Malorean Corp.

A novel to identify with

A panorama to understand

Developers, AI and small business : an (almost) true story of transformation

What SaaS ERP teams are really experiencing with AI today



LAURENT DESÉCHALLIERS

Malorean Corp

> *Not long ago, turning an idea into a prototype could take weeks. Today, someone working with artificial intelligence can do it in just a few days. That is a remarkable change. But the real transformation may lie elsewhere: in how we work together.*

Laurent Deséchalliers

A Note on the Translation

This edition is a machine translation. An AI produced it from the French text.

No human translator has been over it. The ideas, the structure and the examples follow the original closely; some of the phrasing, though, may read awkwardly, and a few passages still carry the shape of the French.

That is the price of having this book exist in several languages rather than in French alone. Where a wording is in doubt, the French edition is the one that counts.

Introduction

What strikes me most about AI is not that it can write code. It is how quickly it lets us do things that, until quite recently, would have taken days or weeks of work.

An idea can become a prototype before the discussion about whether it is feasible has even finished. A developer can explore an approach without waiting for an entire team to become available. Someone who cannot code can sometimes build a first version of what they have in mind. None of this means that a prototype is a reliable product, or that complexity has disappeared. But simply being able to try things changes our sense of what is possible.

Yet our companies were not designed for a world in which a first solution can be produced so quickly.

They have organised themselves around people's skills, the time available, handovers and the decisions needed to keep work moving. These arrangements exist for good reasons. But as people's capabilities change, they begin to raise new questions. Do we still need to work in the same order? Wait for the same approvals? Reserve the same activities for the same professions?

These questions do not concern developers alone. Product Managers, designers, salespeople, team leaders and many

others are also discovering new ways of working. Some can now do directly part of what they previously asked colleagues to do. Others are seeing the nature of their work change. What seemed to be a question of tools is beginning to affect how we collaborate and make decisions.

It would be tempting to announce a revolution whose outcome we already know. I do not think we are there yet.

Some companies are experimenting with new ways of distributing work and responsibility. Others are moving more cautiously, sometimes for good reason. Certain practices look promising, but we do not yet have enough experience to know which will stand up to the pressures of everyday work. And a solution that suits a small team will not automatically be right for a larger, more regulated or more established company.

That is why I did not want to write a manual prescribing how companies should organise themselves around AI. This book is an attempt to understand what is really changing: what the tools make possible, how companies respond, what they gain, what they risk losing and the questions they will have to decide.

To do that, I have chosen two approaches that intersect.

The first is observation. We will look at real companies, publicly documented practices, studies and experiences. I will try to distinguish what is established from what remains a hypothesis, and not confuse an appealing announcement with a demonstrated result.

The second is a story. We will follow Malorean Corp., a fictional company whose employees gradually discover what these new possibilities mean for their work. They have no blueprint for reinventing their organisation. They have customers, habits, urgent demands, convictions and, sometimes, disagreements. Fiction proves nothing; it lets us see these changes through the eyes of the people experiencing them.

Both perspectives matter. Real examples help us avoid mistaking our intuitions for facts. The story reminds us that organisational change is not just about reporting lines: it affects trust, skills, responsibilities and everyone's place in the company.

Ultimately, one question runs through the pages that follow: **what happens to a company when the people who work there discover that they can do far more than their jobs have allowed them to do until now?**

About the Author

I have worked in the software industry for several decades and am now a Product Manager. My career has taken me from development and software architecture to team organisation, and then into product management, product operations and product marketing.

This experience puts me in a good position to observe how artificial intelligence is evolving and how it affects our work. I encounter these changes every day in my professional life, but also through conversations with fellow Product Managers. We share our experiences, discoveries and questions. Those discussions often lead me to look beyond the tools themselves: what is really changing in our professions, in the decisions we make and in the way we work together?

As I observed and discussed these changes, more and more questions emerged. I wanted to take a step back, compare our impressions with what was happening in other companies and share what I was learning. That is how this book came about.

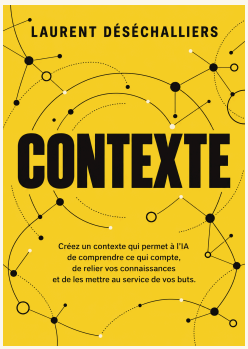
Writing has long been part of my career. I have contributed to specialist publications and worked as a technical reviewer for M2 Édition and Eyrolles. Today, I write at malorean.net about Product Management, AI, ways of working and information systems.

I also contribute to *Proposition Marketing*, an independent magazine covering strategic and operational marketing. Writing for readers in marketing and company leadership encourages me to make product-related questions accessible without assuming that everyone shares the same professional vocabulary.

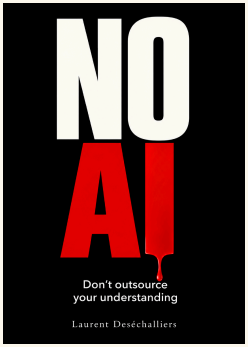
Laurent Deséchalliers

- laurent@malorean.net
- https://malorean.net

More by the Author



Le Contexte — How to build a context that lets AI understand what matters, connect your knowledge and build on what you learn in service of your goals. *Released: October 2026.*



NO AI — *Don't outsource your understanding.* Methods without AI to maintain our memory, deepen our thinking and exercise our critical sense. Practices to develop our own abilities and make better use of AI without handing it our understanding. *Expected release: late 2026.*

More information at https://malorean.net.

Acknowledgements

First, I would like to thank the Product Managers I regularly exchange ideas with. We exchange views on how Product Management and software development roles are evolving and, more broadly, on the effects of artificial intelligence on work and the way companies are organised. These conversations, sometimes lively, are a constant source of questions, challenges to our assumptions and fresh ideas. Some of those reflections have found their way into this book. Thank you for your curiosity, for sharing your experiences and for being willing to debate.

And then there is Christine, my wife, whom I want to thank. At the end of my first book, *Le Contexte*, I promised her a break. Instead, I started on this one almost straight away. Fortunately, this second adventure has taken up less of our evenings and weekends than the first. Thank you, Christine, for your patience and support, even when my thoughts end up turning into another book.

A Note on the Examples

The ideas and examples in this book draw on more than thirty years of work in technology. They come from situations I have encountered across a range of projects and professional settings, and from many conversations with product managers, CTOs, engineering managers and others at different companies. In other words, they span several organisations and several stages of my career.

To serve the purpose of the book, I have altered, generalised or combined these situations. The use cases, figures and metrics illustrate recurring problems and what I have learnt from them. They are constructed examples: none describes a real client, product or organisation, or reflects my employer's data, practices or positions.

The analysis, methods and opinions expressed in this book are mine alone and do not represent anyone else.

Was This Book Written with AI?

Yes. Entirely in collaboration with artificial intelligence.

I know the question carries particular weight today. People wonder whether an article, a novel or a book was written with AI. Some even run texts through detection tools, as though it were essential to separate what came from a human from what came from a machine.

I prefer to be open about it: AI has been involved in every stage of this book, from researching sources and organising ideas to drafting, checking references and looking for blind spots. But this collaboration deserves an explanation, because it does not, on its own, tell you who did the thinking behind the book.

For a long time, undertaking a project of this scale could mean working with people who helped find documents, challenge ideas, review text, structure material or prepare it for publication. AI now gives me access to some of those capabilities. It works quickly, suggests ways of phrasing things, points out inconsistencies and sometimes asks questions that had not occurred to me. It lets me explore far more than I could manage alone in the time I have available.

To write this book, I used a context engine built on ChatGPT technology, with Notion to retain information over time.

Sources, ideas, decisions and working versions can therefore be stored and retrieved across conversations. I have already described this approach in my first book, *Le Contexte*. In it, I explain how such a system can make AI something like a supercharged personal secretary: rather than merely answer a question, it keeps track of projects and helps me move them forward over time.

There is, however, an essential distinction: **the ability to carry out work is not the responsibility for understanding and choosing what to do.** The questions guiding this book, the professional experience behind them, the interpretations, the decisions and what I ultimately choose to argue for or leave aside are mine. AI can disagree with me, challenge my assumptions and help me express a thought more clearly. It cannot replace my judgement or my responsibility as an author.

I also draw a distinction between this book and works in which the writing itself is central to the work. Whether we read Shakespeare in François-Victor Hugo's translation, Cervantes, or Cizia Zykë in the original, the voice, rhythm and style matter as much as the story being told. In a translated work, the translator's choices are part of the experience too. The role we might want AI to play in that kind of literature raises other questions, which I do not claim to settle here.

This book includes fiction, but its main purpose is to understand a technological and organisational transformation.

The writing should be enjoyable and clear; above all, it serves the ideas, facts and questions I want to share.

AI has made this project possible on a scale I probably could not have reached with my own resources, alongside my professional work. It allows me to pursue personal projects like this book without giving up the depth of thought they demand.

And there is something rather fitting, in the end, about writing a book on the changes brought about by AI while working in exactly this way.

Laurent Deséchalliers

Contents

A Note on the Translation 3

Introduction 4

About the Author 7

More by the Author 9

Acknowledgements 10

A Note on the Examples 11

Was This Book Written with AI? 12

Malorean Corp. 17

1 Malorean Corp. — An Extraordinary Day 18

2 AI has dramatically increased developers' capacity to build software 25

3 Malorean Corp. — The Day After 35

4 New Organisational Models Are Emerging 48

5 Malorean Corp. — What We Can Do, and What We Should Do 59

6 Blurring Roles, Enduring Responsibilities 73

7 Malorean Corp. — What Alice Knew but Had Never Written Down 84

8 Greater Autonomy Requires Clearer Boundaries, Not Fewer 103

9 Malorean Corp. — Two Reasonable Decisions 119

10 Will AI Make Middle Management Disappear? 135

11 Malorean Corp. — Time Regained 153

12 Conclusion 172

Malorean Corp.

1 — Malorean Corp. — An Extraordinary Day

Malorean Corp.‘s headquarters were in Le Havre. But since Covid, that address had said little about where the work actually happened. Employees were scattered across France. They met on camera, talked over shared screens and, most of the time, nobody needed to be in the same room to get things done.

Malorean Corp. developed an online ERP system: the kind of software that gradually finds its way into almost every part of a company’s operations. One customer might be a small business with a handful of employees and very practical needs. Another might be a large group with complex processes, particular requirements and little patience for anything imprecise.

Most customers were French. There were a few users in Germany and Spain, but mainly in subsidiaries of groups that were already customers in France. Hubert Félix Duca, the CEO, had bigger ambitions. He wanted Malorean Corp. to establish a real presence across Europe.

The difficulty was that an ERP system did not cross borders as easily as a brochure website. Rules differed from one country to another. So did payroll practices, administrative obligations and how companies worked. Hubert understood these

constraints. His background was in business operations, management, accounting and finance, not software development. To him, a feature was valuable only if customers could use it without running into trouble.

Within the company, everyone's role was fairly clear.

Two teams of five developers worked on the software. Alice and Bob, the two Product Managers, worked alongside them to understand customers' needs, choose worthwhile improvements and give the product direction. Alice had once been a developer; she could join technical discussions without losing track of what mattered. Bob had a business-school background and approached things differently. They brought different experience to the job, but worked very well together.

A sales team sought new customers and maintained existing relationships, supported by one person in marketing. Five specialists worked with prospects before a sale and supported implementations afterwards, either on site or remotely. They did more than install the ERP: they helped companies clarify their needs, adjust their processes and often continued supporting them for the six months following a sale.

The support team had twelve people. Seven looked after smaller accounts and five handled larger ones. They were often the first to hear that a feature was forcing a customer to work around the software rather than use it as intended. Two lawyers looked after Malorean Corp.'s legal affairs but also contributed their expertise to product design, particularly

when a change affected sensitive rules such as payroll regulations.

That was how the company worked. Not perfectly, of course. Software used by so many customers accumulates requests, exceptions, habits that are hard to change and features that remain in place because nobody has had time to rebuild them. But everyone had a reasonable idea of what they were supposed to do.

Then ChatGPT began appearing in conversations.

At first, there was no grand transformation programme. A few months earlier, two particularly curious employees had tried it: Julie Morel in sales and Lucas Martin in development. They had started using it for their own purposes, then shared what they were managing to do.

For Julie, the benefits were immediately obvious. She would draft a difficult email, ask for a clearer way to put something, organise her notes before a meeting or turn a string of ideas into a presentation outline. Sometimes she would even prepare content for a few slides. ChatGPT did not decide what she should tell a customer. She supplied the substance, reviewed the suggestions, corrected the wording and sent only what she was prepared to stand behind.

She never claimed the tool knew her customers better than she did. She was simply glad not to spend half an hour formatting something she already knew.

Philippe Garnier, her colleague, watched from a distance. An experienced salesperson used to dealing with major accounts, he knew the people behind the contracts, the disagreements that never appeared in meeting notes and the little remarks that could change the meaning of a conversation.

‘You should try it for your emails,’ Julie had told him one morning. ‘Even just to tidy them up.’

‘I know how to write my own emails.’

He had not meant to be rude. He simply could not see why he should hand part of his job to a tool when, to his mind, his value lay precisely in knowing his customers and knowing how to talk to them.

Julie had not pressed the point. She had a meeting to prepare for.

Lucas used ChatGPT in a completely different way. An experienced developer based in Lyon, he was not merely looking to save a few minutes on repetitive tasks. He spent a great deal of time experimenting, asking increasingly demanding questions and starting again when an answer did not satisfy him. Julie chiefly wanted a useful result; Lucas wanted to find out just how far he could push the technology.

Word of these early experiences eventually reached Hubert. At around twenty euros per person per month, the calculation seemed straightforward: if everyone saved even a little time on tedious tasks, the investment would more than pay for

itself. Malorean Corp. therefore decided to make ChatGPT available to everyone.

There was no revolution proclaimed through an endless series of presentations. People could use it when they saw a benefit. Some opened it every day, others rarely, and Philippe not at all. For most of them, AI remained an assistant: you explained what you wanted, it made suggestions, you checked them and then you decided.

Life at the company went on.

On Wednesday, during a discussion between the Product and Development teams, an old issue came up again: leave management.

Nobody particularly liked that module. It was old, inflexible and offered few options. It worked well enough for customers to keep using it, but not well enough for them to stop complaining about it.

An employee who wanted to book time off had to use their work computer. There was no simple way to do it on a phone on a Sunday evening, while thinking about an upcoming holiday. And the calculation of leave had an even more embarrassing flaw: the module did not handle public holidays properly. For a request covering Monday to Friday, if Wednesday was a public holiday, it could deduct five days instead of four.

Support knew about the problem. So did Sales. For some customers, the module had become a recurring source of

frustration. They kept it because it was there, not because it really met their needs.

‘We ought to rebuild it properly,’ someone had concluded.

Nobody disagreed. But ‘properly’ meant redesigning the interface, fixing the calculation rules, making the whole process usable on mobile and replacing parts of the module that had aged badly.

They were talking about roughly two months’ work. There were other priorities. Once again, it went onto the list of things that needed doing.

On Friday afternoon, at the end of a video call, Lucas asked for a few minutes.

‘I’ve got something to show you.’

He had not said what it was. Nobody knew he had been working on the leave-management module since Wednesday’s discussion.

His screen appeared in the meeting. An interface opened: clean, modern, with the important information exactly where people expected it.

Lucas showed them how to submit a leave request. Then he narrowed the window to the width of a phone. The interface adjusted. You could carry on, choose your dates and confirm the request without going back to a desktop computer.

He moved to a second example: a week from Monday to Friday, with a public holiday on Wednesday.

The total showed four days.

He changed the dates. The result recalculated. He returned to the mobile view, scrolled through a few screens and continued clicking around. These were not a series of static mock-ups: the forms responded, the calculations worked and the whole process was functional.

At first, a few colleagues had asked ordinary questions: where was a particular option, what happened if someone changed a date? Then the questions stopped.

In the small video-call windows, people's expressions had changed.

On Wednesday, they had discussed two months of work. On Friday, Lucas was showing them a working first version. He had not asked for help. He had told nobody. And, for the moment, he was not explaining how he had done it.

'Hang on,' someone finally said. 'Is that the module we were talking about the day before yesterday?'

Lucas confirmed it.

Silence fell.

On their screens, everyone was looking at the same thing: work they had expected to plan over several weeks, appearing in just two days.

Nobody understood how it had been possible.

2 — AI has dramatically increased developers' capacity to build software

For decades, the organisation of software development has been shaped by one simple constraint: **building software is expensive**.

Turning an idea into a product took time, scarce skills and a great deal of coordination. A feature had to be scoped, broken down, prioritised, developed, tested, reviewed and deployed. The number of developers available, their expertise and the time they could devote to a subject placed practical limits on what a company could undertake.

That scarcity shaped organisations. Companies learnt to protect developers' time, prepare their work in advance, minimise interruptions, organise backlogs and make difficult choices long before development began. Much of modern product management evolved in that environment: the aim was to avoid wasting expensive engineering capacity on the wrong idea.

The arrival of coding assistants, followed by agents capable of carrying out broader tasks, is beginning to change that equation.

This is no longer just better autocomplete

Early AI tools for developers could be seen as another generation of productivity software: completing a function,

explaining a piece of code, writing a test or suggesting a refactor.

The next generation changes the nature of the work far more substantially. An agent can be given a task, explore a codebase, modify several files, run tests, fix certain errors and return a change ready for review. A developer can therefore delegate a substantial share of the implementation and, crucially, set far more work in motion than before.

The result is not merely that ‘developers write code faster’. **The amount of software work one person can start and move forward increases dramatically.**

Some companies are beginning to report striking results. AMPECO, describing its own move to an AI-native engineering model, reports a fourfold increase in engineering velocity and 50% fewer bugs per story. These figures should not be treated as market averages: they are results reported by one company, in a particular setting. But they illustrate the scale of the change being pursued. The aim is no longer simply to save a few minutes when writing a function, but to substantially increase a team’s capacity to get work done. [1]

Those striking claims from one company are not the only kind of evidence available. I have experienced the shift myself with Claude Code. In four days and more than 180 commits, I built an initial version of a product, complete with screens, APIs and tests. Using my previous methods, I estimate that it would have taken me one or two months. I wrote about the

experience on Malorean.net in July 2026. It plainly cannot establish a general productivity gain. What it does show is what a product manager with some development experience can now achieve — and the challenges of quality, maintainability and keeping control of AI-generated code. [7]

This changes the scale of the question.

If AI made coding only a few per cent faster, an organisation could largely stay as it was and absorb the improvement. But when the capacity to build software grows much more quickly, other constraints become relatively more important.

Faster coding does not automatically mean a faster company

A software team is not a production line with only one stage: writing code.

Before building anything, people need to understand a problem, decide whether it deserves investment, choose a direction and clarify the constraints. Afterwards, they need to review, test, secure, deploy, observe and maintain what they have built. Throughout the process, they must keep making trade-offs.

If just one of these stages speeds up dramatically, it merely sends more work towards the others.

