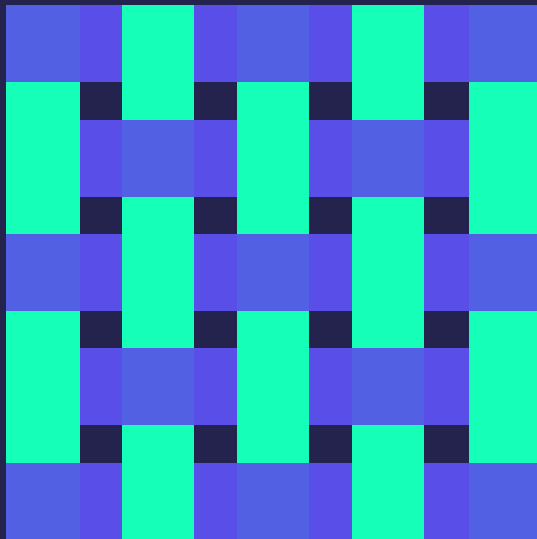


Loomery



THE **PRODUCT ENGINEERING** GUIDE BOOK

A practical guide to Product Engineering, helping you navigate the AI-driven transformation of industry and build high-impact products with smaller, autonomous teams.

TABLE OF CONTENTS

01

What Product Engineering is and why it matters

02

02

The new tooling stack that helps engineers move at the speed of thought

09

03

The impact on teams and the roles within them

14

04

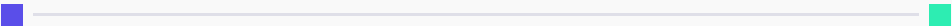
The Product Engineering skillset

20

05

Getting started and the barriers you might face

26



Chapter 1: What Product Engineering is and why it matters

BY TIM CHECKLEY, CO-FOUNDER AT LOOMERY

For the majority of my career, the software industry has operated on a hand-off model. Product managers like me defined the *What*, designers defined the *How*, and software engineers were the specialised mechanics who translated those requirements into code. Their value was measured by their ability to navigate complex syntax and manage infrastructure.

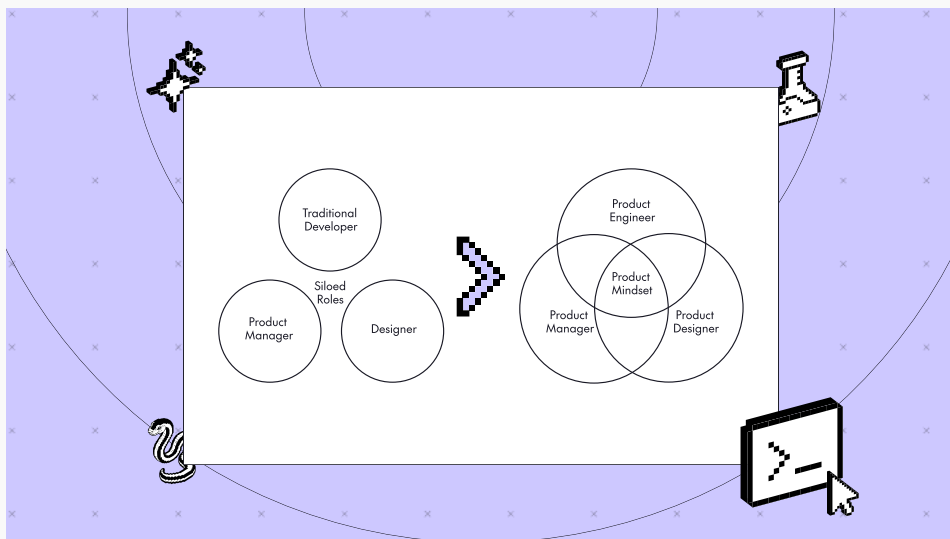
Those traditional roles, and the walls between 'thinking' and 'doing', are collapsing. Coding assistants such as Claude, GitHub Copilot and Cursor are writing a large proportion of the code for developers. One of our clients, BeZero, recently commented that they've never seen their engineers ship so much (at such high quality) and that engineers are racing ahead of product managers (PMs) and designers. We're even seeing companies like Spotify where engineers no longer write code at all.¹ With the arrival of LLMs, 'doing' (the boilerplate, the refactoring) is becoming a commodity. If anyone can generate code, then the act of writing code is no longer the primary value driver of an engineer.

Welcome to the era of the Product Engineer.

At Loomery, we're seeing the nature of engineering change rapidly thanks to AI - and that's what this Product Engineering Guide Book is about.

CHAPTER 1

Here in **Chapter 1**, I'll explain what a Product Engineer is and how to develop a Product Engineering mindset. In **Chapter 2**, Will, Loomery's Head of Engineering, delves into the new tooling stack that helps engineers move at the speed of thought. In **Chapter 3**, I explore the impact on teams and other roles. **Chapter 4** will cover the technical and soft skills that matter. And to close, **Chapter 5** visits the cultural and institutional barriers to overcome in order for Product Engineering to truly succeed.



