How AI can free time for Product Managers to accelerate their discovery, collaboration, and deep thinking

EQUAL EXPERTS

TABLE OF CONTENTS

Abstract

Product Management is an inherently human endeavour; the best discovery, collaboration, and strategic thinking takes time, and this has always been in short supply.

So while the advent of AI-powered software delivery is accelerating coding for engineers, it's also exacerbated a key challenge for Product Managers, which amplifies across teams and releases as the velocity increases.

Based on recent practical experience, we believe AI can be applied to product discovery and specification itself to not only reset this balance between cross-functional delivery teams, but also give Product Managers even more time than they had before.

This document explores a file-based, shared-context approach that enables product and engineering teams to work from a common source of truth.

We outline an iterative workflow spanning problem discovery, solution discovery, and delivery, showing how lightweight orchestration, human-in-the-loop checkpoints, and curated context can make AI a force for clarity and acceleration across the broader software development lifecycle.

AI Accelerates Delivery. It Also Exposes a New Business Constraint.

The adoption of AI in software delivery is becoming more and more common; it's seen as a panacea to reduce costs, increase throughput, and shorten time-to-market. In practice, many are discovering that while engineering capacity has expanded dramatically, overall business outcomes have not improved at the same rate.

What seemed like a productivity gain actually creates a new organizational bottleneck: the ability to decide what should be built, and why, does not scale as quickly as the ability to build it.

> **"The focus on only accelerating coding for engineers has exacerbated an existing issue for product managers: insufficient time."**

Product people share a frustration that their time to focus on more strategic 'human' activities is already pinched: discovery, experimentation, deep thinking, and deep collaboration with engineering teams—all essential to optimizing value delivery—are put aside as a result. Too often they're reduced to mere ticket writers.

AI-accelerated engineering teams are now demanding ever higher rates of PM inputs simply to stay busy.

Andrew Ng recently illustrated this shift when he observed early-stage teams proposing a move from a 1:4 product-manager-to-engineer ratio to 1:0.5.[1] While extreme, it signals a broader trend that will increasingly affect large enterprises: Uneven adoption of AI introduces new bottlenecks and pressure to generate output regardless of its alignment to strategic value.

> **"When delivery is slow, poor alignment is painful but contained. When delivery is fast, small misunderstandings and imperfections are amplified across teams and releases."**

This shifts the risk profile for the business. Speed without alignment increases the likelihood of:

- Building the wrong capabilities quickly and at scale
- Eroding trust between business, product, and engineering
- Optimizing for throughput rather than value creation

The challenge, therefore, is not merely to "speed up product" to match engineering, but to facilitate better decision-making despite this acceleration by ensuring humans have enough time to think deeply. AI can create that time.

Shared Context and Key Principles

The growing velocity gap between product and engineering highlights a second, equally persistent challenge: **the communication gap between these disciplines**. If product intent, technical constraints, and evolving context are not shared effectively, from the outset, this gap widens rapidly with the addition of AI.

As we've shown in building proven patterns within AI-enabled engineering teams, a **shared context** across discovery and specification is the mechanism through which product and engineering teams remain aligned as speed increases.

Beyond this foundation, we have devised an additional set of guiding principles to shape the solution:

- **Portable design** – Works across LLM ecosystems
- **Transparent artifacts** – Everything is human-readable and editable
- **Scalable structure** – Multiple teams can operate from the same foundation
- **Iterative process** – Supports incremental discovery and development
- **Human-in-the-loop checkpoints** – Critical milestones require explicit review

"These principles point toward a concrete question: how should product discovery artifacts be structured so they can be shared, evolved, and reliably used by both humans and AI?"

Putting our principles into practice

Foundation

To explore this approach in practice, the team used Claude Code[2], drawing inspiration from Teresa Torres' recent work[3] with the tool. Claude's strength in both writing and coding makes it well suited to product teams operating across discovery and delivery.

A file-based structure was implemented to support transparency and portability between LLM ecosystems. Rather than introducing a separate product tooling layer, discovery artifacts were stored directly alongside engineering assets in the existing code repository in GitHub. This ensured that product and engineering operated from the same environment and reduced handoff friction.

