Prototype-first PMing: AI design workflows, builder role signals, and faster feedback loops

PM Daily Digest

2026-02-21

Prototype-first PMing: AI design workflows, builder role signals, and faster feedback loops

By PM Daily Digest • February 21, 2026

This edition focuses on prototype-first product work: a practical AI design framework for PMs (with repeatable workflows and evaluation checks), plus signals on the emerging "builder" role shift. It also includes feedback analysis tactics, game-design-inspired UX lessons, and career guidance on collaboration, pricing exposure, and job-search tradeoffs.

Big Ideas

1) AI design is a full system, not "just prompting"

Xinran Ma frames "designing with AI" as five categories: **Prompting** (prompt engineering, context, iteration) [1], **Ideation** (divergent thinking) [2], **Design & Prototyping** [3], **Workflows** (how you work day-to-day) [4], and **Staying conscious** (risks, biases, unintended consequences) [5]. He also argues PM/designer/engineer roles are merging as AI lowers the barrier to prototyping—*if* you understand the broader workflow, not just prompts [6].

Why it matters: If prototyping gets cheap and fast, your advantage shifts to how quickly you can move from an idea to something real *and* validate whether it actually helps users (not just whether it looks good). [7][8]

[1] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[2] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[3] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[4] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[5] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[6] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[7] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[8] How to Design with AI | The Complete Guide for PMs with Xinran Ma

How to apply: Treat AI design like a looped product workflow: generate prototypes quickly, then validate and refine rather than polishing "final" mockups up front [9].

---

2) Prototypes are increasingly replacing long docs as the fastest alignment tool

Andrew Chen's thesis is explicit:

> "the prototype is the new PRD" [10]

He adds that if your team needs a "20-page product strategy doc," you're already behind someone with a weekend prototype [11]—and that actually using the product experience can beat theorizing, market analysis, and user research in assessing quality [12].

Why it matters: AI makes "weekend prototypes" more achievable for more teams, raising the baseline for speed and clarity in product communication.

How to apply: Use prototypes early to test whether the experience *feels right*, then use user validation to confirm it solves the real problem [13][14].

---

3) As delivery gets cheaper, discovery discipline becomes more important

Teresa Torres argues that as "the cost to delivery approaches zero," product teams will spend more time on discovery—and need to make that time effective [15].

Why it matters: Faster shipping can increase the cost of shipping the wrong thing.

How to apply: Use continuous discovery habits (structured, sustainable routines) to stay aligned with customer needs while driving business outcomes [16][17].

---

[9] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[10] post by @andrewchen
[11] post by @andrewchen
[12] post by @andrewchen
[13] post by @andrewchen
[14] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[15] post by @ttorres
[16] post by @ttorres
[17] post by @ttorres

4) The "builder" shift: value moves from writing code to orchestrating + verifying

Aakash Gupta summarizes remarks from Boris Cherny (creator of Claude Code) that the title "software engineer" gets replaced by "builder" or "product manager" [18]. In the same write-up, Cherny is described as running **10–15 concurrent Claude sessions** and treating AI like "compute you schedule" [19]. The post also claims Anthropic moved from **$1B to $7B revenue run rate in nine months** and has **50+ engineering roles open** [20].

The implied job shift: engineers building systems that make AI reliable—e.g., a "verify-app" agent for end-to-end tests, a "code-simplifier" agent for cleanup, and automation hooks for formatting [21].

Why it matters: Teams may hire more engineers even as coding gets automated—because the hard part becomes reliability, sequencing parallel work, and catching the "last 10%" of failures [22].

How to apply: If you're adopting agentic tooling, explicitly build a verification loop (tests, review hooks, sanity checks) alongside the "generate code/design" loop [23].

---

5) Product-market timing still beats "if you build it, they will come"

Tony Fadell notes that "If you make it they will come" doesn't always work: the technology must be ready, timing must be right, and customers need to see you solving a **real problem they have today** (not a distant-future one) [^5].

Why it matters: Faster building increases the risk of building something premature.

[18] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."

[19] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."

[20] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."

[21] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."

[22] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."

[23] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."

How to apply: Use "real problem today" as a gating question before investing heavily in polishing or scaling [⌢5].