Code can be produced faster without customer understanding improving at the same rate. Prototypes can proliferate without anyone becoming better at deciding which are worth pursuing.

More changes can arrive for review without the human attention needed to assess them growing proportionately.
The point is not that AI suddenly makes all the other activities unnecessary.
It is that **AI shifts the bottlenecks**.
What used to be obscured by the slow pace of implementation becomes more visible: whether the right problem has been chosen, whether the intention is clear, how trade-offs are made, how changes are validated, how everything fits together and whether the team learns after release.

Building more is not the same as creating more value

This distinction is becoming essential.
When building software was difficult, the volume delivered could seem like a reasonable proxy for an organisation's ability to create value. That shortcut was already imperfect.
With AI, it becomes dangerous.
A team can now generate more code, more pull requests, more prototypes and more features without making the product any better. It can even make its overall performance worse if the extra volume brings more defects, rework, costs or complexity to maintain.
In other words:
Being able to build something is not the same as knowing what is worth building.
That is why activity metrics are even less adequate than before. The number of changes, the amount of code written or

the mere use of an AI tool tells us nothing about whether a team has created more value.

Canva offers an interesting example of this shift towards impact. The company developed a system that enables Product, Engineering and Data teams to estimate the commercial impact of their experiments and connect local decisions more directly with their business consequences. [2]

The question itself changes.

Before:

How much can we build?

Now:

Of all the things we can build, which ones are genuinely worth building?

> ***The problem of software productivity is more than forty years old***
>
> *In 1986, Fred Brooks presented* No Silver Bullet*, published in* IEEE Computer *in 1987. He distinguished the* ***accidental*** *difficulties of software development — programming languages, tools and translating a design into code — from its* ***essential*** *complexity: understanding, designing, connecting and evolving a system. In his view, no single technical or managerial breakthrough could be expected to increase software productivity, reliability and simplicity tenfold within a decade. [3]*

> *The essay was later included in the 1995 anniversary edition of* The Mythical Man-Month*, first published in 1975. [4] Generative AI may change the scale of implementation, but it does not automatically make the chosen problem the right one. That is a contemporary reading of Brooks,* ***not a prediction he made about AI agents****.*

Judgement is the new bottleneck

As implementation becomes cheaper, several other capabilities become relatively more valuable: knowing the customer, understanding the problem, grasping the strategic context, expressing a clear intention, assessing risk, having a feel for what makes a good product, knowing when to say no and taking responsibility for decisions.

In an article published on Malorean.net in March 2026, I made a related argument: as technical tools become more accessible, competitive advantage shifts towards understanding context — the business, users, customers, technical constraints, competitors and regulations. This is a strategic argument, not an experimental finding. It makes the meaning of good judgement more concrete: knowing a situation well enough to choose the right thing to build, rather than merely knowing how to build it. [8]

AI makes the ‘how’ more accessible.

It does not automatically make the ‘why’ obvious.

It enables a solution to be built before an organisation has finished understanding the problem. It can quickly deliver a

customer request that the company should have recognised as outside its target market. It can generate multiple versions of a feature without anyone knowing which one actually leads to a better outcome.

The easier building becomes, the more important it is to be disciplined about what *not* to build.

Ramp illustrates this well. Its CPO emphasises the need to be clear about priority ideal customer profiles, the problems the company wants to solve particularly well, and the importance of turning down certain requests — even profitable ones — when they pull the product in another direction. [5]

In July 2026, Marty Cagan described this mismatch as the *AI productivity paradox*. His argument is that building faster does not solve the problem of ideas that were never worth building. He distinguishes what we build to learn from what we release to achieve a result. This is a product practitioner's analysis, not a measurement proving that every company is affected in the same way. [6]

The paradox of AI is therefore this:

It greatly increases our capacity to act and, as a consequence, makes choosing our actions more important than ever.

A transformation of the system, not just the tools

One response would be simply to give every developer a coding assistant and ask teams to carry on as before.

That produces local improvements.

But as those gains grow, an organisation runs into its other limits: its ability to understand, decide, coordinate, check and absorb the work being produced.

At that point, the issue is no longer just the adoption of a tool. It becomes an organisational question.

An organisation designed for a time when engineering capacity was its scarcest resource has no reason to remain optimal as that scarcity diminishes.

One question then becomes unavoidable:

What happens to an organisation designed to protect a scarce resource — engineering capacity — when that capacity becomes far more abundant?

Bibliography

[1] How we built an AI-native engineering system — AMPECO

- URL: https://www.ampeco.com/blog/how-we-built-an-ai-native-engineering-system/
- First-hand account of an engineering organisation rebuilt around agents, reporting a substantial increase in capacity.

[2] Measuring Commercial Impact at Scale — Canva

- URL: https://www.canva.dev/blog/engineering/measuring-commerical-impact-at-scale/
- Explains how Canva connects Product and Engineering teams' experiments to accessible estimates of their commercial impact.

[3] No Silver Bullet: Essence and Accidents of Software Engineering — Fred Brooks

- URL: https://www.cgl.ucsf.edu/Outreach/pc204/NoSilverBullet
- Essay presented in 1986 and published in 1987, distinguishing essential complexity from accidental difficulties in software development.

[4] The Mythical Man-Month, anniversary edition, chapter 16 — Fred Brooks

- URL: https://www.oreilly.com/library/view/mythical-man-month-the/0201835959/ch16.xhtml
- Publisher's contents for the 1995 edition, confirming the later inclusion of *No Silver Bullet*; chapter preview only.

[5] 5 insights from Ramp's CPO on building for hypergrowth — Ramp

- URL: https://ramp.com/blog/5-insights-from-ramps-cpo-reddit-ama
- Ramp's CPO stresses focus, priority customer profiles and the need to refuse requests that would pull the product off course.

[6] The AI Productivity Paradox — Marty Cagan, Silicon Valley Product Group (2026)

- URL: https://www.svpg.com/the-ai-productivity-paradox/
- July 2026 practitioner's analysis distinguishing AI-enabled speed from actual outcomes; not a causal demonstration.

[7] Quatre jours de vibe coding dans la peau d'un PM rouillé — Laurent Deséchalliers, Malorean.net (2026)

- URL: https://malorean.net/articles/2026-07-28-quatre-jours-de-vibe-coding-dans-la-peau-d-un-pm-rouille.en.html
- Personal account of building with Claude Code: speed, technical difficulties, testing and code quality; neither a controlled study nor a generalisable finding.

[8] Dans le logiciel, l'avantage ne sera plus la technologie. Ce sera la compréhension du contexte — Laurent Deséchalliers, Malorean.net (2026)

- URL: https://malorean.net/articles/2026-03-23-dans-le-logiciel-lavantage-ne-sera-plus-la-technologie-ce-sera-la-comprehension-du-contexte.en.html
- Author's strategic essay on the growing importance of customer, business, regulatory and technical context; not a quantitative demonstration.

3 — Malorean Corp. — The Day After

‘I used AI.’

Lucas said it almost matter-of-factly, as though those three words explained everything everyone had just seen.

He was still sharing his screen. The new leave-management module filled most of the window. A few minutes earlier, he had shown how someone could request time off on their phone, and how the software took public holidays into account in its calculations.

‘Hang on,’ Alice said. ‘When you say you used AI, do you mean it helped you write the code?’

‘Yes. Well, not just that. I used it to design the module, understand the existing code, build the screens, do the calculations, write the tests… Pretty much everything, really.’

‘And you did all that since Wednesday?’

‘Wednesday afternoon. I finished the last few bits this morning.’

Alice looked at the faces on her screen. A few people were smiling. Others seemed to be trying to work out what they had missed.

‘But you didn’t have a spec,’ she pointed out.

‘No. We’d talked about it on Wednesday. I knew the module and its problems. I started by asking the AI to analyse what was already there. Then we worked through it bit by bit.’

‘And it all works?’

Lucas gave a short laugh.

‘All of it? I couldn’t say. What I showed you works. There are probably still things we need to check.’

Thomas, who worked on the same team, spoke for the first time.

‘And what’s the code behind it like?’

‘You can have a look. I’ve written quite a few tests.’

‘I will,’ Thomas replied.

He didn’t look especially impressed. Or perhaps he was, but didn’t want to show it.

Hubert Félix Duca had stayed quiet until then.

He had sat through enough demonstrations to know that software which works for ten minutes in a demo is not necessarily ready to be used by hundreds of companies. He knew about the exceptions, the particular demands of large customers, the problems that only surfaced once the software was in their hands.

But he also remembered what he had heard on Wednesday.

Two months.

And here, on his screen, was something that already looked remarkably like what they had imagined.

‘Lucas,’ he asked, ‘remind me how long you spent on this?’

‘About two days.’

Hubert leant a little closer to his screen.

‘Right.’

He couldn't think of anything else to say.
The meeting ended a few minutes later. Everyone went back to their work, but nobody really talked about what they were supposed to do that afternoon.

On Saturday morning, Alice found herself thinking about the demonstration as she made her coffee.
It wasn't just Lucas's speed that unsettled her. It was the fact that he hadn't needed anyone.
She knew her job as a product manager well. Before starting development, she tried to understand the problem, weighed up requests, spoke to customers and worked with developers to find a sensible solution.
For something like the leave-management module, she would probably have arranged several conversations. She would have helped clarify the rules, work through the special cases and decide what needed building first.
Lucas had done none of that.
He had heard a discussion, taken the initiative and built a solution.
Part of her admired him. Another part wondered what would happen if every developer started working this way.
And yet her own working life had changed enormously over the past few months.
At first, Alice had used AI to summarise meeting notes or reword documents. Then she had begun entrusting it with

bigger tasks. She had automated some of the work involved in processing customer feedback, preparing summaries, grouping similar requests and carrying out administrative tasks that had once taken her hours.

These days, she could sometimes finish in minutes something that used to take up much of her morning.

She didn't miss any of those tasks.

Quite the opposite.

She finally had more time to talk to customers, understand the problems beneath their requests and think about what they really ought to build.

AI hadn't made her job less interesting.

It had taken away some of the work that stopped her doing it well.

So why should developers be expected to carry on working exactly as before?

Alice put down her cup.

The question seemed obvious.

The answer much less so.

8

Hubert, too, had spent much of the weekend thinking.

His company had always had more ideas than the capacity to deliver them. Features had sat waiting for months. Improvements requested by customers were repeatedly put off. And there was the plan to expand across Europe, which would require a great deal more work on the software.

He knew the obstacle all too well: they didn't have enough time to do everything.

On Friday, Lucas might have shown that the obstacle wasn't quite as insurmountable as it had seemed.

If a project estimated at two months could yield a first version in two days, how many other projects had suddenly become possible?

Ten developers worked at Malorean Corp. Hubert tried to imagine what would happen if every one of them could move that quickly.

Then he tried to imagine ten developers each launching their own features without any coordination.

That second picture appealed to him rather less.

He had considered calling Alice on Saturday, but stopped himself.

On Sunday evening, he simply sent her a message.

Have you got a moment tomorrow morning? I'd like to talk about Friday.

Her reply arrived a few minutes later.

Yes. Me too.

At nine o'clock on Monday morning, Alice and Hubert met on a video call.

'Well?' Hubert asked.

'Well, what?'

'Have you worked out what happened on Friday?'

Alice smiled.

‘Technically, more or less. Lucas used AI to develop something far faster than we’d have thought possible.’

‘And the rest?’

‘That’s exactly the problem.’

Hubert nodded.

‘I’ve spent the whole weekend thinking about it. We’ve been short of capacity for years. We turn down requests, put projects off and spend half our lives choosing what we’re not going to do. And then Lucas comes along and produces in two days something we’d estimated at two months.’

‘A first version,’ Alice pointed out. ‘We don’t know yet whether it’s ready for production.’

‘Yes, I know. But even if it takes another two weeks to finish, it changes a great deal.’

Alice couldn’t argue with that.

‘What bothers me,’ Hubert went on, ‘is that he did all of this without involving you.’

Alice didn’t answer straight away.

‘That bothered me too,’ she admitted.

‘Do you think we should remind him how we work?’

‘I thought about it.’

‘And?’

‘And I think that would be a mistake.’

Hubert frowned.

‘But if he starts deciding for himself what to develop, we could be in for a few surprises.’

‘Of course. But look at what happened. On Wednesday, we said the leave-management module needed rebuilding. If we’d followed our usual process, I’d have started gathering requirements, we’d have organised meetings, prepared the work, broken it down… We probably would have spent weeks before showing anyone anything.’

‘That isn’t necessarily wasted time.’

‘No. And I’m not saying we should stop understanding what people need. But Lucas has shown us he can make progress without waiting for all that.’

‘So you’re suggesting we let developers decide what to build for themselves?’

‘No, quite the opposite.’

Alice thought for a few seconds.

‘I think we’re mixing up two things: deciding *what* we should do and deciding *how* to do it.’

‘The two are connected.’

‘Yes. But that doesn’t mean I need to be involved in every decision.’

She sat up a little straighter.

‘You know, my job has changed enormously over the past few months. I use AI every day. I’ve automated all sorts of things I used to do by hand. I spend much less time preparing documents or organising information.’

‘I know. You’ve told me about it.’

‘But I don’t think you realise just how much it’s changed my work. These days I have far more time for what really matters: understanding customers, identifying problems and deciding where we should put our energy.’

Hubert listened.

‘Before, a big part of my job was keeping things moving through the organisation. Preparing tickets, clarifying requests, answering questions, coordinating conversations. What if, tomorrow, developers needed me much less for all that?’

‘You’d have more time.’

‘Exactly. And so would they.’

She paused.

‘We accept that AI is transforming my job. So why would we want developers’ jobs to stay exactly the same?’

Hubert leant back in his chair.

‘All right. But somebody still has to decide what we build.’

‘Of course. And, more importantly, *why* we’re building it.’

‘So what do you have in mind?’

Alice hesitated.

She hadn’t prepared a presentation. She had no organisational model to put forward, no new process ready to roll out.

She had only a hunch.

‘I wonder whether we ought to try something.’

Hubert waited for her to go on.

‘We could give Lucas a problem to solve. A real customer problem. And give him much more freedom to work out how.’

‘Without specifications?’

‘Without handing him a ready-made solution. We explain the problem, give him access to the people who understand it and let him investigate.’

‘And you?’

‘I’m still there. I’ll help him understand what customers need, challenge his ideas and steer him away from building something just because he can. But I don’t dictate every step.’

Hubert considered it.

‘So you want him to do some of a product manager’s work?’

‘Maybe. Or maybe we should simply stop dividing every bit of work between us.’

‘And if it doesn’t work?’

‘We’ll find out.’

Hubert smiled.

‘That’s not much of a method.’

‘I know.’

Alice smiled too.

‘But I’d rather try it on something small than spend three months talking about a new organisation.’

Hubert didn’t reply immediately.

Then he seemed to remember something.

‘We’ve actually got a problem that’s been coming up for several weeks with some of our food-manufacturing customers.’

Alice nodded. She knew exactly which one.

Malorean Corp.‘s ERP system handled stock and manufacturing operations, but it couldn’t properly track which batches of raw materials had gone into which finished products.

When a supplier warned a food manufacturer that a batch of raw material was defective, the manufacturer needed to identify quickly all the products made using that batch.

As things stood, Malorean Corp.‘s customers had to piece that information together themselves.

‘I had more feedback about this last week,’ Hubert explained. ‘One customer told me he’d spent an entire day with his quality manager trying to track down the products affected by a supplier alert.’

‘Yes. Support has raised several similar cases with us.’

‘Do you think Lucas could work on it?’

Alice thought about it.

This was trickier than the leave-management module. A few screens wouldn’t be enough. They needed to understand what actually happened in the factories, how raw materials were used and how customers identified their products.

But that was precisely what interested her.

‘I think we could give him the problem, yes.’

‘Do you think he knows enough about the industry?’

‘Probably not.’

Hubert raised his eyebrows.

‘That’s reassuring.’

‘Neither do I, not in every detail. But I know who to talk to. We have customers who understand their constraints inside out. We have people in support and implementation who see the difficulties every day. Lucas can go and speak to them.’

‘You want him to talk directly to customers?’

‘Why not?’

‘That’s usually your job.’

‘Exactly.’

Hubert still didn’t look entirely convinced.

‘And what if he builds something that doesn’t meet the need?’

‘Then it’s my job to help him avoid that. But not by writing him a fifty-page specification.’

‘And if he wants to put his solution into production?’

‘It will have to be checked and approved, just like any other change to the software. Giving him more freedom doesn’t mean giving him permission to take every risk.’

Hubert was silent for a moment.

‘Do you know what worries me?’ he finally asked. ‘It’s not just that he might make a mistake. It’s that we won’t know who’s doing what any more.’

‘That worries me too.’

‘And you still want to try?’

‘Yes.’

She took a moment before adding:

‘Because the alternative is to carry on exactly as before, only asking developers to write code faster. I’m not sure that’s what Lucas showed us on Friday.’

They talked for another twenty minutes or so.

Hubert wanted to avoid disrupting projects already under way. Alice didn’t want the experiment to turn into some grand initiative announced to the whole company before it had even begun.

So they agreed to keep things simple.

Lucas would work on the batch-traceability problem. He would be free to explore solutions and speak directly to the people who could help him understand the need.

Alice would be his product point of contact. She wouldn’t prepare a detailed backlog for him, but she would stay involved in important decisions.

They would start with a limited scope, without promising a delivery date to any customer.

Above all, they would take the time to observe what happened.

‘We should still decide how we’ll know whether the experiment has worked,’ Hubert said.

‘Yes. And I don’t want us to measure only how many features he’s built.’

‘What would you measure, then?’

‘First, whether he can understand the problem and come up with something useful. Then how far he can get without us having to organise everything for him.’

‘And how much of our time it takes.’

‘That too.’

Hubert glanced at the time.

‘Right. Let’s try it.’

Alice smiled, though she didn’t feel entirely reassured.

‘Let’s try it.’

‘Will you speak to him today?’

‘Yes.’

The video call ended.

Alice sat looking at her screen for a moment. She opened her messaging app and found Lucas.

She started to type:

Hi Lucas. Have you got a moment today? I’d like to suggest something a bit different from the way we usually work.

She read the message again before sending it.

For the first time in a long while, she was about to give a developer a problem to solve without first having imagined the solution herself in any detail.

She didn’t yet know whether it was a good idea.

And she was no longer sure it was her job to know before he did.

4 — New Organisational Models Are Emerging

AI makes it possible to get more work done. But how are companies reorganising their teams and decisions in response? Their approaches differ.

Some strengthen Product Management around developers. Others give developers a broader remit, provide more opportunities for non-technical teams to build things, or organise work around small teams supported by agents.

The traditional division of labour is no longer a given.

That traditional model had its logic. Product Management held much of the customer and business context; Design turned the intended experience into something tangible; and Engineering turned those intentions into software. Tickets, specifications, handovers and meetings helped these disciplines work together around limited development capacity.

That way of working has not suddenly become wrong.

But as implementation capacity grows, some companies are beginning to redistribute people, context and work in different ways.

