# Agent distribution, AI-era trio collaboration, and pricing/packaging traps

PM Daily Digest

2026-03-07

## Agent distribution, AI-era trio collaboration, and pricing/packaging traps

*By PM Daily Digest • March 7, 2026*

A field guide to what's changing for PMs as AI reshapes distribution, collaboration, discovery, and monetization: agent-facing product surfaces (docs/CLI/MCP), AI-era PM–UX–Tech workflows, pragmatic AI use for messaging critique, and pricing pitfalls (especially AI credits). Includes concrete checklists, examples, and links to tools and standards.

### Big Ideas

#### 1) A new distribution channel: agents discover products programmatically

Aakash Gupta frames a shift from human-facing discovery (search/app stores/websites) to **agent-facing discovery**, where agents connect, authenticate, execute, and move on—discovering tools through **CLIs, MCP servers, and machine-readable documentation** [1][2].

> "If your product cannot be parsed, authenticated, and executed by an agent, you are invisible in the fastest-growing software channel." [3]

**Why it matters:** This changes what "shipping" means for many B2B/dev tools: not only UI/UX, but also whether an agent can reliably *find and use* your product [4][5].

[1] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[2] [Your product's next million users won't have eyes.
[3] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[4] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[5] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md

**How to apply:** Build an "agent-accessible stack" on top of a solid API (docs → CLI → MCP) [6][7][8][9]. Treat tool naming/selection as product work: the PM's judgment helps decide what features to expose, which to expose first, and how to describe them so agents select correctly [10].

---

### 2) AI didn't make the PM-UX-Tech trio obsolete; it changed *when* collaboration matters

Bandan argues that AI made solo work more viable in small moments—but made **collaboration more important in the moments that matter** [^3]. AI also blurs lanes (PMs can generate wireframes, designers can prototype, engineers can ship UI without design review), creating the temptation that one person can do it all [^3].

The catch: collapsing roles reduces self-challenge—"some friction was load-bearing" for catching bad assumptions before they ship [^3].

**Why it matters:** AI can accelerate execution, but it doesn't automatically create the *perspective diversity* needed when interpretation and tradeoffs drive outcomes (core journeys, architectural choices, decisions that are expensive to undo) [^3].

**How to apply:** Use AI to arrive prepared, then collaborate where interpretation and ownership matter. One suggested AI-era workflow:

```
PM brings prototype  →  Trio reacts together  →  UX generates directions  →  Tech stress-tes
```

[^3]

---

### 3) "AI customer simulation" is the wrong argument; the right one is: what job are you hiring AI for?

Leah Tharin calls the debate a false binary. If you hire AI to **predict what customers will do next**, it will fail; if you hire it to give "fresh pairs of eyes" on a homepage quickly, it can be "shockingly good" [^4].

She distinguishes: - What AI *can't* do: simulate real behavior over time, predict churn, model willingness to pay, understand buying-committee politics, or replace talking to real customers [^4]. - What AI *can* do: heuristic evaluation—spot confusing messaging, contradictions between pages, or mismatched CTAs/forms [^4].

---

[6] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[7] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[8] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[9] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[10] [Your product's next million users won't have eyes.

**Why it matters:** Teams risk over-trusting "plausible personas" that won't surprise you like real interviews—and cannot tell you whether people will buy [^4].

**How to apply:** Use AI as a fast heuristic pass (especially pre-traffic) to catch messaging blind spots and stress-test positioning across segments, then validate with real customer conversations [^4][^4].

---

### 4) Pricing and packaging in the AI era: customers want control and predictability

In an a16z interview, Atlassian's CEO argues usage/outcome-based pricing won't be the majority for all SaaS, partly because customers "hate it" when usage isn't clearly tied to value and isn't in their control [^5]. He highlights how **AI credits/tokens** can feel unpredictable ("casino chips"), and feature additions can unexpectedly increase customers' usage without the customer choosing it [^5].

