Taste at speed, "GitHub for PM" context layers, and prototype-first workflows

PM Daily Digest

2026-03-06

Taste at speed, "GitHub for PM" context layers, and prototype-first workflows

By PM Daily Digest • March 6, 2026

This digest focuses on what changes when prototyping and code generation become cheap: PM leverage shifts toward fast judgment, context management, and quality measurement. It also includes practical playbooks for vibe prototyping, building persistent AI assistants (Gems/Projects), and a case-study-driven look at alignment, hardware constraints, and career tactics in AI-era product roles.

Big Ideas

1) When building gets cheap, the bottleneck becomes judgment ("taste at speed")

Teams are increasingly using AI to prototype so fast that the core constraint shifts from *"can we build it?"* to *"should we ship it?"* [1]. Aakash Gupta highlights Anthropic's Claude Code team as an extreme example: they build hundreds of working prototypes before shipping a single feature, with Boris Cherny reportedly shipping **20–30 PRs/day** across parallel Claude instances, and building "Cowork" in about **10 days** [2].

This shows up in the broader PM community too: one Reddit post describes moving from a weeks-long spec → align → build → measure loop [^2] to putting rough versions in front of clients the same day, shrinking feedback loops from **weeks to hours** [^2].

[1][The PM skill that matters in 2026 is taste at speed. Boris Cherny just showed everyone what that looks like.

[2][The PM skill that matters in 2026 is taste at speed. Boris Cherny just showed everyone what that looks like.

Why it matters: As prototyping cost collapses, PM leverage moves to rapid evaluation, ruthless focus, and decision quality—especially when stakeholders can react to a demo instead of a doc [^2].

How to apply (this week): - Run a prototype-first cycle: build a rough demo, test it, then document decisions *after validation* (not before) [^2]. - Treat the PRD as a *source of truth after learning*, not an authorization artifact [^2].

Vibe Coding: The New Product Team Superpower? (0:00)

---

2) Alignment is becoming an AI problem: "GitHub for product management"

Teresa Torres spotlights Momental's vision of a **"GitHub for product management"**: ingest org documents/transcripts/recordings and use AI agents to map them into a structured context layer, then surface **"merge conflicts" in strategy** (e.g., one team prioritizing retention while another prioritizes conversion) for humans to resolve [^3][^3].

Momental frames an internal "product chain" (signals → learnings → decisions → principles) and models org context as three trees (product tree, wisdom tree, people/time tree) [^3]. They emphasize metadata (who said it, when, and in what context) as critical for preventing hallucinations [^3].

Why it matters: Even if engineers ship faster, PMs still spend large amounts of time coordinating alignment; Momental cites the reality that you “don’t know what you don’t know” when conflicts are implicit or distributed [^4].

How to apply: - Treat misalignment like a first-class defect: explicitly track decisions *with reasoning* (not just outcomes), and make conflicts visible for resolution [^3]. - When adopting AI for org context, prioritize provenance/metadata over “just summarization” to reduce ambiguity [^3].

---

3) Accurate AI agents require domain knowledge + proprietary data, plus a hybrid architecture

In high-stakes domains (fintech, legal, healthcare), accuracy is “the product,” and out-of-the-box LLMs aren’t naturally reliable enough [^5]. Two advantages help close the gap:

- **Domain knowledge:** map workflows, stakeholders, and where a “90% answer” is acceptable vs. a failure [^5].
- **Proprietary data:** transaction-level data, interaction history, domain corpora for personalization and insights a general model can’t produce [^5].

On architecture, Lisa Huang recommends a **hybrid system**: LLMs (including multi-agent workflows) where they fit, but deterministic code where you need reliability and control [^5].

Why it matters: Without domain constraints, data advantage, and deterministic guardrails, teams can build fast but ship unreliable behavior in the places users care most [^5].

How to apply: - Before building: map tasks/subtasks and define explicit accuracy thresholds by step [^5]. - Build hybrid: identify components that must be deterministic and keep them in code [^5].

