Debugging team performance, scaling discovery, and building for habits

PM Daily Digest

2026-03-04

Debugging team performance, scaling discovery, and building for habits

By PM Daily Digest • March 4, 2026

This edition highlights two complementary levers for PMs: debugging team performance with the Waterline Model (structure → dynamics → people) and increasing discovery speed as AI accelerates delivery. It also includes a widget-first B2C habit case study, practical tactics for closing the insight and feedback-to-build gaps, and a roundup of PM automation skills worth exploring.

Big Ideas

1) Debug underperformance by checking systems before people (the Waterline Model)

When timelines slip and execution feels messy, it's tempting to jump to people-based explanations—but the Waterline Model is designed to help you diagnose **where** the problem is coming from before deciding what to do [1][2]. The rule of thumb is:

- **"Snorkel before you scuba"**—start with shared systems first (goals, roles, decision-making) before diagnosing personalities [3][4].
- Work through four layers in order: **structure → dynamics → interpersonal → individual** [5][6].

[1] How to debug a team that isn't working: the Waterline Model
[2] How to debug a team that isn't working: the Waterline Model
[3] How to debug a team that isn't working: the Waterline Model
[4] How to debug a team that isn't working: The Waterline Model
[5] How to debug a team that isn't working: the Waterline Model
[6] How to debug a team that isn't working: The Waterline Model

How to debug a team that isn't working: The Waterline Model (3:07)

Why it matters: The model aims to prevent misdiagnosis (e.g., cycling through people unnecessarily) and focus fixes where they have the most leverage [7][8].

How to apply: Start by evaluating **structure** (vision, goals, context, expectations, role clarity, org design), then **dynamics** (decision-making, conflict, information flow), before moving into interpersonal/individual causes [9][10].

---

2) AI is accelerating delivery, but discovery is the bottleneck

Sachin Rekhi argues that AI has handed engineering a "jetpack" (tools like Cursor, Codex CLI, Claude Code), but **discovery** is now the constraint—building faster than you can learn risks "expensive mistakes, sooner" [11]. His framing: the next wave of winning PMs will run **10x the customer learning with the same team** [12].

[7] How to debug a team that isn't working: the Waterline Model
[8] How to debug a team that isn't working: The Waterline Model
[9] How to debug a team that isn't working: the Waterline Model
[10] How to debug a team that isn't working: the Waterline Model
[11] post by @sachinrekhi
[12] post by @sachinrekhi

Why it matters: If shipping gets cheaper/faster, the cost of building the wrong thing becomes the primary failure mode.

How to apply: Pick one discovery workflow where AI can compress cycle time—e.g., - Analyze thousands of NPS verbatims/support tickets/app reviews into structured themes/quotes in an afternoon [13]. - Set up "feedback rivers" that continuously monitor channels and surface actionable signals [14]. - Scale interviews (what used to be 10 interviews becomes 100) via AI-moderated interviews [15]. - Prototype to collect real behavioral data (heatmaps, drop-offs, in-product surveys) before production code [16]. - Ask your database plain-English questions and get charts back—collapsing the hypothesis→answer loop from days to minutes [17].

---

3) "What won't change" is still a product advantage: human behavior

Ryan Hoover highlights a Bezos-style lens: amid accelerating change, it can be more useful to ask what **won't** change—he points to **human nature** and the evergreen value of understanding psychology [18][19][20].

In a related conversation, Nir Eyal distinguishes **persuasive technology** (helping people do what they want to do) from **coercive technology** (getting people to do what they don't want to do, which he calls unethical) [21]. He also emphasizes:

- Reducing friction increases the likelihood a behavior occurs [22].
- Habits are common (he says ~50% of daily actions are habitual), while addiction is a harmful compulsive dependency—and "there's no such thing as a good addiction" [23].

