Agent-First Product Strategy, Evals for AI Features, and Better 1:1s

PM Daily Digest

2026-03-22

Agent-First Product Strategy, Evals for AI Features, and Better 1:1s

By PM Daily Digest • March 22, 2026

This issue covers a shift toward agent-first product design, why structured evals are becoming core PM infrastructure for AI features, and a practical 1:1 framework that surfaces blockers and growth conversations. It also includes growth defensibility guidance, eval-driven case studies, and a short list of resources worth reviewing.

Big Ideas

1) Agent-first products may win as callable primitives, not destinations

Andrew Chen's argument is that many current AI chat panels and copilots are a transitional local maximum. The longer-term end state may look more like invisible infrastructure that agents orchestrate, with the human UI acting as a debug layer. In that world, products are better thought of as composable APIs or CLIs that expose a narrow, high-leverage capability agents can repeatedly choose [1].

Why it matters: - **Distribution shifts** from top of funnel to top of call stack; the winner is the default callable primitive in agent-generated plans [1] - **Product surface area may shrink** toward tighter interfaces with opinionated defaults and structured outputs [1] - **Brand becomes partly machine-legible** through reliability, latency, error rates, schema clarity, and integration ease [1] - **Moats may come from integration depth** with agent ecosystems and becoming the sticky default in templates and workflows [1]

How to apply: ask what minimal capability your product can expose that an agent would repeatedly select, then optimize for clean interfaces, structured

outputs, and reliability at scale [1].

2) Paid acquisition can hide weak defensibility

> "Paid acquisition is a tax on your product's defensibility." [2]

Chen's warning, aimed at AI companies experimenting with paid marketing, is that if you cannot keep outspending incumbents and competitors, you are renting growth rather than building it [2].

Why it matters: growth quality determines whether scale improves your position or just increases your spend burden [2].

How to apply: treat paid as a tactic, not proof of product strength, and pressure-test whether your main channels get cheaper as the product grows [2].

3) For AI features, evals are becoming core PM infrastructure

Aakash Gupta's key point is that the farther a team is from the end user, the more it needs structured evals. Teams where engineers are also the users can sometimes rely on intuition; teams farther from the user cannot [3].

> "Evals are the new PRD" [3]

Why it matters: - evals bridge the distance between builder and user [3] - they create a repeatable quality bar instead of one-off demos [3] - preserved failing evals become a durable asset when models change [3]

How to apply: build evals that include failure cases, rerun those failures first when new models arrive, and improve the full loop of dataset, tool access, scoring, and prompting rather than only tuning the prompt [3].

Tactical Playbook

1) A fast eval loop for AI features

1. Start with a concrete task and a simple system prompt [3]
2. Generate a test dataset for that task [3]
3. Connect the model to the real tools it needs, rather than judging it without working access [3]
4. Replace vague grading with a clearer scoring function; Gupta's example used three levels instead of a fuzzy numeric scale [3]
5. Add few-shot examples and rerun [3]
6. Keep the failing evals and use them as your first regression suite when models change [3]

In Gupta's demo, that loop moved performance from **0 to 0.75** in about **20 minutes** [3].

> "If you only have evals that succeed, you don't know what problems there are." [3]

Why this matters: it turns AI product iteration into something PMs can measure, compare, and revisit [3].

2) Rebuild 1:1s so the direct report frames the conversation first

Productify's framework recommends that the direct report's agenda comes first, unless the manager has something genuinely time-critical [4].

A practical flow: 1. Start with progress on key priorities and blockers that need manager help; this is not a status meeting [4] 2. Move into cross-functional relationships and team issues that need coaching [4] 3. Cover goals, aspirations, well-being, and leadership growth [4] 4. Put feedback for the manager on the agenda explicitly, rather than as an afterthought [4] 5. Handle administrative items [4] 6. Then have the manager share updates, context, specific goal follow-ups, and feedback [4]

Two useful operating details: - if both sides arrive with full lists, write them independently and share them at the same time so the agenda comes from the overlap and gaps, not from whoever spoke first [4] - if either side sends context beforehand, the other side should read it; that makes the conversation sharper and shorter [4]

Why this matters: when the first part of the meeting clearly belongs to the direct report, issues surface that might otherwise never come up, and the goal is for the person to leave feeling seen, supported, and taken seriously [4].

Case Studies & Lessons

1) An AI assistant improved when the team fixed the whole eval system, not just the prompt

Gupta describes an assistant answering questions from Linear. The first run failed completely: when asked, "How many tasks are assigned to me?", it responded with a generic offer to help, scoring **0 across the board** [3]. The improvement came from several coordinated changes: connecting Linear's MCP server, giving access to real tools, telling the model to use them, creating a better scoring function, and adding few-shot examples [3]. About **20 minutes** later, the score reached **0.75 across the board** [3].

Lesson: if an AI feature is underperforming, do not assume the prompt is the only problem; the dataset, tool access, task setup, and evaluator may be where the real gains are [3].

2) Closed-loop experimentation produced gains on code humans had already optimized

Karpathy's system runs a tight cycle: read the code and instructions, form a hypothesis, make one change, train, check the score, then either **git commit** or **git reset** based on the result [5]. In Gupta's examples, that loop found **20**

improvements on code Karpathy had already optimized by hand, yielding an **11%** speedup [5]. Tobi Lutke applied the same pattern to Shopify's Liquid templating engine, where **93 automated commits** led to **53% faster rendering** [5].

Lesson: autonomous experimentation is most compelling when the objective is clearly scorable and the system can safely keep or discard each change [5].

Career Corner

1) Evals are becoming a PM skill, not just an ML concern

If your team is not close enough to "vibe check" with the actual end user, designing structured evals becomes part of product judgment [3]. That includes deciding what success looks like, which failure cases to preserve, and what to retest when models change [3].

How to apply: treat your eval suite as a living product artifact alongside the spec, not a one-time launch task [3].

2) PM strategy in AI is shifting from screen design to capability design

As agents start selecting tools based on reliability, latency, error rates, and schema clarity, PMs may need to think less about destination UX alone and more about the minimal callable capability their product exposes [1]. That is a different product design skill: making your product easy for machines to choose and compose [1].

How to apply: when reviewing a roadmap, ask not only "what is the new feature?" but also "what is the reusable capability?" [1].

3) Listening is a leadership skill, and 1:1 structure signals whether you mean it

Productify's 1:1 advice is less about etiquette than about who gets to frame the conversation. Explicitly putting the direct report first, adding feedback for the manager to the agenda, and reading pre-shared context all signal openness rather than control [4].

How to apply: use the agenda structure itself to show that blockers, relationships, and growth matter, not just updates [4].

Tools & Resources

- Aakash Gupta on evals for PMs — useful if you are building AI features and need a practical example of dataset creation, tool connection, scoring design, and failure-case management [3]

- Autoresearch guide for PMs — a follow-on resource if you want to explore scorable experimentation loops more deeply [5]
- Most 1:1s are run the wrong way — a reusable template for direct-report-led 1:1s, including simultaneous agenda setting and pre-read norms [4]
- Andrew Chen on agent-first products — a compact strategy prompt for thinking about callable capabilities, agent distribution, and machine-legible product quality [1]
- Andrew Chen on paid acquisition — a short but useful gut check for growth plans that depend heavily on paid spend [2]

Sources

1. X post by @andrewchen
2. X post by @andrewchen
3. substack
4. Most 1:1s are run the wrong way
5. substack