---

4) “Personalized AI” beats one-off chats: build persistent context assistants (Gems/Projects)

Lisa Huang argues the core issue with typical LLM usage is starting from scratch each chat—role, strategy, writing style, product history all reset [^5]. Gemini Gems (and analogous Claude Projects / custom GPTs) aim to retain context across work so you don’t re-brief every time [^5].

Why it matters: Persistent context makes AI useful as a daily collaborator for writing, strategy, and synthesis—not just a “glorified search engine” [^6].

How to apply: Start with three “foundation” assistants: - **Writing clone:** upload PRDs/emails/Slack messages for drafts in your voice [^5]. - **Product strategy advisor:** feed strategy docs, positioning, competitor analysis; use as a thought partner (not a replacement for judgment) [^5]. - **User research synthesizer:** upload transcripts/surveys/support tickets to extract themes you can't manually read at scale [^5].

Tactical Playbook

1) Prototype fast without derailing the org (vibe prototyping change management)

A practical framing: call prototypes what they are—*not deployable code*, but a substitute for a "clickable Figma," and pilot with one team first [^7].

Step-by-step: 1. **Prototype for yourself first:** expect to revise requirements **5–15 times** after seeing the first version and noticing what you forgot to specify [^7]. 2. **Bring it to the team:** use it to get engineering/design feedback without claiming it replaces their work [^7]. 3. **Use it for stakeholders:** prototypes create shared understanding; senior stakeholders often won't read PRDs, but they will react to a working flow [^7]. 4. **Then validate with users:** test with customers to learn quickly [^7].

Prompting discipline (avoid "degrees of freedom"): - Provide enough context so the tool doesn't guess across thousands of possibilities [^7]. - Include an **object model** (high-level entities/relationships) so the prototype isn't built on wrong assumptions [^7]. - Use a "Goldilocks" amount of context—too little causes wrong guesses; too much can overwhelm context windows [^7].

Build advice for speed: stay front-end as long as possible; delay auth/DB, and "fake it" with sample data (CSV/local storage) until needed [^7]. If you realize you're on the wrong path, restart ("nuke from orbit") because regenerating is cheap [^7].

---

2) A quick-start map for the "vibe coding" tool landscape

Dan Olsen's "Vibe Coding Spectrum" organizes tools from less technical (browser/UI-first) to more technical (IDE/CLI/code-first), with suggested entry points by role [^7]:

- **Designer-friendly:** Figma Make, Magic Patterns [^7]
- **PM-friendly:** Lovable, Bolt, Base44 [^7]
- **More technical:** Replit, V0 [^7]
- **Developer tools:** Cursor, GitHub Copilot (and others) [^7]

How to apply: start on the left where you can iterate quickly, then migrate right only when you hit constraints [^7].

---

3) Build a Gem (persistent copilot) with a PM-friendly workflow

Step-by-step: 1. **Write detailed instructions:** a full page of context (role, audience, format preferences). Avoid vague prompts like "help me write better" [^5]. 2. **Upload your knowledge files:** PRDs, emails, competitor teardowns, roadmaps; Gemini Gems rely strictly on instructions + files, so update the files as context changes [^5]. 3. **Iterate like a mini product:** refine instructions/knowledge over time [^5].

Lisa Huang's suggested scale: a PM may end up with ~**20** Gems/Projects across workflows [^6].

---

4) Measuring AI agents: a three-layer scoreboard (in order)

1. **Quality:** ask "is the AI doing what it's supposed to do?" via evals, human annotators, and LLM judges (each scales differently) [^5].
2. **Product metrics:** adoption, usage, retention, CSAT; also track qualitative signals (social, customer conversations, support tickets) [^5].
3. **Business impact:** revenue attribution, retention influence, ARR contribution—tracked consistently on the business scorecard [^5].

The sequence matters: jumping to business impact without a quality foundation is unstable measurement [^5].

Case Studies & Lessons

1) Building AI into hardware changes the design space (Meta Ray-Ban)

Lisa Huang describes constraints that "pure software" teams often don't face: weight, battery life, privacy, bystander concerns, and even partner pace differences (e.g., Luxottica vs. a Silicon Valley engineering org) [^5]. She flags an important trade-off: cloud processing is the default today, but on-device is positioned as the future—especially because "privacy wins over performance" for a device worn on your face all day [^5].