Artifacts were primarily markdown files, organized into three categories:

- **Context** – Curated inputs for the LLM, including product vision, personas, technical architecture, backlog, interview notes, and stories
- **Instructions** – Prompts, templates, and rules guiding LLM behavior
- **Orchestration** – Mechanisms controlling sequencing and coordination, including slash commands and files such as `CLAUDE.md` and `README.md`

With minor adjustments, this structure can be adapted for other AI tools such as OpenAI Codex[4]. The key architectural decision is not the tool itself, but the **shared source of truth:** product and engineering work from the same repository, source files, and context.

Because LLM performance still degrades as tokens are added to the context window of each conversation, the architecture deliberately keeps inputs lean and targeted, providing an index for the model to find what it needs rather than loading everything at once.

With shared context established at the file and repository level, the next step was to define how discovery and delivery activities would evolve throughout the delivery lifecycle.

The Loop of Loops Product Workflow

Building on this foundation, the team defined a three-phase iterative workflow: **Problem Discovery, Solution Discovery**, and **Delivery**. Each phase operates as its own refinement loop, with explicit checkpoints for human review; overall these are contained within the larger interactive lifecycle.

This 'loop of loops' reflects established agile practice and other frameworks such as the Double Diamond Design Process.[5]

AI fundamentally changes the economics of each loop. Activities that were previously time-intensive become fast enough to repeat multiple times without too much effort. Teams can explore more options and make larger bets within the same time, without increasing risk.

The workflow begins with **problem discovery**, where AI is used to:

- Transcribe and summarize interviews
- Synthesize interview outputs alongside curated business and technical inputs

Once a clear understanding of problems and needs emerges, the process moves into **solution discovery**, where teams:

- Translate synthesis into product stories with acceptance criteria
- Define user journeys and refine stories accordingly
- Prioritize stories and load them into a task tracking system

To support multiple initiatives running concurrently, iteration-specific directories are introduced to compartmentalize discovery work.

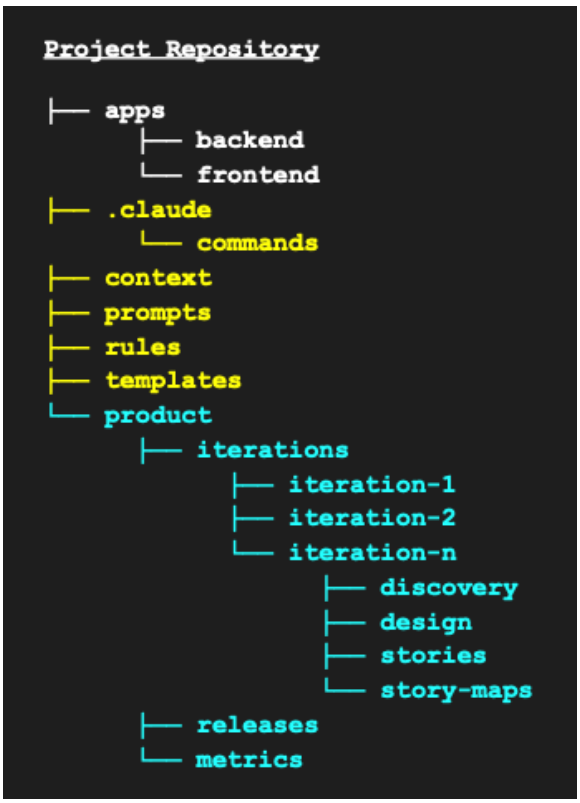


Figure 2: An example repository directory structure, with shared context in yellow, and product-specific content in blue

The final phase is the **delivery loop**, where teams:

- Confirm what was actually built
- Update backlog and story status
- Produce an as-built product specification

"When AI is generating artifacts, unchecked drift compounds quickly. Explicit human review at each checkpoint prevents this, creates space for judgment and correction, and ensures that accountability for outcomes remains with the product manager."

