Production-Ready GenAI, Faster Discovery, and the Agentic PM Role

PM Daily Digest

2026-03-18

Production-Ready GenAI, Faster Discovery, and the Agentic PM Role

By PM Daily Digest • March 18, 2026

This issue focuses on a five-pillar framework for shipping GenAI products, the shift toward agentic PM workflows, and practical playbooks for faster discovery, metric triage, and adoption. It also covers two detailed case studies—Amazon collaboration spaces and LennyRPG—and closes with career signals for discovery skills, interviews, and freelance positioning.

Big Ideas

1) Production-ready GenAI is a systems problem, not a feature problem

The strongest framework this week comes from a Product School talk by an Amazon AI product leader, who cites Gartner's estimate that **85% of AI projects never make it to production** and argues that the gap is usually caused by missing system design, not missing features [1]. The proposed five-pillar framework is:

1. **User-centric design** grounded in real pain points and jobs-to-be-done [1]
2. **Robust evaluation** across trust, usefulness, adoption, and business impact—not accuracy alone [1]
3. **Governance and safety** with guardrails, transparency, and compliance built in from the start [1]
4. **Scalable architecture** for performance, cost, reliability, and extensibility [1]
5. **Adoption strategy** with pilots, enablement, community, and feedback loops [1]

> "AI products succeed when PMs design systems and not features" [1]

Why it matters: This reframes the PM job for GenAI from shipping a capability to designing the full operating system around that capability—evaluation, trust, scale, and adoption included [1].

How to apply: Before calling a GenAI initiative "ready," force a launch review that answers five questions: what user job is being solved, how success will be measured, what guardrails exist, how the system scales, and how adoption will be driven after launch [1]. The same speaker's guidance is to **move fast, but build right**, with 3–6 months achievable when teams do the upfront work that avoids re-architecture later [1].

2) The PM operating system is shifting from meetings and docs to loops, logs, and simulations

Andrew Chen argues that in an agentic world, the product role splits into two jobs: organizing **humans** and organizing **agents** [2]. In his framing, standups become anomaly and run-log reviews, OKRs become continuous agent-based grading, PRDs give way to living agentic loops, and product reviews become simulations that test agent behavior under different constraints [2].

This future-facing view also fits the more current GenAI PM description from the Amazon talk: PMs already bridge AI capabilities to user problems, shape ethical and trustworthy AI use, and align technical and non-technical stakeholders [1].

Why it matters: The PM surface area is expanding from persuasion and coordination into instrumentation—prompts, evals, workflows, feedback loops, and behavior review [2].

How to apply: Treat agent behavior as something that needs product management, not just engineering. Build explicit prompts and evals, review deltas and failures, and make simulation or scenario testing part of pre-launch review for agentic systems [2].

3) As AI speeds up engineering, discovery is becoming the bottleneck

Sachin Rekhi's concise diagnosis: **engineering velocity has 10x'd with AI coding tools, but customer discovery hasn't kept pace**, so PMs are increasingly the constraint in deciding what to build, how to design it, and how to validate it before shipping [3].

He responds with **10 AI-powered discovery workflows** spanning surveys, feedback streams, interview scripting, interview synthesis, AI-moderated interviews, prototype-based discovery, metrics analysis, and automated metric analysis [3].

Why it matters: Faster build loops create more pressure on PMs to improve discovery throughput and decision quality, not just documentation quality [3].

How to apply: Audit your current discovery flow and identify the slowest step. Then add AI support there first—survey analysis, interview synthesis, prototype discovery, or metric analysis—rather than trying to automate everything at once [3].

4) Messy docs can be a strength—if you design a clean interface to the organization

The Beautiful Mess argues that many high-performing product teams rely on **freeform, manually migrated documents** filled with links, flags, checklists, copied data, comments, and repeated context [4]. The point is not formal structure; it is externalizing working memory so teams can reason through customer signals, hypotheses, dependencies, and half-formed ideas together [4].

The tension is that teams need local emergence, while organizations still need legibility about progress, risks, and focus [4]. The preferred answer in the essay is not to eliminate the mess, but to design **intentional interfaces**: the smallest shared routines, objects, and language that let the rest of the org understand what is happening without crushing the frontline work [4].