The task, then, is not yet to find ‘the right model’.

It is to examine the main approaches now emerging.

Strengthening Product Management around developers

The most intuitive response is to preserve a fairly conventional separation between Product and Engineering while increasing the product-management capacity available to developers.

Gamma offers an interesting example. The company reports a ratio of roughly one Product Manager to four engineers, compared with a commonly cited historical ratio closer to one in ten. [1]

In this model, greater engineering capacity is matched by more Product Managers.

The principle is straightforward: when more can be built, more effort goes into understanding, prioritising and directing that work.

Giving developers more ownership of the problem

Another strategy is to broaden the engineer's remit.

Developers no longer necessarily wait until a problem has been fully translated into a solution and then into tickets. They may take a greater part in understanding the need, prototyping, making local trade-offs and tracking outcomes.

In its job advertisement for a Product Engineer, Linear sets out the responsibilities it expects: owning a feature from initial idea to shipped product and helping to define the problem without constantly relying on Product Management. [2]

This model aims to reduce some handovers by bringing more context and more ability to act together in the same person.
It does not mean that every developer becomes a Product Manager.
It means broadening the engineering role on certain kinds of work.

Letting disciplines cross their old boundaries

This shift is not limited to developers.
In a study of changing roles, Figma reports that 64% of respondents identify with at least two roles and that more than a third say their responsibilities span several disciplines. This cannot be attributed solely to AI, but it does describe less rigid boundaries between roles. [3]
Atlassian also documents workflows in which designers work directly in production code and open a pull request themselves before an engineering review. [4]
Cursor describes an organisation in which Product, Design and Engineering are all more closely involved in building things. [5]
The point is less that ‘everyone does everything’ and more that **more people work beyond the boundaries of their traditional roles**.
That raises a separate question: when several disciplines can act in the same area, how can responsibilities remain clear?
We will return to it.

Giving small, accountable teams a wider remit

Another group of approaches concentrates a broader scope of work in fewer hands.

AMPECO describes ‘alignment syncs’ in which an engineer and a Product Owner agree on the intention before agents take on much of the analysis, development and testing. [6]

Taito describes a process in which a team approves a plan and then one or two people oversee the build with the help of agents. [7]

In these models, a small group of people can set far more work in motion than before.

The organisational change is therefore a possible reduction in handovers and a concentration of ownership in one person or a small group.

What remains to be defined is what those people can decide on their own and when they need to involve someone accountable for a particular domain.

Automating some of the checks

Faster delivery also puts pressure on the ways changes are approved.

Some companies respond by automating more checks or varying the review process according to the risk involved.

Meta uses a Diff Risk Score to estimate the likelihood that a change will cause an incident. [8]

Cloudflare formalises engineering standards that can be checked automatically before a merge. [9]

Taito also combines deterministic rules, review agents and risk-based triggers. [7]

For now, the important observation is simply that **not every change needs to follow the same validation process**.

We will examine how automation, human accountability and escalation thresholds can work together later in the book.

Providing context rather than prescribing every step

Other companies go further, questioning whether a detailed ticket should remain the main way to coordinate work.

Linear explicitly criticises the model in which a Product Manager scopes a problem, writes it up in a system and waits for Engineering to carry it out. It stresses the need to make customer feedback, ideas, strategic direction, decisions and code accessible to both people and agents. [10]

Cursor likewise emphasises setting sufficiently clear goals to enable greater local initiative. [5]

What we are seeing is not the disappearance of coordination. Rather, part of that coordination is shifting away from detailed instructions towards **richer context that is more directly accessible**.

A dedicated chapter will consider what that context should contain, how to maintain it and how to define the limits of autonomy.

Letting the person who knows the need build the solution

Some companies even allow teams that understand a particular need to create small tools themselves, without always joining a developers' queue.

Shopify built Quick, an internal platform that enables more employees to publish websites and small tools quickly. [11]

Vercel describes a model in which non-technical employees build internal tools in a secure environment without distracting developers from priorities on the main product. [12]

Atlassian also shows designers working in code before a technical review. [4]

In these examples, AI does not merely help developers work faster.

Sometimes it shifts the actual implementation to **the person who already understands the need**.

An idea older than AI

As early as 2018, Marty Cagan was advocating what he called *empowered product teams*: multidisciplinary teams given problems to solve by their leaders, rather than lists of features to deliver. He argued that they should bring Product, Design and Engineering skills together to find solutions, and be accountable for outcomes rather than output volume. [13]

He went on to develop this approach with Chris Jones in the 2020 book *EMPOWERED*. The model did not imply interchangeable professions or the end of management. On

the contrary, it required a shared vision, strong skills and leaders who helped teams develop. [14]

So the idea predates AI agents. What AI may change is how quickly and directly an empowered team can implement some of the solutions it chooses to explore. **Product autonomy is an older ambition; AI changes the tools and constraints surrounding it.**

Choices that have not yet converged

These responses are not stages in a single transformation. Strengthening Product Management and giving developers more product responsibilities may even appear contradictory. Yet either can work in different companies, and they may coexist within one organisation.

They share something else: **when building things becomes more accessible, the way a company distributes context, decisions and responsibility becomes a choice again.**

We still need to see what that shift means in the everyday lives of those involved, beyond announcements and organisation charts.

We will return to Malorean Corp. shortly to see how these models affect a small French company. Then we will examine the responsibilities and rules that autonomy requires.

Bibliography

[1] The bottleneck keeps shifting: What AI is changing about how we build — Atlassian / Gamma

- URL: https://www.atlassian.com/blog/how-we-build/the-bottleneck-keeps-shifting-what-ai-is-changing-about-how-we-build
- Describes how Gamma increased the density of Product Managers as engineering capacity grew.

[2] Product Engineer — Linear Careers

- URL: https://linear.app/careers/0c7c2e26-0a98-42cf-a47c-9a3999fb513b
- Job advertisement describing a Product Engineer's responsibilities, from defining the problem to shipping features.

[3] Are Roles and Responsibilities a Thing of the Past? — Figma

- URL: https://www.figma.com/blog/2025-shifting-roles-report/
- Documents growing overlap between roles and responsibilities spanning traditional professional boundaries.

[4] Designers' workflow for shipping code — Atlassian

- URL: https://www.atlassian.com/blog/ai-at-work/designers-workflow-for-shipping-code
- Describes designers changing real product code before an engineering review.

[5] How Cursor Is Navigating the New Shape of Product Craft — Atlassian / Cursor

- URL: https://www.atlassian.com/blog/how-we-build/how-cursor-is-navigating-the-new-shape-of-product-craft
- Describes Product, Design and Engineering working more closely on implementation.

[6] How we built an AI-native engineering system — AMPECO

- URL: https://www.ampeco.com/blog/how-we-built-an-ai-native-engineering-system/
- Describes a model in which a small number of people agree the intent before agents carry out much of the implementation.

[7] Inside an AI-native engineering organization — Swarmia / Taito.ai

- URL: https://www.swarmia.com/blog/inside-an-ai-native-engineering-organization/
- Describes a small team overseeing agents, with much of the work coordinated by one or two people.

[8] Diff Risk Score: AI-driven risk-aware software development — Meta

- URL: https://engineering.fb.com/2025/08/06/developer-tools/diff-risk-score-drs-ai-risk-aware-software-development-meta/
- Introduces a risk score used to vary validation processes according to how likely changes are to cause incidents.

[9] How Cloudflare enforces engineering standards using AI — Cloudflare

- URL: https://blog.cloudflare.com/engineering-standards-enforcement/
- Describes how explicit engineering standards become automated checks in a development workflow.

[10] Issue tracking is dead — Linear

- URL: https://linear.app/next
- Argues for sharing more context across customer feedback, strategy, decisions and code, beyond isolated tickets.

[11] Quick: An internal hosting platform for the AI era — Shopify

- URL: https://shopify.engineering/quick
- Describes an internal platform enabling more employees to build and publish tools directly.

[12] Stopping the slow death of internal tools — Vercel

- URL: https://vercel.com/blog/stopping-the-slow-death-of-internal-tools
- Explains how business users can build internal tools without diverting engineers from core product work.

[13] Empowered Product Teams — Marty Cagan, Silicon Valley Product Group (2018)

- URL: https://www.svpg.com/empowered-product-teams/
- Foundational account of teams responsible for solving problems rather than simply delivering features.

[14] EMPOWERED: Ordinary People, Extraordinary Products — Marty Cagan and Chris Jones (2020)

- URL: https://www.svpg.com/books/empowered-ordinary-people-extraordinary-products/
- Overview of the book on empowered teams, shared strategic context and leaders' responsibility for developing their teams.

5 — Malorean Corp. — What We Can Do, and What We Should Do

On Tuesday morning, Lucas began by asking Alice a question she hadn't expected.

'Can I speak to the customer?'

The day before, he had received the few reports Support had kept about the batch-traceability problem. They mentioned manufacturing numbers, raw materials, Excel spreadsheets and supplier alerts. He had spent the evening organising the information with AI.

'Of course,' Alice replied. 'Why?'

'Because I don't quite understand what they're missing.'

Alice smiled. A week earlier, she would probably have handled the meeting herself before passing her conclusions on to Development. But she had now promised to let Lucas take ownership.

'I'll introduce you to the person in Support who looks after that account. They can arrange a meeting.'

On Wednesday, Lucas discovered a situation far more complicated than the phrase 'manage batches' suggested. The customer's quality manager showed them a letter from a supplier. A raw material had to be withdrawn. The supplier knew its batch number. The manufacturer knew which deliveries it had received. But to identify exactly which

finished products were affected, its teams had to cross-reference several documents.

‘Last time,’ the quality manager explained, ‘there were four of us checking files for most of the day. Even then, we had to go back onto the factory floor to make sure.’

Lucas asked to see how the information was recorded during production. The customer showed him a screen in the ERP, then a spreadsheet.

‘But why do you also record that number in Excel?’

‘Because your software can’t tell us what we need to know.’

The answer wasn’t aggressive. That made it no easier to hear. By the end of the call, Lucas had three pages of notes and fewer certainties than when he started.

‘Well?’ Alice asked when he called her back.

‘I’d started imagining a screen where they could record batch numbers.’

‘And now?’

‘Now I think we first need to understand how they record what they’ve actually used. Otherwise we’ll have a lovely screen and still won’t be able to answer their question.’

Alice nodded. She could have told him that on Monday. But she wasn’t disappointed that he’d worked it out for himself.

The following Friday, Lucas showed her a first version running on a demo database.

You could link a batch of raw materials to a manufacturing run, then find the finished products made from it. He had added a small example to trace a batch through to products affected by an alert.

Alice clicked around.

‘You did this during the week?’

‘Yes. There’s still a lot that’s wrong or missing, but at least we’ve got something to show the customer.’

‘And what if we added the ability to trace things in the other direction too?’

‘It’s already there.’

He opened another view.

Alice burst out laughing.

‘Of course it is.’

Since the leave-management demonstration, her sense of what projects might involve had shifted. Things she would once have put off until the following quarter now seemed within reach.

‘We could search by supplier as well. And put alerts directly on the affected orders. And why not prepare recall documents?’

Lucas didn’t answer immediately.

‘We can write those ideas down,’ he eventually said. ‘But I’d like to go back to the customer with what we’ve got first.’

‘Surely we’re not going to wait a fortnight before showing them something more complete?’

‘It’s not the demo I’m worried about.’

‘What are you worried about?’

This time, Lucas shared a rather less appealing screen. Old database tables, fields whose names meant very little, several connections between manufacturing and stock.

‘We can build as many screens as we like. The problem is that our ERP was never designed to preserve the full history of these batches. There are stock movements going back years, some corrected by hand. Some of the information simply doesn’t exist.’

‘Can’t AI help us sort all that out?’

‘It can help me write a migration. It can’t invent information we never recorded.’

Alice fell silent.

‘If we want to do this properly,’ Lucas continued, ‘we’ll need to change part of the data model and check what that affects in other modules. We can’t afford to corrupt a customer’s stock records just to save two weeks.’

‘I’m not asking you to break the software.’

‘I know. But ever since Friday, every time I show you something, you ask for three more things.’

He hadn’t said it aggressively. All the same, it stung.

‘That’s true,’ Alice admitted.

‘I want to move fast too. You know that. But just because I can generate code in an hour doesn’t mean fifteen years of ERP behaviour suddenly becomes simple.’

After a moment’s hesitation, he added:

‘Thomas looked at some of the leave-management code. He found two cases we’d handled badly. He was right. We’ve fixed them, but it reminded me of something: we can build much faster and still need somebody who knows where the traps are.’

Alice remembered Thomas asking to see the code on the day of the demonstration.

‘So what do you suggest?’

‘We carry on with the demo using fictional data. We check that we’ve understood what the customer needs. But before we touch real stock records, I want us to take the time to understand the existing system and run the checks we need.’

Alice looked at the first version, still open on her other screen. The previous week, she had defended Lucas’s autonomy to Hubert. She was suddenly discovering that autonomy included the right to tell her no, and she didn’t much like it.

‘All right,’ she said at last. ‘We won’t touch real data for now.’

The following Monday, Hubert opened their weekly meeting with what was becoming a familiar question.

‘So, how’s our experiment going?’

‘It’s moving forward,’ Alice replied. ‘Faster than expected in some areas, slower in others.’

‘Meaning?’

Alice showed him Lucas’s demo, then summarised Friday’s discussion.

Hubert paid particular attention to the technical reservations.
‘He’s right to be careful,’ he said. ‘I don’t want us to solve a traceability problem by creating stock problems for every customer.’
‘Yes. And I think I pushed him a bit.’
‘You?’
‘For the past two weeks, I’ve felt as though anything is possible. As soon as an idea comes up, I think we can try it straight away.’
Hubert smiled.
‘That’s more or less how I felt after his first demo.’
‘The trouble is, it’s easy to confuse how quickly we can get something onto a screen with how quickly we can make something reliable.’
‘At least Lucas knows when to slow down.’
‘Yes. And I need to trust him on that.’

Two days later, Julie Morel asked Alice whether she had five minutes.
Alice knew that phrase. With Julie, five minutes could easily become half an hour.
‘I met a fairly important prospect last week,’ Julie began. ‘They’ve got several factories and they’re looking for an ERP that can adapt to the way they work. Look what I’ve prepared.’
She shared a series of polished mock-ups. They showed different approval stages depending on the type of order, rules

specific to each site, and screens that changed according to the user.

‘You made these?’ Alice asked.

‘With ChatGPT. I worked from my meeting notes, asked it to suggest a user journey, then we refined it together. Took me an evening.’

Alice immediately saw how useful the images could be. In a few hours, Julie had given concrete form to a need that had previously existed only in sales meeting notes.

‘Has the customer seen the mock-ups?’

‘Not yet. I wanted to show you first. But he told me pretty clearly that if we could do this sort of thing, it could change the conversation.’

Alice scrolled through the screens.

‘It’s an interesting idea. You should speak directly to Lucas.’

Julie looked surprised.

‘Directly? I thought you’d prepare the work.’

‘Why? You know the customer, you’ve spoken to him and you’ve even made the mock-ups. You can explain what he’s looking for better than I can.’

‘And Lucas can see whether it’s feasible?’

‘Yes. But be careful, Julie: checking whether something’s feasible doesn’t mean we’re committing to build it.’

‘Of course.’

Alice knew that ‘of course’. It wasn’t a lie. It was a salesperson’s enthusiasm at the prospect of a sale.

The following day, they met with Lucas. Julie took them through her mock-ups, explained the prospect's requests and answered their questions.

Lucas was immediately interested in the differences between sites.

'We've already got some of that mechanism in the ERP,' he said. 'Not to do exactly this, but it exists. We could probably put something together to test the idea.'

'How long would it take?' Julie asked.

'For a demonstration? Not necessarily very long.'

Julie looked at Alice. She was smiling almost as though the contract were already won.

'If we show him this next week, it could really make a difference.'

Alice felt her own eagerness to move forward trying to answer before she had thought it through. She considered the traceability project, the waiting requests, this prospect's unusual rules and all the other customers who might have no use for them.

'No,' she said.

The other two turned towards her.

'No to what?' Julie asked.

'No to starting development on this feature in our product just to win that contract.'

'But you've just said it's interesting.'

‘It is. But we don’t know whether this customer needs something that ought to be available to all our customers, or a very specific way of working for his factories.’

‘We might lose the sale.’

‘Perhaps.’

Julie clearly hadn’t expected that answer.

‘We finally have the means to respond quickly to requests like this,’ she insisted. ‘It would be a shame not to use them.’

‘I don’t want to give that up. I want us to understand what we’re offering. If we put every special request into the heart of the ERP, we’ll end up with software that’s impossible to develop further.’

‘But Lucas says he can do it.’

‘Lucas says he can prepare a demo. That’s not the same thing.’

Silence followed. Julie looked at her mock-ups one last time.

‘So what do I tell him?’

‘That we understand his problem and want to show him an approach. Not that we’ve decided to add the feature to the product.’

Julie nodded. She looked disappointed, but not discouraged.

‘All right. I can arrange a meeting so he can explain the differences between his factories in more detail.’

‘Perfect. And keep the mock-ups. They’ll help us ask the right questions.’

When Julie left the call, Alice stayed with Lucas for a few seconds.

‘Could you really have built that quickly?’

‘A first version, yes.’

‘And would you have wanted to?’

‘Frankly? Yes. It looked fun.’

Alice laughed.

‘I wanted to say yes too.’

‘What stopped you?’

‘The fact that we were both talking about what we *could* do, and barely talking about what we *should* do.’

The following week, Lucas sent Alice a link and a message.

I threw something together yesterday. Can you have a look?

Alice opened the page. She recognised their ERP interface, but there was a small panel on the side for changing certain rules. She could select a site, alter an approval threshold, show or hide a step, then run the demonstration again.

She called Lucas back.

‘What’s this?’

‘A little tool for preparing demos with different rules. I reused our old configuration mechanism. It’s only on a test instance.’

‘How long did it take?’

‘Two hours, maybe. I thought of it after looking at Julie’s mock-ups.’

‘But that’s not even what you’re meant to be working on!’

‘No. I wanted to see if we could avoid changing the code every time we wanted to test a variation.’

Alice began adjusting the settings. The screen responded. She tried another configuration, then a third.

‘Hang on. You mean I can prepare several different versions of a demo myself?’

‘For the rules we’ve made configurable, yes. Not just anything. And definitely not in customer environments.’

‘But I could show Julie two different journeys and ask which one actually matches her prospect’s needs.’

‘Exactly.’

Alice sat looking at the page for a moment. The first leave-management module had made Lucas faster. This tool gave *her* a possibility she hadn’t had the day before.

She could try out an idea without asking a developer to implement it especially for her.

‘Can you imagine what we could do with this?’

Lucas smiled.

‘I think I can. But before you ask me to turn it into a full product, remember it’s something I threw together in two hours.’

‘Promise.’

She ended the call, then went back to playing with the settings.