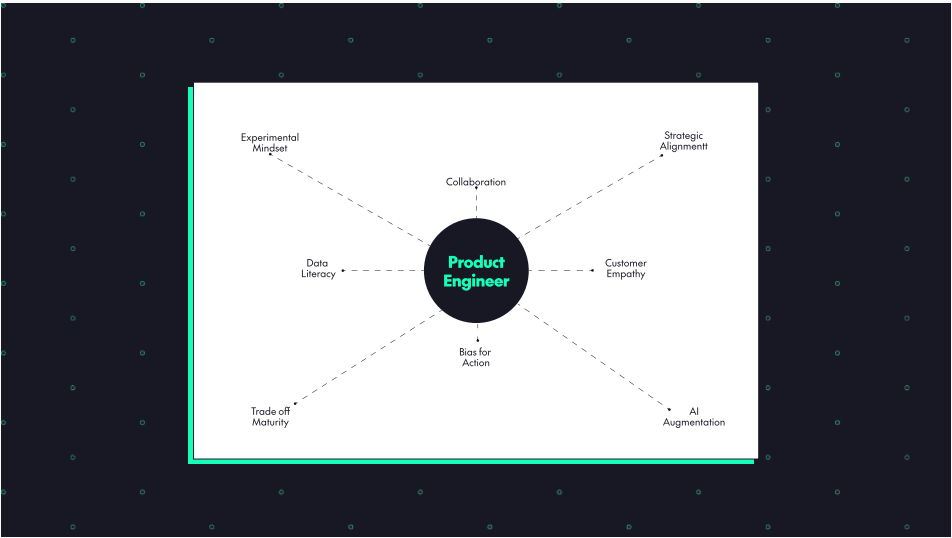
The broadening of the engineering role

What's the impact of this shifting shape for developers? Naturally, it leads to a broadening of the role. It's no longer enough for engineers to write React or Python. Increasingly, engineers only succeed if they possess a 'product mindset': meaning they're focused on outcomes, thinking through the lens of customers, and considering a product's continuous improvement. It's worth saying, this

mindset isn't brand new - and great engineers have always been 'T shaped' in their skillsets. However, we're seeing AI-enabled engineers breaking down the traditional silos at a pace never seen before. A team is no longer about **x developers** and **y designers** working at arms' length from research or sales roles; roles overlap and are more closely-knit and collaborative.

Shopify and Atlassian have been hiring and developing Product Engineers for the last couple of years. We've been experimenting with the role at Loomery, and the impact for clients has been huge. Today, a Product Engineer at Loomery can use AI to stand up a functional MVP, integrate complex APIs, and iterate based on real feedback twice as fast as we're used to. That's not hyperbole; we've measured it. This truly is the 'Age of Software Abundance'.

The anatomy of a Product Engineer



CHAPTER 1

Let's be a bit more specific about the traits and behaviours Product Engineers model:

Customer empathy:

They don't just build to spec; they 'feel' the user's pain points, ensuring every line of code solves a genuine human problem rather than just satisfying a technical requirement.

Close to strategy:

They understand the 'North Star' metrics of the business, allowing them to align their technical design with long-term commercial goals and market positioning.

AI-augmented execution:

By offloading everyday code generation, syntax debugging, and unit testing to AI assistants, they reclaim the mental bandwidth needed for high-level system design and creative problem-solving.

Data-informed decision making:

They look beyond the IDE, using product analytics, user research, and competitive teardowns to validate their technical direction with objective evidence.

Always prototyping and experimenting:

They treat the codebase as a laboratory, frequently deploying 'lo-fi' versions of features to gather real-world feedback before committing to a full-scale build.

Bias for action and quick wins:

They prioritise momentum over perfection, identifying 'low-hanging fruit' that can deliver immediate value to the user while the larger architecture evolves.

Balancing trade-offs:

They possess the maturity to know when 'good enough' is better than 'perfect,' expertly navigating the tension between technical debt, speed to market, and feature richness.

Strong communicator and collaborator:

They act as the 'connective tissue' of the team, translating complex technical constraints into clear business trade-offs for stakeholders and designers alike.

Product Engineering at Loomery

At Loomery we're investing in broadening the role and upskilling our engineers to succeed as Product Engineers. Whilst it does ask a lot of the engineering role, our engineers are excited by the challenge. Ultimately, the rise of Product Engineering, the broadening of roles, and the breaking down of silos mean we're building higher quality products for our clients, faster. This is exciting to us at Loomery as moving and shipping fast has been our promise from Day 1, now it's being realised day in, day out.

For broadband operator and Loomery client Vorboss, our Product Engineers have been executing multiple engineering initiatives at a rapid pace.

We're big advocates for this new shape of engineering role - and we're helping clients adopt a Product Engineering mindset too.

"The result was a level of ROI that would be difficult to match with in-house resource or other agencies."

Aaron Rice, Chief Information Officer at Vorboss

How to develop a Product Engineering mindset

Transitioning from a traditional 'software engineer' to a Product Engineer is less about learning a new language and more about shifting cultural attitudes and role boundaries.

It requires moving technical roles from thinking '*How do I build this?*' to '*Should we build this, and what happens once we do?*'

Here are a few pointers on how to bridge that gap for your team:

1. **Expose engineering teams to customer pain:** Stop filtering feedback through PMs and multiple layers of management. Have engineers watch a recording of a user failing to find the 'Sign Up' button. Seeing someone struggle with their work creates a drive to fix it that no Jira ticket can replicate.
2. **Ditch the 'dev only' meetings:** Invite your engineers to strategy and co-design sessions. If they understand the business goals early, they can suggest technical shortcuts that save weeks of work while still hitting the objective.
3. **Become truly 'AI-first':** Encourage the team to use LLMs for the repetitive parts of coding - like setting up data structures or writing basic tests. This isn't laziness; it's efficiency. It clears their schedules so they can spend more time thinking about user flow and system logic.
4. **Measure outcomes, not velocity:** Move away from counting story points. Instead, ask the team: 'Did that feature actually reduce the churn rate as we hoped?' When the team shares the business goals, they stop acting like a 'feature factory.'