Defining the workflow established the "what" of each phase; the next challenge was ensuring "how" each step could be executed consistently and repeatedly.

Orchestration

Each activity in the workflow requires explicit guidance on what context the LLM should read, what work it should perform, and what outputs it should produce. This guidance was encoded using Claude Code's slash commands.

For example:

- `/iter` creates a new iteration and captures its intent
- `/synth` produces a structured synthesis from discovery artifacts

Each command applies consistent context and instructions, enabling repeatable execution (with an audit trail).

Command	Phase	Description
`/iter`	Setup	Start a new iteration, interview the PM to create a README
`/synth`	Problem Discovery	Synthesize discovery materials into themes and opportunities
`/req`	Solution Discovery	Extract stories from synthesis
`/map`	Solution Discovery	Generate a Miro story map
`/demap`	Solution Discovery	Sync prioritization changes from Miro back to story files
`/jira`	Delivery	Load stories to the issue tracker
`/rel`	Delivery	Create a release and update the product spec and the backlog

Table 1: Workflow Command Reference Across Discovery Phases

As the workflow matured, the underlying orchestration mechanisms evolved as well.

Skills and Future Orchestration

Since this work began, Anthropic introduced **skills** within Claude Code—a more structured evolution of slash commands that bundle templates, rules, and scripts into reusable units. Unlike explicit commands, skills can be inferred and applied automatically based on user intent, making them well suited to iterative and agentic workflows where tasks do not follow a fixed sequence.

For clarity, this article references slash commands, but in practice future iterations of the approach will increasingly rely on skills as the primary orchestration mechanism, especially as work becomes automated with **agents**.

With orchestration in place, the workflow can be examined phase by phase, starting with problem discovery.

The Loop of Loops: #01 Problem Discovery

Problem discovery is hands-on and social, encompassing interviews, observation, and research into user and stakeholder needs.

Primary artifacts include interview notes, journey maps, and business observations. These are synthesized into actionable insights that guide decision-making and are typically expressed as an initial set of user stories.

Within the file-based framework, discovery artifacts live inside iteration directories created via `/iter.` This structure provides initial context for the LLM while remaining navigable and reviewable by humans.

The quality of problem discovery depends heavily on how insights are captured, synthesized, and shared.

Accelerating Insights from Discovery Interviews

Discovery interviews provide some of the most valuable context available to product teams. They surface business drivers, success metrics, and how users experience their work and tools.

Modern AI transcription tools for interviews and meetings have matured significantly. In this workflow, interviews are captured using Granola[6], chosen for its ability to:

- Combine transcripts with private annotations
- Record across platforms and devices
- Generate structured summaries aligned to discovery goals

These summaries provide concise, consistent inputs to Claude Code while preserving nuance and reducing processing time.

Synthesis Through Instructions and Orchestration

Once interviews are complete, Claude Code ingests structured summaries and produces a synthesis in minutes. A predefined template specifies the output structure, while the `/synth` command applies the appropriate context and instructions to generate the file.

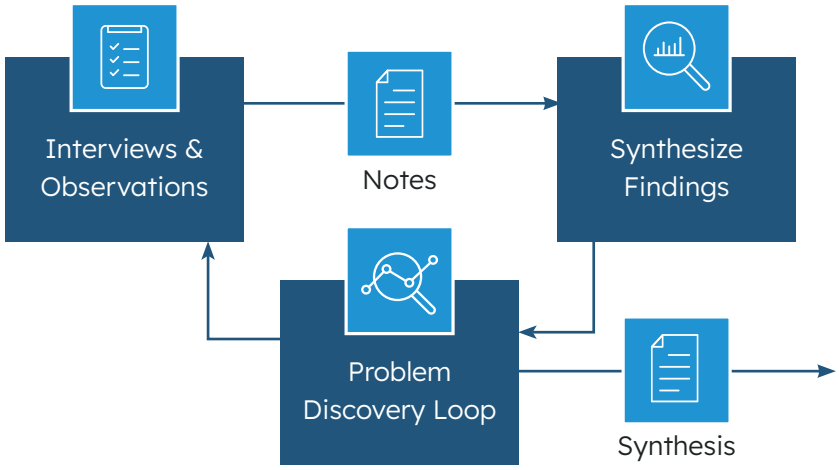


Figure 3: The Problem Discovery Loop

Each synthesis is timestamped and records its source materials, enabling iterative refinement and historical review. The LLM can also propose follow-up questions to address gaps in understanding. All artifacts remain human-editable markdown files, preserving transparency and accountability.

