Automation Boundaries, Compounding AI Skills, and Doist's Ramble Playbook

PM Daily Digest

2026-04-17

Automation Boundaries, Compounding AI Skills, and Doist's Ramble Playbook

By PM Daily Digest • April 17, 2026

A practical framework for where AI actually helps PM teams, a detailed case study on how Doist shipped Todoist's voice-to-task feature, and concrete guidance on launch diagnosis, discovery, and career management.

Big Ideas

1) Automate friction, not glue work

AI is most useful when a behavior is already understood and valued, but too painful or easy to deprioritize—tailoring release notes, keeping status documents current, or cross-referencing risk logs [1]. The same framework warns that AI is a poor substitute for glue work, because the visible artifact is only part of the job; the real work includes timing, legitimacy, translation across functions, informal feedback loops, and surfacing hidden coordination problems [1].

Why it matters: PM teams can reduce real friction with AI, but they can also create convincing artifacts that nobody trusts or uses if they automate the output and ignore the judgment behind it [1].

How to apply: - Diagnose whether the bottleneck is **capability, opportunity, or motivation** before automating [1]. - If the real barrier is unclear quality standards, missing ownership, or social risk, fix the system before adding AI [1]. - After AI makes production easy, check the next bottleneck: who consumes the output, and who acts on it [1].

2) The compounding AI skill for PMs is question volume

> "I will ask Claude about anything I don't understand and ask it to teach it to me." [2]

Aakash Gupta's takeaway from Hannah Stulberg's **1,500 hours** in Claude Code is that task-based AI usage plateaus, while using AI for unknowns compounds [2]. The differentiator is not prompt cleverness; it is how often you reach for the tool when you hit something you do not understand [2]. He also notes that many PMs quit too early, while the compounding starts around hour 20 rather than day one [2].

Why it matters: In technical product roles, self-serve learning can change your slope faster than another prompt template.

How to apply: - Ask AI to review recent chats and identify your prompting blind spots [2]. - Use Socratic teaching for technical systems you use but do not fully understand [2]. - Stress-test a PRD by asking for skeptical engineering pushback before the real review [2].

3) AI shortens the middle of the work and raises the bar on taste

Andrew Chen's observation is that AI-assisted workflows reduce time spent on the middle 80% of the work and push more of the saved time to the first 10% and last 10%—idea generation, validation, and iteration [3]. PMs in technical domains describe the same pattern in practice: using AI for rapid prototypes and diagrams to validate customer feedback the same day, debating why a strategy will or will not work, and translating technical concepts into plain language for GTM and enablement audiences [4, 5, 6].

Why it matters: Faster execution does not reduce the need for product judgment; it concentrates it at the beginning and end of the workflow [3].

How to apply: Spend less effort on first-draft production, and more on framing the problem, validating the output, and adapting the story for different stakeholders [3, 6].

Tactical Playbook

1) Run a pre-automation checklist before handing work to AI

1. Start with the behavior, not the tool: what should be happening today that usually does not? [1]
2. Ask whether people actually know what *good* looks like. If not, it is a capability problem [1].
3. Ask whether the workflow, incentives, and ownership support the behavior. If not, it is an opportunity problem [1].
4. Ask whether people believe in the behavior or identify with doing it. If not, motivation is the constraint [1].
5. Respect the *iceberg*: the visible artifact may be only 20% of the underlying work [1].
6. Before declaring success, identify the next wall after generation becomes cheap [1].

Why it matters: This sequence helps PMs separate work that AI can legitimately accelerate from work that still depends on human judgment and organizational design.

2) Diagnose a weak launch by connecting six signals

When a launch underperforms, practitioners point to six signal sources: **sales calls, win/loss notes, support tickets, customer success conversations, product usage, and scattered internal feedback** [7].

A practical sequence: 1. Pull all six signals into one view [7]. 2. Decide who owns connecting them [7]. 3. Explicitly rank which signals you trust most [7]. 4. Ask where the truth usually gets lost [7]. 5. Ask how long it takes before you feel confident in the real cause [7]. 6. If you are **PLG/self-serve**, use instrumentation to locate the drop in success metrics like activation [8]. 7. If you are **sales-led**, separate **usage vs. adoption vs. exposure**, then use qualitative research to learn why [8].

Why it matters: Product usage rarely explains a weak launch on its own; the diagnosis often lives in the gaps between signals [9, 8].