Why it matters: PM teams often over-rotate toward official artifacts and lose the sense-making layer where important work actually happens [4].

How to apply: Let teams keep their working scratchpad, but define a minimal interface outward: a small set of recurring rituals, a few shared objects, and consistent language for status, risks, and decisions [4].

Tactical Playbook

1) Turn AI discovery into a repeatable four-stage workflow

A practical way to apply Rekhi's 10 workflows is to map them into four stages:

1. **Collect signals**: analyze customer surveys, automate survey programs, and automate feedback rivers [3]
2. **Run interviews faster**: generate interview scripts, conduct AI-moderated interviews, and synthesize interview feedback [3]
3. **Test concepts**: use prototypes for discovery and, where useful, generate synthetic user feedback [3]
4. **Close the loop with numbers**: analyze metrics and automate metric analysis [3]

Why it matters: This creates a discovery pipeline that can better match faster engineering cycles [3].

How to apply: Start with one workflow per stage. For example, automate survey analysis, draft interview guides with AI, test concepts via prototypes, and then automate recurring metric readouts [3].

2) When a metric drops, require diagnosis before brainstorming

A useful community workflow for analytics triage starts with a strict rule: the agent does **not** get to brainstorm until it can identify **where the delta is coming from**—specifically the step, segment, and time period involved [5]. Only after that first pass does it move to generating **2–3 experiments** and a **tracking checklist**, with every idea mapped to a measurable metric [5].

Why it matters: It prevents the common PM pattern of spending 30 minutes in dashboards without notes, structure, or a clear next step [5].

How to apply: Standardize a three-step triage: - First answer: **what changed, where, and since when** [5] - Then propose: **2–3 experiments** tied to the diagnosed step or segment [5] - Then create: a **tracking checklist** so engineering gets a concrete handoff and each idea is measurable [5]

The same thread raises two good discipline questions for teams to adopt: what are your first three checks when a metric drops, and do you document what you ruled out or rediscover it every time [5]?

3) Plan adoption before launch, not after it

The Amazon talk is especially strong on adoption mechanics. The suggested sequence is:

1. **Run pilots with a defined population** and gather real feedback before global launch [1]
2. **Build success stories** so launch materials show concrete use cases, not just product claims [1]
3. **Invest in documentation, tutorials, and training** so users can self-serve and leaders understand the rollout [1]
4. **Create a community** where users can share tips, ask questions, and report issues [1]
5. **Maintain a transparent roadmap** and keep feedback loops active after launch [1]

Why it matters: The speaker explicitly argues that building is only half the battle; without discoverability, enablement, and change management, even strong AI products fail to get used [1].

How to apply: Add adoption work to the launch checklist itself—pilots, champions, docs, training, community, and roadmap visibility—rather than treating them as marketing extras [1].

Case Studies & Lessons

1) Amazon collaboration spaces: a full-stack GenAI rollout

Problem: Teams across Amazon needed AI systems with their own knowledge bases, documents, settings, and tools; generic systems did not understand team-specific context [1].

Product decision: The team built **collaboration spaces** where teams could upload documents, customize prompts, integrate with other Amazon tools, and control access and permissions [1]. They validated the concept with user research before writing code, built evaluation in from day one, treated governance and safety as core features, architected for scale, and paired the product with pilots, documentation, and community [1].

Outcomes: The rollout went from an initial **12–18 month** timeline to **six months from concept to global launch** [1]. Reported results included **40–50% faster prompt creation**, **3x higher engagement** for role-specific content, **2x retention** on repeat user rates, and **five major feature announcements in the first two months post-launch** because the architecture was extensible [1].

Key takeaway: Enterprise GenAI speed came from doing more product work upfront, not less—especially on evaluation, governance, architecture, and adoption [1].