This synthesis marks the transition from understanding problems to deciding what to build.

The Loop of Loops: #02 Solution Discovery

To make more progress towards actually delivering software, we need to start breaking our insights down into the actual work that should be done, and what the user experience should be.

Claude has plenty of context to help us make those decisions, from architectural documents provided by the engineering team, to our discovery synthesis.

The process of deciding what to build is historically a hard one for product teams. A tight iteration loop involving human collaboration and review ensures we get to the right package of work to provide to the engineering team. Requirements captured during problem discovery are best documented as stories in separate markdown files within our file structure. If preferred, product requirements documents (PRDs) can also be generated for larger feature narratives using similar techniques.

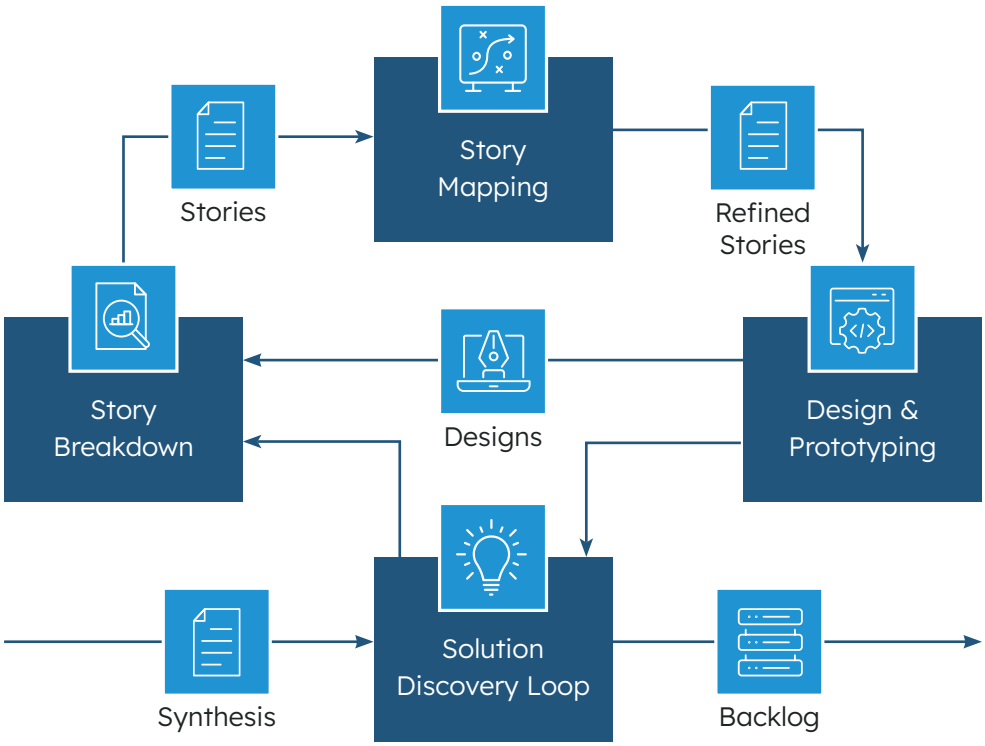


Figure 4: The Solution Discovery Loop

Story Format Depends on Engineering Methodology

Initial story templates reflected traditional product management best practices and dictated small stories with high levels of detail. In an AI-accelerated delivery environment, this proved counterproductive. Rich templates consume unnecessary context tokens when architectural rules and delivery conventions already exist elsewhere.

Agentic coding also changes assumptions about granularity. Larger-grained stories—three to five times the size of traditional stories—can be delivered quickly and safely when supported by shared context.