3) Design natural-language features for correction, not perfection

Doist's Ramble offers a clear execution pattern for AI capture workflows: 1. Keep the model's toolset narrow: **add task, edit task, delete task** [10]. 2. Process input while the user is still speaking so partial results appear immediately [10]. 3. Let users correct the output in real time, instead of waiting for a perfect final pass [10]. 4. Use non-verbal confirmation cues for low-attention contexts like driving [10]. 5. Inject the current date and user context such as projects and labels, rather than expecting the model to infer account-specific state [10]. 6. Test across languages, accents, and recording conditions, then use the evals mainly to catch regressions [10].

Why it matters: Natural-language interfaces will make mistakes. A product that is easy to correct can still feel trustworthy.

Case Studies & Lessons

1) Doist's Ramble is a strong example of constrained AI product design

Doist describes Ramble as Todoist's first pure AI feature, but it was built on an older product strength: fast task capture and years of natural-language parsing [10]. The team did not start with a fixed Ramble feature. It ran a two-to-three month AI exploration, then converged on Ramble after research exposed a real behavior: some users brainstorm tasks on paper or with ChatGPT voice before committing them to Todoist [10]. Ramble emerged because it solved that brain-dump use case, not because the team wanted AI in the product for its own sake

[10].

On the product side, the architecture stayed narrow. The app streams raw microphone audio to a backend service, forwards it through Google Vertex, and lets the model make real-time tool calls while the user is still speaking. There is no conversational text or audio response from the model—only task actions surfaced in the UI [10]. For project and label matching, Doist found that directly injecting the user's project list into the prompt worked well enough and avoided the latency and complexity of extra calls or a RAG pipeline [10].

Building Todoist Ramble: How Doist Turned Voice Braindumps into Real-Time Task Capture (11:31)

The UX was built around confirmation and correction. Tasks appear live on screen, users can edit as they go, and audio cues confirm adds and edits for people using the feature while driving [10]. The team explicitly chose not to use text-to-speech because of latency and multilingual complexity [10].

> "We don't want the model to try to be overly smart. It just needs to fit within the boundaries that we are setting." [10]

That constraint showed up in the prompt work. The team tuned temperature and instructions so Ramble would capture tasks literally, avoid inventing plans or subtasks, but still make tasks actionable by adding verbs when helpful [10]. Date handling was another major source of complexity: they injected the current date, pushed the model to express dates in days rather than months or weeks,

and had it output dates in English for parser compatibility [10].

Quality assurance became part of the product system. Doist built an LLM-judge eval setup, collected recordings in **20+ languages** from **100+ employees** across **35 countries**, and used the system to replay sessions, assess tool-call quality, and catch regressions [10]. Even with that system, quality still varied significantly by language [10]. User feedback inside the experience drove ongoing prompt iteration [10]. The feature moved from prototype to launch, and Doist is now extending task capture to text, files, and images in beta, along with work on assignees, automations, and integrations such as meeting notes [10]. The team also reports that model upgrades improved speed and task understanding over time [10].

Key takeaway: The most convincing AI case studies in product right now are not the most open-ended ones. They are the ones with a clear behavior to support, a narrow action surface, and a fast correction loop.

2) Put customer experience first, then invite hard feedback

One product-development view in this batch argues for betting on the approach that best reduces customer pain, even if it requires a strategic leap. In this case, the bet was on a cloud model because maintaining back versions and multiple ports wasted innovation and led to shelfware and misuse; the belief was that better customer experience should come first and the business model would follow [11].

The same source argues for recruiting the **hardest, smartest customers** as design partners. Tough feedback is treated as a gift, and if early customers will not become references, that itself is a signal to investigate [11].

> "Feedback's a gift. And when you don't get hard feedback is when you actually need to worry." [11]

Key takeaway: Friendly customers are easy to please. Demanding customers are more useful when the goal is to harden a product and earn peer advocacy.

Career Corner

1) If the remit sounds impossible, rewrite the timeline before the company does it for you

One PM candidate who applied for a senior role but was offered a director-level seat was told to, within the first month, improve GTM efficiency, speed delivery with near-immediate MRR impact, and understand platform limits; by month two, extend the roadmap to six months without adding headcount, improve internal and external communication, and draft vision, mission, and strategy [12]. The same candidate also learned that tracking was weak and the feedback system was flawed [12].

The strongest community advice was not to accept those dates literally. Instead: - Build a reasonable timeline around the goals, then discuss it with leadership [13, 14]. - Start with a month of listening and understanding [15]. - Own expectations through a **30-60-90** readout covering what was expected, what you learned, the deltas, and what you are doing next [16]. - Treat sudden role inflation and compressed timelines as a possible org-risk signal, and keep other options open [17, 18].