The following Monday, Hubert noticed that Alice hadn't opened her usual progress document.

'Haven't you prepared anything today?'

'I have. But there's something I'd like to show you.'

She presented Lucas's tool and created two variations of the same demo in front of him.

Hubert understood quickly enough.

'You didn't ask the developers to make those changes?'

'No. Lucas built a little tool. I can test certain rules on my own.'

'Could we let Sales use it?'

'Perhaps later. For now, it's not robust enough. But Julie can already work with me on her prospect's use cases.'

Hubert studied the screen with interest.

'And the batch project?'

'Lucas is making progress. He went back to the customer with his first version. We're beginning to understand what we need to do. But he refuses to touch the historical data until we've made sure we can handle it safely.'

'Sounds sensible.'

'Yes. On Friday it annoyed me. Today I'm glad he insisted.'

Alice twirled her pen between her fingers.

'Something strange is happening, Hubert. Lucas is talking to customers and making decisions about the solution. Julie is turning up with mock-ups that are almost usable. And I can

now test some variations of our ERP without needing a developer to build them.'

'Does that worry you?'

'Sometimes. I wonder what happens to our jobs when everyone starts doing part of someone else's work.'

'Have you found an answer?'

'Not really.'

She thought for a moment.

'I just know I don't want to be the person every idea has to pass through. There was no reason Julie should wait for me to turn her mock-ups into tickets. Lucas doesn't need me to explain how to build every screen.'

'But you said no to developing the prospect's request.'

'Yes. Because that was a decision about our product. I might have been wrong, and I can still change my mind, but it was my decision to own.'

'And Lucas refused to take a shortcut with the stock records.'

'Because he knows what we might break. On that, his judgement is something I have to listen to. Even when I'm in a hurry.'

Hubert looked at the small window containing Alice for a moment.

'So we can let people do more things, but we can't just let all the decisions float around between them.'

'That's what I'm starting to think.'

She hesitated, then added:

‘Before, I sometimes felt my work was important because so many things came through me. Now I’m wondering whether what really matters is what comes out the other end. Are we solving the right problem? Are we building something reliable? Is it worth doing at all?’

Hubert nodded.

‘And if nobody knows who’s supposed to answer those questions, we’ll have a problem.’

‘Yes.’

They were quiet for a few seconds. Neither of them wanted to turn the conversation into an announcement about a new organisation.

Their experiment had only just begun. It had opened up more possibilities, but it had also given them more decisions to make.

Alice received a notification. A message from Julie: the prospect had agreed to review the demo variations. He also wanted to know when they could be made available to his teams.

Alice didn’t reply immediately.

It was now possible to produce answers much faster.

But someone still had to decide which ones Malorean Corp. was prepared to promise — and stand behind.

6 — Blurring Roles, Enduring Responsibilities

For a long time, the boundaries between Product, Design and Engineering provided simple points of reference.
Product understood the problem and decided what to build. Design shaped the experience. Engineering turned that intention into software. In reality, the division was never quite so neat, but it still shaped teams, recruitment, processes and responsibilities.
That division was being questioned even before AI arrived. The empowered product teams described in Chapter 1.4 already brought Product, Design and Engineering together around a shared problem. AI is now making it easier to cross another boundary: that of implementation, as people can take on tasks once reserved for another profession.
Our survey of organisational models illustrated this, notably through Figma's study. [1] On Malorean, I have also described developers moving towards understanding problems while Product Managers increasingly take part in building solutions. [2]
It would be tempting to conclude that roles are going to disappear. I believe the change instead calls for us to rethink each person's responsibility: **the more roles overlap, the more explicit responsibilities must become.**

The ability to act no longer belongs to one profession

The first thing AI redistributes is not necessarily decision-making authority. It is the ability to act.

Consider designers modifying code themselves, a practice documented by Atlassian: an engineering team still checks technical quality and the conditions for releasing the change. [3] That is not a retreat to the old division of labour. It shows that being able to change code and being accountable for its reliability are different things.

The same distinction applies to Linear's Product Engineer vacancy: the company expects someone in that role to take ownership of a feature from the initial problem to the shipped result. [4] That does not automatically give them authority over every product-strategy decision.

We can therefore reduce handovers without discarding the expertise that justified some of them. The question is no longer how many professions are involved, but **who is accountable for what when several professions work in the same area**.

Being able to contribute does not mean owning responsibility for a domain

8

We need to distinguish two things that organisations often confuse: permission to act and accountability.

Someone in Sales can build an internal tool. That does not make them responsible for the platform's security.

A Product Manager can change an interface. That does not mean they become responsible for the consistency of the Design System.

A developer can decide on a small, local aspect of product behaviour. That does not automatically make them responsible for the overall strategy, positioning or roadmap.

Many teams already recognise these distinctions, but AI forces them to make them explicit.

In a highly specialised organisation, the profession and the responsibility often overlapped. The person who could do something was also the person whose domain it was. As the ability to act spreads, that overlap breaks down.

The danger is confusing autonomy with blurred accountability.

I have made a similar argument about quality: developers cannot ship software and then assume its defects belong to a QA team, a bug-fixing rota or the next developer who happens to be free. Technical quality must remain someone's identifiable responsibility. [5]

Blurring the boundaries between roles does not change that principle. It makes it more important.

The concept of the Guarantor

To address this problem, I use the concept of a **Guarantor**.

For every important domain, a person or team remains accountable for its coherence and quality.

The Guarantor is not necessarily the person who does all the work in that domain.

Nor must they approve every minor decision.

They are the person the organisation turns to when an important choice affects their domain, a boundary is crossed or something goes wrong.

In a simple organisation, we might establish at least three explicit areas of accountability:

- **Product Guarantor**: product coherence, major product choices, target customers, value proposition, important trade-offs and alignment with strategy.
- **Design Guarantor**: Design System, consistency of the user experience, accessibility, new patterns and UX standards.
- **Engineering Guarantor**: architecture, security, performance, maintainability, technical quality and release conditions.

The principle is straightforward:

Anyone can contribute across domains, but each domain retains someone who is accountable for it.

This allows roles to overlap without responsibilities disappearing.

Sales can build things without becoming accountable for their technical quality

Take a deliberately simple example.

A salesperson spots a need, uses an AI-assisted tool and builds a small application or makes a useful change. Provided the company has a sufficiently secure environment, there is no

reason to assume that only Engineering should be allowed to build things.

But that freedom does not make the salesperson accountable for technical quality.

If the change affects security, permissions, sensitive data or architecture, the Engineering Guarantor must be able to request a review, insist on a correction or block it.

The salesperson remains responsible for the need they identified.

Engineering remains accountable for what the company agrees to maintain and operate.

The same logic applies to Design.

A Product Manager might assemble a screen from existing components. As long as they stay within the Design System and use established patterns, requiring formal design approval every time would needlessly restrict autonomy.

But if that Product Manager goes beyond the Design System, invents a new pattern or creates a complex, fundamental interaction, the Design Guarantor must be able to step in.

The boundary is therefore no longer: ‘Who is allowed to touch what?’

It becomes: **‘At what level of risk or novelty must the person accountable for the domain become involved?’**

The owner of a problem is not the same as the Guarantor of a domain

This is probably one of the most important distinctions in emerging organisations.

The **owner** is accountable for a particular piece of work.

They need to understand the problem, move the work forward, make local trade-offs, coordinate when necessary, check the result and ensure the work reaches its intended outcome.

The **Guarantor** carries a broader responsibility across different pieces of work.

They do not necessarily own the individual initiative. They safeguard a level of consistency, quality or acceptable risk within a domain.

A Product Engineer can therefore own a feature without becoming responsible for the company's entire architecture.

A Product Manager can own a launch without becoming accountable for all of Design.

A salesperson can own an internal business tool without becoming responsible for the security of every internal application.

The distinction makes accountability for the outcome clearer, without expecting one person to be an expert in every field.

AMPECO illustrates a version of this arrangement: a Product Owner and an engineer align on the 'what' and the 'why', then agents undertake much of the analysis, development and

testing. Execution is spread more widely, but a human remains accountable for final validation. [6]

AI may therefore reduce the number of people needed to carry out a piece of work while making it all the more important to know **who is accountable for it**.

Job titles may no longer be the best way to understand an organisation

This also raises a broader question: should companies still organise themselves primarily around job titles?

The debate about the Product Owner is revealing. On Malorean, I have argued that Product Owner is first and foremost a function — a mandate to make decisions and take responsibility — rather than a profession whose purpose is maintaining a backlog. [7]

That distinction becomes particularly useful with AI.

Define a role by the tasks its holder performs and AI makes the definition unstable. Many tasks become accessible to more people.

Define a responsibility by what a person or team must answer for, and it proves far more resilient.

A Product Manager may write fewer specifications without ceasing to be accountable for good product choices.

A developer may write less code by hand without ceasing to be accountable for technical quality.

A designer may no longer be the only person adjusting pixels without ceasing to safeguard a consistent user experience.

At Cursor, where boundaries between Product, Design and Development have also become more fluid, the head of Product emphasises that the essential task remains choosing what is worth building and providing enough direction to stay on course. [8]

A profession changes not so much because it loses all its responsibilities, but because it loses its monopoly over certain ways of carrying out work.

The Guarantor must not become another layer of bureaucracy

There is, however, an obvious trap.

If every change a Product Manager makes to code needs engineering approval, every local decision by a developer must go back to Product, and every screen created by anyone other than a designer has to wait for a design committee, we have solved nothing.

We have simply recreated the old bottlenecks under another name.

The Guarantor must not turn into a universal approval process. Their role exists precisely to enable greater autonomy. People can act within known boundaries without constantly asking permission, because the organisation knows both where accountability lies and when the relevant people need to be involved again.

In real life, of course, this model never works perfectly.

Changes reach production that the Guarantor would not have approved in quite that form. Local decisions go unnoticed. Teams sometimes take more freedom than expected. Conversely, some people escalate too early out of caution.

That does not necessarily mean the model has failed.

An organisation that genuinely grants autonomy accepts that some decisions will be made without prior inspection. Otherwise, it is not autonomy.

The goal is not to ensure that no deviation ever happens.

It is to define **which deviations are acceptable, which the system must detect, and which require a Guarantor to step in before they become costly**.

This brings us to a different question.

We have distinguished two kinds of responsibility: that of the person who owns an initiative and that of the Guarantor of a domain.

One question remains:

When should an owner be free to decide, and when should they bring in the Guarantor?

At Malorean Corp., this boundary between initiative and accountability is starting to cause friction. We will see it in the next episode, before examining safeguards and escalation rules.

Bibliography

[1] Are Roles and Responsibilities a Thing of the Past? — Figma

- URL: https://www.figma.com/blog/2025-shifting-roles-report/
- Examines broader responsibilities and increasingly fluid professional boundaries in modern product teams.

[2] PM, développeurs et IA : les rôles se brouillent, les responsabilités restent — Malorean

- URL: https://malorean.net/articles/2026-08-01-pm-developpeurs-et-ia-les-roles-se-brouillent-les-responsabilites-restent.en.html
- The author's account of AI broadening who can act without eliminating Product, Design or Engineering accountability.

[3] Designers' workflow for shipping code — Atlassian

- URL: https://www.atlassian.com/blog/ai-at-work/designers-workflow-for-shipping-code
- Describes designers changing product code directly while Engineering continues to review quality, performance and release readiness.

[4] Product Engineer — Linear Careers

- URL: https://linear.app/careers/0c7c2e26-0a98-42cf-a47c-9a3999fb513b
- Job description covering a Product Engineer's responsibility from defining the problem to delivering features.

[5] La qualité appartient à ceux qui livrent — Malorean

- URL: https://malorean.net/articles/2026-06-03-la-qualite-appartient-a-ceux-qui-livrent.en.html
- Argues that teams delivering software retain technical responsibility rather than handing defects to a repair team.

[6] How we built an AI-native engineering system — AMPECO

- URL: https://www.ampeco.com/blog/how-we-built-an-ai-native-engineering-system/
- Describes agents taking on much of the work while intent, architecture and validation remain human responsibilities.

[7] Le PO n'est pas un métier, c'est une fonction — et l'IA n'y change rien, elle accélère — Malorean

- URL: https://malorean.net/articles/2026-08-03-le-po-nest-pas-un-metier-cest-une-fonction.en.html
- Distinguishes a decision-making and ownership mandate from a job defined mainly by backlog administration.

[8] How Cursor Is Navigating the New Shape of Product Craft — Atlassian / Cursor

- URL: https://www.atlassian.com/blog/how-we-build/how-cursor-is-navigating-the-new-shape-of-product-craft
- Describes broader role boundaries while Product judgement, goals and safeguards remain central.

7 — Malorean Corp. — What Alice Knew but Had Never Written Down

On Thursday morning, Lucas called Alice for the third time that week.

‘Two minutes, I promise.’

Alice had learnt to be wary of that phrase. Lucas never asked her to attend long meetings. He would arrive with a precise question, they would talk it through and then he would go back to work. That was exactly what made the interruptions so difficult to refuse.

‘Go on.’

‘On batch traceability, the customer wants to search by supplier. I can add that to the demo. But if a supplier has delivered several batches with the same internal number, should we show everything together or ask the user to choose?’

Alice thought for a few seconds.

‘We mustn’t give the impression we’ve identified a batch when several match. Show the results separately. We can work out with the customer how they distinguish between them.’

‘That’s what I thought. But I wasn’t sure.’

‘Then why are you calling me?’

Lucas gave an embarrassed laugh.

‘After what happened with Julie’s mock-ups, I’d rather check before making something that looks like a product decision.’

The previous week, Alice had asked him not to turn a prospect’s request into a standard feature. Now she regretted that he had taken away the lesson that he should ask her opinion whenever he was unsure.

‘All right. But you didn’t need me for that one.’

‘It’s reassuring to hear you say it.’

They hung up. The call had lasted seven minutes. Alice returned to the document she was writing, read her last sentence and could no longer remember what she had wanted to add.

Ten minutes later, Lucas sent a message.

Another question: for older manufacturing runs where no batch information exists, do we show “unknown” or hide the column?

She started typing a reply. Then she stopped.

The question was different, but the underlying principle was the same: never present information as certain when the ERP didn’t have it. She could have given him that rule instead of deciding two separate cases.

Even so, she replied.

Make it clear when the information is missing. Don’t hide the problem.

Lucas responded with a thumbs-up.

Alice looked at her diary. A meeting with Support had been moved. She had a customer call that afternoon. Between the two, she needed to review Julie's proposals for her multi-site prospect.

She liked the fact that Lucas spoke to her directly. That was precisely what she had wanted when she gave him the project. But he moved so quickly that every new possibility created a new question. And the more he consulted her, the less time she had for the things she was best placed to do: understand customers and think about the product's direction.

He was losing time as well. Sometimes he waited for her reply before carrying on.

An unpleasant thought crossed her mind: if this continued, perhaps they would end up putting every decision back into tickets. It would be easier to manage. She would write the specifications, Lucas would implement them and nobody would have to wonder where his autonomy ended.

Exactly the arrangement they had been trying to move beyond.

At lunchtime, Alice met Bob on a video call. They had got into the habit of speaking without an agenda, simply to compare what was happening in their work.

'How are things going with Lucas?' Bob asked.

'Too well, in a way.'

'That doesn't sound like a compliment.'

‘He’s moving incredibly fast, noticing all sorts of things, talking to customers. But he calls me constantly about decisions he could make on his own.’

‘And do you answer him?’

‘Of course.’

‘Then he’s right to call you.’

Alice rolled her eyes.

‘Thanks, Bob.’

‘I’m only half joking. If I can get your answer in five minutes instead of risking making the wrong decision, why wouldn’t I?’

‘Because we can’t possibly carry on working like this.’

Bob asked for an example. She told him about the two questions that morning, then the conversation with Julie, then a recent uncertainty Lucas had raised: should they add a setting requested by a customer if it would probably only be useful to that one customer?

‘Do you see?’ Alice said. ‘They’re always variations on the same question.’

‘And what’s your answer?’

‘It depends.’

Bob smiled.

‘There you go. That’s exactly the sort of answer that makes people call you again.’

Alice was about to object. But he was right.

‘It’s not just “it depends” at random,’ she went on. ‘I look at whether it’s a recurring problem, whether it helps our priority

customers, whether it'll make the product more complicated for everyone, whether we'll be able to maintain it…'

She stopped.

'See?' Bob said. 'You've just given me part of the rule.'

'It's not really a rule. More a way of thinking.'

'Even better. Perhaps that's what you should pass on to Lucas.'

Alice fell silent.

For years she had been writing tickets. She described expected behaviours, special cases and acceptance criteria. But she almost never wrote down the reasoning that helped her choose between several possible behaviours. That reasoning stayed in her head, or disappeared into a meeting once the work was delivered.

'I must have hundreds of things like that,' she said.

'Probably. Me too.'

'And we never had a reason to write them down.'

'We had reasons. We didn't have the time. And nobody ever opened a ticket called "Explain why we sometimes turn down a good idea".'

They laughed.

After the call, Alice opened a new document. She typed: *Product decision-making principles*.

She added a first sentence, deleted it and tried another. Fifteen minutes later, she had a paragraph that sounded like an internal policy. It was perfectly reasonable. It sounded nothing like what she had explained to Bob.

She closed the document.

Then she opened ChatGPT.

‘Help me get what I know about our product out of my head,’ she dictated. ‘I don’t want a theoretical procedure. Ask me questions, help me remember examples and distinguish genuine rules from things that are simply my habits.’

The first response suggested starting with recent decisions.

Alice talked.

She talked about Julie’s prospect, the appealing mock-ups and the risk of turning one customer’s unusual need into a promise to all customers. She explained that an important account might be worth a lot of money without automatically setting the right direction for the product.

‘Hang on, I need to explain why,’ she added aloud. ‘Our ERP serves very different businesses. If we add every special rule to the heart of the software, we make life harder for everyone. But sometimes a specific request reveals a much wider problem. We can’t just say no to everything either.’

ChatGPT asked how she distinguished one case from the other.

Alice described conversations with Support, situations seen across several customers, requests from Sales and implementation constraints. She recalled old discussions of which no notes remained. She explained why some options had been abandoned, why a configuration setting had been

preferred to a new feature, and why a payroll exception could not be treated like a preference about the interface.

At times, she caught herself answering a question she had never thought to ask explicitly.

The AI rephrased, grouped ideas and suggested a principle. Alice corrected it.

‘No, it’s not “we never develop anything customer-specific”. We can under certain conditions, but we have to make a conscious decision about the cost. Above all, we mustn’t treat a commercial assumption as a product decision.’

A little later, ChatGPT wrote that a traceability feature could guarantee the reconstruction of every historical batch.

‘Absolutely not,’ Alice said to her screen. ‘If the data was never recorded, we can’t magically reconstruct it. That’s a limitation to make clear, not a capability to promise.’

She had the passage corrected. She asked the document to distinguish what had been decided, what she thought probable and what still needed checking with Support, Lucas or customers. She avoided pasting in confidential information from actual accounts; the examples were reworded.