Tactical Playbook

1) A fast AI design workflow (idea → clickable prototype → code)

Xinran's demonstrated workflow:

1. **Custom GPT to force clarity**: ask focused questions to define (a) who you're designing for, (b) their core need, and (c) the first experience to build; then generate a lightweight **markdown spec** (screens/components/interactions) [24].
2. **Claude as a "mock run" sanity check**: paste the spec for a quick visual preview to verify screens/flows roughly make sense before spending Lovable credits [25][26].
3. **Lovable for the real prototype**: paste the same spec; Lovable generates a working prototype in ~60 seconds (demo included add-expense, summary, confirmation states, navigation) [27].
4. **Iterate quickly**: refinement loops take ~20–30 seconds; after ~5–10 iterations you can reach something polished enough to share or test [28].
5. **Add real logic**: ask for functional behaviors (auto-suggest categories, real-time totals, bank API imports) and it generates working code [29].
6. **Export clean code**: Lovable outputs clean React code you can hand to engineers and deploy as a test version [30].

Why it matters: It compresses a "weeks-long" linear design path into minutes/hours and makes stakeholder alignment possible earlier [31][32].

How to apply: Adopt the "sanity check → prototype → fast iteration" rhythm so you don't over-invest in the wrong direction [33][34].

---

2) Divergent exploration from an existing design (Stitch → AI Studio)

Workflow for exploring alternatives quickly:

[24] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[25] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[26] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[27] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[28] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[29] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[30] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[31] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[32] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[33] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[34] How to Design with AI | The Complete Guide for PMs with Xinran Ma

1. In Stitch, **paste a screenshot** of the existing design and write a prompt with (a) context, (b) business goal, and (c) your ask [35].
2. Generate and review **2–3 variants** by default [36].
3. Use the creativity slider ("refined → YOLO") and specify what to vary (layout, color schemes, text); generate **3–4** wide variants to break assumptions [37].
4. Pick the best direction and **export to Google AI Studio** as an HTML reference + prompt; if you select multiple screens, you get a multi-screen prototype [38].

Why it matters: It's a fast way to explore directions you wouldn't reach in a short brainstorm [39].

How to apply: Use "YOLO variants" intentionally in early-stage discovery, then converge into a prototype you can test or annotate collaboratively [40][41].

---

3) A 4-layer checklist for evaluating AI-generated designs

Xinran's four layers:

1. **Visual representation**: brand-fit and visual pleasantness [42].
2. **Problem solving**: does it actually address what users need? validate with real users [43].
3. **Design principles**: accessibility checks (contrast, readability, screen reader compatibility, keyboard navigation, touch target sizes) [44].
4. **Implementation feasibility**: confirm it fits your tech stack/architecture; review with engineering so you don't design something that can't ship [45].

Why it matters: Many AI designs look good at layer 1 but fail deeper checks [46].

How to apply: Don't treat AI output as shippable until it passes user validation, accessibility review, and feasibility review [47][48][49].

---

[35] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[36] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[37] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[38] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[39] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[40] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[41] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[42] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[43] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[44] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[45] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[46] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[47] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[48] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[49] How to Design with AI | The Complete Guide for PMs with Xinran Ma

4) Customer feedback analysis when volume overwhelms you (pattern > anecdotes)

A lightweight process shared on r/ProductManagement:

1. Collect feedback from all channels in one place [^6].
2. Group comments by the underlying problem (not by source) [^6].
3. Track recurring issues weekly [^6].
4. Separate emotional language from usability friction [^6].
5. Prioritize based on frequency + impact [^6].

The "biggest shift": stop reacting to individual comments; focus on recurring issues to make roadmap conversations clearer and less emotional [^6].

Why it matters: Multi-channel feedback breaks the "read everything" approach as you scale [^6].

How to apply: Set a recurring cadence (weekly) for pattern review, so prioritization is anchored in recurrence and impact rather than volume and intensity [^6][^6].

---

5) Tackling the fuzzy front-end: from vague direction to something spec-able

A thread on r/prodmgmt highlights three complementary moves:

- Start with an AI-assisted rough pass (the OP uses **Figr AI** for user flows/issues), then bring it into **Miro** for team discussion—having a starting point beats a blank canvas [^7].
- Write it down: the act of documenting how it will work forces deeper thinking; AI can create an "illusion" of substance without it [^8].
- Collaborate with designers and engineers to co-create solutions; spec-driven development misses more often than collaborative design [^9].