[13] post by @sachinrekhi
[14] post by @sachinrekhi
[15] post by @sachinrekhi
[16] post by @sachinrekhi
[17] post by @sachinrekhi
[18] post by @rrhoover
[19] post by @rrhoover
[20] post by @rrhoover
[21] Why We Don't Do What We Know We Should: Beliefs, Habits, and AI Practice with Nir Eyal 8|4
[22] Why We Don't Do What We Know We Should: Beliefs, Habits, and AI Practice with Nir Eyal 8|4
[23] Why We Don't Do What We Know We Should: Beliefs, Habits, and AI Practice with Nir Eyal 8|4

Why We Don't Do What We Know We Should: Beliefs, Habits, and AI Practice with Nir Eyal 8|4 (34:10)

Why it matters: New tooling changes what's possible, but many product problems still come down to behavior change, ethics, and repeatable usage patterns.

How to apply: When you're designing engagement, explicitly choose (and document) whether you're reducing friction toward behaviors users *want*, and avoid framing "addiction" as a goal [24][25].

Tactical Playbook

1) A practical Waterline diagnostic (structure → dynamics → interpersonal → individual)

Use the model in order, and stop as soon as you find a plausible upstream cause.

1. **Structure check (start here):** Confirm people know what they're doing and how success is defined (vision, goals, context, expectations, role clarity,

[24] Why We Don't Do What We Know We Should: Beliefs, Habits, and AI Practice with Nir Eyal 8|4

[25] Why We Don't Do What We Know We Should: Beliefs, Habits, and AI Practice with Nir Eyal 8|4

org design) [26]. One practical prompt is to ask each person how they describe their role, what goals the team owns, and which numbers they personally own [27].

2. **Dynamics check:** Look at how decisions get made, how conflict shows up and gets resolved, and how information flows (or doesn't) [28][29]. Dynamics are "experienced, not written down," and can bottleneck decisions or create confusion about who decides [30].
3. **Interpersonal check (only after ruling out above):** Interpersonal tension (trust, unresolved conflict, style clashes) can be real, but is often caused/amplified by unclear roles/overlapping ownership/incentives [31].
4. **Individual check (last):** Only after goals/roles/dynamics are sound: evaluate whether the gap is coachable in the time the business can afford; if not, change the role or make a clean exit—avoid "lingering in ambiguity" [32].

---

2) Shrink the "Insight Gap" by naming where the lifecycle breaks

A Reddit thread frames the "Insight Lifecycle" as three common time sinks—even with tools like Amplitude, Stripe, Salesforce, and Gong [33]:

- **Collection ("Fetch")**: chasing data via BI tickets, exports, or locating where a metric is tracked [34].
- **Synthesis ("Tying it together")**: manual CSV cleaning or dashboard-building to correlate data across sources [35].
- **Interpretation ("So what?")**: deciding which signal matters and how it affects prioritization [36].

Step-by-step way to apply this: 1. **Label your current bottleneck** (collection vs. synthesis vs. interpretation) [37][38][39]. 2. If your bottleneck is **interpretation speed**, trial faster loops like natural-language metric analysis (ask in plain English, get a chart back) [40]. 3. If your bottleneck is **input sprawl**, consider automation that continuously monitors feedback channels and surfaces

[26] How to debug a team that isn't working: the Waterline Model
[27] How to debug a team that isn't working: The Waterline Model
[28] How to debug a team that isn't working: the Waterline Model
[29] How to debug a team that isn't working: The Waterline Model
[30] How to debug a team that isn't working: The Waterline Model
[31] How to debug a team that isn't working: The Waterline Model
[32] How to debug a team that isn't working: The Waterline Model
[33] r/prodmgmt post by u/MeeshManne
[34] r/prodmgmt post by u/MeeshManne
[35] r/prodmgmt post by u/MeeshManne
[36] r/prodmgmt post by u/MeeshManne
[37] r/prodmgmt post by u/MeeshManne
[38] r/prodmgmt post by u/MeeshManne
[39] r/prodmgmt post by u/MeeshManne
[40] post by @sachinrekhi

signals (so you're not constantly triaging manually) [41].