Tool: Product Engineering scorecard

You can use this tool to evaluate your current team or your own Product Engineering mindset. Rate each category from 1 (Traditional Dev) to 5 (Product Engineer).

Characteristic	1: Traditional Developer	5: Product Engineer	Score
Customer Empathy	Focusses Strictly on technical specs; rarely thinks about end-user	Regularly talks to users; identifies friction points before they are reported	/5
Strategic Alignment	"I just build what I'm told."	Understands the business model and North Star metrics.	/5
AI Augmentation	Writes all code manually; views AI with skepticism or as a "cheat"	Uses AI to handle 50%+ of boilerplate; focusses on systems design and logic	/5
Data Literacy	Relies on PMs to interpret data; rarely looks at analytics	Queries the data themselves; uses evidence to suggest feature pivots	/5
Experimental Mindset	Relies on PMs to interpret data; rarely looks at analytics	Queries the data themselves; uses evidence to suggest feature pivots	/5
Bias for Action	Gets stuck in "analysis paralysis" or over-engineering	Identifies the "Minimal Viable Move" to unblock the team or add value	/5
Trade-off Maturity	Always chooses the most "correct" technical path, regardless of cost	Weights technical debt against market windows and business needs	/5
Collaboration	Prefers to be left alone to code; avoids "business" meetings	Leads cross-functional workshops; bridges the gab between design and tech.	/5

Wherever your team lands on that scale, the direction of travel is the same. The rest of this guide book is about how to move along it, starting with the tooling stack that makes it possible.

Chapter 2: The new tooling stack that helps engineers move at the speed of thought

BY WILL ROCHA-THOMAS, HEAD OF ENGINEERING AT LOOMERY

The engineer's role is broadening, and writing code is becoming a commodity. As we've argued in our piece on the Age of Software Abundance, the value is shifting to deciding what to build and why. This chapter is about the stack that makes that shift possible: the tooling that lets a Product Engineer move at the speed of thought.

In a traditional hand-off model, the 'tooling stack' was designed to isolate the engineer from the product strategy. The Product Engineering stack does the opposite: it removes the friction of implementation so the engineer can spend their time in the problem space.

Coding assistants are the foundation

LLMs have transformed engineering. Claude, Codex, Cursor, Github Copilot and Gemini, the coding assistants most teams reach for, have moved well past autocomplete: they write code at scale, refactor across files, and pair-program through ambiguous problems. They behave less like clever editors and more like junior engineers we can delegate to and then review.

At Loomery, we've moved to Claude, mainly because of the strength of the integration ecosystem. It works for Product Engineers and, just as importantly, for the non-engineers across the business who increasingly rely on agents in their own day-to-day.

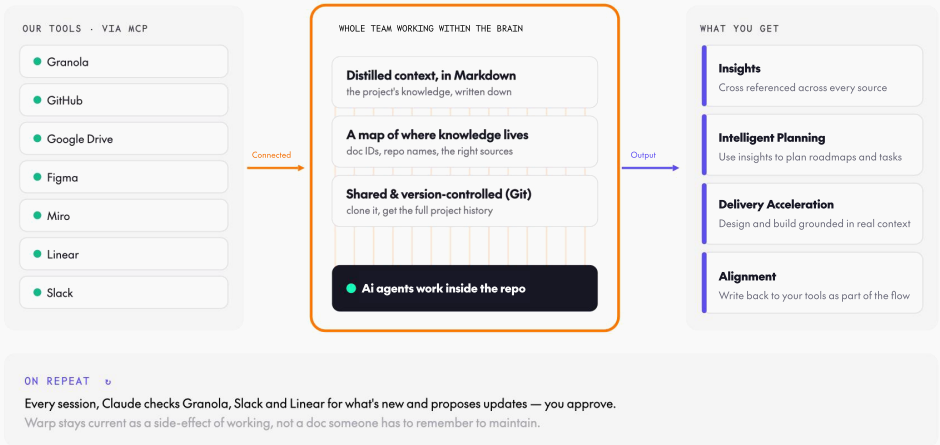
Tooling should act as a decision-support system

Whilst the speed of coding has undoubtedly increased, we think for Product Engineers that's the wrong metric to fixate on. Shipping fast matters, but shipping the right thing matters just as much, and you can only do that if you have the context to make good decisions. If your tooling doesn't make it easy to interrogate product metrics, strategy or user feedback, it's hard to be an effective Product Engineer. You just fall back into the silo and code blindly.

AI can collapse the information gap and enable autonomy

Context is key. At Loomery, we've been integrating tools like Granola, Linear, Figma and Notion with AI to build a 'Product Brain'. Using a well-crafted `agents.md`, hooks and skills that pull from all those places, we encode product and organisational context into repositories directly or tools a Product Engineer can reach via MCP. The engineer can ask questions about a user feedback session, product analytics, a recent release or the design system without switching tabs. That's what lets them make real-time trade-off decisions, like simplifying a feature to launch sooner.

