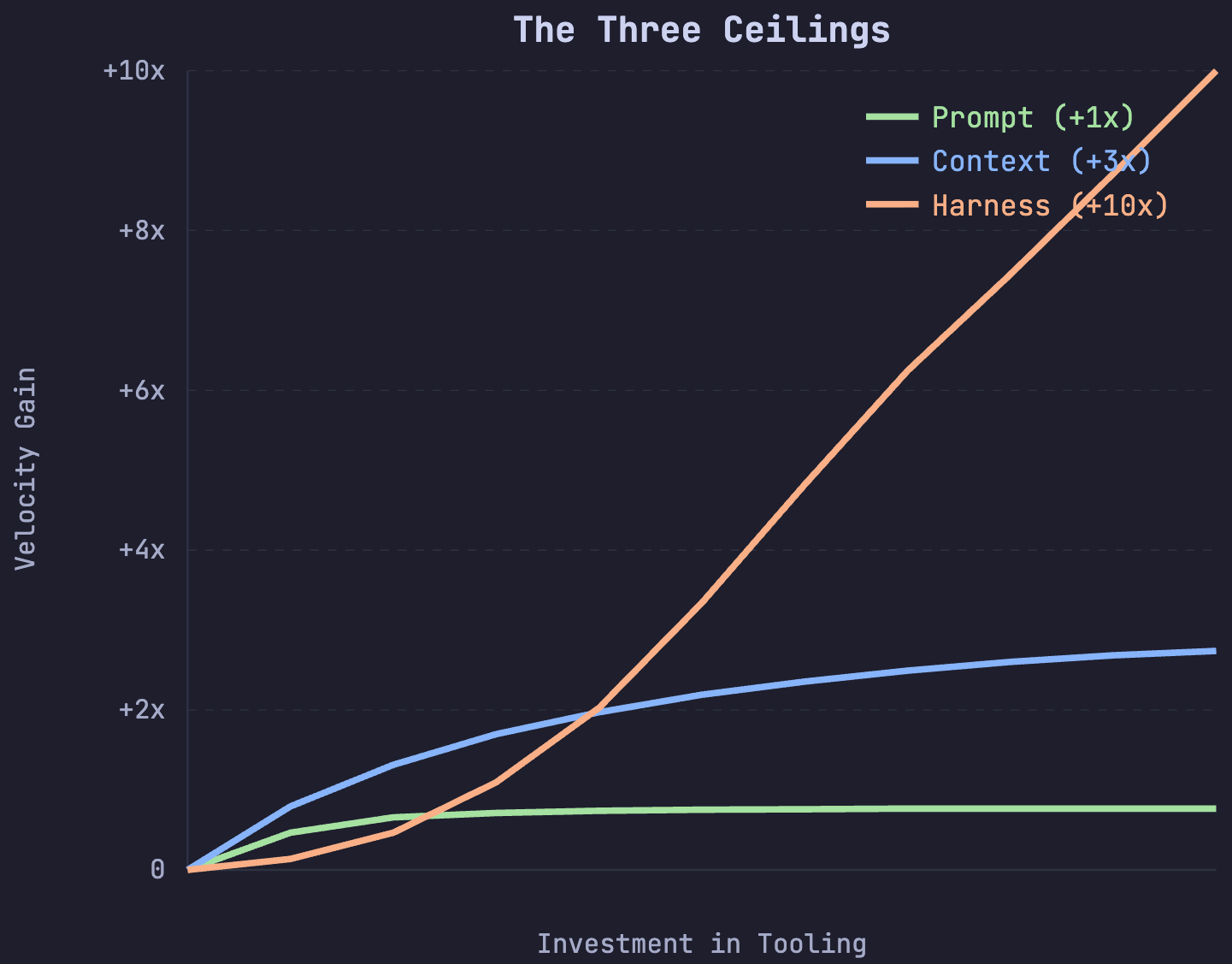
// odradek: impact on the backlog

Odradek impact on the backlog

- → **P1s** (ship-within-a-week): used to sacrifice one engineer per sprint on rotation. Odradek bought back ~half an engineer.
- → **P2s** (fix-within-a-quarter): used to stack for months. Now addressed as they come in.
- → **P3s** (nice-to-fix): were permanent backlog residents. Some are actually getting fixed now.