An hour passed. Then two.

The document filled with things Alice had known for years: the customers Malorean most wanted to serve, recurring problems, trade-offs they had accepted, promises they should not make, the limits of old data and the reasons a prototype wasn’t a feature ready to sell.

She added links to information they already had: Support feedback, decisions that could still be traced, meeting notes and product documentation. Where evidence was missing, the text had to say so.

Near the end of the third hour, ChatGPT proposed a summary. Alice read it. Some pages were too long. Others mixed a general principle with an example. She softened a few over-categorical statements one last time, then added a date and defined the document's scope.

Finally, she exported it as a PDF.

She scrolled through the pages, surprised by how many there were.

Three hours earlier, she had mostly wanted to explain to Lucas when he could decide something without her. Now she was holding an imperfect but surprisingly rich picture of what the team knew about its product without ever having brought that knowledge together.

It was not the final truth. Some passages would need confirming or updating. The PDF would not replace customers, data or the judgement of those accountable.

But at least the next person would not have to start from scratch.

Lucas opened the file the next morning.

'You wrote all this for me?'

‘Not entirely on my own,’ Alice replied. ‘I spent three hours talking to ChatGPT. It helped me ask questions, organise my answers and notice what was missing. I reviewed and corrected it. But I already knew most of these things.’

‘Then why had I never seen them?’

Alice smiled, a little sheepishly.

‘Because they didn’t belong in any ticket.’

Lucas looked through the contents. He found principles, examples of past decisions, business constraints, customer information, links to sources and questions that remained open.

‘Can I give this to my AI when I’m working on traceability?’

‘Yes, in our authorised environment. But tell it to use the document as context, not as proof that everything in it is true. And if a rule matters, go back to the source or the person accountable for it.’

‘That could save me an enormous amount of time. Sometimes I ask AI to help me choose between two solutions, but it only knows what I’ve told it. It can give me an incredibly convincing answer that ignores a decision we made three years ago.’

Alice knew the feeling. A perfectly reasonable answer in isolation could make no sense at all in their ERP.

Lucas stopped at a passage.

‘Here you say we must never hide missing business information if that could make a result look complete.’

‘That’s what I told you yesterday.’

‘Yes. And if I’d had this sentence yesterday, I wouldn’t have called you.’

He carried on reading. Then he looked up.

‘All right, but I’ve still got a problem.’

Alice felt a stab of discouragement.

‘What is it?’

‘Now I understand better how you think. I still don’t know where the boundary is between decisions I can make on my own and decisions I can’t.’

She put down her coffee.

He had found the biggest gap in the document.

They went through it together. Alice shared a blank page.

‘Let’s try something simple. Before calling me, ask yourself three questions: Do I understand the outcome we’re looking for? Does my decision stay within the boundaries we’ve agreed? And what could go wrong if I get it wrong?’

‘We’ll have to define “within the boundaries”.’

‘Exactly.’

They wrote down three levels.

Green: Lucas decided on his own when the choice was local, reversible, consistent with known rules and involved no new promise to a customer. He had to be able to explain his reasoning and check the result.

Amber: he consulted Alice when he encountered something new, conflicting objectives or a significant doubt about product behaviour. For a technical risk, he also involved the people with the relevant expertise, such as Thomas when it concerned the ERP's history.

Red: nobody decided alone when a change involved a fundamental rule, a commercial commitment on Malorean's behalf, sensitive data or a major risk that would be difficult to reverse. An explicit decision was needed from the relevant people accountable, sometimes including Hubert.

Lucas read through the framework.

'So if I change how the results appear in the demo to avoid confusion, that's green.'

'Yes.'

'If I want to add behaviour that could affect all our customers, I come to you.'

'We discuss it, yes. Especially if we haven't understood the need.'

'And if I want to change historical data to record batches?'

'It's not for me alone to give you the go-ahead. We need technical analysis and the right checks. You've reminded me of that often enough.'

Lucas smiled.

'I just wanted to be sure you'd remembered.'

They added another rule: if Alice did not reply, an amber or red decision would not magically become green. Equally, he should not ask her permission for every green decision.

‘And what if I get the colour wrong?’ Lucas asked.

‘Tell me. We’ll adjust the boundaries. I’d rather improve the rule than spend all our time blaming you for not guessing what I was thinking.’

Lucas nodded.

‘I could even ask my AI to check my choices against the framework, including the reasons and any missing information.’

‘Yes. Provided it doesn’t decide on our behalf whether a risk is acceptable.’

Alice added the framework to the reference document. She was beginning to understand the difference between sharing information and really enabling someone to make decisions.

They could have stopped there. But Lucas returned to the traceability demo.

‘We’ve spent an hour talking about my autonomy. Let me ask you something about the work itself: how will we know we’ve succeeded?’

Alice replied almost automatically:

‘If the customer can find the products made using a particular batch.’

‘Yes, but how do we check that? And how reliable does it need to be?’

This time she didn’t rush to answer.

They had spent days exploring screens, rules and data structures. They had a promising demo. Yet nobody had written down precisely how to judge the result.

Alice remembered the quality manager who had spent most of a day cross-checking files after a supplier alert.

‘We should start from there,’ she said. ‘Given a batch number flagged by a supplier, can they find the affected manufacturing runs without doing all that manual detective work again? How long does it take? Is the result complete? And when we don’t have the information, do we say so clearly?’

‘We could test it with them. Not just ask whether they like our demo.’

‘Exactly.’

Alice opened another page.

‘Actually, we should have started here before we did anything else.’

Lucas burst out laughing.

‘If you’d asked me to do this two weeks ago, I’d have accused you of trying to turn the work back into a ticket.’

‘Maybe. But look at what we’re writing.’

They recorded the observed problem, the users affected, the expected outcome, how to measure it, the limits of the data,

what was out of scope and which decisions still needed approval. One page, no more.

Lucas suggested calling it a *Mini PRD*.

‘Nobody’s telling me which screen to build,’ he observed.

‘We’re agreeing on what would make the work useful.’

‘And on the limits we mustn’t cross.’

Alice thought aloud:

‘For future projects, this should be our starting point. Before asking what we’re going to develop, we ask what outcome we want and how we’ll know we’ve achieved it. Only then do we define the scope and decide what is worth building.’

‘And if I can fill in the document with the right information, I can start without waiting for you to prepare a backlog.’

‘You can propose the scope. That doesn’t mean we’ll approve any commitment without looking at it. But yes, that’s the idea.’

They completed the traceability Mini PRD, leaving open the questions they could not yet answer. Some cases still needed checking with the customer, and technical constraints remained to be confirmed.

The document gave no false impression of certainty. It stated what they were trying to learn.

Alice found that more useful than a list of features to tick off.

The following Monday, Alice’s weekly meeting with Hubert was different.

She shared neither her usual tracking dashboard nor the list of features Lucas had built.

She showed him the decision-making framework, followed by the Mini PRD.

Hubert read the first few lines.

‘You’ve spent all that time writing documents when Lucas can work so much faster now?’

Alice recognised the objection. She had made it herself only a few days earlier.

‘I think we actually didn’t write down enough. We wrote what to build, but rarely the information we needed to make choices. So Lucas kept asking me to decide things. With this, he can make some of those decisions without me.’

‘And how do you know he’s staying within the boundaries?’

‘We’ve defined when he can decide on his own and when he needs to consult us. We’re not going to inspect everything else. We’ll check the results and adjust the rules when necessary.’

Hubert paused at the line about old data.

‘You won’t be able to guarantee traceability for something that was never recorded.’

‘No,’ Alice replied. ‘And the document says so. We need to make that clear to the customer.’

Hubert approved.

‘I’d rather know that before a salesperson promises the opposite.’

‘Exactly,’ Alice said. ‘The context won’t be just for Lucas. Julie could use it too, to see what we know and what we haven’t decided. But we’ll have to keep the information up to date, and everyone will need to know which parts are authoritative.’
They agreed to keep an editable working version, with the PDF serving as a dated snapshot. Important decisions would be recorded along with their rationale. Repeated principles could, where appropriate, become instructions for agents or checks built into tools. None of that would replace technical testing or the approvals needed before a release.
‘I don’t want to find out in six months that AI is applying an old rule nobody agrees with any more,’ Hubert said.
‘Neither do I,’ Alice replied.
For the first time, Hubert seemed most interested in the Mini PRD.
‘And you’re going to check whether the customer can actually trace their batches?’
‘Yes,’ Alice said. ‘We’ll give them cases to test with us. If the result isn’t reliable, the number of screens we’ve finished won’t matter.’
Hubert closed the document.
‘Good. I want to know what you’re learning, not just what you’re shipping.’

On Wednesday, Alice noticed something odd in her messaging app.

Not a single message from Lucas all morning.

She checked the project channel. There had been plenty of activity: Lucas had tested several ways of displaying results in the demo environment. In one case, he explained that he had chosen to show batches with identical references separately, so that the interface would not suggest a connection that did not exist. He cited the relevant product-context principle, specified the data used and proposed checking the customer's understanding of the screen.

He hadn't asked permission.

Alice read his explanation, then returned to work. She had nothing to add.

Towards the end of the afternoon, Lucas called.

'I've got an amber that might be turning red,' he said.

'I'm listening.'

'The customer wants to search through their older manufacturing runs too. For some of that history, we don't have a reliable link between raw materials and finished products. I can display what we do know, but if we present it as a complete search, they might assume that no result means no products are affected.'

Alice sat up straighter.

'You're right. We can't leave that ambiguity.'

'I could add a warning and distinguish between periods where the data can and can't be relied on. But I'd like us to decide together what we can promise the customer.'

‘Yes, this is one we should discuss. And we need to check with Support how a customer would understand that limitation.’

Lucas shared his screen. They began working on the problem. This was not approval of a button or an answer Alice already knew. They faced a decision that could change how much trust a customer placed in the software.

During the conversation, Alice realised that she had not once thought about how much of her time it was taking.

Before hanging up, Lucas hesitated.

‘You know, your PDF hasn’t answered all my questions.’

‘Thank goodness,’ Alice said.

‘Thank goodness?’

‘If I’d managed to put every answer in it, that would mean you had nothing left to discover.’

Lucas smiled.

‘All right. But I think I understand better when I need to come to you.’

After the call, Alice opened her diary again. She had a free hour before her next customer meeting.

A whole hour.

She still didn’t know whether their experiment would work at a larger scale, or how many mistakes they would make along the way. All she knew was that Lucas had moved forward without her on several decisions, then returned at the point where her judgement really mattered.

She added a new question to the shared document about how to present historical data. Not a ticket telling Lucas what to code. A decision to investigate with the people concerned. Their way of working had not become less demanding. They were simply beginning to focus that effort where it belonged.

8 — Greater Autonomy Requires Clearer Boundaries, Not Fewer

An organisation may decide that more people should be able to act without waiting for a specialist.
The increasingly fluid roles and responsibilities examined earlier create a new coordination problem: several people can now work on the same subject without necessarily having the same points of reference for making decisions.
But if every new ability to act requires routine human approval, autonomy remains theoretical.
The old handovers have simply moved from one profession to another.
The real challenge is therefore to create a system in which more decisions can be made locally without losing consistency, security, quality or product direction.
This reveals an important paradox:
the more autonomy we want to distribute, the more explicit the boundaries must become.
Autonomy does not come from having no rules.
It comes from being able to act without constantly asking permission because the goals, limits, risks and responsibilities are sufficiently clear.

If everything needs approval, there is no autonomy

Imagine a Product Engineer given ownership of a problem.

They speak to users, explore a solution, prototype, develop and prepare for launch.

On paper, they are autonomous.

But if they have to ask the Product Manager about every minor product trade-off, the designer about every interface detail and the Tech Lead about every technical choice, the organisation has not really redistributed responsibility. It has simply asked one person to navigate more approval stages.

Other professions face the same obstacle: allowing someone to build something achieves little if every action still triggers the old processes and approvals.

The question, then, is not simply **who can act**.

It is **within what boundaries that person can decide for themselves**.

At Cursor, this idea is expressed through goals and safeguards clear enough for builders to act without drifting away from the product vision or roadmap. [1]

The organisation replaces some detailed instructions with a decision-making framework.

Replacing some tickets with a framework for decisions

We saw with Linear why fully prescribed tickets can restrict initiative. [2] Now the question becomes practical: what information does a team need to make trade-offs for itself rather than wait for another specification?

This does not mean abolishing tickets. On Malorean, I have argued that a backlog should be a tool for action and coordination, not the complete memory of a product. [3]

A framework for autonomy could therefore make the information guiding local decisions explicit without fixing every solution in advance.

It might include:

- the priority customer or ideal customer profile (ICP);
- the problem to solve;
- the expected outcome;
- the limits of the scope;
- important constraints;
- the standard of quality expected;
- what the organisation has explicitly decided not to do;
- the thresholds beyond which a decision must be escalated.

Ramp illustrates this in practice: being clear about core customer profiles, core problems and what the company wants to do particularly well also makes it possible to turn down profitable opportunities that would pull the product in a different direction. [4]

Figma, meanwhile, reminds us that moving rapidly from zero to one is not enough. Real-world use cases, exceptions and quality standards still determine whether a quick prototype becomes a good product. [5]

This framework illustrates an organisational choice: not merely stating **what to do**, but making the boundaries that must be

respected visible. It is not presented here as a universal method.

Context becomes infrastructure

To make good local decisions, people need access to the information that helps them understand the problem.

That may be one of the most significant organisational changes of the age of AI agents.

A company cannot ask more people to make decisions while keeping most of the relevant context in the heads of a few specialists. Linear argues that customer feedback, strategic direction, decisions and code should be accessible to humans and agents alike. [2]

On Malorean, I have described the Product Manager as an **architect of context**: someone who turns scattered knowledge — interviews, sales objections, support feedback, decisions, data, sources and weak signals — into a reusable organisational asset. [6]

This fundamentally changes what documentation is for.

The goal is no longer to produce more pages.

It is to make available what someone needs to decide without reconstructing all the reasoning from scratch.

Good context allows someone to understand:

- what customers actually said;
- which assumptions have already been tested;
- which constraints matter;
- which important decisions have been made;

- which trade-offs have already been accepted;
- which metrics will tell us whether the result is successful.

The more autonomy increases, the more reliable and widely shared that context must be.

Otherwise, the company distributes the ability to act without distributing the ability to understand.

Preserving the reasons behind decisions

This redistribution creates another problem.

More decisions are made locally, sometimes by different people and sometimes with agents that were not present during earlier discussions.

Yet a decision without a record of its reasoning soon becomes open for debate again.

People remember the conclusion but not the context.

They know a rule exists but no longer know why it was chosen.

A new customer arrives, a new manager joins, a team changes, and a settled discussion starts all over again with less information than the first time.

This is exactly the purpose I assign to **Product Decision Records**. [7]

A PDR does not document every minor trade-off.

It records product decisions important enough to guide several teams, products or future situations.

In particular, it preserves:

- the context;

- the alternatives considered;
- the decision taken;
- the reasons for it;
- the positive and negative consequences;
- the risks accepted;
- the conditions that would justify revisiting it;
- the people who made the decision or were consulted.

The aim is not to add administration.

On the contrary, a good PDR should prevent the same discussion from being reopened fifty times.

A useful distinction emerges:

Code records what was built; decision records should preserve why certain important rules were chosen.

Sustained autonomy requires both.

Moving safeguards into the system

Even with good context, asking every employee to memorise every organisational rule would be inefficient.

Some companies are therefore moving repetitive rules out of manual approval processes and into their tools. This is an observable mechanism, not a guarantee that automation alone can ensure quality.

Meta uses a Diff Risk Score to estimate the likelihood that a software change will cause an incident and to inform risk-aware workflows. [8]

Cloudflare turns explicit engineering standards into automated checks. Some mandatory rules can prevent a

change from being approved until it is fixed or an explicit exception is granted. [9]

This matters because it changes the role of experts.

An Engineering Guarantor need not repeat the same rule manually across fifty pull requests if it can be checked automatically.

A Design Guarantor need not approve every button when the Design System makes the right choice easy and mistakes visible.

A Product Guarantor need not explain the same constraints every week when key decisions, customer targets and objectives are accessible.

The framework becomes infrastructure.

Tests, continuous integration, the Design System, standards, security rules, templates, instructions for agents, metrics and shared context take on some of the checking once carried out by people.

Human expertise can then focus on exceptions.

Not every decision deserves the same level of scrutiny

This is where risk becomes central.

Changing a piece of text and changing a permissions system should not go through the same approval process.

A local, reversible change does not require the same scrutiny as a data migration or a decision that alters the company's positioning.

One way to understand these organisations is to replace the yes-or-no question — 'Does this need approval?' — with a graded one: **How much control does this level of risk justify?**

The following three levels are **a framework proposed in this book**, not a practice established across every company cited.

Local autonomy

The decision stays within known objectives and standards.

It is reversible, its possible consequences are limited and it does not change an important rule.

The owner decides.

Targeted approval

The decision introduces something new, an exception or a significant risk within a domain.

The relevant Guarantor gets involved — Engineering, Design, Product or another domain.

They do not necessarily take ownership of the initiative. They contribute the specialist judgement it lacks.

Escalation

The decision changes the boundaries of the system itself.

On the technical side: fundamental architecture, security, sensitive data, critical migrations or regulatory risk.

On the Design side: a significant new interaction, a break with the Design System or major consequences for other user journeys.

On the Product side: changes to the target market, value proposition, pricing, major investment, roadmap or strategic trade-offs.

Here, the owner is no longer simply deciding **within** the existing framework.

They are proposing to change it.

In the model put forward here, that is the kind of situation that calls for a Guarantor or a more senior level of decision-making.

Without explicit boundaries, two opposing risks emerge

Without clear boundaries for decision-making, an organisation faces two opposite dangers.

The first is bureaucracy.

Everything gets escalated. Experts become queues. Managers must approve trivial decisions and autonomy disappears.

The second is diluted accountability.

Everyone makes local decisions without knowing when a decision stops being local. Contradictory choices accumulate until they create a visible problem for customers or the company.

On Malorean, I have described several tools for organisational coherence designed to make these boundaries clearer: shared language, a one-page brief, a hierarchy of metrics, a decision log and escalation rules. [10]

Their value does not lie in producing more documentation.

It lies in forcing certain questions into the open:

- who makes the final call?
- on which subjects?
- at what point?
- what can remain local?
- what must be escalated?
- under what conditions can a decision be reopened?

These two risks suggest an organisational hypothesis: **putting decisions in the right place** may matter more than multiplying or eliminating approvals.

Deciding also means accepting what we give up

Safeguards cannot be purely technical.

An essential part of the framework is the ability to make genuine trade-offs.

Organisations often make decisions so vague that everyone can maintain the appearance of agreement. Each person can project their own priority onto the outcome, and implementation later reveals that the constraints were incompatible.

I call these **soft decisions**: decisions that avoid making a real choice. [11]

They are particularly dangerous in highly autonomous systems. When the objective is ambiguous, each team can make a locally reasonable decision that undermines the whole.