---

3) Close the feature-request → implementation gap by staying in the loop

One PM described a flow where users submit requests → PM writes tickets → devs code → a **2–3 week cycle**, by which time momentum is lost; they're exploring whether AI can bridge the gap [42]. A reply argues this resembles a waterfall-style handoff and warns it's a "recipe for disaster" [43].

Step-by-step adjustment to apply (process, not tooling): 1. Treat your job as confirming the solution meets user needs across **discovery → definition → delivery and beyond**, not just ticket-writing [44]. 2. During the 2–3 week cycle, repeatedly check what's being built against what the user asked for (instead of waiting until the end) [45]. 3. Increase customer involvement frequency—framing this as the original goal of Agile: bringing customers into the loop to ensure what's delivered meets their needs [46].

---

4) Use AI prototyping tools *after* research (don't skip the problem space)

Aakash Gupta shares a warning from Nadav Abrahami (Wix co-founder, now building the AI prototyping platform Dazl): he's seen PMs rush straight into building and skip the step that determines whether a feature succeeds [47].

> "You need to understand what problem you're solving, what user story, and the rough shape of the feature… you can't just jump in immediately to the solution space." [48]

Itamar Gilad similarly flags the risk of using prototyping for ideation in a way that moves too fast into the solution space [49].

How to apply (minimum viable sequence): 1. Understand the **problem**. 2. Map the **user stories**. 3. Define the rough **shape** of the feature. 4. Then prototype—"the tool is a hammer; make sure you've found the right nail first" [50].

---

[41] post by @sachinrekhi
[42] r/prodmgmt post by u/Ill_Interaction3910
[43] r/prodmgmt comment by u/holyelvis
[44] r/prodmgmt comment by u/holyelvis
[45] r/prodmgmt comment by u/holyelvis
[46] r/prodmgmt comment by u/holyelvis
[47] [Everyone is vibe coding. Almost no one is doing the work that comes before it.
[48] [Everyone is vibe coding. Almost no one is doing the work that comes before it.
[49] [Everyone is vibe coding. Almost no one is doing the work that comes before it.
[50] [Everyone is vibe coding. Almost no one is doing the work that comes before it.

Case Studies & Lessons

1) Underperformance that was actually structural: roles/goals weren't coherent

In a Waterline Model example, a leader taking over a struggling marketing team avoided diagnosing people first and instead asked each person how they described their role, what goals they owned, and what metrics they were expected to move [51]. The answers were "wildly inconsistent," and leadership's view of responsibility and success measurement differed from how individuals understood their jobs [52].

Lesson: Structural clarity (mandate, goals, role definitions, success criteria) can improve performance quickly, before personnel changes [53].

---

2) A "speed problem" that was actually dynamics: decisions weren't stable

Another example describes a founder frustrated by slow execution—yet when the team tried to move quickly, the founder "swoops in" and unmakes decisions or second-guesses judgment, making it feel unsafe to move quickly [54]. The team adapted rationally by slowing down, adding extra alignment layers, and escalating decisions that didn't need escalation [55].

Lesson: Dynamics can create rational "self-protection" behaviors that look like performance issues from the outside [56].

---

3) B2C habit design via a widget-first product (Bible verse app MVP)

A startup founder (not religious) reports competitor research across top Bible apps and found a repeated review theme: users "keep forgetting to open it," which they interpreted as a **habit problem** rather than a content problem [^11]. Their solution: a home screen widget that shows the daily verse automatically—"the widget is the product; the app is secondary" (journaling, verse history, exploration live in-app) [^11].