Why it matters: AI can accelerate structuring, but it doesn't remove the need to do the thinking and alignment work [^8][^9].

How to apply: Use AI to draft flows, then force the "writing pass" and run a collaborative review to pressure-test feasibility and user impact [^7][^8][^9].

---

6) Reducing coordination drag on small copy fixes (while keeping dev review)

A PM built a Chrome extension to address a common workflow: spotting copy issues leads to screenshot → Slack/Jira → dev handling → review → ships next sprint, even when the fix is seconds and coordination is days [^10].

The tool: click any text element in the live product, edit it, and it creates a GitHub PR with the exact file/language/line location; devs still review and merge [^10]. It also supports flagging UI elements as GitHub issues with precise source + screenshot [^10] and inspecting who built an element via commit/PR/ticket context [^10].

Why it matters: Many teams' bottleneck isn't fixing copy; it's the handoff and context reconstruction [^10].

How to apply: If you try a workflow like this, explicitly evaluate whether the dev review step still creates too much friction for your team's speed goals [^10].

Case Studies & Lessons

1) "Idea → prototype in 60 seconds" (and why iteration is the real skill)

In Xinran's expense-tracker demo, copying a markdown spec into Lovable produced a working clickable prototype in about 60 seconds [50]. The improvement came through rapid iteration—20–30 seconds per refinement round, repeated 5–10 times to reach something ready to share/test [51].

Takeaway: The skill isn't the first output; it's iteration speed and refinement depth [52][53].

---

2) Modern game design lessons that translate to product work (and pitfalls)

Cheryl Platz (Riot Games, formerly Microsoft) describes a "motivators of play" framework—classic motivators (fun, mastery, competition, immersion, meditation, comfort) [^11] and modern motivators (companionship, self-expression, education) [^11].

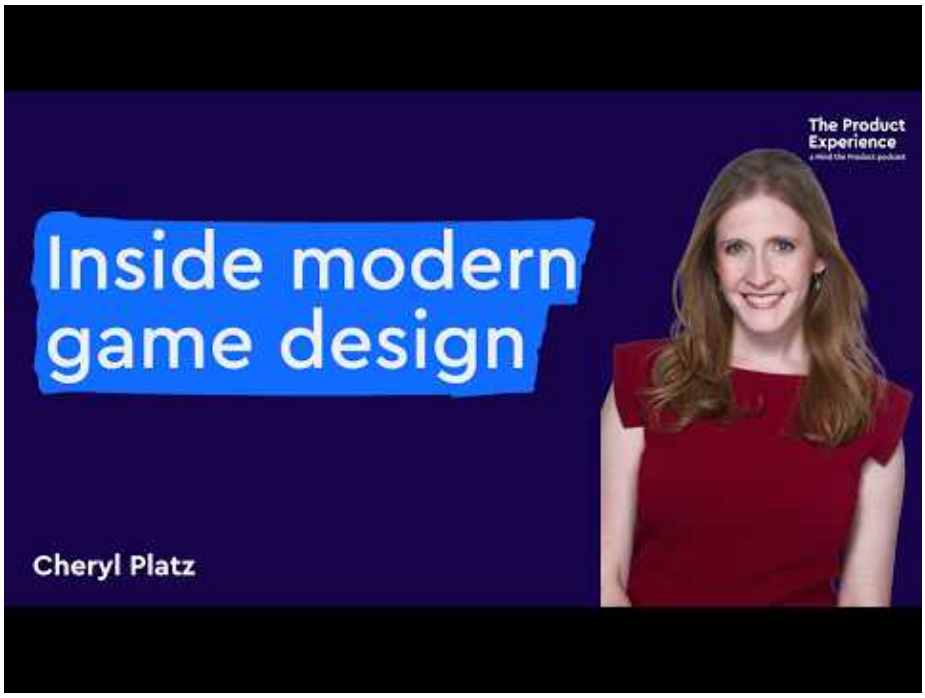
Two applied examples:

- **Disney Friends (Nintendo DS):** some players (boys) lacked clear next steps and didn't know what to do [^11][^11]. The team added visible feedback (sparkles) and a daily friendship-points meter [^11], and the change increased engagement for those players while keeping the experience working for others [^11].
- **Marvel StrikeForce outage:** an anniversary event drove mass logins and orb-redemption activity that took down servers [^11]. The

[50] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[51] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[52] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[53] How to Design with AI | The Complete Guide for PMs with Xinran Ma

team later introduced a scalable "logarithmic" UI for opening orbs (10/100/1,000/10,000), improving player agency and addressing a UX problem that existed before the outage [^11][^11].