Giving people more autonomy therefore also requires making the accepted losses visible.

Which customer comes first?

Which market segment will not be served for now?

Which quality standard is non-negotiable?

Which constraint can temporarily be relaxed?

Which objective takes precedence when it conflicts with another?

Autonomy does not require rules that cover everything.

It requires a framework that makes genuine trade-offs clear enough for local decisions to point in the same direction.

Measuring the outcome is the final safeguard

Autonomy does not end when something is shipped.

If a person truly owns a problem, they also need to see what happens afterwards.

The first chapter distinguished output volume from value created. Here the question is different: **how does someone who has made a decision later find out whether it was the right one?**

Canva provides a concrete mechanism: its teams can link their experiments to an estimate of commercial impact. [12]

For each initiative, this suggests defining:

- what change was expected;
- which measure would show it;
- which standard of quality had to be preserved;
- what users actually did;
- whether to continue, correct, abandon or investigate further.

Measurement closes the loop of accountability.

Without it, autonomy can become little more than permission to produce.

With it, autonomy becomes responsibility for an outcome.

Autonomy is not the withdrawal of management

Marty Cagan's model of empowered teams, introduced in Chapter 1.4, helps clarify management's role here. Giving a team a problem to solve without a predetermined solution requires clear objectives and strategic context. [13] Cagan also emphasises developing skills, supporting teams and providing leadership: allowing a team to decide does not mean abandoning it. [14]

There is an important limit to the comparison, however. Cagan describes the accountability of product teams with complementary skills. The Guarantor concept proposed in this book addresses another problem: how to preserve accountability for a domain when implementation moves from one profession to another.

The Guarantor protects the framework rather than approving every action

Chapter 1.6 introduced the Guarantor as accountable for a domain's coherence even when they do not personally carry out every task. Here, that role becomes practical: define understandable rules, automate repetitive checks and step in when risk or novelty goes beyond what a team can decide alone.

If the Guarantor must inspect everything, they become another bottleneck. If nobody is accountable for anything, responsibility dissolves.

Autonomy is therefore neither approving every action nor removing all controls: it is making more local decisions possible while preserving clear accountability.

But this transformation raises a question that goes beyond Product and Engineering.

If context circulates directly, decisions can be made closer to the work and agents absorb some of the coordination, **what happens to the layers of management that previously performed those functions?**

Malorean Corp. will first show us what this question means for the people involved. We will then examine its implications at company level.

Bibliography

[1] How Cursor Is Navigating the New Shape of Product Craft — Atlassian / Cursor

- URL: https://www.atlassian.com/blog/how-we-build/how-cursor-is-navigating-the-new-shape-of-product-craft
- Shows how goals, safeguards and Product judgement help people in overlapping roles act without losing the intended direction.

[2] Issue tracking is dead — Linear

- URL: https://linear.app/next

- Makes the case for sharing customer feedback, strategy, decisions and code within a richer context than ticket-centred workflows.

[3] Le backlog n'est pas un dépotoir : c'est un outil d'action — Malorean

- URL: https://malorean.net/articles/2026-06-03-le-backlog-nest-pas-un-depotoir.en.html
- Distinguishes the backlog as a temporary coordination tool from the lasting memory of product decisions and reasoning.

[4] 5 insights from Ramp's CPO on building for hypergrowth — Ramp

- URL: https://ramp.com/blog/5-insights-from-ramps-cpo-reddit-ama
- Emphasises priority customer profiles, core problems and turning down opportunities that would distract the product.

[5] What It Takes to Build Great Products Now — Figma

- URL: https://www.figma.com/blog/what-it-takes-to-build-great-products-now/
- Notes that quick prototypes still need real-world use cases, attention to exceptions and quality to become lasting products.

[6] Le PM comme architecte du Contexte — Malorean

- URL: https://malorean.net/articles/2026-06-08-le-pm-comme-architecte-du-contexte.en.html

- Proposes turning dispersed signals, sources and decisions into reusable context for future judgement.

[7] Product Decision Record : tracer les choix produit qui structurent l'entreprise — Malorean

- URL: https://malorean.net/articles/2026-06-03-product-decision-record-tracer-les-choix-produit.en.html
- Describes a concise record of important decisions, alternatives, consequences and conditions for reconsideration.

[8] Diff Risk Score: AI-driven risk-aware software development — Meta

- URL: https://engineering.fb.com/2025/08/06/developer-tools/diff-risk-score-drs-ai-risk-aware-software-development-meta/
- Presents a risk score used to adapt software review workflows according to the likelihood of an incident.

[9] How Cloudflare enforces engineering standards using AI — Cloudflare

- URL: https://blog.cloudflare.com/engineering-standards-enforcement/
- Shows how automated checks can enforce engineering standards before changes are released.

[10] Les outils de cohérence organisationnelle — Malorean

- URL: https://malorean.net/articles/2026-06-02-les-outils-de-coherence-organisationnelle.en.html

- Connects shared language, objectives, metrics, decision memory and escalation rules to clearer organisational trade-offs.

[11] Pourquoi les organisations préfèrent les décisions molles — Malorean

- URL: https://malorean.net/articles/2026-06-02-pourquoi-les-organisations-preferent-les-decisions-molles.en.html
- Analyses vague consensus decisions that conceal sacrifices and defer contradictions until implementation.

[12] Measuring Commercial Impact at Scale — Canva

- URL: https://www.canva.dev/blog/engineering/measuring-commerical-impact-at-scale/
- Describes self-service measurement of commercial impact linking experiments to business outcomes.

[13] Team Objectives – Empowerment — Marty Cagan, Silicon Valley Product Group (2020)

- URL: https://www.svpg.com/team-objectives-empowerment/
- Supports giving teams problems and strategic context rather than predetermined solutions.

[14] Product Leadership Is Hard — Marty Cagan, Silicon Valley Product Group (2020)

- URL: https://www.svpg.com/product-leadership-is-hard/
- Explains why empowered product teams need active leadership, skill development and support rather than less management.

9 — Malorean Corp. — Two Reasonable Decisions

On Monday morning, Vivien, who coordinated the two development teams, opened his meeting with the same question he had asked for years.

‘Can someone tell me where we are on this week’s work?’

His tracking board was already open. Tickets, owners, deadlines and blockers sat in columns he had patiently refined. He knew both teams of five developers, their strengths, their habits, the topics that needed pushing and those best left to develop in their own time.

Nobody answered immediately.

Then Lucas spoke.

‘I’ve updated the Mini PRD for traceability on Friday. We tested the batch search with the customer. Next we need to validate the cases where the data is incomplete. I’ve put the open questions in the context document.’

‘And your delivery date?’ Vivien asked.

‘That still depends on checking the historical stock data. Thomas needs to look at something with me.’

Thomas nodded. He was working on an improvement to the process for preparing shipments. He too had updated his progress in their tool.

Vivien moved on to the other developers. Their answers followed much the same pattern. Each understood their problem better, had made some decisions and could explain the next step. A few still needed help breaking down their work. The others had made progress without waiting for the meeting.

At the end, Vivien looked at his board.

Everything he had just heard was already there, sometimes in greater detail.

‘Right,’ he said. ‘That’s everyone.’

The meeting had taken twelve minutes.

Previously it would have taken forty-five.

Vivien wasn’t entirely sure whether he should be pleased.

He had been at Malorean long enough to remember all the developers working together in the Le Havre office. He had organised releases, managed delays, negotiated priorities with Product and explained to the CEO why a request promised for Friday would not be ready before the following Tuesday.

When the company moved to remote working after Covid, he had adapted his approach. Corridor conversations were no longer enough. He had asked people to write down decisions, make blockers visible and record what they had agreed with other departments.

That had become his daily work: making information circulate when nobody had the whole picture.

He liked it. What he particularly enjoyed were the moments when a struggling project got back on track because somebody finally brought the right people together.

But since AI tools began appearing in the teams' work, something had changed. Developers were producing things faster, but, more importantly, some had begun taking ownership in different ways. Lucas was talking directly to customers. Alice was sharing product context that explained why decisions had been made, not just the list of features to build. Bob was starting to use the same approach on his own projects.

And now an agent prepared some of the summaries Vivien used to compile by hand.

The work hadn't disappeared. Pieces of it were slipping out of his hands, or happening somewhere else.

He hadn't mentioned it to anyone yet.

The following Wednesday, Lucas showed Alice an updated traceability demo.

'We've tested what happens when a supplier flags a potentially defective batch. We can find the manufacturing runs we know about. And I've added an "under review" status so it doesn't look as though the batch has been cleared.'

Alice studied the screen.

A banner indicated that the situation needed checking. For older manufacturing runs, a second message explained that some information was unavailable.

‘That’s more honest than the first version,’ she said.

‘Yes. The customer’s quality manager specifically asked us not to confuse “no problem identified” with “we don’t have the information we need”.’

Alice approved. They had already captured that distinction in the context document.

‘Did you put it in the Mini PRD?’

‘Yes. And I added test cases.’

‘Then carry on.’

She didn’t need to decide where the banner went or the precise wording of the message. Lucas knew the outcome they were trying to achieve. He had chosen how to present it and planned to test the approach with the customer.

A little later, Thomas showed Vivien an improvement to the shipping module.

It addressed a different problem. At some customers, an order could become stuck because of intermediate or incorrectly recorded statuses, even when no explicit prohibition applied. Logistics teams wasted time having records corrected when no risk had been identified.

Thomas had therefore clarified the preparation rules: a batch marked ‘blocked’ prevented dispatch; other situations

remained visible and followed the normal process, with existing checks.

‘Are you sure you haven’t changed how the other modules behave?’ Vivien asked.

‘For the existing cases, yes. I’ve added regression tests. I haven’t changed the blocking rules.’

Vivien had no reason to doubt Thomas’s caution. Thomas sometimes knew about exceptions that were no longer even mentioned in the old documentation.

‘Good. Have the change reviewed before we go any further.’

Thomas nodded.

The two changes had not yet been brought together in a single scenario.

The problem surfaced on Friday, in an integrated test environment.

Someone from Support had prepared a complete journey: receipt of a raw material, manufacturing, a supplier alert, identification of the affected products and finally preparation of a shipment.

Lucas followed the demonstration on a video call. Thomas was there too. Alice had asked to join so she could see how someone who had not designed the new workflow would understand it.

Everything went well for the first few minutes.

The batch number was found. The manufacturing runs appeared. The ‘under review’ banner was displayed.
Then the Support colleague moved into the shipping module. The delivery note was still in preparation. No warning prevented them from continuing.
‘Hang on,’ Alice said. ‘We’re talking about the same batch, aren’t we?’
Lucas leant towards his screen.
‘Yes. Why?’
‘It’s under review up there, but down here we can carry on as though nothing’s wrong.’
Thomas opened the order details.
‘It isn’t blocked. It’s under review. Those aren’t the same status.’
‘To me,’ Lucas replied, ‘“under review” meant we shouldn’t treat it as approved until somebody had made a decision.’
Thomas turned towards his camera.
‘Then why didn’t you mark it as “blocked”?’
‘Because I didn’t want to impose a blanket ban. Some customers need to be able to investigate without stopping their entire operation. It’s for their quality manager to decide what’s actually affected.’
‘And in my module,’ Thomas replied, ‘we’ve got old batches with incomplete information. If I start blocking everything that hasn’t been explicitly cleared, some orders will never move forward.’

Alice looked at both screens.

She understood Lucas's reasoning. She understood Thomas's. The problem was that a user could see 'under review' in one part of the ERP and find no warning preventing them from proceeding in another.

'Let's stop the demo there,' she said.

Nobody objected. No real customer data had been touched, and neither change had gone into production. That was exactly what the test was for.

But the silence afterwards was less comfortable than the one following a successful demonstration.

Vivien joined the call when they told him what had happened. He let Lucas explain his decision, then Thomas explain his. He asked a few questions about tests and the scope of each change.

'You've each got a Mini PRD?' he asked.

'Yes,' Lucas replied.

'And your decision rules?'

'Yes,' said Thomas. 'I didn't alter the existing blocking behaviour. This was a local improvement to the preparation process.'

'Neither did I,' Lucas said. 'I added information so we wouldn't hide uncertainty. We'd explicitly agreed that mattered.'

Vivien opened both documents side by side.

Neither was wrong. They simply addressed different things. The first described how to communicate uncertainty in traceability. The second described when a shipment should be stopped. Neither had been written to answer the question connecting them: what should the software do when a batch was suspect but had not been explicitly blocked?

‘There’s a gap between these two projects,’ Vivien said.

Thomas nodded.

‘Technically, we have several options. But we need to know which behaviour we want.’

‘And we need to ask the customer what their quality manager actually does in that situation,’ Lucas added. ‘I’ve got hypotheses, not a rule that works for everyone.’

Alice spoke.

‘I’ll clarify the product behaviour we want. Not on my own. We need Support, Lucas, Thomas and probably a customer to check the consequences.’

Vivien looked up.

‘I’ll organise a meeting for tomorrow, then.’

Alice hesitated.

‘You can help us get the right people together, yes. But you don’t have to own the decision on our behalf.’

She wasn’t saying it to exclude him. Even so, Vivien felt the shift.

For years, he had been the one who got problems back into the right process. This time, the people concerned already

knew they needed to talk to one another. They didn't need him to turn their disagreement into a series of tickets.
They needed something else: a decision neither developer could make alone.

The next day, Alice began by asking everyone what they were trying to avoid.
Lucas didn't want a quality manager to believe they had complete traceability when data was missing.
Thomas didn't want an imprecise rule to stop operations unnecessarily for customers whose historical records were incomplete.
Support pointed out that a supplier alert did not always mean an immediate recall. Some customers first had to determine what was affected, consult their quality teams and decide what action to take.
'Right,' Alice said. 'Perhaps our mistake was trying to make the same status mean two different things.'
She shared a proposal.
A batch could be marked **for investigation** without the ERP pretending to make a regulatory or quality decision on the customer's behalf. However, any process that could use or dispatch that batch had to make the alert visible and require the prescribed confirmation process, rather than letting it vanish.

Uncertain historical information had to remain clearly flagged. And cases where the software lacked the necessary data could not be presented as risk-free.

‘We can check that in the shipping module,’ Thomas said. ‘But I also want us to test older orders.’

‘And I want to check the wording with the customer,’ Lucas added. ‘“For investigation” might not mean quite the same thing to their quality manager.’

Alice agreed.

They hadn’t finished designing the solution. But at last they had established who needed to decide what.

She recorded the provisional decision and open questions in the shared context, along with the reasons and cases to investigate. She added a condition to their decision framework: when a local decision changed the meaning of information used by several modules, it was no longer automatically green, even if the code change looked reversible.

‘We should have spotted that earlier,’ Lucas said.

‘Yes,’ Alice replied. ‘But that’s why we test. Now we know what to look for next time.’

Thomas studied the two documents.

‘We also need a test that goes across modules. We’ve got lots of tests checking each behaviour separately. Far fewer checking what happens when they meet.’

Vivien, who had been listening for several minutes, made a note.

For the first time since the call began, he recognised a problem he knew well: nobody had a complete view of all the dependencies.

But what struck him was how the solution was emerging. Alice hadn't dictated the screens. Lucas and Thomas weren't waiting for someone to break down the tasks. Support brought the customer's experience. His own role had been to make the missing connection visible and help the conversation happen.

That had been useful.

It had almost nothing to do with the release meeting he organised every Monday.

That afternoon, Thomas received a message from a less experienced developer on the other team.

The developer couldn't understand why an old stock-processing routine made two apparently identical checks. The coding agent suggested deleting one.

Thomas opened the file.

'Don't touch that until we've looked at it together,' he replied.

They spent half an hour sharing a screen. The second check covered a case left over from an older product change, poorly explained in the code. Thomas could no longer remember exactly which release it dated back to, but he knew where to look.

‘Your tool hasn’t invented a bad answer,’ he explained. ‘It found two similar checks and suggested simplifying them. Without knowing the software’s history, I can see why.’

‘Do you really never use AI for coding?’ the developer asked.

‘Not to suggest blind changes in areas like that.’

‘But you’ve just spent thirty minutes on something Lucas might have checked in five.’

Thomas thought for a moment.

‘Perhaps. And perhaps Lucas would have found the same thing. What matters is that you understand why the check exists. Next time you’ll know how to ask a better question, both of your tool and of me.’

The developer asked whether he could document the explanation.

‘Please do. I’m tired of being the only person who remembers.’

Thomas ended the call, surprised by what he’d just said. For years, his knowledge of exceptions had made him indispensable. He had sometimes feared that AI would make his experience less valuable. The discussion showed him something different: he could help others acquire the judgement they lacked, instead of keeping every answer to himself.

He opened the shared document and wrote a few lines.

At their Monday meeting, Hubert asked Alice how the experiment was going.

She told him about the two incompatible decisions, the test that revealed the problem and how they had adjusted their framework.

‘So you let two developers decide for themselves, and they almost produced an inconsistency?’ Hubert summarised.

‘Yes.’

‘That’s hardly reassuring.’

‘We caught it before anything went into production,’ Alice replied. ‘The problem isn’t that two people made decisions. It’s that none of our rules explained how those decisions should work together.’

Hubert looked at the incident notes.

‘Previously, we’d probably have held an extra approval meeting.’

‘Perhaps. But another meeting wouldn’t necessarily have exposed the question. We needed somebody to test the complete journey and clear accountability when a decision affects several domains.’

‘And who makes the decision now?’

‘Lucas owns his problem. Thomas owns his, and helps us understand technical risks. I’m the Product Guarantor, responsible for overall product coherence. For sensitive cases, we consult the relevant specialists. We can’t delegate all that to a tool.’

Hubert nodded slowly.

‘And Vivien?’

Alice took a few seconds to answer.

‘He helped bring the teams together and work out where the problem lay. But he didn’t need to assign their tasks or decide on the product behaviour.’

Hubert seemed to be thinking about something else.

‘I saw him on Friday,’ he said. ‘He showed me the new project report. It’s very good. But I reckon a machine could have prepared eighty per cent of what he told me.’

‘He knows that too,’ Alice replied.

‘What I don’t know is what we’re supposed to conclude.’

Alice had no ready-made answer.

‘Not that he’s no longer useful,’ she finally said. ‘We’ve just seen that he knows how to get people working together. It’s just that his job may no longer need everything to go through him.’

Hubert kept looking at his screen.

‘We’ll have to talk to him.’

Vivien received the invitation for the following day.

His first thought was reorganisation. Over the past few weeks, his days had contained more empty spaces. Developers owned their own work. Tools collected automatically some of the information he used to spend evenings assembling. He still handled people problems, helped new arrivals settle in and unblocked complex situations, but those tasks no longer filled his diary as they once had.

He had liked being indispensable to Malorean's releases.

He wasn't sure he liked discovering that he was becoming less so.

Yet the meeting began with a question that surprised him.

'What would you like to carry on doing, Vivien?' Hubert asked.