Takeaway: Don't "fall in love with the technology." The best AI products sit at the intersection of what users need and what the tech can reliably do today; build fast, observe behavior, and update assumptions [^5].

---

2) Standing out in AI PM interviews: do the work before you're asked

Aakash Gupta relays a hiring story from Lisa Huang: a candidate with zero AI experience stood out by watching **three hours of TikTok videos** from coaches working with small businesses, then bringing synthesized user needs into the interview. No other candidate did comparable pre-work [^6].

Takeaway: The differentiator wasn't AI credentials—it was initiative and user-centric research depth [^6].

---

3) Even agents need aligned context (Momental's pivot)

Momental's founders described building a "product team of agents" (developer agent, PM agent doing slides/sprint planning), but discovered the agents asked endless sensible questions—mirroring the same alignment problems real teams have. The insight: they hadn't solved alignment; they needed a context foundation first [^4].

Takeaway: Multi-agent systems can amplify the demand for clear shared context—speed doesn't remove coordination problems [^4].

---

4) Cheap code can lead to "shipping slop" unless strategy and focus stay sharp

Casey Winters argues code is now "incredibly cheap," which can become an excuse for a lack of strategic thinking about what's worth building [^8]. He describes incumbents "DDoS'ing" customers with too many features and notes that running multiple agents doesn't guarantee value—often it produces "slop" without product sense and business strategy [^8].

Takeaway: Higher build throughput increases the penalty for weak focus: customers get overwhelmed and teams lose clear signal on what's working [^8].

Career Corner

1) The PM role is shifting toward hybrid builders (judgment stays core)

Aakash Gupta's framing: AI won't replace PMs, but it will automate or accelerate execution work (PRDs, mocks, roadmaps, data pulls). Product judgment—deciding in ambiguity what's worth doing—remains core [^5]. Structural changes follow: PM-to-engineer ratios compress and PM expectations shift toward prototyping/design/coding enough to communicate intent [^5].

How to act on it: choose one build-adjacent skill (rapid prototyping, lightweight coding, or system prompt + eval design) and ship artifacts regularly [^5].

---

2) Breaking into AI PM: remove the "I don't work on AI" excuse

Gupta's roadmap includes: - Get direct AI experience in-role if possible; otherwise build on the side [^5]. - Invest in network and referrals (he emphasizes referrals still matter) [^5]. - Treat interview prep as a skill: practice out loud, get mocks, drill the format (product sense, execution, behavioral, case questions) [^5].

He also argues you don't need permission, budget, or a team to build AI products—consumer tools provide access to the same models many companies build on, and many companies aren't fine-tuning at all [^5].

---

3) Product-manage your career (and keep empathy as the strategy anchor)

Deb Liu recommends treating your career with the same intentionality PMs apply to product roadmaps [^9]. She also anchors product strategy in empathy—"vision without customer pain is theater" [^9][^9].

---

4) Job market signal (EU): Technical Project Manager (AI & Web Infrastructure), Frankfurt

A Frankfurt-based technology startup is hiring a **Technical Project Manager** to coordinate product strategy and execution for the European market and translate technical capabilities into market-ready products [^10]. Responsibilities include market/competitor research, structuring product priorities, coordinating development cycles, supporting validation/evaluation, and exploring AI-based workflow tools [^10][^10][^10][^10][^10]. The post lists requirements like CS/technical background, web/cloud fundamentals, structured thinking, and interest in emerging AI tools [^10][^10][^10][^10]. Apply via **careers@novada.com** [^10].

Tools & Resources

- **Claude Code webinar recording (Sachin Rekhi):** Rekhi hosted a live session with **1,500 PMs**, covering why he views Claude Code as highly productive for PMs, showing 13 automation skills, and walking through setup (editors/terminals/voice tools) [^11][^11][^11][^11]. Video link: https://www.youtube.com/watch?v=zsAAaY8a63Q [^11].
- **Gemini Gems masterclass (Lisa Huang):** Podcast episode URL: https://www.news.aakashg.com/p/lisa-huang-podcast [^5]. (Key build steps: detailed instructions, upload knowledge, iterate) [^5][^5][^5].