· *Work that would never have happened at the current team size*

// the three ceilings



// act III take-away

Act III take-away: where to invest your tokens for optimal returns

→ **Beginner:** Start with prompt engineering. Short feedback loop, transferable skills, necessary foundation

→ **Intermediate:** Context engineering. Build the knowledge layer. Start with PILRs where agents are most confused

→ **Advanced:** Harness engineering. Pick a narrow, repetitive workflow. Measure the one-shot rate

→ **The balance:** increasing velocity - maintaining quality

// what about product & design?

Product Convergence: The new bottleneck

- → **Prototyping**
- → **Spec writing**
- → **Feature work** - TBD

// close

Next week

- → **Set a goal:** identify an outcome & turn it into a goal
- → **Focus on fundamentals:** leverage the goal to prioritize tech debt & quality initiatives
- → **Invest in Agentic Dev maturity:** focus on prompt, context & harness engineering in that order. Read the docs @ claude.md
- → **Try Superpowers:** github.com/obra/superpowers - it's easy!

· *None of this requires a BHAG or an executive mandate*

QUESTIONS?

ASK ME IN THE APP

Download on the App Store

GET IT ON Google Play

PLEASE RATE MY SESSION

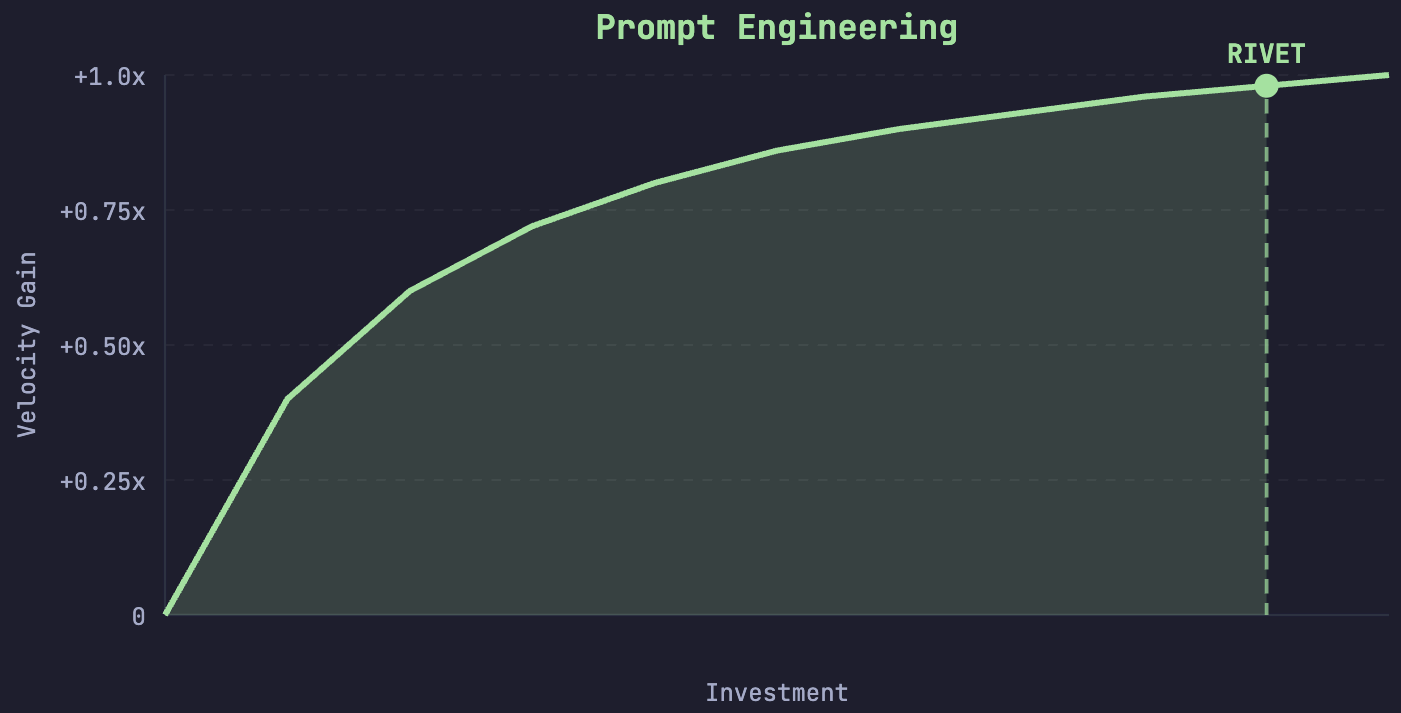
// appendix

Appendix

Reference Material

Extra slides for Q&A

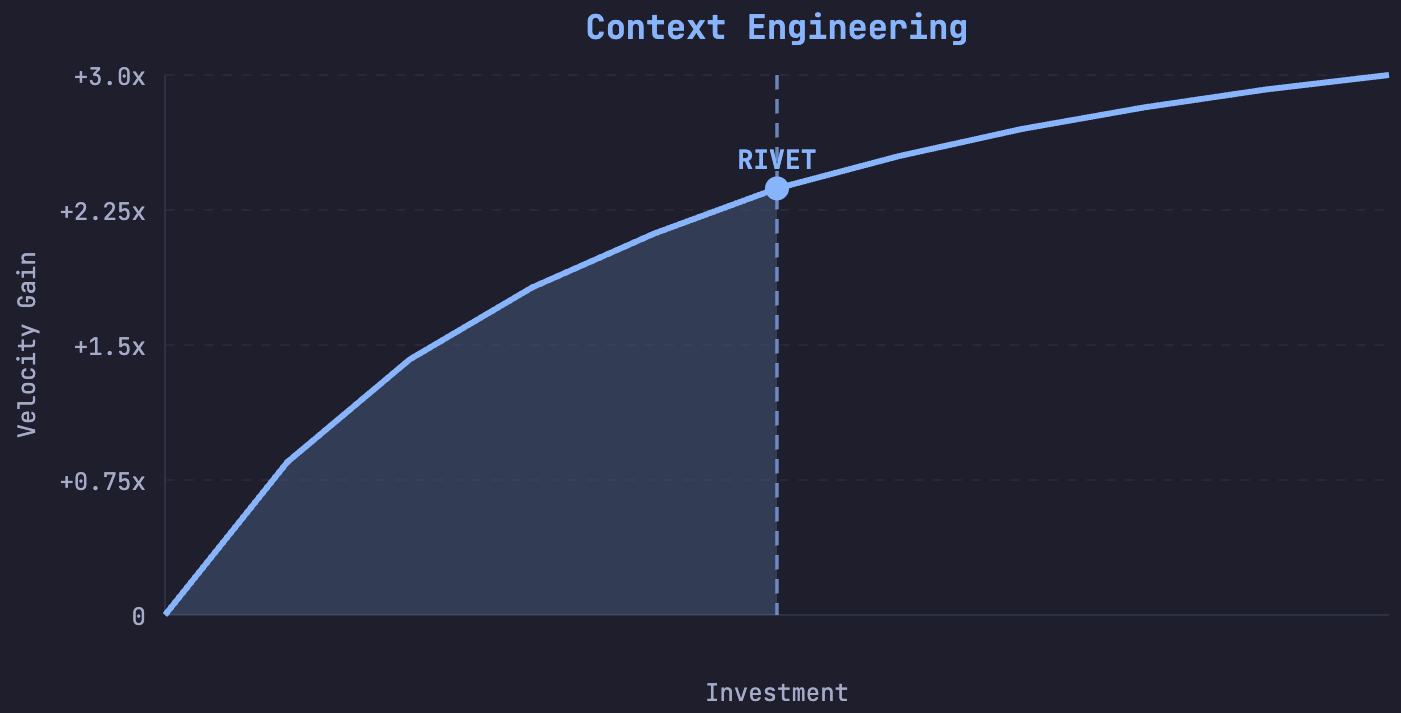
// RIVET - prompt engineering



Far along, near the ceiling

- → Deep investment in CLAUDE.md, rules, skills
- → Almost entirely markdown optimization
- → Tight feedback loop - edit, run, observe
- → Most of the returns are already captured