He also offers two useful frames: - **Input-constrained vs output-constrained work**: some processes have fixed demand (customer service, legal), where AI mainly improves efficiency; others (creative marketing, software development) can scale output as efficiency rises [^5][^5]. - A simplified **SaaS classification**: some seat-based businesses are vulnerable if AI reduces the need for seats tied to doing the work (he uses Zendesk as an example), while others (e.g., Workday as a system of record) may be more resilient [^5][^5].

**Why it matters:** Monetization discussions (credits vs seats vs outcomes) often fail when they ignore what the customer can actually control—and what will feel fair/predictable [^5].

**How to apply:** When proposing AI packaging, pressure-test whether customers can manage cost drivers (and understand them), and whether added AI features change bills in ways customers didn't "choose" [^5].

---

### 5) Procurement can be a moat (not just product)

A post by Josh Kale claims Anthropic introduced a marketplace that lets companies route their *existing Anthropic budget* to third-party tools (e.g., GitLab, Snowflake, Replit) under **one contract**, reducing procurement friction—and potentially creating a moat independent of model quality [^6]. April Underwood called the concept "super smart," noting she wanted to reach something similar with Slack Platform [^7].

**Why it matters:** Distribution and adoption can hinge on non-technical constraints (budgeting/procurement). If true, "contract aggregation" becomes part of the product strategy surface area [^6].

**How to apply:** When evaluating partnerships/marketplaces, model adoption friction explicitly: what can you bundle into existing procurement pathways, and what requires net-new approvals? (Keep this grounded in how your buyers actually buy.) [^6]

## Tactical Playbook

### 1) Build for agents: a practical docs → CLI → MCP sequence

Gupta's recommended build order (on top of a solid API) is: 1) **Documentation** (AGENTS.md + OpenAPI + Agent Skills) 2) **CLI** 3) **MCP server** [11][12][13][14]

**Step-by-step (start this sprint):**

1. **Make your API machine-contractible**
   - If your docs are scattered, agents can't parse them; create a single OpenAPI 3.0 spec as the "machine-readable contract" [15].
2. **Add an agent-facing instruction surface**
   - Draft an **AGENTS.md** describing how agents should work with your codebase/product (executable commands early, boundaries on what agents should never do, exact framework versions) [16].
3. **Wrap for composability with a CLI**
   - Treat the CLI as a structured wrapper around your API that supports Unix-style composability (e.g., JSON output, env-var auth, chaining) [17].
4. **Expose "tools" via an MCP server**
   - Use MCP to expose product capabilities as tools AI clients can discover/call through a standard protocol [18].
5. **Apply MCP quality guardrails (where many teams fail)**
   - **Tool descriptions:** avoid vague descriptions ("manages tasks"). Research cited by Gupta suggests agents start failing at **30+ tools** when descriptions overlap and are "virtually guarantee[d]" wrong at **100+**; reducing Playwright's MCP server from **26 tools to 8** improved accuracy [19].
   - **Auth without a browser:** use OAuth device flow (URL + code) or API keys; don't make browser-dependent auth part of the critical path [20].

[11] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[12] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[13] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[14] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[15] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[16] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[17] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[18] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[19] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[20] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md

- **Structured errors:** make errors actionable (e.g., "API_TOKEN is invalid…") [21].
- **Idempotent endpoints:** agents retry; handle duplicates gracefully [22].
- **Clear rate limits:** return 429 with Retry-After headers [23].

---

**2) Run an AI-era "trio kickoff" that starts *in the middle***

Bandan's suggested shift: instead of sequential handoffs, each function arrives with an AI-accelerated artifact so the conversation begins with shared, concrete inputs [^3][^3].