Monetization + distribution choices (as shared): - Pricing: **$4.99/month or $39.99/year**; free tier includes unlimited daily verse + 3 extra exploration

[51] How to debug a team that isn't working: the Waterline Model
[52] How to debug a team that isn't working: the Waterline Model
[53] How to debug a team that isn't working: the Waterline Model
[54] How to debug a team that isn't working: The Waterline Model
[55] How to debug a team that isn't working: The Waterline Model
[56] How to debug a team that isn't working: The Waterline Model

verses/day; premium removes limits/ads and adds customization; AdMob on free tier [^11]. - Launch focus: US/UK/Canada/Australia/Philippines day one; "available in 175 countries technically" but focusing on English markets [^11]. - Go-to-market: an "ASO/Reddit/Product Hunt" lane plus "Christian micro-influencers" given premium access + small gift cards to create authentic content [^11].

Lesson (connected to a general behavior principle): Nir Eyal notes that reducing friction increases the likelihood a behavior occurs [57]—and this product's core design puts the habit on the home screen, before other apps [^11].

Career Corner

1) Don't let "people" be your default diagnosis

Lenny Rachitsky amplifies a leadership trap: when a team underperforms, most people's first instinct is to blame the people—and he argues that's "almost always wrong," pointing instead to structural problems below the surface [^12].

How to apply: When you're escalating performance concerns, bring a Waterline-style writeup: what you checked in structure and dynamics before attributing issues to individuals [58][59].

---

2) The differentiator in the AI era: higher learning velocity

Rekhi's claim is directional: the PMs who "win in the next wave" won't be the ones who only learned prompting to build—they'll be the ones who learned to run **10x the customer learning with the same team** [60]. Separately, Jason Long predicts AI will widen the gap: driven, entrepreneurial people with time and resources will accelerate, while others may be pushed further back; he also expects job losses and a "greater consolidation of power and wealth" [^13].

How to apply: Treat discovery leverage as a core skill—e.g., shorten cycles on feedback analysis, interviews, prototype-based learning, or metric analysis (pick one and systematize it) [61][62][63].

---

[57] Why We Don't Do What We Know We Should: Beliefs, Habits, and AI Practice with Nir Eyal 8|4

[58] How to debug a team that isn't working: the Waterline Model

[59] How to debug a team that isn't working: the Waterline Model

[60] post by @sachinrekhi

[61] post by @sachinrekhi

[62] post by @sachinrekhi

[63] post by @sachinrekhi

3) Scope signals: senior PMs being asked to own P&L

One PM with 10 years of experience shared that they were asked to handle **P&L statements for their charter** for the first time and asked if that's now common for senior PMs [^14].

How to apply: Use this as a prompt to clarify expectations in your org: whether "charter ownership" includes financials, and what support (Finance partnership, tooling, cadence) comes with that responsibility [^14].

Tools & Resources

1) OpenClaw: a PM-oriented "skills catalog" (with automation examples)

Jason Long (a former product owner) shares a custom OpenClaw skills catalog for PM workflows, distributed as a zip file he describes as safe [^13][^13]. Example skills he describes include:

- **Second brain**: builds an ontology by reading emails/Google Docs/Asana (read-only); maps projects, team members, strengths/weaknesses, and connections to improve the quality of generated work (e.g., more informed emails) [^13].
- **Competitive monitor**: tracks competitors' websites/releases and posts updates day-by-day; can centralize outputs into a Notion database/wiki [^13].
- **Feedback aggregator**: reads 7 days of Slack/Discord channels, extracts themes, maps to product areas, counts frequency, and generates a structured Monday digest (with charts/graphs) [^13].

OpenClaw Security for Product Managers - AI PM Community Session #86 (37:09)

He also notes a security posture choice: he uses Discord instead of Slack for certain workflows due to sensitive data and not feeling "comfortable enough" securing the system in Slack yet [^13].

---

2) Waterline Model (read + share)

- Lenny's guest post: How to debug a team that isn't working [^12]

---

3) AI discovery workflows (live session)

- Rekhi is sharing his "10 AI discovery workflows" at the Lean Product Meetup in Mountain View on March 5 (registration link provided) [64].

