Unhobbling AI Products While Preserving Product Ownership

PM Daily Digest

2026-07-28

Unhobbling AI Products While Preserving Product Ownership

By PM Daily Digest • July 28, 2026

An AI-product brief on finding capability hidden by product scaffolding, competing with entrenched workflows, and updating agentic products through observation and verification. It also covers clearer product ownership with agencies and a practical skill-development lens for PMs.

Big Ideas

AI product opportunity can be in the harness, not the model

A YC discussion frames **product overhang** as capabilities a current model already has but that products have not yet elicited. In this view, a product can "hobble" a model when its scaffolding gets in the way; the opportunity is to remove unnecessary constraints and let the model express more of what it can do. [1]

Why it matters: AI product strategy is not limited to waiting for the next model release. It also means testing whether the interface, prompts, tools, and workflow are preventing useful behavior.

Adoption competes with existing habits

AI startups often face entrenched alternatives: long-lived tracker sheets, multi-tab tasks, undocumented copy-paste processes, and employees who know how to get the work done without a new tool. Fear of change and inertia are part of that competitive set. [2]

Apply it: in discovery, identify the actual workflow being replaced—including its informal steps—before positioning an AI feature as an alternative.

Tactical Playbook

Refresh the AI harness empirically with each model release

1. **Remove inherited prompt scaffolding.** Claude Code changes its system prompt and tools frequently because behavior can differ substantially between models; on one release, its team removed 80% of the system prompt. [1]
2. **Run real tasks before prescribing fixes.** Observe where the product works and where it repeatedly fails rather than guessing the instruction a model needs. [1]
3. **Add back only recurring constraints.** Use higher-level task descriptions, guardrails, and exit criteria instead of rigid step-by-step instructions. [1]
4. **Build verification into the workflow.** The discussion identifies making it possible for the model to check its work as a critical capability for harder tasks. [1]
5. **Maintain—and eventually renew—evals.** Keep appending to useful evals across model generations, but replace them when model progress saturates the set. [1]

Make product ownership explicit when working with an agency

Development agencies can provide guidance on implementation effort, technical debt, testing, and platform health, but they lack the daily customer, business, usage, and team context required to decide what should be built. [3]

Apply it: assign someone to answer four questions: where the business is headed, what the product must do to support it, what to build now, and what can wait. Options are an internal or fractional product leader, an agency that fully embeds in the product role, or a founder who takes on PM responsibilities. [3]

Case Studies & Lessons

Claude Code pairs broad delegation with repeatable maintenance routines

The team describes giving modern models harder, higher-level tasks with guardrails rather than overly specific instructions. For complex work, dynamic workflows can break a task into stages of agent execution, verification, and summarization. [1]

It also reports running 20–30 recurring maintenance routines across codebases—such as dead-code cleanup, test generation, and unifying near-duplicate abstractions—with hundreds or sometimes thousands of agents daily. The system is described as not fully automated yet. [1]

Lesson: agentic execution is not simply delegation. Product teams need clear

completion criteria, verification, and narrowly defined recurring loops before expanding scope.

Career Corner

Build practical product judgment alongside technical skill

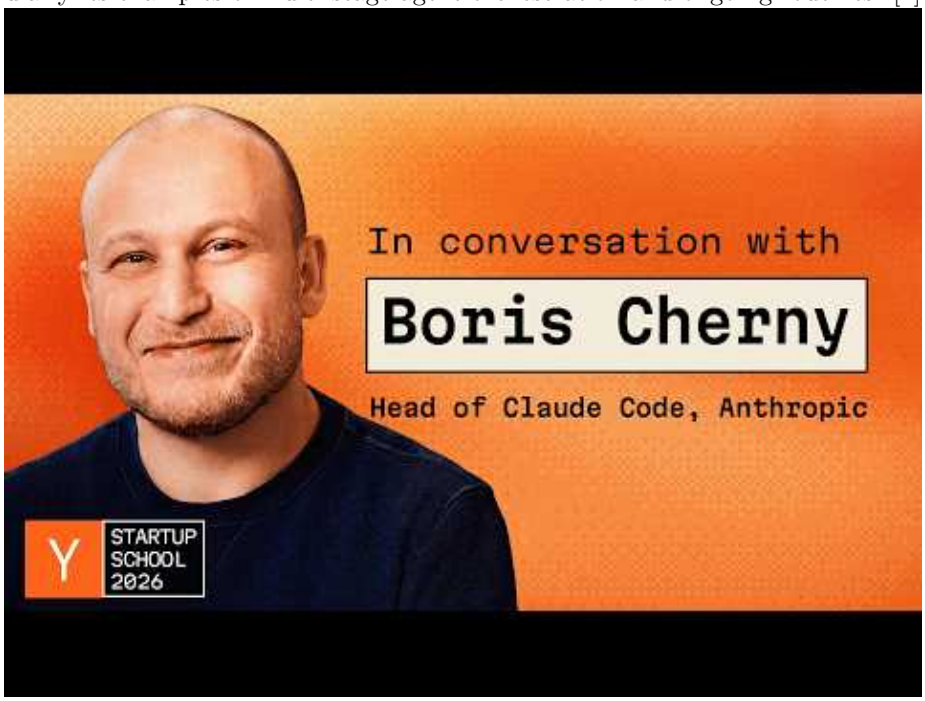
The career advice in the YC discussion is to learn computer science through practical problem-solving: build products, talk to users, and develop design, business, and data skills in addition to engineering. [1]

Apply it: choose work that forces both technical execution and user contact. The goal is not theory alone, but learning to connect a real problem, a product decision, and an implemented solution.

Tools & Resources

YC: *Boris Cherny: Building Claude Code*

The conversation is a useful resource for PMs designing agentic workflows, particularly its examples of multi-stage agent orchestration and ongoing routines. [1]



Boris Cherny: Building Claude Code (25:20)

Sources

1. Boris Cherny: Building Claude Code
2. X post by @andrewchen
3. r/startups post by u/mahow