Context Layers and Expert Workflows Become Core PM Work

PM Daily Digest

2026-05-13

Context Layers and Expert Workflows Become Core PM Work

By PM Daily Digest • May 13, 2026

This issue focuses on a shift from generic AI usage to explicit system design: PMs are deciding memory, workflow context, and skill routing while translating expert judgment into reusable playbooks. It also covers partial-data MVPs, operational triage, AI hiring signals, and concrete resources to try.

Big Ideas

1) Context is becoming a product surface

Two separate threads point to the same shift: AI systems break when they lack the right context. In agent products, continuity does not appear automatically; by default, agents forget prior turns unless teams intentionally design memory using the context window, a task-level scratchpad, and a cross-session vector store [1]. In enterprise software, Scribe argues the equivalent problem is missing workflow context: AI cannot see thousands of workflows or trace the decisions behind them, so it produces generic or wrong output [2]. Scribe says its answer is a new context layer that maps how work gets done, and it reports crossing **$100M ARR** with **90,000 enterprise customers**, including nearly half the Fortune 500 [2].

- **Why it matters:** Without continuity or workflow context, AI cannot reliably act on behalf of users or inside an organization [1, 2].
- **How to apply:** Separate **session context**, **task memory**, and **cross-session memory** up front, then decide what parts of the business workflow need to be legible to both humans and agents [1, 2].

2) Encoded judgment beats generic prompting

Julie Zhuo describes AI product building as turning analysts' expert art into explicit playbooks or skills for an LLM: how to inspect a metric move, separate signal from noise, and decide whether a product change actually moved the needle [3]. Her lesson is that the jump from acceptable to excellent output is rarely a handful of big rules.

> "The gap between 70% quality and 95% quality is not 3 or 4 big things. It's more like 100s of small things." [3]

Aakash Gupta makes the same point from a tooling angle: Claude chooses skills from the **name and description** alone, so overlapping skills fail unless the routing logic is explicit [4]. Sachin Rekhi's guide extends this into PM practice: his 15+ AI workflows are the ones that survived a year of daily use, span both macro work like strategy and customer discovery and micro work like status updates and slides, and include failures as well as wins [5].

- **Why it matters:** Better AI output depends on externalizing expert taste, boundaries, and reusable workflows, not just writing longer prompts [3, 4].
- **How to apply:** Document what great practitioners actually notice, define where one skill should stop and another should start, and keep refining the workflows that hold up in daily use [3, 4, 5].

3) Simplicity is still a strategic advantage

Tony Fadell's reflection on the iPod is a reminder that product strategy is not getting easier just because technology is more capable. He says the goal was for the technology to disappear so people could have their music anywhere: "1,000 songs in your pocket" [6]. His broader lesson is that constraints create freedom, removing features often creates a better product than adding them, and the best technology understands when to step back [6].

- **Why it matters:** New capability can easily turn into more screens, menus, and notifications unless teams keep subtracting [6].
- **How to apply:** When evaluating an AI feature, ask whether it reduces complexity for the user or simply adds another layer to manage [6].

Tactical Playbook

1) Build agent memory in three layers

1. Use the **context window** for what the model needs right now, knowing it has a token limit and older turns disappear when the session gets long or ends [1].
2. Add a **scratchpad** for task-level working memory so the agent can track what it has already done across tool calls or multiple steps [1].
3. Add a **vector store** for cross-session continuity by summarizing important user facts, storing them with a user ID, and retrieving them next session

[1].

4. Treat the summary rules as a PM decision: what gets captured, what gets ignored, and when it should be pulled back into context [1].

Why this matters: The default behavior is forgetting. Continuity only exists if the product defines it [1].

2) Rewrite skill descriptions for routing, not documentation

1. Start by listing where skills overlap, such as `/weekly-review` versus `/stakeholder-update` or `/activation-analysis` versus `/retention-analysis` [4].
2. Assume Claude will inspect only the **skill name and description** before deciding relevance [4].
3. Put **3 trigger phrases**, a clear boundary, and an explicit alternative such as use `/Y` instead in the description [4].
4. Spend effort on the first two sentences of routing logic before polishing long instruction bodies [4].

Why this matters: A perfect skill body is effectively invisible if the description does not make the routing decision obvious [4].

3) Launch an MVP with partial data without hiding the trade-off

1. Clarify the launch scope first: is this for friendlies or the full customer base, and what are the consequences of missing or bad data [7]?
2. If the goal is learning, use a **high-confidence, representative subset** rather than waiting for full completeness [8, 9].
3. Design the system for future scale and let workflows degrade gracefully when edge cases are missing [8, 9].
4. Add post-launch feedback loops so coverage gaps surface quickly through customer feedback, support signals, and usage patterns [8].
5. Keep the roles of discovery and MVP clear: discovery helps scope the opportunity, while the MVP reveals actual demand and behavior [10, 11].

Why this matters: The trade-off is not simply 70% versus 100%; it is speed of learning versus the risk of designing too tightly around incomplete data [8, 9].