"Incremental development remains essential, but its value lies in risk reduction, not scope reduction. If larger stories can be delivered within the same elapsed time and assumptions are validated, the value-to-risk ratio increases."

Multiple story templates support different delivery styles. The `/req` command prompts the product manager for preferences, surfaces open questions, and generates stories aligned with the chosen approach.

Even with well-structured stories, refinement remains a collaborative and iterative activity.

Accelerated Refinement

Product managers, engineers, stakeholders and users must share an understanding of why features should be built and which assumptions require validation. Story mapping helps identify gaps and prioritize delivery for larger features. AI can assist by generating visual maps to facilitate human collaboration. It can then incorporate changes back into the written stories.

Generative AI prototyping tools further accelerate validation by producing clickable demos. They reduce the cost of exploration and speed up feedback. Screenshots or raw HTML captured from a selected prototype can be stored alongside stories and indexed for LLM access, further informing its context.

While this work only included a cursory examination of AI techniques for story mapping and prototyping, this is an area rich for deeper future exploration.

Building a Roadmap

As discovery spans multiple iterations, the LLM must understand what has already been captured. A story backlog file outside iteration directories provides this orientation.

The backlog indexes unbuilt stories with IDs, priorities, and source iterations. Stories are added automatically via `/req`, and the LLM consults the backlog to avoid duplication and surface dependencies.

This shared awareness improves accuracy and efficiency as discovery scales across teams and iterations, and transitions into delivery.

The Loop of Loops: #03 Delivery Loop

The delivery loop connects product intent to what engineering actually builds. Stories are synchronized to an issue tracker via `/jira`, preserving traceability between markdown artifacts and tickets. Story IDs are applied as labels, acceptance criteria populate descriptions, and dependencies are represented as linked issues.

While implemented with Jira in this instance, the approach is tool-agnostic. Story files remain the source of truth; trackers provide alternative views for teams with different preferences, leadership visibility, or automation needs.

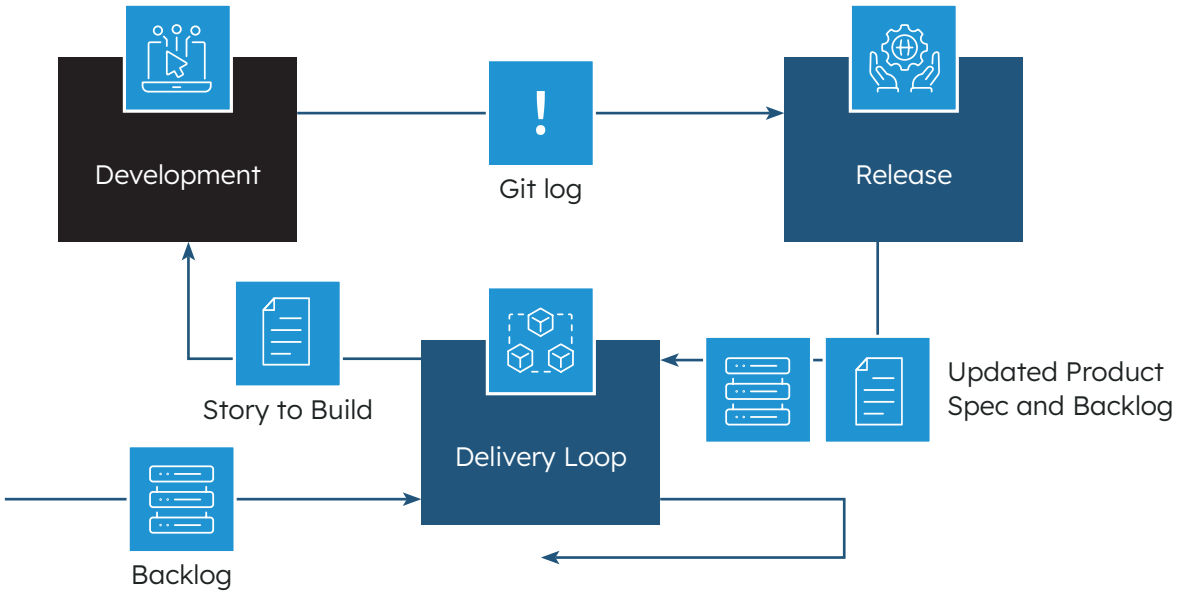


Figure 5: The Delivery Loop

This article is intentionally not covering the engineering parts of the delivery lifecycle to focus on product management activities, so it's represented here by a literal black box.