**Step-by-step meeting recipe:**

1. **PM pre-work:** bring a rough AI-generated prototype so the problem is visible—but stop once it starts answering UX questions and hand off [^3].
2. **UX pre-work:** bring AI-generated user flows/rough concepts/research synthesis and multiple directions to explore [^3].
3. **Engineering pre-work:** bring a quick AI-assisted spike/proof-of-concept that clarifies feasibility, risk, and "hard edges" early [^3].
4. **In-meeting:** react together, surface disagreements faster, kill bad ideas earlier, sharpen good ones sooner [^3].

**Rule of thumb:** AI-enabled solo work is fine for low-stakes/small-scope validation (internal tools, quick experiments, one flow, proof-of-concept) [^3]. Bring the trio in when interpretation and reversibility risk dominate (core journeys, architecture, shared ownership) [^3].

---

**3) Use AI for messaging clarity—then validate with real customers**

Leah Tharin's tool "RoastMyWebsite" simulates five ICP personas visiting a homepage for the first time and outputs a grade, a bounce rate, and specific insights quoting the site's copy [^4][^4].

**Step-by-step (60 minutes):**

1. Paste your homepage URL (it may also scrape pricing if found) [^4].
2. Review the five persona reactions (gut reaction, confusion point, objection, and action like sign-up vs close tab) [^4].
3. Classify feedback into:
   - **Contradictions** (e.g., messaging says "simple" vs pricing complexity) [^4]
   - **CTA friction** (e.g., CTA vs form complexity) [^4]

[21] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[22] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[23] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md

- **Unclear positioning** (what it is / who it's for) [^4]

4. Make the minimal edits that improve clarity.
5. Follow with real customer conversations—because personas are "plausible, not real," and AI can't tell you if people will actually buy [^4].

## Case Studies & Lessons

### 1) Tool naming is product: why Stripe-style descriptions beat vague ones

Gupta's example: "review payments, troubleshoot declines, process refunds" is specific enough that an agent knows what to do; "manages payment operations" is vague and can be skipped [24].

**Takeaway:** "Product judgment" increasingly includes tool taxonomy: the words you choose determine whether an agent can correctly select and execute the right capability [25].

---

### 2) Atlassian's AI in existing workflows: summarize tickets without changing the workflow

In Jira/service workflows, Atlassian describes ticket summarization as a high-leverage insertion point: when a new collaborator joins a ticket with lots of attached files and conversation, summarization can reduce the time to understand context (without changing the underlying workflow) [^5].

**Takeaway:** Look for "brain bootload" moments in workflows: places where context ramps are costly but the workflow itself doesn't need to change to realize value [^5].

---

### 3) "Create with Rovo": a UI paradigm shift is also an adoption challenge

Atlassian describes "Create with Rovo" as a shift from blank-page document creation to starting with a prompt/template, with a document pane and a chat pane for operations across the doc (including broad commands like changing headings) [^5]. They note power users "love it," while many regular business users struggle with the new paradigm at first [^5].

**Takeaway:** AI UX isn't only model capability—it's teaching new mental models. Plan explicitly for onboarding users into the new creation/editing paradigm [^5].

[24] [Your product's next million users won't have eyes.
[25] [Your product's next million users won't have eyes.


*Is This the Biggest Software Shift of the Decade? | a16z Interview - Atlassian CEO (51:00)*

---

**4) Procurement as product surface: Anthropic marketplace (as reported)**

Josh Kale claims Anthropic's marketplace could let companies allocate existing Anthropic budget across third-party tools under one contract, reducing procurement friction and creating a moat beyond model quality [^6]. April Underwood endorsed the approach as "super smart" [^7].

**Takeaway:** If your GTM depends on enterprise budgets, distribution may hinge on contracting mechanics as much as feature differentiation [^6].

## Career Corner

**1) The durable PM value is decisions, not deliverables**

Aakash Gupta argues AI will increasingly automate/accelerate "execution layer" deliverables (PRDs, mocks, roadmaps, pulling data), compressing PM-to-engineer ratios; the PMs who struggle are those whose value was the deliverables, while those who thrive create value through decisions under ambiguity [^8][^8].