// RIVET - context engineering



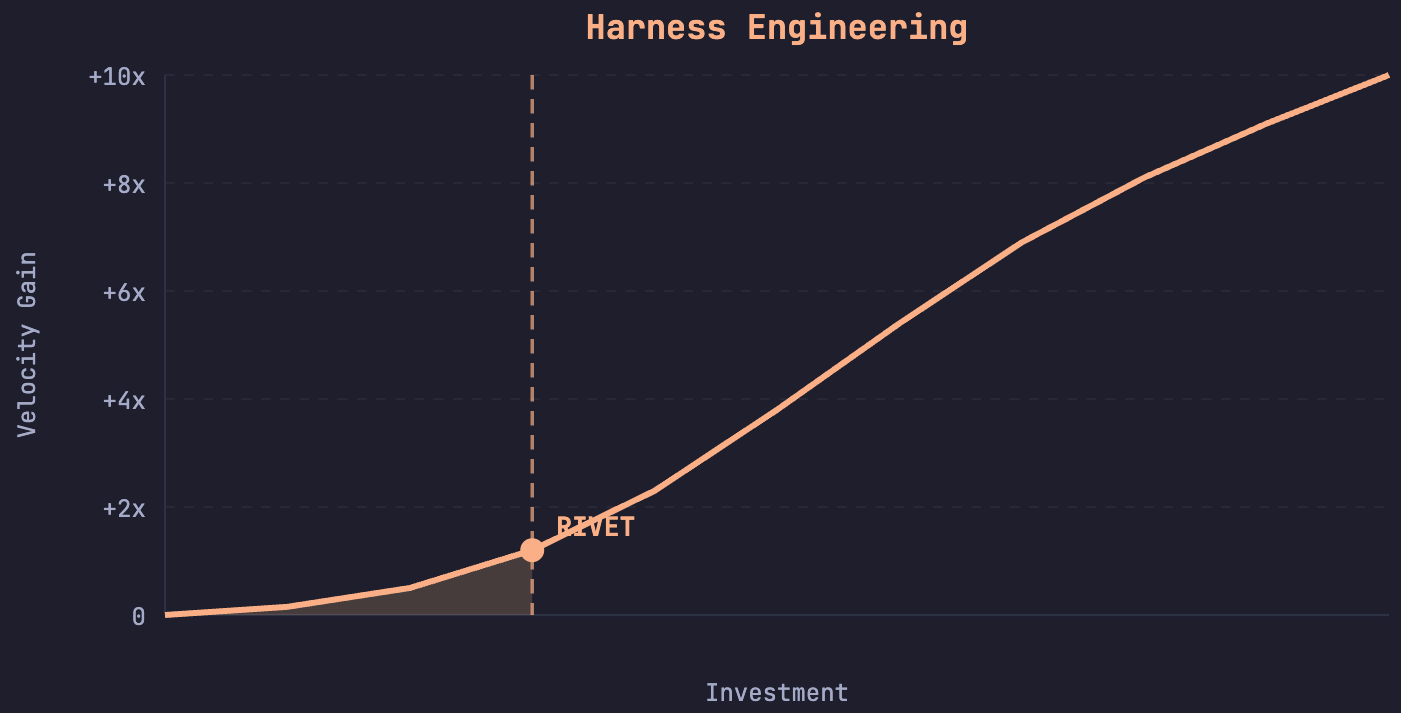
Early-mid, significant upside

→ PILRs pattern is right; infrastructure isn't finished

→ **Type 1 (Ephemeral)** solid, **Type 2 (Evergreen)** in progress, **Type 3 (Cumulative)** nascent

→ Starting to invest in shared hosting & data infra

// RIVET - harness engineering



Very early, asymmetric upside

→ Odradek is our first real harness - 80% one-shot rate

→ S-curve: slow start, steepest returns ahead

→ This is where we're investing

// appendix: odradek roadmap

Odradek Roadmap

- → Event-driven triggers - auto-fire on bug ticket creation
- → Cloud-hosted dashboard - non-engineers see work in progress & resolution status
- → MS Teams integration - ask about bugs & initiate workflows from chat
- → Parallel issue processing - git worktrees for concurrent work
- → Ephemeral test environments - CS/product verify fixes themselves
- → Model routing - right model tier for each task; balances cost & effectiveness