Recording Releases

After deployment, the `/rel` command analyzes git history to identify delivered work. Commits are matched to stories where possible, and unmatched changes e.g. bug fixes, are flagged for review.

Once confirmed, the command updates story status, removes completed stories from the backlog, and generates release documentation. It updates the product specification, keeping shared context current, ready for subsequent work.

The As-Built Product Spec

The product specification records what is actually running in production: delivered capabilities, architectural characteristics, and known limitations.

When the LLM references this specification, it avoids re-specifying existing functionality and can reason accurately about integration and extension. Where appropriate, screenshots of live systems can be generated and linked.

This artifact supports LLM-assisted discovery and delivery, living alongside code in a concise, structured format.

> **"By maintaining this shared, up-to-date context, teams ensure that each subsequent cycle of discovery and delivery is informed by an accurate picture of what already exists. Together, these practices reinforce a tight feedback loop between discovery, delivery, and learning."**

Historically, maintaining documentation reflecting the current state of software is a difficult task. AI can not only take away this burden, but leverage the resulting artifacts in a feedback loop to accelerate and improve the quality of downstream enhancements.

Key Takeaways: making AI a force for clarity rather than fragmentation

"Organizations that treat AI purely as a delivery accelerator jeopardize the very outcomes they seek. Those that redesign product discovery alongside AI-enabled engineering can turn speed into a durable competitive advantage without increasing risk."

AI must free product leaders from low-leverage work—manual ticket writing, repetitive clarification, and administrative coordination—and create space for higher-value activities: discovery, experimentation, synthesis, and judgment.

At the same time, organizations must put deliberate mechanisms in place to preserve shared context and intent as delivery accelerates.

In this new operating model, alignment becomes a first-order concern.

EE's approach to AI-powered discovery and specification is grounded in a small number of core ideas:

- Shared context is foundational
- File-based artifacts enable transparency and portability
- Loop-based workflows preserve agile discipline
- Commands and skills encode repeatable processes
- Human judgment remains central

What's Next?

"Our approach continues to evolve. Skills and agentic workflows are enabling more autonomous orchestration, while highlighting the limitations of text-only collaboration.

Future work will explore how LLMs can better support visual collaboration alongside textual artifacts, and incorporate proactive solicitation of humans during agentic workflows to ensure appropriate oversight.

The discovery and specification approach described here provides the stability needed to adopt more autonomous capabilities without relinquishing control over outcomes that matter."

References

[1] Ng, A. (2025). Building Faster with AI
[2] Anthropic. (2024). Claude Code.
[3] Torres, T. (n.d.). Product Talk.
[4] OpenAI. (2024). Codex.
[5] Design Council. (2019). The Double Diamond.
[6] Granola. (2024). Granola AI Meeting Assistant.