**How to apply:** Audit your week: - List deliverables you produce that AI could accelerate. - For each, define the *decision* it supports (what gets built/killed, what tradeoff gets made), and practice making that call explicitly [^8].

### 2) Owning outcomes + shipping as a "super IC" matters more as teams shrink

Shreyas Doshi says owning outcomes and shipping as a super individual contributor has always mattered—and will matter even more as teams get smaller due to AI [^9][^9].

**How to apply:** Pick one outcome you own end-to-end this month, and ship at least one artifact that directly moves it (prototype, workflow change, or an agent-facing surface like docs/tooling), not just coordination.

### 3) Build skill in "restraint" as AI expands your reach

Bandan's warning: AI gives every role "a longer reach," but not a better reason to overstep; each role needs to know when to stop and hand the problem back to the right lane [^3].

**How to apply:** In reviews, add one explicit question: "Where should I stop, and who should take it from here?" [^3].

## Tools & Resources

- **Agent distribution deep dive (Aakash Gupta):** "The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md" https://www.news.aakashg.com/p/master-ai-agent-distribution-channel [26]
- **AGENTS.md standard:** https://agents.md/ [27]
- **MCP tool selection research link (as cited):** https://www.speakeasy.com/mcp [28]
- **Model Context Protocol video (linked in post):** https://www.youtube.com/watch?v=a9wO6GSAo [29]
- **RoastMyWebsite (free):** https://tear-my-site-down.vercel.app/ [^4]
- **a16z interview (Atlassian CEO):** https://www.youtube.com/watch?v=0lzo2tFBFy8 [^5]

Andrej Karpathy said it last week: "It's 2026. Build. For. Agents." He was quote-tweeting a Polymarket CLI that lets autonomous software access prediction markets through the terminal.

Most product teams have zero surface area for this.

[26] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[27] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[28] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
[29] The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md

MCP went from zero to 97 million monthly SDK downloads in its first year. 10,000+ active servers. OpenAI, Google, Microsoft, Cloudflare all adopted it. Anthropic donated it to the Linux Foundation because the standard had already won. Gartner projects 40% of enterprise apps will embed AI agents by end of 2026, up from less than 5% in 2025.

Every generation of software has a dominant distribution channel. Retail (Microsoft won shelf space). Web (Salesforce killed on-premise with a URL). Mobile (Instagram ate companies that tried to port desktop). AI Discovery (if ChatGPT describes your competitor, you lose the click). Now: Agent Distribution. Agents discover through CLIs, MCP servers, and machine-readable docs. They don't onboard. They connect, authenticate, execute, move on.

The companies that build for the new interface first capture outsized share. The ones that retrofit lose ground they never recover.

Three layers matter. Documentation (%5Bhttp://AGENTS.md%5D(http://AGENTS.md), already adopted by 60,000+ projects). CLI (the entire Unix philosophy was accidentally designed for agents decades before they existed). MCP Server (Stripe lets agents review payments and process refunds, Cloudflare exposes 2,500 endpoints through just two tools).

I talked to Todd Olson at Pendo and Brian Helmig at Zapier for the podcast. Zapier exposed 30,000 actions across 8,000 apps as MCP tools. Brian was honest: "It'll work one time and then it'll go off the rails the next time." That's why tool descriptions matter more than having a server at all.

Engineers build the MCP server. But the quality of tool descriptions? Which features to expose first? Starting read-only? That's product judgment.

Stripe wrote "review payments, troubleshoot declines, process refunds." An agent knows exactly what to do. Compare that to "manages payment operations." The first gets picked. The second gets skipped. Research shows agents start failing at 30+ tools when descriptions overlap.

I wrote the most practical guide I could on owning this shift: PM's role, strategy meeting playbook, the first PRD template for agent access, a Claude Code sprint from zero to working MCP server, five production teardowns, and seven mistakes that kill adoption.