A broader warning from Platz: gamification can be misused; you need to understand what you're motivating because rewards shape future behavior [^11].



Inside modern game design - Cheryl Platz (Riot Games, Microsoft) (23:27)

Takeaway: Borrow the "clarity of core loop + visible progress" instincts from games, but be deliberate about the behaviors you reinforce [^11][^11].

---

3) A costly experiment reminder: beware pinpoint changes in complex systems

A PM shared that an experiment launch degraded metrics during the busiest time of year, likely costing "7 figures," and reinforced a lesson about avoiding pinpoint changes in complex end-to-end systems [^12].

Takeaway: Speed (especially with experimentation) increases the need for system-level thinking and guardrails [^12].

---

4) UI trust for AI agents: make the interface match user expectations

Teresa Torres shared a case where ShowMe's AI sales agents could demo products and close deals on video calls, but visitors underestimated them and immediately asked for a human [^13]. When the interface was redesigned to feel like a real video call (Google Meet/Zoom), users engaged with the AI more like a colleague and conversions followed [^13].

Takeaway: Trust is often an interaction-model problem, not just a capability problem [^13][^13].

---

5) Distribution as product strategy: Timex changed the channel to change the economics

Paul Graham notes Timex cut retail markup in half to make watches cheaper; when jewelers resisted, Timex sold watches off racks in drugstores [^14].

Takeaway: Sometimes the lever isn't the product—it's the path to market [^14].

Career Corner

1) "Work with anyone" becomes a decisive skill as you get more senior

Deb Liu argues that as careers progress, technical excellence becomes table stakes and the ability to work with almost anyone becomes decisive [^15]. A key dynamic: in senior roles, you often can't choose your peers (e.g., boards, C-suite), so you must learn to collaborate effectively with the people in front of you [^15][^15].

Practical behaviors she highlights:

- Start with **incentives** (what they value, how they're measured, where they want to go) rather than personalities; aligning incentives can shift a tense relationship quickly [^15].
- Find genuine **common ground** to build trust (shared backgrounds, hobbies, life context) [^15].
- Make other people look good—publicly credit enabling teams whose work is essential but invisible [^15].
- Change "posture": move from "across the table" debate to "same side" shared problem-solving [^15].

How to apply: Treat incentive-mapping and shared-success framing as first-class PM work, not "soft skills" [^15][^15].

---

2) When you're a de facto PO on legacy systems: document reality and tech debt

A new, untrained de facto PO described managing critical legacy tools with highly variable dev availability (often ~30% time), a massive unprioritized backlog, significant tech debt, no PM culture, and constant urgencies [^16]. One piece of advice: use Claude Code to **document what exists now**, then document tech debt [^17].

How to apply: Start by making the current system legible (what exists + what's broken), so prioritization and stakeholder conversations have a concrete foundation [^17].

---

3) Pricing exposure is uneven—many PMs learn it on the job

A thread highlights PM anxiety about never having owned pricing for a new feature [^18]. Replies emphasize learning through research and iteration [^19], drilling into core product pricing dimensions and correlating them to feature usage [^20], and leaning on guidance docs/courses and senior gut checks [^21].

How to apply: If pricing is new to you, expect an "apprenticeship" period—ground yourself in how the core product prices today, then map the feature to those dimensions [^20].

---

4) Resume signals: senior PMs are still debating ATS vs. readability tradeoffs

A PM with 12 years of experience condensed a resume from two pages to one, removed a professional summary and skills section (previously 20–30 skills), and worried about balancing a human tone with ATS scoring [^22].

How to apply: Use this as a prompt to pressure-test your own resume choices (format, skills list depth, summary/no-summary) against the roles you're applying for [^22].

Tools & Resources

1) AI design tool picks (by use case)

From Xinran's stack:

- **Prompt generation:** build a custom GPT that knows your design system, product, and user base [54].

[54] How to Design with AI | The Complete Guide for PMs with Xinran Ma

- **High-quality prototypes:** Lovable for design quality + clean code [55]; v0 as a close second with a different aesthetic and free code editing [56].
- **Fast variations:** Magic Patterns for generating multiple divergent directions quickly [57].
- **Free prototyping:** Google AI Studio as a capable free option [58].
- **Full-stack prototypes:** Cursor for real databases/APIs/complex logic (more technical) [59].
- **Sanity checks:** Claude as a "mock-run" tool before dedicated prototyping [60].