Naturally, we've branded up our Product Brain, calling it the Warp, which is the term used for the set of threads that provide the structure in a loom, acting as the backbone of the fabric you're weaving.



Here's a real example. One of our engineers recently asked Warp to explain the context behind a past incident where a single user flooded the system with thousands of files at once and overwhelmed it, then to assess how recent code changes would handle that scenario differently. That's not a question you answer just by reading code. It needs the incident history, the original decisions and the current implementation, all in one place. Warp pulled that all together.

AI shouldn't just mean we write more code faster; it should result in more time to think.

Offloading implementation (boilerplate, technical migrations, simple refactoring) allows a Product Engineer to focus more on the 'what.' They can reclaim the mental bandwidth required for high-level system design and analysis.

The feedback loop is the 'product' part

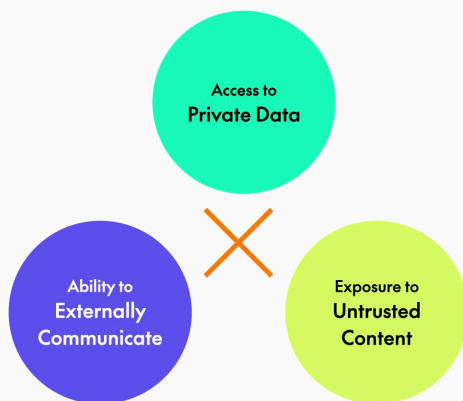
Many engineers ship a feature and move on. A good Product Engineer ships a feature and then lives in the observability/analytics platform, tools like PostHog or Datadog, to see if it actually solved the user's problem. These are tools that help the engineer close the loop on the 'why' behind what they built in the first place.

Increasingly, our engineers are getting closer to the user with session replay and user research tools like Dovetail and Lookback.

On top of that, robust CI pipelines, feature flagging, and canary releases are not just 'DevOps'; they are experimental infrastructure. All of this together is what allows you to safely put code in front of users, measure the impact, and iterate.

Defining the boundaries (the human-in-the-loop)

Product Engineers are starting to look less like authors of code and more like mini-managers of groups of agents. Software engineering as a role is becoming orchestration: deciding which agent gets which job, what context it needs, how to verify its output, and how the pieces fit into a system that actually works. Whilst agents can now handle all kinds of tactical work, strategic judgement such as knowing when to ship, when to pivot, and when to simplify, still rely on a human. That's not only because humans-in-the-loop lead to better decisions. We're also the ultimate security circuit breaker.



Consider the 'lethal trifecta' for AI agents, a security concept coined by developer Simon Willison¹. If an agent has access to private data, exposure to untrusted content from the internet, and the ability to communicate externally, it's extremely vulnerable to attack. This is why we use a combination of codified rules in markdown and agent settings, segregated accounts and mandatory human reviews with agents. These stop the tools, or anyone commandeering the tools, from going rogue. It sounds counterintuitive, **but a scalable human-in-the-loop is what makes the rest of the toolstack safe to use at all.**

Product engineering is hard, and that's the point

Our engineers have risen to the challenge of the broader skillset that Product Engineering requires. But holding a product mindset and an engineering mindset at the same time is genuinely hard, and the temptation, depending on which side you came from, is to drift back to your original speciality. To hold both at once, you have to treat the friction between them as the work, not as a problem to be automated away or solved.

Chapter 3:

The impact on teams and the roles within them

BY TIM CHECKLEY, CO-FOUNDER AT LOOMERY

The product teams I've worked in have generally followed a familiar shape: a product manager, a designer and an engineer (or several). A lovely clear triad, with specified roles, and handoffs between them. The broadening of engineering roles, and AI more generally, is quietly redrawing that picture. The roles still exist - for the time being at least - but the edges have gone soft. With the help of our robot friends, every individual can get more of 'their' work done. But they can also get other things done that require new skills, beyond their typical role. And when that happens, teams get smaller.

The engineering role keeps broadening

When an AI assistant writes a lot of the code, the scarce skill stops being typing and starts being judgement. Product leader Martin Eriksson talked about this at our conference Weave:London and in his great book, *The Decision Stack*.¹ Product Engineers feel this acutely and move sideways into product and design, because that is where the unanswered questions now sit.

"AI will give you speed, but you need clear strategic direction...and this comes from all those soft skills that we're really good at."

Martin Eriksson at Weave:London, 2026

We see a Product Engineer as an engineer who owns outcomes, not just code. Rather than waiting for a spec to build to, they work across the whole problem: understanding the user, the data, and the business goal, then using AI to handle much of the implementation so they can spend their time deciding what's worth building and whether it actually worked. In short, they hold a product mindset and an engineering mindset at the same time, blurring the old line between 'thinking' and 'doing'.

Engineers and designers are collaborating in code far more than they used to. For Loomery client Bite, an AI orchestration platform for structural engineers, our AI engineer coded up a working prototype for a piece of UX, a way to visualise overlays onto files, and sent it to the designer with a simple note: here is what we could do. That is a different conversation to a static mockup or crude sketch. Engineers can now put a real design concept in front of a designer, who takes it further. The feedback loop is undoubtedly quicker as a consequence.