Nadav Abrahami co-founded Wix and scaled it to a $5.5B public company. He's now building Dazl, an AI prototyping platform. He's seen hundreds of PMs rush straight into building - skipping the step that actually determines whether the feature succeeds.

[64] post by @sachinrekhi

Even Itamar Gilad, one of the most respected voices in product management, has flagged this problem: when we jump into prototyping for ideation, we move too fast into the solution space and don't spend enough time in the problem space.

Here's what Nadav said:

"You need to understand what problem you're solving, what user story, and the rough shape of the feature. So these three elements, if you truly want to master AI prototyping, you can't just jump in immediately to the solution space."

More from Nadav: %5Bhttps://www.news.aakashg.com/p/nadav-abrahami-podcast%5D(https://www.news.aakashg.com/p/nadav-abrahami-podcast)

And Itamar: %5Bhttps://www.news.aakashg.com/p/make-more-of-your-product-launches%5D(https://www.news.aakashg.com/p/make-more-of-your-product-launches)

My take: %5Bhttps://www.news.aakashg.com/p/ai-prd%5D(https://www.news.aakashg.com/p/ai-prd)

Lesson: AI prototyping tools are not the starting line. Research is. Understand the problem, map the user stories, define the rough shape - then prototype. The tool is a hammer. Make sure you've found the right nail first.](https://substack.com/@aakashgupta/note/c-222772416) [^11]: r/startups post by u/Ok_Measurement8264 [^12]: post by @lennysan [^13]: OpenClaw Security for Product Managers - AI PM Community Session #86 [^14]: r/ProductManagement post by u/Dizzy_Lifeguard_674

Sources

1. How to debug a team that isn't working: the Waterline Model
2. How to debug a team that isn't working: The Waterline Model
3. post by @sachinrekhi
4. post by @rrhoover
5. post by @rrhoover
6. Why We Don't Do What We Know We Should: Beliefs, Habits, and AI Practice with Nir Eyal 8|4
7. r/prodmgmt post by u/MeeshManne
8. r/prodmgmt post by u/Ill_Interaction3910
9. r/prodmgmt comment by u/holyelvis
10. [Everyone is vibe coding. Almost no one is doing the work that comes before it.

Nadav Abrahami co-founded Wix and scaled it to a $5.5B public company. He's now building Dazl, an AI prototyping platform. He's seen hundreds of PMs rush straight into building - skipping the step that actually determines whether the feature succeeds.

Even Itamar Gilad, one of the most respected voices in product management, has flagged this problem: when we jump into prototyping for ideation, we move too fast into the solution space and don't spend enough time in the problem space.

Here's what Nadav said:

"You need to understand what problem you're solving, what user story, and the rough shape of the feature. So these three elements, if you truly want to master AI prototyping, you can't just jump in immediately to the solution space."

More from Nadav: %5Bhttps://www.news.aakashg.com/p/nadav-abrahami-podcast%5D(https://www.news.aakashg.com/p/nadav-abrahami-podcast)

And Itamar: %5Bhttps://www.news.aakashg.com/p/make-more-of-your-product-launches%5D(https://www.news.aakashg.com/p/make-more-of-your-product-launches)

My take: %5Bhttps://www.news.aakashg.com/p/ai-prd%5D(https://www.news.aakashg.com/p/ai-prd)

Lesson: AI prototyping tools are not the starting line. Research is. Understand the problem, map the user stories, define the rough shape - then prototype. The tool is a hammer. Make sure you've found the right nail first.](https://substack.com/@aakashgupta/note/c-222772416) 11. r/startups post by u/Ok_Measurement8264 12. post by @lennysan 13. OpenClaw Security for Product Managers - AI PM Community Session #86 14. r/ProductManagement post by u/Dizzy_Lifeguard_674