- **Vibe Brief template + tool starting point:** Dan Olsen recommends starting with Lovable by default and sharing a lightweight "vibe coding brief" at "bitly slash vibebrief" [^7].
- **AI PM feedback loop (community writeup):** Reddit thread link includes: https://www.clawrapid.com/en/blog/ai-pm-feedback-loop [^2]. One described workflow: rough requirements doc for Claude → prototyping → experimenting → PRD (source of truth) → ship [^12].
- **A caution on "auto-invoked" AI skills:** Rekhi noted that installing an auto-invoked "frontend-design" skill made his monthly NPS trend visualizations harder to read, and he prefers skills he can invoke manually [^13][^13].

His Claude Code team at Anthropic doesn't write PRDs. They build hundreds of working prototypes before shipping a single feature. Boris personally ships 20-30 PRs a day running 5 parallel Claude instances. They built Cowork, a full product for non-engineers, in about 10 days.

Everyone is debating whether PRDs should die. They shouldn't in big companies and established products, where understanding the hypothesis and success metrics is critical. But the real question is: what happens to the PM who can't evaluate 15 prototypes and pick the 3 worth shipping?

Because here's what changes when building costs near zero: the bottleneck moves from "can we build it" to "should we ship it." PRDs existed because building was expensive and you needed sign-off before committing resources. When a prototype takes 45 minutes instead of 6 weeks, nobody needs a document to authorize exploration. They need someone who can look at working software and say "this one, not that one" in real time. If anything, the PRD now comes after the prototype.

On the Claude Code team, PMs code. Data scientists code. User researchers code. In fact, now everyone has the same title ("Member of Technical Staff"), and it's by design. Boris said productivity per engineer grew 70% even as Anthropic tripled in headcount. The coordination cost of translating specs into code disappears when everyone can build. And that changes what a PM is actually good for.

Boris said it himself: "There's just no way we could have shipped this if we started with static mocks and Figma or if we started with a PRD." The old process would have spent more calendar time documenting Cowork than his team spent building it.

This is the Claude Code team today. It will be most fast-moving teams within 18 months. The PMs who thrive will be the ones reviewing prototypes at 9am, killing 80% of them by noon, and shipping the survivors by end of week. Pattern matching across user research, technical feasibility, and business model simultaneously while staring at working software.

So here's how to get there:

1. Understand Modern PRDs: %5Bhttps://www.news.aakashg.com/p/ai-prd%5D(https://www.news.aakashg.com/p/ai-prd)
2. Master Claude Cowork: %5Bhttps://www.news.aakashg.com/p/you-should-be-using-claude-cowork%5D(https://www.news.aakashg.com/p/you-should-be-using-claude-cowork)
3. Learn Claude Code: %5Bhttps://www.youtube.com/watch?v=4nthc76rSl8%5D(https://www.youtube.com/watch?v=4nthc76rSl8)
4. Build your OS: %5Bhttps://www.news.aakashg.com/p/pm-os%5D(https://www.news.aakashg.com/p/pm-os)
5. Watch Boris' interview with @Gergely Orosz: %5Bhttps://www.youtube.com/watch?v=julbw1JuAz0%5D(https://www.youtube.com/watch?v=julbw1JuAz0)

The PMs who struggle will be the ones still writing 15-page specs for features that could be prototyped, tested, and validated before the doc hits its first review cycle. Taste at speed is the new moat.

Boris Cherny

Creator of Claude Code ($2.5B+ ARR)

On building Claude Cowork **in 10 days**:

“There’s **just no way** we could have shipped this if we started with **static mocks in Figma** or if we **started with a PRD**.”

Repost this

](https://substack.com/@aakas
223484179) [^2]: r/prodmgmt post by u/Itchy-Following9352 [^3]: post by @ttorres [^4]: Building GitHub for Product Management: How Momental Uses AI to Find Merge Conflicts in Strategy [^5]: Gemini Gem Masterclass From the Creator Lisa Huang [^6]: If you aren’t using Gems or Projects, you’re not using AI. **You’re using a glorified search engine.** [^7]: Vibe Coding: The New Product Team Superpower? [^8]: How SuperMe is Building the Professional Network for the AI Era [^9]: 50 Lessons From 50 Years Lived [^10]: r/ProductManagementJobs post by u/CleanButterfly4532 [^11]: post by @sachinrekhi [^12]: r/prodmgmt comment by u/jetf [^13]: post by

@sachinrekhi

Sources

1. [The PM skill that matters in 2026 is taste at speed. Boris Cherny just showed everyone what that looks like.