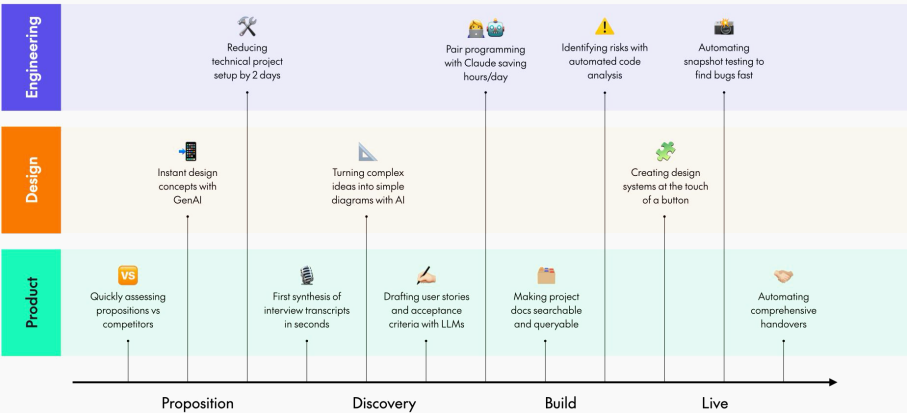
The PM and designer roles change too

But this is not only an engineering story. AI adoption is happening across every role. Product people like me are using AI throughout discovery and execution every day. Designers are getting from brief to prototype far faster than the old process allowed. At Monzo, a new client of ours, the design team is using AI to

create life-like coded prototypes - and even ship to production (with constraints and guardrails in place, naturally!)

It has also exposed something that was already happening. Some PMs had drifted into managing people and tickets rather than understanding the data, the competitive landscape, and the product strategy. When the mechanical work compresses, the manager-of-tickets product owner becomes pretty redundant.

What we're actually seeing is all the roles reshape. PMs move from spec-writers to context-providers and prioritisers: making sure the team understands the why, then deciding what matters most. Designers move from pixel-perfect handoff to setting the systems and the direction that engineers can run with on their own. The classic triad blurs. Smaller teams mean fewer handoffs, and fewer handoffs mean less translation loss between the person who understood the problem and the person who built the solution.



More can be done with less

We're seeing meaningful productivity gains across Loomery teams. Product Engineers who genuinely understand and own the why can make hundreds of small decisions a day without escalating. That is the source of the speed, and it only works if context is genuinely shared. Without that, empowerment just means building the wrong thing faster.

Sometimes the result is a team that ships more, and to a higher standard. Sometimes it means the team gets smaller. Increasingly we're forming product teams with fractional design and product roles, and sometimes neither at all. For business broadband provider Vorboss, a Loomery team shipped quickly without dedicated PMs or designers on the team. Product and design decisions still had to be made, of course, but they were absorbed by strategically-thinking Product Engineers.

Agents become team mates

The bigger shift underneath all of this is the move from AI as a tool to AI as a team mate, that you delegate tasks to

A tool is something you operate; a colleague is something you hand a task to, give context, and come back to check on.

Loomery teams are starting to work with agents in that second way, and it changes how a team runs day to day.

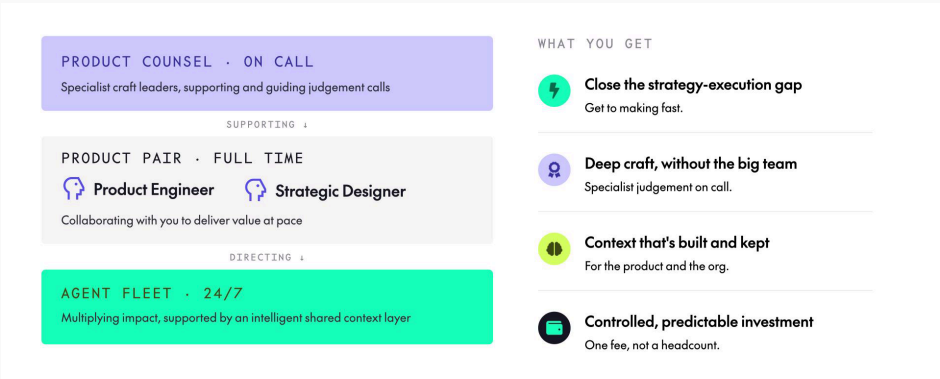
Agents still need a human in the loop, which makes a new set of skills important. I love the metaphor of us all becoming software pilots.² The teams getting real value are not the ones with the cleverest prompts, they are the ones who have got good at piloting the agents: overseeing, intervening and taking responsibility. Friend of Loomery Charlotte Roach (Product Director at Motorway) now runs a product team of 50, the majority of which are agents - a live example of what a 'team' looks like when agents become teammates.³

This raises a question we genuinely cannot answer yet. If some of your most productive team mates are not people, what is a team? How do you size one, how do you trust its output, and where does accountability sit when an agent gets it wrong? We are working that out as we go.

We are discovering the impact in real time

None of this is as clear-cut as the optimistic version suggests. A recent longitudinal study of AI coding assistants concluded that development work is being reorganised rather than simply accelerated.⁴ Most engineers reported spending less time on the traditional development tasks, with writing code declining most, and the balance tilting from creation towards verification. There is also anecdotal evidence that with less hands-on coding, engineers find it harder to get into flow state, as constant context switching eats into the gains. That may just prove to be teething pains, as we all adjust to new, better ways of working.