The agents are already looking for your product. Whether they find it is up to you.

%5Bhttps://www.news.aakashg.com/p/master-ai-agent-distribution-channel%5D(https://www.news.aakashg.com/p/master-ai-agent-distribution-channel)

https://www.news.aakashg.com/p/master-ai-agent-distribution-channel](https://substack.com/@aakashgupta/note/c-224165610) [^3]: How does AI change PM-UX-Tech workflow? [^4]: I said AI can't simulate your customers. Then I built a tool that does. [^5]: Is This the Biggest Software Shift of the Decade? | a16z Interview - Atlassian CEO [^6]: post by @JoshKale [^7]:

post by @aunder [^8]: "I don't pay PMs to write PRDs. I pay them for their product judgment." [^9]: post by @shreyas

---

**Sources**

1. The PM's Guide to Agent Distribution: MCP Servers, CLIs, and AGENTS.md
2. [Your product's next million users won't have eyes.

Andrej Karpathy said it last week: "It's 2026. Build. For. Agents." He was quote-tweeting a Polymarket CLI that lets autonomous software access prediction markets through the terminal.

Most product teams have zero surface area for this.

MCP went from zero to 97 million monthly SDK downloads in its first year. 10,000+ active servers. OpenAI, Google, Microsoft, Cloudflare all adopted it. Anthropic donated it to the Linux Foundation because the standard had already won. Gartner projects 40% of enterprise apps will embed AI agents by end of 2026, up from less than 5% in 2025.

Every generation of software has a dominant distribution channel. Retail (Microsoft won shelf space). Web (Salesforce killed on-premise with a URL). Mobile (Instagram ate companies that tried to port desktop). AI Discovery (if ChatGPT describes your competitor, you lose the click). Now: Agent Distribution. Agents discover through CLIs, MCP servers, and machine-readable docs. They don't onboard. They connect, authenticate, execute, move on.

The companies that build for the new interface first capture outsized share. The ones that retrofit lose ground they never recover.

Three layers matter. Documentation (%5Bhttp://AGENTS.md%5D(http://AGENTS.md), already adopted by 60,000+ projects). CLI (the entire Unix philosophy was accidentally designed for agents decades before they existed). MCP Server (Stripe lets agents review payments and process refunds, Cloudflare exposes 2,500 endpoints through just two tools).

I talked to Todd Olson at Pendo and Brian Helmig at Zapier for the podcast. Zapier exposed 30,000 actions across 8,000 apps as MCP tools. Brian was honest: "It'll work one time and then it'll go off the rails the next time." That's why tool descriptions matter more than having a server at all.

Engineers build the MCP server. But the quality of tool descriptions? Which features to expose first? Starting read-only? That's product judgment.

Stripe wrote "review payments, troubleshoot declines, process refunds." An agent knows exactly what to do. Compare that to "manages payment operations." The first gets picked. The second gets skipped. Research shows agents start failing at 30+ tools when descriptions overlap.

I wrote the most practical guide I could on owning this shift: PM's role, strategy meeting playbook, the first PRD template for agent access, a Claude Code sprint from zero to working MCP server, five production teardowns, and seven mistakes that kill adoption.

The agents are already looking for your product. Whether they find it is up to you.

%5Bhttps://www.news.aakashg.com/p/master-ai-agent-distribution-channel%5D(https://www.news.aakashg.com/p/master-ai-agent-distribution-channel)

https://www.news.aakashg.com/p/master-ai-agent-distribution-channel](https://substack.com/@aakashgupta/note/c-224165610) 3. How does AI change PM-UX-Tech workflow? 4. I said AI can't simulate your customers. Then I built a tool that does. 5. Is This the Biggest Software Shift of the Decade? | a16z Interview - Atlassian CEO 6. post by @JoshKale 7. post by @aunder 8. "I don't pay PMs to write PRDs. I pay them for their product judgment." 9. post by @shreyas