His Claude Code team at Anthropic doesn't write PRDs. They build hundreds of working prototypes before shipping a single feature. Boris personally ships 20-30 PRs a day running 5 parallel Claude instances. They built Cowork, a full product for non-engineers, in about 10 days.

Everyone is debating whether PRDs should die. They shouldn't in big companies and established products, where understanding the hypothesis and success metrics is critical. But the real question is: what happens to the PM who can't evaluate 15 prototypes and pick the 3 worth shipping?

Because here's what changes when building costs near zero: the bottleneck moves from "can we build it" to "should we ship it." PRDs existed because building was expensive and you needed sign-off before committing resources. When a prototype takes 45 minutes instead of 6 weeks, nobody needs a document to authorize exploration. They need someone who can look at working software and say "this one, not that one" in real time. If anything, the PRD now comes after the prototype.

On the Claude Code team, PMs code. Data scientists code. User researchers code. In fact, now everyone has the same title ("Member of Technical Staff"), and it's by design. Boris said productivity per engineer grew 70% even as Anthropic tripled in headcount. The coordination cost of translating specs into code disappears when everyone can build. And that changes what a PM is actually good for.

Boris said it himself: "There's just no way we could have shipped this if we started with static mocks and Figma or if we started with a PRD." The old process would have spent more calendar time documenting Cowork than his team spent building it.

This is the Claude Code team today. It will be most fast-moving teams within 18 months. The PMs who thrive will be the ones reviewing prototypes at 9am, killing 80% of them by noon, and shipping the survivors by end of week. Pattern matching across user research, technical feasibility, and business model simultaneously while staring at working software.

So here's how to get there:

1. Understand Modern PRDs: %5Bhttps://www.news.aakashg.com/p/ai-prd%5D(https://www.news.aakashg.com/p/ai-prd)

2. Master Claude Cowork: %5Bhttps://www.news.aakashg.com/p/you-should-be-using-claude-cowork%5D(https://www.news.aakashg.com/p/you-should-be-using-claude-cowork)
3. Learn Claude Code: %5Bhttps://www.youtube.com/watch?v=4nthc76rSl8%5D(https://www.youtube.com/watch?v=4nthc76rSl8)
4. Build your OS: %5Bhttps://www.news.aakashg.com/p/pm-os%5D(https://www.news.aakashg.com/p/pm-os)
5. Watch Boris' interview with @Gergely Orosz: %5Bhttps://www.youtube.com/watch?v=julbw1JuAz0%5D(https://www.youtube.com/watch?v=julbw1JuAz0)

The PMs who struggle will be the ones still writing 15-page specs for features that could be prototyped, tested, and validated before the doc hits its first review cycle. Taste at speed is the new moat.

Boris Cherny

Creator of Claude Code ($2.5B+ ARR)

On building Claude Cowork **in 10 days**:

“There’s **just no way** we could have shipped this if we started with **static mocks in Figma** or if we **started with a PRD**.”

Repost this

](https://substack.com/@aakas
223484179) 2. r/prodmgmt post by u/Itchy-Following9352 3. post by @ttorres 4. Building GitHub for Product Management: How Momental Uses AI to Find Merge Conflicts in Strategy 5. Gemini Gem Masterclass From the Creator Lisa Huang 6. If you aren’t using Gems or Projects, you’re not using AI. **You’re using a glorified search engine.** 7. Vibe Coding: The New Product Team Superpower? 8. How SuperMe is Building the Professional Network for the AI Era 9. 50 Lessons From 50 Years Lived 10. r/ProductManagementJobs post by u/CleanButterfly4532 11. post by @sachinrekhi 12. r/prodmgmt comment by u/jetf 13. post by @sachinrekhi