4) Make unplanned work visible on Kanban boards

1. Separate **outcome work** from **unplanned work** instead of mixing both into one queue [12].
2. Use **Epic containers** or an **Expedite bucket** so the team can see what is interrupt-driven versus planned [12].
3. Set up the board to support refinement, triage, and clear priority signals [12].

Why this matters: On teams like DevSecOps, operational security work can derail outcome work unless the board makes the trade-off visible [12].

5) Find early adopters where they already gather

1. Go beyond friends and coworkers when you need signal from a niche audience [13].
2. Start with digital communities that already organize around the problem space, such as relevant Facebook groups or subreddits [14].
3. Add physical environments, like dog parks in the example discussed, when in-person interviews are feasible [14].

Why this matters: Early validation gets stronger when the audience self-selects around the problem rather than your personal network [13, 14].

Case Studies & Lessons

1) Scribe: a workflow-context thesis at $100M ARR

Scribe says it has grown from a **$7K first deal** to **$100M ARR**, with **90,000 enterprise customers** and adoption in nearly half the Fortune 500 [2]. Alongside that milestone, the company argues that enterprise AI needs a context layer that makes workflows legible; otherwise models cannot see the recipes of work and become generic or confidently wrong [2, 15].

Key takeaway: If AI is being added to an enterprise product, workflow context may matter as much as model quality [2].

2) TeamSundial: distilling analysts' judgment into LLM skills

Julie Zhuo says a large share of the team's time now goes into converting analysts' tacit judgment into LLM playbooks and skills [3]. She identifies two bottlenecks: seeing what better performance looks like, and then systematically articulating it for a model [3]. That is why the last mile from decent to great quality is made up of hundreds of small judgments rather than a few obvious prompt tweaks [3].

Key takeaway: If you want AI products to outperform generic tools, invest in capturing expert standards explicitly, not just adding another model call [3].

3) iPod: make the technology disappear

Tony Fadell frames the iPod's success around a simple goal: technology should disappear into the experience, not compete with it [6]. His follow-on point is especially relevant to AI products: not every problem needs another screen or menu, and removing features can create a better product than adding them [6].

Key takeaway: Product progress can come from subtraction when new capability threatens to make the experience heavier [6].

Career Corner

1) AI fluency is showing up in PM hiring, but depth is still a choice

One PM publication argues that, in 2026, top PMs at Meta, Google, and AI-native companies treat **Claude Code mastery** as table stakes, and says hiring loops now ask how candidates use AI day to day, include prototype rounds from PRD to evals, and value portfolios with AI-built agents or prototypes [16]. Teresa Torres offers a useful counterbalance: the product-builder trend is a tool, not a mandate, and enjoyment and skill should guide who on a team leans into it [17].

> "It's a tool in our toolbox. We can decide who on our team has fun with it, wants to do it, wants to contribute." [17]

How to use this: Be ready to show at least one concrete AI artifact or workflow, and be equally clear about where your contribution is strongest [16, 17].

2) High agency has upside and cost

Shreyas Doshi says he is a huge fan of high agency while also emphasizing two realities: not everyone can develop it, and it brings major professional upside alongside chronic anxiety in situations you cannot control [18].

How to use this: Recognize the upside-cost trade-off when evaluating your own working style [18].

Tools & Resources

- Sachin Rekhi's "How I use AI as a product manager": useful when you want PM-native AI workflows rather than generic examples; it covers **15+** daily workflows, spans macro and micro tasks, includes failures, and argues the practice can sharpen product thinking [5].
- Aakash Gupta's note on skill descriptions: useful when overlapping Claude skills keep producing the wrong artifact; it gives a simple routing pattern built around trigger phrases, boundaries, and explicit alternatives [4].
- Claude Code for PMs: The Beginner's Guide: useful as a fast start if you want a working setup; it begins with Visual Studio Code, the Claude Code extension, and `/login`, then offers a downloadable template with preconfigured plugins, skills, and a minimal `CLAUDE.md`. The author says the setup can cut costs by at least **50%** and work with free frontier models [16].
- The same Product Compass post also lists upcoming session topics that may be useful for structured PM upskilling, including **AI skills every professional needs in 2026**, **How to use Claude to get your dream job**, **Building your AI Chief of Staff from scratch**, **How to Build Agentic Products**, and **The context engineering and agentic memory** [16].

Sources

1. r/prodmgmt post by u/InfamousInvestigator
2. X post by @scribeceo
3. X post by @joulee
4. substack
5. X post by @sachinrekhi
6. X post by @tfadell
7. r/ProductManagement comment by u/londongastronaut
8. r/ProductManagement post by u/Humble-Pay-8650
9. r/ProductManagement comment by u/SynthBoard
10. r/ProductManagement comment by u/ConstantKooky3329
11. r/ProductManagement comment by u/Nottabird_Nottaplane
12. r/prodmgmt post by u/Tayto95
13. r/ProductManagement post by u/AdIntelligent495
14. r/ProductManagement comment by u/AdOrganic299
15. X post by @scottbelsky
16. Claude Code for PMs: The Beginner's Guide
17. X post by @ttorres
18. X post by @shreyas