# Artifact-Led PM Work, Embedded Customer Learning, and a Practical Career Reset

PM Daily Digest

2026-06-01

## Artifact-Led PM Work, Embedded Customer Learning, and a Practical Career Reset

*By PM Daily Digest • June 1, 2026*

This brief covers a sharper view of AI-era PM craft: reviewing artifact layers instead of code, keeping strategy in repo-native context for agents, and embedding customer learning directly into delivery. It also highlights a more hands-on PM skill bar and a grounded lens for mid-career satisfaction.

### 1) Big Ideas

- **PM review is moving up a layer—from code to artifacts.** In one agentic workflow, the PM reviews plans, decisions, strategy docs, and other "load-bearing" artifacts instead of reading code diffs [1]. **Why it matters:** accountability stays with the PM, but the unit of review becomes what changed, why it changed, and whether it should ship [1]. **How to apply:** keep strategy, decisions, constraints, and plan docs in the repo; review those first and only escalate to code when the decision requires it.

  > "You're not approving lines of code. You're approving a description of what changed, why, and whether it should ship." [1]

- **Build before agreement.** The old PRD-first motion made sense when implementation was the expensive step. The source material argues that a first working version is now often cheaper than the meeting about it, so prototypes, screen recordings, failed paths, confusing labels, and observed user behavior produce better alignment than imagined specs [1]. **Why it matters:** teams can collapse ambiguity faster with something inspectable than with a document everyone interprets differently. **How to apply:** use rough prototypes as the first alignment artifact, then discuss evidence from what people actually do.

## 2) Tactical Playbook

1. **Create an agent-ready context layer.**

   - Put target users, trade-offs, non-goals, and failure conditions in repo docs like `strategy.md` or `CLAUDE.md` that agents read repeatedly [1].
   - Keep one source of truth and point every agent to it with a lightweight `AGENTS.md` redirect [1].
   - After each bug, add the smallest test, eval, or policy that would have caught it; one example in the notes is a `fuckups.md` file that stores scar tissue from repeated mistakes [1].

   **Why it matters:** a shared working-context layer keeps multiple agents from developing different realities and lets the system improve from failure instead of repeating it [1].

2. **Separate preferences from non-negotiables.**

   - Put preferences in prompts.
   - Put non-negotiables in hooks, tool permissions, and branch protections [1].
   - Spend manual skepticism on irreversible changes and high-impact claims, then cross-review them with a different model [1].

   **Why it matters:** the source notes argue that soft instructions can be reinterpreted, while mechanical limits hold when the model is confident and wrong [1].

## 3) Case Studies & Lessons

- **Lorikeet keeps discovery inside engineering.** Every engineer is expected to answer a weekly question: "What's one thing you learned from a subscriber?" [2] Releases then move through alpha, beta, and launch with checkpoints to confirm the product actually solves the problem [2]. **Key takeaway:** customer learning is not a separate research ritual; it is part of delivery. **How to apply:** ask every team member to bring one customer learning each week, then use staged checkpoints to test whether that learning changed the release.
- **One PM artifact system grew into 44 markdown files around a single `CLAUDE.md`.** The author says that structure emerged from use and became the layer that held strategy, decisions, constraints, and shipping context together [1]. **Key takeaway:** the durable asset is often not the spec itself; it is the inspectable context that humans and agents can both use [1].

## 4) Career Corner

- **The AI-era PM bar is more hands-on.** April Underwood, former product leader at Slack and Twitter, is seeking a part-time internship because the PM job "has changed a lot" [3]. In the same set of notes, Aakash Gupta says the old PRD → design → engineering handoff model is collapsing, and the new bar includes prototyping in code, writing and running evals, reasoning about model tradeoffs, and shipping with AI agents [3]. **How to apply:** if you need a reset, the practical path in the source material is to build something end to end, learn Claude Code, get AI foundations, build taste at speed, and create a PM GitHub that shows shipped work [3].
- **For mid-career PMs, optimize for fit—not optics.** Shreyas Doshi argues that satisfaction depends less on how your LinkedIn looks and more on resisting career envy [4]. Title, money, and scope may feel good when you accept a role, but they stop delivering happiness once the job starts; competence, flow, culture fit, work-life harmony, and how Sunday evenings feel matter more over time [4]. **How to apply:** evaluate roles against your own criteria, because when it comes to your career, "you are the user," and the true audience is you and those dependent on you [4].

## 5) Tools & Resources

- **Four lightweight repo templates stand out:** `strategy.md` for target users and trade-offs [1], `CLAUDE.md` as a central instruction file agents can read [1], `AGENTS.md` as a pointer so every agent shares the same context [1], and `fuckups.md` for policies learned from repeated failures [1]. These are small, practical ways to make agent-assisted product work easier to inspect and reuse.

---

**Sources**


1. I Don't Review the Code. I Review the Artifacts.
2. X post by @ttorres
3. substack
4. On mid-career satisfaction