For individuals, the open question is whether we all become some flavour of cross-functional makers, with specialism collapsing into general capability. I don't think that's likely. My bet is that breadth becomes more important and specialism still matters: depth does not disappear, it gets rarer and more valuable. For teams, my bet is we'll continue to see smaller teams, a few Product Engineers plus a single product person or design strategist, achieving more than the cross-functional squad of a few years ago ever could.



From cross-functional team to product pair - the new team model we're testing at Loomery

I'll miss the traditional trio but equally, the change is exciting to experience. And the core product work for people like me, deciding what is worth building and making sure it actually delivers, is not going anywhere. It is just being shared out differently, and increasingly with team mates who are not human!

Chapter 4: The Product Engineering skillset

BY TIM CHECKLEY, CO-FOUNDER AT LOOMERY

In this chapter I want to talk about the skills that Product Engineers require to be successful; the skills that hold those smaller, blurrier teams together.

The first thing to acknowledge is it's a broad role! LLMs have proven their worth most clearly in software development; for engineers and non-engineers alike, they've 'raised the floor' and let everybody do things they previously couldn't. As a consequence, the Product Engineering role has absorbed parts of a traditional PM and design role. So, naturally, it suits T-shaped people: engineers who have real craft depth but also broad knowledge of other areas, together with the appetite to learn what they don't already know.

I want to cover here both the technical competencies central to Product Engineering (as opposed to traditional engineering) as well as the soft skills that make Product Engineers successful.

Soft skills, for those who haven't come across the term, are the interpersonal and self-management capabilities that shape how someone works with others and handles judgement calls. Think communication, collaboration, adaptability, emotional intelligence, feedback and judgement, versus knowing a programming language or a framework. We're of the opinion that the teams that will thrive in the future are those who nail these skills, made up of the best communicators, those with the sharpest judgement and the people who do the 'glue work' without being asked.

I've boiled it down to eight skills in total: four technical skills and four soft skills that have the biggest impact on successful Product Engineering.

Technical and craft skills

1. Designing and using AI harnesses

The first and most obvious skill is strong comfort and capability working with AI: simply put, being brilliant on the platforms that matter, whether that's Claude, ChatGPT, Perplexity, or Cursor. That means building comprehensive knowledge of effective prompting, prompt engineering techniques, tool use within LLMs, RAG and agentic search, MCP and so on. People often refer to this as building and managing your AI harness. It's worth saying Anthropic and OpenAI have some great training material on this; ten members of the Loomery team have been through the Anthropic Academy and are now Claude-certified developers.

The deepest part of this skill isn't actually that technical. It's the ability to orchestrate or 'pilot' agents: configuring them, delegating to them, knowing when to intervene. That's really closer to managing a team than prompting a tool, which I think tells you something about where this is heading.

And whilst Product Engineers are necessarily pro-AI, I do also think great Product Engineers have a healthy scepticism of AI output. The shift from creation to verification I flagged in Chapter 3 is a discipline: actually interrogating what the agents produce instead of assuming everything will go swimmingly first time.

2. Architecture and operability design

Product Engineering requires the ability to take ideas from vague requirements or signals through to working solutions you can release to users. A key part of that requires designing solutions that consider the wider systems architecture and balance all the trade-offs that solution architects have historically had to consider. This is particularly true when delivering at scale.

It also means being comfortable deploying and monitoring services on AWS, GCP, Vercel or wherever your stack lives, managing databases and data models, and keeping CI/CD pipelines reliable rather than treating them as someone else's job.

3. Discovery and experimentation

Product Engineers are most effective when they're comfortable undertaking product discovery and experimenting based on what they learn. These areas have traditionally been seen as product and design craft skills. Good Product Engineers are comfortable speaking to users directly. Fortunately, user research is a discipline which can be learned - and tools like Dovetail and Lookback help here.

When turning problems into solutions, quickly experimenting in the frontend means having enough UX fundamentals to build something coherent when there isn't a designer sitting next to you.¹

Product Engineering puts more emphasis on understanding product usage and using it to inform development decisions. At a minimum, that means being comfortable in a tool like Mixpanel or PostHog: reading funnels and retention, building a basic tracking plan, watching session replays to see how people actually use what you built. Fortunately, most engineers I know are pretty happy working with data, so this isn't a big leap.

Product Engineers should also know how to test safely in production using a combination of real user monitoring, observability tools, feature flags and canary releases. As Will explored in Chapter 2, that's not 'DevOps'; it's experimental infrastructure, and it's what lets you put code in front of real users quickly without betting the business on it.

4. Refactoring

Refactoring is another big skill that becomes more important, and by that I mean a solid understanding of how to improve code without breaking it, so you can guide AI to not just replicate patterns already there (which may add to tech debt) but also improve it and make it better.

At Loomery we're big fans of the 'boy scout rule': always leave the code cleaner than you found it.

Whenever you touch a file, you do a bit of unforced tidying, even if it's not related to the task you're actually there for. That might be a better variable name, a broken-up function, or a stale comment removed. Those improvements compound and make it easier to collaborate not just with humans, but with agents too.