'My job,' he replied.

Hubert smiled.

'That's exactly what we're trying to understand.'

Vivien looked at Alice, who had been invited to talk about how the teams worked, then turned back to Hubert.

'I don't want us to pretend. The teams don't need me to ask every week where they are. And I don't want to spend my time reviewing summaries an agent has already prepared.'

'What do they still need?' Alice asked.

'The conflicts that never get written down. The people who are reluctant to admit they're lost. New starters who don't know whom to ask. Decisions that cross several teams. And sometimes just someone who takes the time to listen.'

He paused.

'But that doesn't mean all of it should still have to go through me.'

Hubert sat up.

'No. And we'll need to be clear about who's responsible for those things if your role changes. I don't want us to abolish a title and then discover that nobody looks after people any more.'

Vivien agreed.

They discussed supporting developers, cross-team decisions and what senior management still needed to oversee. Alice spoke about how Thomas was helping colleagues understand the old code.

No definitive solution emerged from the call.

When it ended, Vivien sat looking at his calendar for a few moments.

He felt less threatened than the day before, but not entirely reassured. His profession hadn't disappeared. It was breaking into several responsibilities, some of which could be handled differently.

That evening, he closed his laptop earlier than usual.

For once, he had no delayed release to catch up on.

And he still didn't know what he was going to do with the time he'd got back.

10 — Will AI Make Middle Management Disappear?

A team can now consult its own decision records, act within known boundaries and entrust some progress tracking to agents. It needs a manager less often to pass information along or organise each step.

So **what happens to the layers of hierarchy that used to fulfil that role?**

The answer may appear obvious: if AI takes over coordination, middle managers lose their purpose. Yet managers do more than pass on instructions. They make trade-offs, support people, connect teams and deal with problems that fit no established procedure.

The real question, then, is not so much how many jobs will disappear as **where their responsibilities and authority will go**.

A hierarchy built to coordinate decisions

An important customer reports a problem. Support passes it to a manager, who contacts Product. Product asks Engineering for an estimate. Several managers pool their information, compare priorities, and the decision may then be escalated another level.

That process is not necessarily irrational. Each manager holds part of the context and can commit part of the company to a

course of action. It becomes burdensome when the chain is the only way to understand a problem and decide what to do. Historically, hierarchy enabled companies to grow despite fragmented information. Middle layers translate senior management's goals into priorities, coordinate teams, monitor dependencies and settle conflicts over resources. They also help assess employees, develop their careers and address people-related difficulties.

But these activities are not all alike. Preparing an incident report is different from deciding to abandon a sales opportunity. Tracking a deadline calls for different judgement from resolving a conflict or helping someone develop.

On Malorean, I have previously made a similar distinction in relation to the Product Owner: a position may cease to justify itself as a simple intermediary without the function it was meant to perform becoming unnecessary. We need to examine the work being done, not just the title of the person doing it. [1]

If the middle layers are reduced, who will perform the functions they used to carry out?

Remote working had already changed the rules of coordination

AI did not invent this question. Companies operating remotely, sometimes across countries and time zones, had to learn to make decisions without relying on corridor conversations or improvised meetings.

At GitLab, the ‘handbook-first’ principle means making rules and context accessible in writing. Anyone can propose a change; those responsible for the relevant domain decide whether to adopt it. An employee can therefore find information without waiting for a manager to provide it. [2]

Alan describes a different but complementary approach. Important decisions are discussed in GitHub threads: one person makes and implements the decision, another remains accountable for the initiative’s success, while those affected are consulted or informed. Discussion can be open without requiring unanimity. [3]

At 37signals, the company behind Basecamp, much of the coordination happens through written exchanges, reviews and information kept with projects rather than scattered across fleeting conversations. [4]

These are not three companies without management. They represent three ways to **reduce dependence on a person who holds the information or has to keep reminding everyone what was decided**.

That precedent matters for AI agents. A system can consult a rule, retrieve a decision and identify the person responsible, provided the organisation has made that information explicit. A decision that remains in a manager’s head is inaccessible to colleagues and agents alike.

But we should be careful not to romanticise remote work. A Microsoft study involving more than 61,000 employees found,

among other things, that collaboration networks became more siloed following the shift to remote working. Writing more does not guarantee that context will be shared or that teams will cooperate. [5]

The real legacy of remote working is not the disappearance of the office. It is the possibility of organising some work around information and decisions that are accessible without a permanent intermediary. AI can build on this achievement. It does not create it by itself.

AI is starting to take on some of this work

A document stores a rule; a dashboard shows a delay. An agent can go further: bring several sources of information together, prepare a progress report, detect a contradiction, suggest a response or carry out an authorised action.

These are precisely the activities that took up part of managers' time: consolidating information, reporting, monitoring and preparing decisions. Their cost may fall. Their disappearance is far from established.

An exploratory analysis by HEC Paris, drawing partly on 45 interviews, suggests that structured, data-rich managerial tasks lend themselves better to automation than tasks dependent on trust, human relationships and judgement. It measures neither the hours of management actually removed nor the jobs rendered unnecessary. [6]

An experiment at Procter & Gamble also shows that, on product-innovation tasks, one person working with AI can

achieve results comparable to those of a two-person team without AI. The tool notably helps combine perspectives from different disciplines. But making collaboration easier in a specific task does not mean replacing a manager over the long term. [7]

In another experiment involving 7,137 employees across 66 companies, regular users of an AI assistant saved around two hours of email time per week during the second half of the study. Yet researchers found no significant change in the composition of work tasks over six months. The study concerned an office assistant, not autonomous agents carrying out entire processes. [8]

That gap is crucial: **an individual's productivity gain is not yet an organisational transformation**.

For a management layer to become less necessary, information must actually reach teams, decision-making authority must change, and unusual situations must have somewhere to be resolved. This is a question of organisation, not just tools.

Three companies, three very different paths

Organisations are not responding in the same way. Three cases help distinguish a reorganisation already under way, an ambition built around AI and the difficulties of rapid transformation.

Bayer: a hierarchy that has genuinely been reduced

Through its *Dynamic Shared Ownership* (DSO) model, Bayer has redistributed some decisions to multidisciplinary teams. In his 2025 letter to shareholders, the group's chief executive reports removing up to six organisational levels and reducing management positions by around two-thirds. He claims faster decisions and shorter development times. [9]

The scale deserves attention, even though Bayer itself is reporting the results. This sort of transformation changes daily responsibilities, not just titles on an organisation chart: teams decide more locally, leaders spend more time supporting people and some coordination functions are reorganised.

But one detail changes how the case should be interpreted: **DSO began before the rise of generative AI agents**. Bayer shows that a large company can substantially reduce its hierarchy; it does not prove that AI caused it.

Block: sharing context without going through the whole hierarchy

Block goes further in the role it wants AI to play. In a piece co-authored by Jack Dorsey and investor Roelof Botha, its leaders describe an organisation supported by a shared information system.

The ambition is for an employee to understand a situation and act without waiting for a manager to relay the context. The authors distinguish individual contributors, people accountable for cross-functional outcomes (DRIs) and leaders who continue producing work while coaching others, known as *player-coaches*. [10]

Managers whose main role was passing information along would become less important. Developing people, accountability for outcomes and leadership decisions would remain.

This is an ambition described by the company and its investor, not an independent assessment proving that the model has succeeded over time.

Meta: when output outpaces the capacity to check it

A counterpoint comes from a Reuters investigation into Meta's 2026 reorganisation plans: smaller teams, extensive use of AI and reductions proposed or undertaken in certain layers. A second wave of restructuring was eventually abandoned. According to internal documents cited by Reuters, changes to internal software and infrastructure rose far more quickly than new features delivered to users. At the same time, technical incidents and the time required to resolve them increased. Meta disputed or qualified aspects of the report. [11]

These figures do not establish that AI or reductions in management caused the incidents: several transformations were taking place at once. But they raise a fundamental question: what is the point of producing more if the company can no longer verify, stabilise and absorb what it produces?

Bayer demonstrates the possibility of flattening an organisation, Block sketches an AI-oriented model, and Meta illustrates the risks when output grows faster than

coordination. None of these paths, on its own, tells us the future of management.

Management functions do not disappear with the people holding those titles

Consider an example: a customer reports a software fault. As long as the problem is familiar and the rules for resolving it are clear, a team can act without waiting for a succession of approvals. With reliable data, it can investigate, apply a fix and inform others.

Then two diagnoses conflict. The agent recommends one action, a developer another. The fix could disrupt a neighbouring service.

At that point, someone must bring the disagreement into the open, assemble the expertise needed and decide which risk to take. Autonomy has not eliminated coordination; it has changed how coordination happens.

An experiment by Martin Krzywdzinski and colleagues offers an interesting clue. In a simulated production line, 87 pairs dealt with failures using an AI system that was sometimes wrong. Self-organising teams communicated more and resolved problems better than pairs with an appointed leader. The analysis emphasises the decisive role of communication. [12]

Two people in a laboratory cannot stand in for a large company. But the study highlights an important distinction: **a**

team can work without a formal leader; it cannot do without the work of coordination.

That work might be done by peers, an expert, someone given a mandate to decide or a line manager. The right arrangement varies with skills, risks and disagreements.

There is also work that no automatically generated report can do on its own: welcoming someone, helping them progress, resolving tension, supporting a difficult decision or recognising a contribution. Not every manager necessarily performs these tasks well, and they can be distributed differently. But someone still needs the responsibility and the time to do them.

Gallup finds that greater use of AI does not eliminate fears about jobs, and that perceived support at work is associated with lower levels of concern. This does not establish the need for a traditional manager, but it reminds us that an organisation is also a human community. [13]

And who will train the people who must make tomorrow's difficult decisions?

On Malorean, I have raised a concern about junior developers: when AI writes code and fixes errors immediately, it may remove opportunities to learn how to debug, understand a failure and develop sound judgement. That is a concern worth investigating, not evidence of a general decline in skills. [14]

The same problem affects future managers. People also learn by observing trade-offs, receiving feedback and seeing how

teams resolve conflict. A company that removes management layers must find other ways to pass on that experience. Ultimately, each function could have several destinations: automation, responsibility taken on by the team, transfer to a specialist, or escalation to senior leadership. **A task may disappear, change hands or become additional work for somebody else.**

To assess whether a flatter organisation is actually more efficient, therefore, counting eliminated positions is not enough. We need to compare the monitoring saved with the new burden of supervising agents, resolving incidents, supporting people and training them. We do not yet have a robust net assessment of that redistribution.

Is a flatter organisation necessarily more autonomous?

One final question remains, less obvious on organisation charts: **who decides when the intermediaries disappear?** We can imagine better-informed teams, free to resolve more problems without escalating every decision. But we can also imagine the opposite: fewer managers on the ground and more decisions controlled directly by senior leadership through digital systems.

This tension existed before generative AI. In research on industrial firms, Nicholas Bloom and colleagues show that technologies which give employees more information are associated with greater local autonomy. Technologies that make communicating with superiors easier can instead favour

centralisation: asking for an instruction from the top becomes simpler. [15]

That earlier research proves nothing about today's agents. Nevertheless, it illuminates a mechanism: **making data available is not the same as distributing authority**.

Algorithmic management systems make the risk more concrete. They can assign tasks, propose priorities, measure performance and trigger alerts. Research shows that such tools can carry out certain management and evaluation functions even when human managers remain to implement them or handle exceptions. [16]

A system does not appear as a superior on an organisation chart. Yet if it determines which tasks people receive, which criteria measure their performance and when an action is allowed, it exercises part of the authority in the organisation. We must therefore look at the decisions behind the screens: who sets agents' objectives? Who chooses the rules, and who can change them? Who can challenge those rules? And when an algorithm makes a mistake, who can genuinely reverse the action it triggered?

One team may have considerable capacity to build things while being unable to change its priorities. Another may retain a manager while genuinely controlling its own methods and local trade-offs.

On Malorean, I have already described the 'soft decisions' that create an appearance of agreement without making the

choice, the accepted sacrifice or the person with final authority explicit. Removing a layer makes no difference if authority remains just as difficult to identify. [17]

We may therefore face a paradox: **less visible hierarchy, but more centralised control embedded in tools**.

That outcome is not inevitable. The same systems can help teams decide more locally and make power structures more transparent. It depends on what a company genuinely chooses to delegate, not on how many boxes it removes from its organisation chart.

What might genuinely disappear

Will AI make the middle layers of companies disappear?

Some layers may become less necessary. But their disappearance will be neither automatic nor equivalent to the end of management.

Functions consisting largely of relaying information, consolidating progress reports or arranging repetitive approvals may become less significant. Others — making trade-offs, resolving conflicts, passing on experience and taking responsibility — will still have to be performed somehow.

Bayer shows that many management levels can be removed without AI being the identified cause. Block proposes entrusting more coordination to a shared system. Meta reminds us that accelerating delivery without measuring reliability or human workload can weaken an organisation.

Nothing yet establishes that agents will cause management to disappear across the board.

The shift could nevertheless be profound. A manager would less often be the mandatory intermediary for information or approval. They might devote more attention to consistency, unusual situations and developing people. Elsewhere, some of those functions would fall directly to teams; others would be built into software or concentrated in the hands of a few decision-makers.

It may not be management itself that becomes unnecessary. It is hierarchy as the compulsory mechanism of coordination that is beginning to be questioned.

For a developer, the consequence is concrete. When work passes through fewer intermediaries, they do not merely get more tools to produce software. They encounter the customer need, the trade-offs and the results for which somebody must be accountable more directly.

This is the shift described throughout this first book: the ability to act increases, and so does the need to understand, choose and take responsibility.

It guarantees neither better decisions nor more humane organisations. But it does require us to design them differently: share context, make authority identifiable, preserve opportunities to pass on experience and accept accountability for outcomes.

The question is not merely what AI now enables us to produce, but what kind of company we want to build around that new capability.

In the final episode, we return to Malorean Corp. to see how its leaders and teams experience the choices they have made, beyond the organisation chart.

Bibliography

[1] Le PO n'est pas un métier, c'est une fonction — et l'IA n'y change rien, elle accélère — Malorean

- URL: https://malorean.net/articles/2026-08-03-le-po-nest-pas-un-metier-cest-une-fonction.en.html
- Author's argument distinguishing a useful function from a job that serves mainly as a link between decisions and implementation.

[2] Handbook First — GitLab

- URL: https://handbook.gitlab.com/handbook/company/culture/all-remote/handbook-first/
- Describes documentation as a shared reference, with open discussions of proposed changes and clearly accountable people.

[3] Comment prendre des décisions dans un environnement de travail asynchrone — Alan

- URL: https://alan.com/fr-fr/blog/healthy-business/a/decisions-asynchrones-methode-alan

- Explains how decisions are discussed in writing, who decides and remains accountable, and who is consulted or informed.

[4] The 37signals Guide to Internal Communication — 37signals

- URL: https://37signals.com/how-we-communicate
- Describes predominantly asynchronous internal communication, with written discussions and durable project context.

[5] The effects of remote work on collaboration among information workers — Microsoft Research

- URL: https://www.microsoft.com/en-us/research/publication/the-effects-of-remote-work-on-collaboration-among-information-workers/
- Studies collaboration networks after the shift to remote work, including the risk of growing silos.

[6] AI Automates Routine Management and Raises the Value of Leadership — HEC Paris

- URL: https://www.hec.edu/en/dare/tech-ai/ai-automates-routine-management-and-raises-value-leadership
- Exploratory analysis of managerial tasks based on interviews and model assessments, without causal measurement of job losses.

[7] The Cybernetic Teammate: A Field Experiment on Generative AI Reshaping Teamwork and Expertise — NBER

- URL: https://www.nber.org/papers/w33641

- P&G experiment on AI's contribution to product-innovation tasks and cross-disciplinary expertise.

[8] Shifting Work Patterns with Generative AI — NBER

- URL: https://www.nber.org/papers/w33795
- Randomised study in 66 companies finding individual time savings without a significant change in task composition over six months.

[9] Chairman's Letter — Bayer Annual Report 2025

- URL: https://reports.bayer.com/annual-report-2025/en/to-our-stockholders/chairmans-letter.html
- Bayer reports a substantial reduction in management levels and roles through DSO, which predates AI agents.

[10] From Hierarchy to Intelligence — Sequoia Capital / Block

- URL: https://sequoiacap.com/article/from-hierarchy-to-intelligence
- Jack Dorsey and Roelof Botha's vision of coordination through a shared system, with DRIs and player-coaches.

[11] Mark Zuckerberg had a bold plan to replace Meta staff with AI. Here's how it imploded — Reuters

- URL: https://www.timeslive.co.za/news/sci-tech/2026-08-26-special-report-mark-zuckerberg-had-a-bold-plan-to-replace-meta-staff-with-ai-heres-how-it-imploded/
- Reuters investigation reproduced by TimesLIVE into organisational and technical tensions surrounding Meta's restructuring.

[12] How team organization influences the ability to solve automation failures — AI & Society

- URL: https://link.springer.com/article/10.1007/s00146-025-02761-5
- Experimental study of self-organising and hierarchical pairs confronting automation failures, highlighting communication.

[13] Using AI More Does Not Reassure Workers, Managers Do — Gallup

- URL: https://www.gallup.com/workplace/713231/ai-not-reassure-workers-managers-do.aspx
- Survey on workers' concerns about AI, respect and perceived support at work.

[14] L'IA ne devrait pas seulement servir à produire dix fois plus. Elle devrait aussi nous obliger à comprendre dix fois mieux ce que nous produisons — Malorean

- URL: https://malorean.net/articles/2026-07-23-lia-produire-dix-fois-plus-comprendre-dix-fois-mieux.en.html
- Author's reflection on learning opportunities lost when AI performs the debugging that once helped junior developers develop judgement.

[15] The distinct effects of Information Technology and Communication Technology on firm organization — NBER

- URL: https://www.nber.org/papers/w14975

- Distinguishes the effects of greater information access and easier communication with superiors on local decision-making authority.

[16] Algorithmic Management: From Technology to Politics and Theory — Weizenbaum Journal of the Digital Society

- URL: https://ojs.weizenbaum-institut.de/index.php/wjds/article/view/4_3_9/163
- Reviews management and performance-evaluation functions performed by algorithms alongside human oversight.

[17] Pourquoi les organisations préfèrent les décisions molles — Malorean

- URL: https://malorean.net/articles/2026-06-02-pourquoi-les-organisations-preferent-les-decisions-molles.en.html
- Essay on superficial consensus that delays real trade-offs and obscures accountability.

11 — Malorean Corp. — Time Regained

Two months after the leave-management demonstration, Hubert Félix Duca opened his Monday meeting without asking how development work was progressing.

‘So, what did we learn last week?’

Alice shared her screen.

She had a summary of recent customer feedback, results from ongoing experiments and a few decisions that remained open. An agent had prepared an initial grouping of the information overnight. She had reviewed it that morning, rejected several misleading connections and added two questions it could not have formulated on its own.