2) LennyRPG: how a non-technical product designer used AI to build a real product

Ben Shi, a non-technical product designer at Miro, built **LennyRPG**, a Pokémon-style RPG based on Lenny's Podcast, as an AI-assisted product build [6]. The process is notable because it mirrors classic product development more than "prompt and pray" building:

1. **Define the core idea** and visualize it for the AI when the product is highly visual [6]
2. **Create a PRD** by having the AI interview the creator, then synthesize answers and artifacts into a single source of truth [6, 7]
3. **Build a POC** around the core loop first [6, 7]
4. **Pivot fast when the stack is wrong**—from RPG-JS to Phaser when the framework fought the quiz-based design [7]
5. **Systematize repetitive work** with CLI tools for quiz generation and avatar creation across hundreds of episodes [7, 6]
6. **Polish and ship** with AI-assisted QA and UI cleanup [7]

Two lessons stand out. First, Shi says getting the **core idea and PRD right determines 80% of how smooth the rest of the build will be** [7, 6]. Second, the early validation was intentionally lightweight: he shared the POC internally to see whether people understood what to do, whether the core loop made sense, and whether it felt fun rather than like work [7].

Key takeaway: AI can accelerate implementation and batch work, but the hard product choices—concept clarity, framework fit, game balance, and what "good" feels like—still required deliberate PM judgment [6].

Career Corner

1) Discovery is becoming a career-defining PM skill

If engineering velocity is increasing much faster than discovery velocity, PM leverage shifts toward faster learning, not just faster execution [3]. Rekhi's 10-workflow list is a useful skills map for PMs who want to stay ahead: survey analysis, feedback automation, interview design, interview synthesis, prototype discovery, and metrics automation [3].

How to apply: Pick one discovery workflow you do repeatedly and learn how to speed it up with AI this quarter [3].

2) Open-ended PM interviews are testing structured thinking under ambiguity

One candidate described repeatedly failing the brainstorming stage of PM interviews despite positive feedback on energy and bias to action [8]. The examples were deliberately broad: propose three products after a data breach, explain how Spotify recommendations work, or organize a folder so others can navigate it easily [8]. The thread's core question was whether experienced PMs rely on a specific framework or thought process in these situations [8].

What to take from it: These rounds appear to reward legible reasoning and repeatable structure, not just raw creativity [8].

How to apply: Practice unfamiliar prompts and focus on making your reasoning easy to follow—problem framing, assumptions, options, and trade-offs—rather than trying to sound instantly brilliant [8].

3) Community signal: freelance PM work may be easier to win as concrete delivery work

In one Product Management thread, a PM with strong **0→1**, **1→10**, and AI prototype-building experience is exploring freelancing while building a portfolio of small AI projects and apps [9]. A reply says Upwork still has some good opportunities, but few are true freelance PM roles; more are narrow tasks such as analytics configuration or effectively full-time work routed through the platform [10].

What to take from it: The clearer the deliverable, the easier the market fit may be for freelance PM work in today's environment [9, 10].

How to apply: If you are testing freelance PM work, package yourself around concrete outcomes—MVPs, prototypes, analytics setup, or specific product problem-solving—rather than a generic "fractional PM" label [9, 10].

Tools & Resources

- Lean Product Meetup recording — Sachin Rekhi's walkthrough of 10 AI-powered discovery workflows [3]
- Building Production-Ready GenAI Products — five-pillar framework plus enterprise GenAI examples from Amazon [1]
- How I built LennyRPG — written breakdown of the AI-assisted six-step build process [7]
- LennyRPG video walkthrough — concise tool stack: Miro, ChatGPT, Claude Code, Cursor, GPT-4, Phaser, Supabase, and Vercel [6]
- ClawRapid analytics skills directory — analytics connectors referenced in the metric-drop workflow [5]

Sources

1. Building Production-Ready GenAI Products | Amazon AI Product and Technology Leader
2. X post by @andrewchen
3. X post by @sachinrekhi
4. TBM 411: Messy Docs As Helpful Pattern
5. r/prodmgmt post by u/Itchy-Following9352
6. How I built LennyRPG
7. How I built LennyRPG
8. r/ProductManagementJobs post by u/Visible_Substance569
9. r/ProductManagement post by u/Old_Leshen
10. r/ProductManagement comment by u/chakalaka13