Soft skills

5. Product judgement

If Product Engineers are owning product changes end-to-end, they need to think and act like PMs too. That means knowing when good-enough beats perfect, and when to ship, simplify or pivot, then owning the outcome past the point of shipping code. As I explored in Chapter 3, some of our teams are now shipping without a dedicated designer, which only works if the engineer has the design taste and product sense to make good calls alone. Empathy is part of it: seeing things from the user's perspective. So is holding a high-quality bar. A huge amount of software is being produced without any of the traditional quality supports, so great Product Engineering is just as much, and if not more, about quality as it is about quantity.

“The shift from software engineer or architect to Product Engineer is a shift in what you optimise for. You need to think beyond just technical excellence, and more about making the right trade-offs, understanding the user, and owning the outcome.”

Uran Fetahu, Lead Product Engineer at Loomery

6. Communication

Product Engineers act as the connective tissue of a team, and ours are client-facing too. Knowing how to write well is important so you can summarise goals and plans, draft docs that move conversations forward, and deliver concise async updates that let smaller teams coordinate without a PM translating intent for everyone. It's also about great verbal comms: particularly giving and receiving feedback well. Feedback becomes critical precisely when you strip out process and run on trust instead, which is exactly the smaller team shape I explored in the previous chapter. It's what makes trust-based teams work rather than collapse.

"One challenge for me has been managing client expectations solo, and getting the most out of the people and time in meetings - both of these are about communication, really."

Tom Holmes, Senior Product Engineer at Loomery

7. Curiosity

There's a lot of ambiguity in software development generally, and I'd suggest there's even more of it when agents play an active role, and teams get smaller. Great Product Engineers need to be comfortable with that ambiguity, but also genuinely curious about how to resolve it, both personally and for the team and product. That means always prototyping and experimenting; it means getting low-fi versions of features out into the world to gather feedback before scaling to a larger build.

“Working with University of the Arts London, we ran near-daily feedback loops, testing multiple options with both their team and their students. That pace sharpened our sense of direction and let us try ideas we'd normally have skipped; e.g. we prototyped a 3D campus map, put it in front of students, but cut it based on testing feedback. We were able to be creative, but also learn quickly.”

Edoardo Trotta, Senior Product Engineer at Loomery

8. Self-awareness

One risk is drifting back to your old specialism when the going gets uncomfortable; it helps to be conscious of that. Will put it well in Chapter 2: the friction between holding a product mindset and an engineering mindset at once is not a problem to solve; it's the work itself. Self-awareness is what stops you quietly retreating to whichever side feels more comfortable. It's also important to be honest with yourself about what you're capable of and what you're not, so you know when to bring someone else in. Craft depth in design and product is still important to making delightful products, and craft specialists will know the answers that you may not!

Can these soft skills be learned, just like the technical skills? I think so. Not through Anthropic or OpenAI courses, sure, but through conscious practice, real-world application, and feedback. And of course, it's this stuff that AI is bad at, so it's only going to get more important.

Chapter 5: Getting started and the barriers you might face

BY TIM CHECKLEY, CO-FOUNDER AT LOOMERY

By now, you should have a good idea of what Product Engineering is, the tools and skills Product Engineers deploy, and the new team shapes that are emerging.

At Loomery we've been investing in broadening the role and upskilling our engineers to succeed as Product Engineers. We've also been helping our clients embed Product Engineering. That's made us aware of the reality that embedding Product Engineering within an organisation is not necessarily a given. It relies on cultural, structural and operational enablers to succeed. This final chapter is about the cultural and institutional barriers you may encounter as a champion for Product Engineering within your organisation - and ways you can overcome them.

Hierarchy and decision rights

Product Engineering needs small empowered teams that can decide and act, but many big companies push decisions up through someone who isn't in the room. By the time the answer comes back, the team's learned new things and the question's out of date. The model only works if leadership is genuinely hands-off on how things get executed while staying close to why they matter.

Tips

- Agree upfront which decisions the team owns outright and which genuinely need escalation, in writing, before work starts. Most things (design direction, which experiment to run, which feature to cut) should belong to the team.
- Reserve escalation for the handful of things that are actually costly or hard to reverse: things like brand-defining decisions, spending that's particularly high, and anything with legal exposure (no one wants to end up in prison!).
- Build trust with evidence, starting with small low-stakes bets and showing the misses as well as the wins.

Risk appetite and the definition of failure

Product Engineering means shipping fast to learn (one of the reasons we love the role at Loomery!). Depending on the size and risk appetite of the organisation, you might be more likely to encounter 'nothing goes out the door until it's been through six sign-offs.' Of course, in regulated industries that's often for good reason: compliance, legal, brand, and security are all built to stop mistakes reaching customers, so a team of Product Engineers optimised to make small visible mistakes on purpose looks, to them, like it's just making mistakes.

The solution here is to start small. Get Product Engineers shipping quickly into a contained space (a closed beta; an internal cohort; a small percentage rollout) where the blast radius is small enough that risk teams can say yes. In the spirit of working cross-functionally, it's also a good idea to get potentially tricky stakeholders in the room at the start so they can help your Product Engineers design the safe way to tackle their concerns.