Podcast episode source: https://www.news.aakashg.com/p/xinran-ma-podcast [61].

---

2) Continuous discovery training

Teresa Torres is running cohorts of **Product Discovery Fundamentals**, a six-week course with hands-on practice in continuous discovery habits [62]. Details: https://buff.ly/2QjolV6 [63].

---

3) Claude Code learning resources (curated list)

From Aakash Gupta:

- https://www.youtube.com/watch?v=YKYQ-z6A9Fs [64]
- https://www.youtube.com/watch?v=4nthc76rSl8 [65]
- https://www.news.aakashg.com/p/how-to-use-claude-code-like-a-pro [66]
- https://www.news.aakashg.com/p/pm-os [67]

[55] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[56] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[57] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[58] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[59] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[60] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[61] How to Design with AI | The Complete Guide for PMs with Xinran Ma
[62] post by @ttorres
[63] post by @ttorres
[64] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."
[65] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."
[66] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."
[67] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."

- https://www.youtube.com/watch?v=PQU9o_5rHC4 [68]

---

4) A "copy-to-PR" workflow for faster fixes

If your team's bottleneck is micro-copy coordination, the Chrome extension that turns live text edits into GitHub PRs is described here (with issue creation + provenance inspection) [^10][^10][^10].

---

5) Reading: collaboration as a career skill

Deb Liu: https://debliu.substack.com/p/how-to-work-with-anyone [^15].

This is Anthropic's head of Claude Code saying his own product eliminates the skill that defined his entire career. Cherny rose to Principal Engineer (IC8) at Meta over seven years. He wrote and reviewed code for a living. Now he runs 10-15 concurrent Claude sessions and treats AI like compute you schedule, not a tool you prompt.

In a podcast with YC, Cherny posted that engineers are "more important than ever." When someone pointed out the contradiction with Dario's Davos prediction that models will do all of SWE end-to-end in 6-12 months, Cherny replied: "Dario is talking about what's next."

This tells you everything about Anthropic's internal positioning. They've gone from $1B to $7B revenue run rate in nine months. They have 50+ engineering roles open right now. And the guy building the tool that automates coding is simultaneously hiring more engineers than ever.

The contradiction dissolves when you realize what those engineers actually do. Cherny uses a "verify-app" agent to run end-to-end tests, a "code-simplifier" agent for architecture cleanup, and a PostToolUse hook that auto-formats the last 10% Claude misses. He's building systems that make AI reliable, not writing application code.

Anthropic is betting that the value shifts from people who can write code to people who can define what to build, sequence the work across parallel agents, and catch the 10% of failures that compound into production disasters. That's the "builder" Cherny is describing.

5 resource to move to this way of working:

1. New tool stack: %5Bhttps://www.youtube.com/watch?v=YKYQ-z6A9Fs%5D(https://www.youtube.com/watch?v=YKYQ-z6A9Fs)

[68] [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."

2. Claude code for PMs: %5Bhttps://www.youtube.com/watch?v=4nthc76rSl8%5D(https://www.youtube.com/watch?v=4nthc76rSl8)
3. Advanced claude code: %5Bhttps://www.news.aakashg.com/p/how-to-use-claude-code-like-a-pro%5D(https://www.news.aakashg.com/p/how-to-use-claude-code-like-a-pro)
4. Claude code PM OS: %5Bhttps://www.news.aakashg.com/p/pm-os%5D(https://www.news.aakashg.com/p/pm-os)
5. Boris' interview: %5Bhttps://www.youtube.com/watch?v=PQU9o_5rHC4%5D(https://www.youtube.com/watch?v=PQU9o_5rHC4)

Every PM reading this should understand: the person who built the most successful AI coding tool in history just said your job title is the future of software. The question is whether you're building the judgment and technical fluency to claim it.