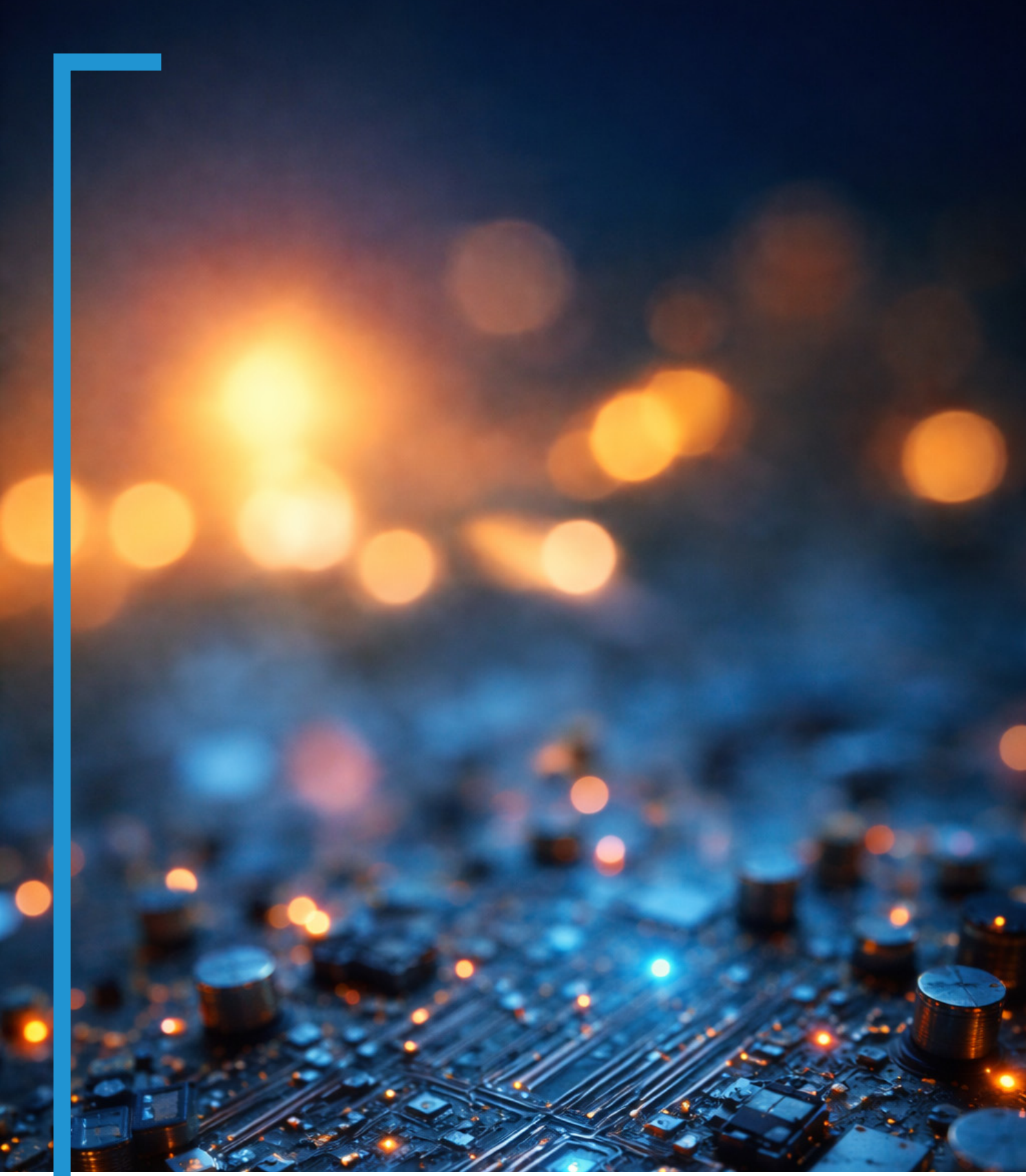
Author

Mike Mitchell
Product & Strategy Principal, Equal Experts
mike.mitchell@equalexperts.com

Mike Mitchell is a Product & Strategy Principal at Equal Experts, helping organizations adopt modern product practices and integrate AI into their product lifecycle in practical, sustainable ways.

His current focus is on helping product teams do more meaningful strategic work with AI, both by accelerating the routine work that crowds it out, and by making AI a more proactive, multi-modal collaborator.

Mike brings 35 years of experience in enterprise software, with a foundation of engineering followed by two decades of product leadership. He is a founder of multiple startups, has advised dozens more, and has led multiple enterprise transformations. Just prior to EE, Mike served as a fractional Chief Product Officer, guiding early- and growth-stage firms to deliver value faster through stronger product strategy and operating models. He holds a degree in Computer Science & Engineering from MIT.



www.equalexperts.com

EQUAL EXPERTS