Why it matters: Scope negotiation is part of product leadership, not a side conversation.

2) PM-owned research looks normal; full UX design still reads as role compression

In a discussion about PMs doing UX work, several practitioners said it is already normal for PMs—especially in startups or FAANG-adjacent environments—to own research and at least some wireframing, and that mid-level PMs should be comfortable designing and running formal or ad hoc studies [19, 20]. The sharper objection was to full UI/UX design responsibility, which one respondent described as a specialized discipline and effectively two employees for one [21]. The question arose in the context of a UI-heavy redesign role asking the PM to do research plus screens and prototypes using a weak design system and AI [22].

Why it matters: AI may blur workflows, but it does not erase the difference between product judgment, research craft, and design craft.

How to apply: Treat PM-owned research as a normal skill expectation, and scrutinize roles that want a PM to absorb deep design execution without design support [20, 21].

3) A long-tenured PM should expect a temporary drop in confidence after switching companies

A PM moving to a new Group PdM role after 10 years in one company asked how to make a strong start [23]. The most grounded advice was to give yourself slack, assume you will need runway before you feel as effective as before, and spend the first week or two learning how the new company's machine really works through informal 1:1s [24].

Why it matters: Domain experience transfers; org-specific pattern recognition does not.

Tools & Resources

1) Claude or Claude Code as a PM learning environment

The most concrete pattern in this batch is not document drafting. It is using Claude as a structured tutor and reviewer: - Teach me anything I don't under-

stand [2] - Show me patterns in how I prompt badly [2] - Teach me a technical concept Socratically, one question at a time [2] - Review my PRD like a skeptical staff engineer [2]

Why it's worth exploring: This turns AI into a compounding skill tool rather than a one-off drafting assistant [2].

2) AI workflows PM peers say are actually moving the needle

Practitioners in technically complex domains described a set of uses beyond PRDs and meeting summaries: - Rapid prototyping and diagramming to validate customer feedback the same day [4] - Strategy debate, market and competitive research, and strategy report generation [5] - Translating technical concepts into plain language for presentations and GTM or enablement [6] - A personal assistant to track, prioritize, and cross-correlate inbound asks across meetings, Slack, and documents [25] - An analyst agent that remembers team-specific data quirks such as time zones, field names, and odd joins [25]

Why it's worth exploring: These are closer to leverage multipliers than generic content-generation tasks.

3) A reusable launch-diagnosis template for PM and PMM teams

If a release misses, use this prompt set as a working template: - Which signals do we have: sales, win/loss, support, customer success, usage, internal feedback? [7] - Who owns connecting them? [7] - Which signals do we trust most? [7] - Where does the truth get lost? [7] - How long until we are confident in the real reason? [7]

Why it's worth exploring: It gives teams a concrete path from underperformance to an evidence-backed diagnosis.

Sources

1. TBM 417: Before You Fire All Your Glue People Because of AI
2. substack
3. X post by @andrewchen
4. r/ProductManagement comment by u/thereasonigotbangs
5. r/ProductManagement comment by u/spoiled___princess
6. r/ProductManagement comment by u/thereasonigotbangs
7. r/ProductMarketing post by u/Ashamed_Listen_1170
8. r/ProductMarketing comment by u/awesomeign
9. r/ProductMarketing comment by u/Slight-Fruit5672
10. Building Todoist Ramble: How Doist Turned Voice Braindumps into Real-Time Task Capture
11. Why your CEO mentality is wrong (with investor Byron Deeter) | Masters of Scale

12. r/ProductManagement post by u/milan92nn
13. r/ProductManagement comment by u/the_new_hunter_s
14. r/ProductManagement comment by u/HurryAdorable1327
15. r/ProductManagement comment by u/Intelligent-Image338
16. r/ProductManagement comment by u/Professional-Run-305
17. r/ProductManagement comment by u/fpssledge
18. r/ProductManagement comment by u/milan92nn
19. r/ProductManagement comment by u/languidlasagna
20. r/ProductManagement comment by u/WholePaycheque
21. r/ProductManagement comment by u/snowytheNPC
22. r/ProductManagement post by u/Appropriate-Dot-6633
23. r/ProductManagement post by u/Blush_and_bashful
24. r/ProductManagement comment by u/NoahtheRed
25. r/ProductManagement comment by u/1_2_red_blue_fish