](https://substack.com/@aakas
217285355) [^5]: post by @tfadell [^6]: r/ProductManagement post by u/manonixx [^7]: r/prodmgmt post by u/AggravatingSlice1 [^8]: r/prodmgmt comment by u/calciphus [^9]: r/prodmgmt comment by u/holyelvis [^10]: r/ProductManagement post by u/seasonh5 [^11]: Inside modern game design - Cheryl Platz (Riot Games, Microsoft) [^12]: r/ProductManagement post by u/fiftyfirstsnails [^13]: post by @ttorres [^14]: post by @paulg [^15]: How to Work With Anyone [^16]: r/prodmgmt post by u/Desparate_Impostor [^17]: r/prodmgmt comment by u/mckirkus [^18]: r/ProductManagement post by u/varbinary [^19]: r/ProductManagement comment by u/Is_ItOn

[^20]: r/ProductManagement comment by u/Sufficient-Rough-647 [^21]: r/ProductManagement comment by u/kreepykrally [^22]: r/prodmgmt post by u/Erynor_

Sources

1. How to Design with AI | The Complete Guide for PMs with Xinran Ma
2. post by @andrewchen
3. post by @ttorres
4. [The creator of Claude Code, which hit $1B ARR faster than almost any dev tool in history, just told you the endgame: the title "software engineer" gets replaced by "builder" or "product manager."

This is Anthropic's head of Claude Code saying his own product eliminates the skill that defined his entire career. Cherny rose to Principal Engineer (IC8) at Meta over seven years. He wrote and reviewed code for a living. Now he runs 10-15 concurrent Claude sessions and treats AI like compute you schedule, not a tool you prompt.

In a podcast with YC, Cherny posted that engineers are "more important than ever." When someone pointed out the contradiction with Dario's Davos prediction that models will do all of SWE end-to-end in 6-12 months, Cherny replied: "Dario is talking about what's next."

This tells you everything about Anthropic's internal positioning. They've gone from $1B to $7B revenue run rate in nine months. They have 50+ engineering roles open right now. And the guy building the tool that automates coding is simultaneously hiring more engineers than ever.

The contradiction dissolves when you realize what those engineers actually do. Cherny uses a "verify-app" agent to run end-to-end tests, a "code-simplifier" agent for architecture cleanup, and a PostToolUse hook that auto-formats the last 10% Claude misses. He's building systems that make AI reliable, not writing application code.

Anthropic is betting that the value shifts from people who can write code to people who can define what to build, sequence the work across parallel agents, and catch the 10% of failures that compound into production disasters. That's the "builder" Cherny is describing.

5 resource to move to this way of working:

1. New tool stack: %5Bhttps://www.youtube.com/watch?v=YKYQ-z6A9Fs%5D(https://www.youtube.com/watch?v=YKYQ-z6A9Fs)
2. Claude code for PMs: %5Bhttps://www.youtube.com/watch?v=4nthc76rSl8%5D(https://www.youtube.com/watch?v=4nthc76rSl8)
3. Advanced claude code: %5Bhttps://www.news.aakashg.com/p/how-to-use-claude-code-like-a-pro%5D(https://www.news.aakashg.com/p/how-

to-use-claude-code-like-a-pro)

4. Claude code PM OS: %5Bhttps://www.news.aakashg.com/p/pm-os%5D(https://www.news.aakashg.com/p/pm-os)
5. Boris' interview: %5Bhttps://www.youtube.com/watch?v=PQU9o_5rHC4%5D(https://www.youtube.com/watch?v=PQU9o_5rHC4)

Every PM reading this should understand: the person who built the most successful AI coding tool in history just said your job title is the future of software. The question is whether you're building the judgment and technical fluency to claim it.

](https://substack.com/@aakas
217285355) 5. post by @tfadell 6. r/ProductManagement post by u/manonixx 7. r/prodmgmt post by u/AggravatingSlice1 8. r/prodmgmt comment by u/calciphus 9. r/prodmgmt comment by u/holyelvis 10. r/ProductManagement post by u/seasonh5 11. Inside modern game design - Cheryl Platz (Riot Games, Microsoft) 12. r/ProductManagement post by u/fiftyfirstsnails 13. post by @ttorres 14. post by @paulg 15. How to Work With Anyone 16. r/prodmgmt post by u/Desparate_Impostor 17. r/prodmgmt comment by u/mckirkus 18. r/ProductManagement post by u/varbinary 19. r/ProductManagement comment by u/Is_ItOn 20. r/ProductManagement comment by u/Sufficient-

Rough-647 21. r/ProductManagement comment by u/kreepykrally 22. r/prodmgmt post by u/Erynor_