‘On traceability, our pilot customer can now find the affected manufacturing runs much faster when the data is complete,’ she began. ‘But there are still historical cases where we have to make the limitations clear. We’re going to test a new presentation with their quality manager.’

Hubert nodded.

‘And the other customers?’

‘Too early to say,’ Alice replied. ‘That’s exactly why we haven’t rolled it out more widely.’

Bob took over. He was working on a different set of product initiatives and had just spoken to several people in Pre-sales and Support.

‘We’ve also got three requests that look similar at first glance,’ he said. ‘The agent grouped them under a single feature idea. But there are at least two different problems behind them. I’d like to clarify whom we’re targeting before deciding whether to invest.’

‘How many developers do you need?’ Hubert asked.

Bob smiled.

‘None for now.’

Hubert smiled back.

Bob glanced at a notification, then added:

‘By the way, Support has been testing a new way of categorising recurring requests since yesterday. Someone on the team put it together with an agent, using anonymised copies, to see whether information can get to the right people faster.’

‘Who started that?’ Hubert asked.

‘Support. Alice confirmed that the scope was internal and reversible, with a human checking the results.’

‘Did you know about it?’

‘About the idea, yes. Not the details of the test.’

Hubert’s first instinct was to ask why nobody had told him before they started. He held back. They had specifically agreed that limited, reversible initiatives no longer required his approval.

‘Let me know if it works,’ he finally said. ‘And if somebody has to maintain it, I want us to know who.’

He had not needed to authorise the test. He was still responsible for the resources they might invest in extending it. Not so long ago, that response would have seemed unusual in a Product meeting. The aim was mainly to work out what developers should build and in what order. Now Alice and Bob spent more time choosing problems, checking that initiatives remained coherent and stepping in when a decision went beyond the agreed boundaries.

Their job titles had not changed. In the company's administrative records, they were still Product Managers.

But their daily work no longer looked quite the same as it had two months earlier.

Perhaps the most visible change was what no longer felt remarkable.

A developer spoke to a customer. Another proposed a Mini PRD for improving a module. A salesperson created a variation of a demo in the environment designed for it. Someone in Support reported an inconsistency in a product rule directly. None of this now sparked a major debate about the boundaries between professions.

Malorean had not announced that everyone could do everything. The company had simply stopped assuming that an action had to pass through the person traditionally assigned to the job.

The two teams of five developers were still there. Most now owned some initiatives end to end: understanding the problem, proposing a scope, speaking to the people concerned, developing, testing and then following the results. Some had adopted this approach faster than others. Two developers still preferred close support when defining a problem. Another was excellent at complex technical challenges but had no desire to fill their diary with customer meetings. Nobody was trying to force them to become copies of Lucas.

There were still tickets, too.

They used them to coordinate an intervention, record a particular task or track a fix. But they no longer tried to squeeze the purpose of an initiative, every earlier decision and all the information an agent might need into a ticket. Instead, each initiative owner maintained a context area in their shared documentation tool. It contained objectives, decisions, sources, risks, open questions and links to things that already existed: tickets, quotes, Support feedback, meeting notes, tests and sometimes old discussions they had managed to retrieve.

There was no need to copy everything. What mattered was that somebody could understand why a decision had been made, where to verify information and who could change it.

Alice's first document had served as a model.

Bob adopted the method, then developers, then several people in Support, Sales and Implementation. Many discovered, as Alice had, that they could spend an hour explaining their choices to ChatGPT aloud when formalising those choices on their own might have taken days.

Not everything in people's heads had miraculously become accessible. But some of the reasoning, exceptions and tacit knowledge were starting to be shared.

Remote working had made that shift feel more natural. Since Covid, Malorean had already learnt to write more so that colleagues spread across France could work without waiting for a corridor conversation. AI accelerated the extraction and organisation of knowledge. It had not invented the need to share it.

Alice opened a page entitled *Product Decisions — Shared Rules*.

'We had another misunderstanding last week,' she told Bob. 'One team's documentation used "approved" to mean "technically reviewed". Support used it to mean "behaviour confirmed with the customer".'

Bob sighed.

'That word has never meant the same thing to everyone.'

'Exactly. I've added a link to the decision we made after the batch incident.'

A few weeks earlier, Lucas and Thomas had nearly allowed two incompatible behaviours to coexist between traceability and shipping. Their changes had been stopped during testing. Since then, the teams had learnt to be wary of words that seemed obvious in one module but meant something different elsewhere.

Context had helped. It had created new problems as well. Some pages were out of date. Provisional decisions had been copied as though they were definitive rules. On one occasion, an agent had retrieved an old commercial hypothesis as though it committed the company to a course of action.

They had had to establish some discipline: every context identified its owner, sources, confirmed decisions and points that remained uncertain. Significant changes had to be recorded, and sensitive information was not made available indiscriminately to every tool.

‘We could have avoided creating so many pages,’ Bob remarked.

‘Perhaps. But the contradictions existed before too. They were just in our heads, where nobody could see them.’

Alice read her colleague’s latest change.

‘I approve the principle. Not the wording about German customers: we don’t know enough yet.’

Bob corrected the line.

Being a Guarantor often looked like that. Not approving every screen, but preventing an assumption from turning into a rule by accident.

On the other team, Thomas had just finished talking with a developer who had joined recently.
He had spent an hour with them going through some of the ERP's legacy code. The developer had used an assistant to understand an old payroll mechanism, then asked why an exception seemed to contradict the general rule.
Thomas had found the reason for the exception in an old release.
'If you rely on the summary alone, you might think we should clean up the code,' he said. 'But before cleaning anything up, you need to understand what you're about to remove.'
'Could you put that in the team's context?'
'That's what I'm doing.'
Thomas shared the new note. He had added the source, an explanation and two test cases.
He still wasn't handing over the implementation of his technical changes to agents the way Lucas did. His scepticism had not disappeared. Nevertheless, he willingly used the new documentation spaces and worked more closely with others to pass on his experience.
Vivien, who had been overseeing the new developer's induction, called Thomas on video a little later.

‘How’s it going?’

‘Well. They’re curious. They ask questions before deleting things.’

‘That’s reassuring.’

‘And I’ve finally taken the time to write down why some of the checks exist.’

Vivien smiled.

‘So you’ve changed your mind about AI?’

‘No. I’ve changed my mind about keeping those explanations to myself.’

He paused.

‘But this takes time. If I’m expected to coach others and you still fill every day with development work, it won’t happen.’

‘I’ve spoken to Hubert. We’ve set aside time for mentoring in the plan. It’s not something we’ll do only when everything else is finished.’

Vivien knew what he was talking about. For years he had seen teams announce that they would train new starters ‘as soon as they had some time’. That moment never arrived.

They spoke a little longer about the developer’s needs, then Vivien ended the call.

He had not opened the release board once.

At lunchtime, Hubert asked Vivien to join him on a call.

‘I’ve got a question,’ he began. ‘How much of your time do you now spend doing what you did last year?’

Vivien thought about it.

‘Not enough to justify my old full-time job, if that’s what you’re asking.’

Hubert did not reply immediately.

For a long time, Vivien had been the point through which two technical teams coordinated. He managed delivery, dependencies, some urgent issues and conversations between developers and business leaders. When the organisation was less autonomous, his role easily filled his days.

That was no longer true.

Every initiative had a clearly identified owner. Developers made many local decisions themselves. Information circulated without him having to relay it. Tools prepared progress reports and flagged dependencies, although their claims still needed checking.

People problems, conflicts and mentoring had not disappeared. Vivien still helped with them. But those responsibilities no longer needed to be concentrated in a post devoted to daily delivery supervision.

‘I don’t mind having fewer pointless meetings,’ he continued. ‘Quite the opposite. But I didn’t spend all these years learning how to manage teams just to become someone who checks whether an agent filled in a spreadsheet properly.’

‘I won’t ask you to,’ Hubert replied.

They had prepared a transition. Important technical decisions belonged to the relevant Guarantors. People responsible for

mentoring had dedicated time. Hubert retained accountability for career decisions and people problems that could not simply be left to a document. Vivien would hand over the remaining coordination work.

It was not perfect. They had already had to revisit a situation where two people each thought the other was looking after an employee who was struggling. Another team had waited too long for a decision because nobody had been named as accountable.

Each time, they had corrected the responsibilities and how they were made visible.

‘There’s one thing I don’t want,’ Hubert said. ‘I don’t want us to make your job disappear and then pretend that the work you’ve done had no value.’

‘Neither do I.’

‘Then we need to work out where your experience could be useful now.’

Vivien looked at his cup.

‘I think I’ve got an idea.’

8

A few days earlier, his wife had brought up Germany again. She was German. For some time, they had been talking about moving there. Not urgently: they had a life in France, routines and friends. But the possibility kept returning, with increasingly concrete details.

Vivien had always put the question off.

He liked his job at Malorean too much. He knew the teams, their history, the software and the customers. Leaving would have felt like abandoning something he had helped build.
Yet he liked challenges too. At this stage of his career, he was surprised to find himself wanting a new one.
The previous Saturday, they had browsed property listings in Germany without quite admitting what they were doing. Then his wife had closed the laptop.
‘You know,’ she said, ‘we don’t have to keep talking about it for another ten years.’
Vivien had laughed.
But he was still thinking about it on Monday.

‘Germany?’ Hubert repeated.
Vivien nodded.
‘We’ve been talking about it in strategy meetings for years. We’ve got a few customers there, but they’re mostly subsidiaries of French groups. We haven’t really learnt how to sell our ERP to German companies yet.’
‘That’s true.’
‘And everyone says we can go when the product is ready. Except we’ll never know what’s missing until we start having proper conversations with companies there.’
Hubert was listening closely.
‘You’d like to lead the market-development work?’

‘Yes. Not by pretending I’ve become an expert on the German market overnight. I know our product, our constraints, our teams and how customers implement it. I know how to bring people together around a complicated problem. I’ll have to learn the rest.’

‘Have you ever worked in sales?’

‘Not like Julie or Philippe. And I’ve no intention of telling them how to do their jobs. I’d like to work with them, with Implementation and with our business specialists to understand what German companies need, then find out whether we can meet those needs.’

Vivien hesitated.

‘My wife is German. We’ve been talking about living there for a while. It would be a chance to do something I genuinely want to do without leaving Malorean.’

Hubert was quiet for a moment.

‘You know this isn’t just a translation project?’

‘Of course. The ERP deals with rules and processes that vary between countries. Translating the screens isn’t enough. We need to know what we can actually promise, and what we’ll need to build, adapt or refuse.’

‘And we could get it wrong.’

‘We probably will get some things wrong. I’d rather find out with customers than after investing a year in features nobody wants.’

Hubert looked at his project list.

For years, Germany had been an ambition for later. First they had to finish overdue work, hire more developers, free up capacity to adapt the ERP, prepare demonstrations and train the teams.

Some of those difficulties remained. But several had changed in scale.

Preparing a demonstration took less effort. Variations on the product could be tested without routinely involving developers. Support and Pre-sales found it easier to produce documents and share knowledge. Developers had more capacity to explore limited ideas.

That did not mean they had run out of work.

It meant they could choose to put some of their energy into something other than working through the queue of requests from France.

‘Put together a proposal for a first phase,’ Hubert said. ‘Not a promise that we’ll enter the market. Meetings, the questions we want answered, the skills you’ll need and a way to decide whether to continue.’

Vivien smiled.

‘A Mini PRD for Germany?’

‘Don’t make fun of me. If we can frame a technical problem that way, we can at least try to spell out the assumptions behind a market project.’

‘All right. But I want to keep the right to discover we were wrong.’

‘I’m counting on it.’

The following week, Hubert brought Alice, Bob, Vivien, Julie and Philippe together.

He presented the proposal without starting by asking how many features they would need to develop.

‘We now have a capability we didn’t have two months ago. I’d like us to devote some of it to understanding what a genuine presence in the German market would involve.’

Julie responded first.

‘I can already prepare demos in German. We could start contacting prospects fairly quickly too. With AI, we can adapt the materials in a few hours.’

‘The materials, perhaps,’ Philippe said. ‘But a German business leader buying an ERP won’t choose us just because our presentation is in their language.’

Julie folded her arms.

‘I didn’t say they would.’

‘I know. I’m just saying we need to understand how they choose suppliers, whom they consult and what kind of trust they need before buying. We don’t know that market well enough yet.’

Vivien stepped in before the conversation turned into an argument about tools.

‘That’s exactly what I want to learn. Julie, we’ll need your ability to prepare approaches and demos quickly. Philippe,

we'll need your experience with major accounts and conversations that can't be settled by a nice-looking screen. And our implementation specialists know the questions customers sometimes discover too late.'

Alice spoke.

'On the Product side, we'll need to be very clear about our limitations. We can explore quickly, but we can't promise regulatory features we haven't validated.'

Bob nodded.

'We'll also have to choose which kinds of company we want to understand first. Germany isn't one type of customer. If we try to serve everyone, we'll repeat the mistakes we've learnt to avoid here.'

Hubert looked at the faces on his screen.

He had asked Vivien to propose an approach. Nobody had asked him to set up a permanent team immediately, reserve the whole development department or promise a commercial launch date.

Yet several people had already begun identifying how they could contribute.

That had become a habit at Malorean.

'Very well,' Hubert said. 'Here's my decision: we'll fund an initial discovery phase. Vivien owns it. It should tell us which problems German companies have that we can already solve, which require adaptation and which we don't want to take on. Then we'll choose whether to invest further.'

He added:
‘And nobody gets assigned to this project simply because a tool has saved them time. We’ll look at what each person can genuinely take on without neglecting the customers we already serve.’
Vivien made a note of the last point. He had spent enough years managing schedules to know that capacity freed up on paper was not always available in practice.

Over the following days, he spoke to everyone in Pre-sales and Implementation.
Some of their work had changed since demos could be prepared from reusable configurations and customer-meeting summaries were easier to produce. Two people explained that they now spent less time recreating the same materials and would like to take part in the first conversations in Germany.
‘I can help you prepare questions about implementation,’ one offered. ‘And join a few meetings remotely.’
‘What about your French customers?’ Vivien asked.
‘I can’t drop everything. But I can free up time if we put it in the plan.’
Another colleague, already busy with several difficult implementations, told him they had no capacity at all.
Vivien didn’t press the point.
They drew up a provisional arrangement: a few slots reserved for understanding German companies’ expectations, meetings

with subsidiaries that were already customers where useful, and no promise of permanent staffing before they had measured the real effort.

The time saved had not magically turned into growth. They still had to choose what to do with it and resist immediately filling every recovered hour with yesterday's requests.

That evening, Alice received a message from Lucas.

I saw the Germany project. We've already got a configuration mechanism that could help us test some differences in processes. I can prepare an exploratory environment without changing the standard product. Would that be useful?

She read the message again.

Two months earlier, a similar proposal would probably have prompted a discussion about developers waiting to be assigned work. Now the first question was simpler: did it serve the strategy, and within what boundaries could he proceed?

Alice opened the context for the first phase of the Germany project. Vivien had already filled it with assumptions, accountable people, limits and open questions.

She replied:

Yes, if you agree the scope with Vivien and we don't turn a business assumption into standard product behaviour without approval. Start with what we want to learn, not what we know how to build.

Lucas replied almost immediately.

Understood. I'll prepare a Mini PRD.

Alice smiled.

She remembered the Friday when he had shown them a leave-management module built in two days. Everyone had tried to understand how he could produce so much, so quickly.

That evening, nobody was surprised any more that he could build a demo quickly.

The question was why they would need one.

Vivien ended his day by showing his wife the dates of the first meetings planned with companies in Germany.

He had signed no contracts yet. Malorean had opened no office, nor adapted its ERP to every requirement of the market. They would need to discover local ways of working, verify legal and regulatory constraints, and learn how to win the trust of companies that did not know them.

There were far more unknowns than certainties.

'Does it scare you?' his wife asked.

'A little.'

'You always seemed so sure of yourself when you were coordinating your teams.'

Vivien smiled.

'I'd spent years learning how to do that job.'

'And now?'

He looked at the document he had begun drafting for his first visit.

Questions. Hypotheses. Names of people to meet. An initial list of things he did not know.

‘Now,’ he replied, ‘I’ve got something to learn again.’

She came over to look at the screen.

‘And what about us?’

‘We can start talking seriously about moving.’

She put her hand on his shoulder.

Vivien thought about his old release board, his Monday meetings and all the work that had held the projects together. He had loved it. He did not regret those years.

Nor did he regret accepting that the teams no longer needed him in quite the same way.

On his screen, the first meeting in Germany was confirmed.

For Malorean, this was not yet a conquest or a success.

It was a beginning.

12 — Conclusion

More fluid roles, clearer responsibilities

Job titles no longer define exactly what each person can do. A developer can explore a customer need, a product manager can build a solution, and a salesperson can create a prototype. AI opens up activities that were once the preserve of other specialisms. That does not make their expertise interchangeable.

We are no longer responsible only for the work we carry out ourselves. We can take ownership of a problem that spans several functions, while the people accountable for each domain — the Guarantors — safeguard its coherence. The ability to act can move between roles; responsibility must not disappear in the process.

There are two risks: continuing to say, ‘That’s not my job’, or assuming, ‘If I can do it, I can make the decision on my own.’ Neither defending professional boundaries nor sweeping them away is enough to organise work.

Autonomy depends on shared context

Greater freedom only makes sense when everyone has the information they need to exercise judgement. Without shared context, a decision can look right in isolation while conflicting with a wider need, an earlier decision or an important constraint.

Documentation is therefore not about piling up meeting notes. It means making the problem, the objective, the assumptions, the reasons behind decisions, the limits and the rules for escalation accessible. When these things are explicit, people can act without having to track down whoever holds the information at every step. Context becomes a decision-making tool, for people and agents alike.

This is where the Guarantors' role changes. They do not inspect every initiative. They make the boundaries understandable, protect consistency and step in when a decision goes beyond those boundaries. Without safeguards, freedom can descend into chaos; if every decision must be sent upwards for approval, freedom is little more than a promise.

Management is changing too. Some monitoring and coordination can be handled differently. But people still need support, experience still needs to be passed on, conflicts still need resolving and difficult decisions still need to be made. Those responsibilities do not vanish when the tasks that once gave them shape change.

Doing more of what matters

It will be tempting to measure this transformation by the number of features built or hours saved. AI really can help us produce more. But immediately filling the time we save with new tasks would merely speed up the old way of working without questioning its purpose.

We could instead use that extra capacity to understand needs better, test our assumptions, improve what already exists and abandon work that creates no value. None of this happens automatically. We have to decide where to invest the attention and resources we free up.

At Malorean Corp., Vivien discovers more than the fact that his old work now takes up less of his time. He discovers that he can put his experience to work on a different problem, with no guarantee of success. Progress lies not only in what has been automated, but in what the change makes possible.

At the beginning of the story, everyone was astonished by how quickly Lucas could build software. By the end, the question has become: why build it, for whom, and with what consequences?

The point of AI is not to make us do ever more. It is to give us the means to do more of what truly matters.