Access to customers and data

In some organisations, the people building the product don't have access to the customers. We've seen customer access get rationed through account managers and research teams, customer data get locked down behind permissions, GDPR caution and over-zealous data governance policies - and that doesn't work for Product Engineering. It means engineers are building blind, and finding out too late if they didn't hit the mark.

It matters to treat proximity to the customer, and the data about them, as something built into the process. **We encourage Product Engineers to observe (and run!) user interviews and usability sessions.** Obviously, they need access to analytics and support data. If any data are particularly sensitive, they can be anonymised or role-scoped as required.

Existing delivery model and tooling debt

Legacy architecture and years of process built around the old way of working create friction a small empowered team can't work around on its own. Product Engineering assumes you can actually ship, which assumes CI/CD, environments, AI licenses (you'd be surprised...), and a level of platform maturity that some tech functions simply don't have.

The best approach here is to treat platform readiness as its own workstream, run in parallel to embedding new role shapes. Give the team a modern slice to build in (a new service, a decoupled front end, a sandboxed environment) so they can move at Product Engineering speed from day one, while a separate effort chips away at the legacy tech debt behind them. Don't wait for the whole estate to modernise before letting anyone go fast.

Functional silos

Product Engineering pods are by definition small teams of generalists who can operate across product, design and engineering. Lots of organisations are staffed and structured by function. Layering an AI-forward, cross-functional pod on top of a siloed org can create confusion about who reports to whom and who's accountable, and the existing functional teams can feel like the new pod is either poaching the fun bits of their job (experimenting with AI!) or showing them up.

A few things can help here. Be explicit about the new pod's mandate and its relationship to existing teams from day one. Also recruit or develop genuine T-shaped people, rather than assuming that any IC engineer can (or wants to) adapt to the role. Give existing teams a stake in your new Product Engineering pod's success (rotations into the new pod, shared roadmap visibility, credit when it's due) so it's not 'them and us'.

How to get started

Here are a few practical steps to get you going

Find your people Pilot Product Engineering with a small pod - identify your early champions. Over time, you can foster a Community of Practice.	Design the role Run a session to design the shape of the Product Engineering role in your organisational context.	Secure a sponsor Find a senior leader who can endorse the Product Engineering shift, give it visibility, and help secure basic resources.
Agree on a goal Tie the pod's objectives directly to business outcomes (like reducing checkout times or improving retention) rather than individual functional tasks. Give the new pod free rein to experiment against this goal.	Shift left Bring engineers into ideation and discovery early instead of waiting until planning implementation.	Build the product mindset. Ensure engineers understand the business model, strategy and immerse themselves in the user's context, not just the 'requirements from the business'.
Focus on outcomes over output Measure success by customer value and behavioural changes rather than lines of code written or features shipped.	Coach and upskill Consider proactive coaching and training, which really accelerates the pace of change	Celebrate the wins This helps tie a Product Engineering culture together, making the transition a more motivating journey. People will naturally gravitate towards the role when they see the value it brings.

We started this Guide Book by saying the walls between 'thinking' and 'doing' are collapsing. Five chapters later, hopefully it's clear that this is an opportunity for product teams. The organisations that figure out how to rebuild their teams, tools and culture around this shift won't just ship faster. They'll build better products because all the people building them will actually understand the *why*.

Naturally, embedding Product Engineering is something we love doing at Loomery, so if that's your goal, please get in touch.

tim@loomery.com

Endnotes

Chapter 1:

1. Lauren Edmonds, 'Spotify CEO says its top developers "have not written a single line of code" in 2026', Business Insider, 2026. [businessinsider.com/spotify-developers-not-writing-code-ai-2026-2](https://www.businessinsider.com/spotify-developers-not-writing-code-ai-2026-2)

Chapter 2:

1. Simon Willison, 'The Lethal Trifecta for AI Agents', Simon Willison's Weblog, 2025. simonwillison.net/2025/Jun/16/the-lethal-trifecta/

Chapter 3:

1. Martin Eriksson, The Decision Stack: How Strategic Clarity Unlocks Organisational Momentum, 2026. thedecisionstack.com
2. Software Pilotry. softwarepilotry.com
3. Charlotte Roach, LinkedIn. [linkedin.com/in/charlotte-roach](https://www.linkedin.com/in/charlotte-roach)
4. Annie Vella and Kelly Blincoe, 'The Impact of AI Coding Assistants on Software Engineering: A Longitudinal Study', arxiv, 2026. arxiv.org/abs/2605.23135

Chapter 4:

1. Peter Makeshoof, '20 Essential UX Principles Every Designer Should Know', Designer Daily, 2025. designer-daily.com/20-essential-ux-principles-every-designer-should-know-193200

Endnotes

Read more from Loomery:

- Brett Thornton, 'You Can Move Faster With Technology, And You Should', Loomery, 2020. loomery.com/insights/you-can-move-faster-with-technology-and-you-should
- George Kean Proudfoot, 'The Age of Software Abundance Is Here. So what do we do now?', Loomery, 2026. loomery.com/insights/the-age-of-software-abundance-is-here
- Milly Allatson, 'The Bottleneck Isn't the Code Anymore', Loomery, 2026. loomery.com/insights/the-bottleneck-isnt-the-code-anymore
- 'Vorboss: Case Study', Loomery, 2026. loomery.com/vorboss



Get in touch: hello@loomery.com