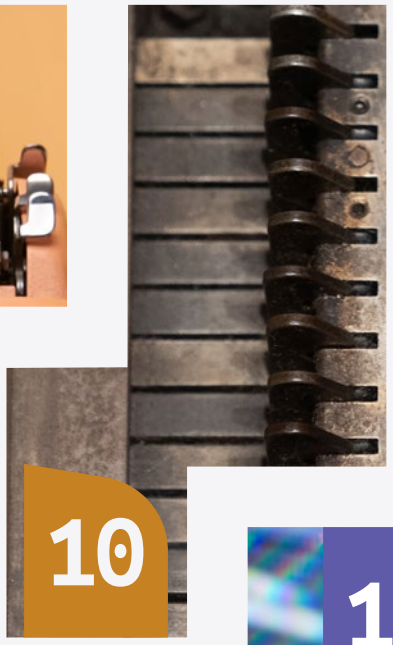
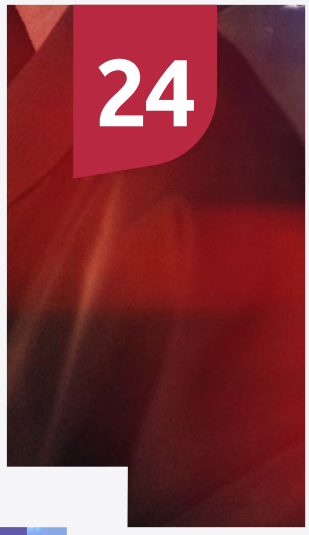
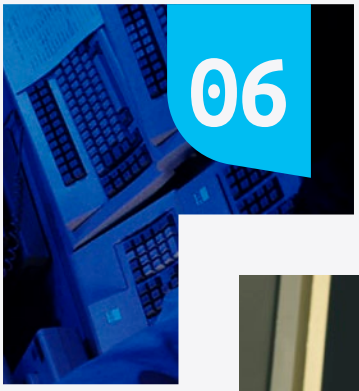
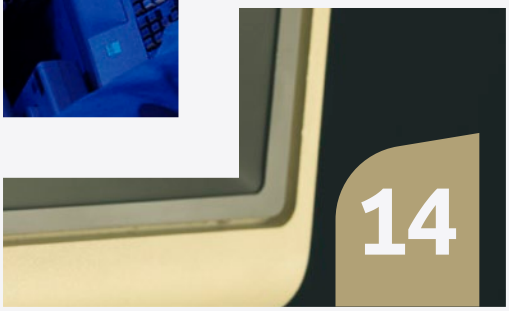

sogeti labs
Part of Capgemini
Vibe coding
Report 2 of the Autopilot series

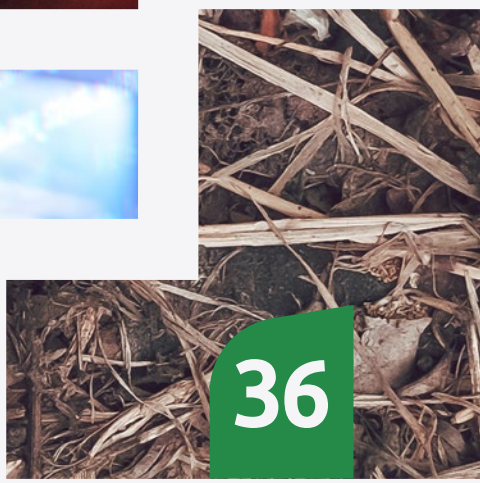
Contents

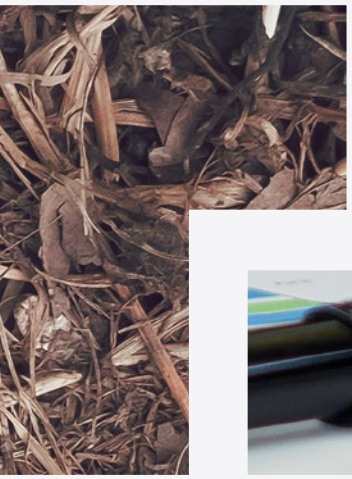


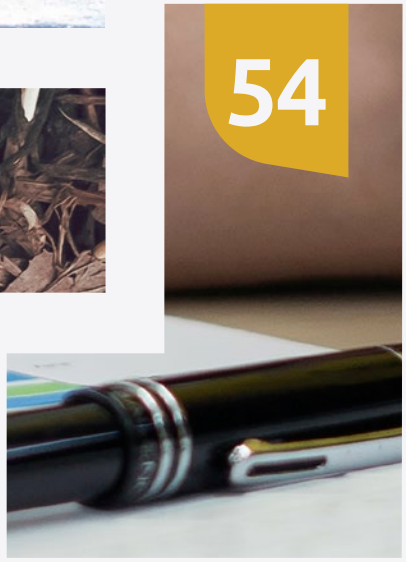


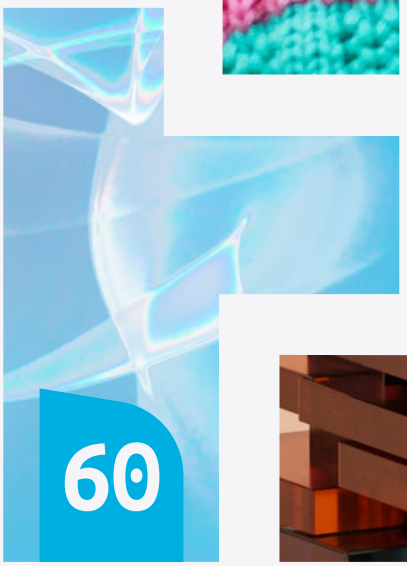


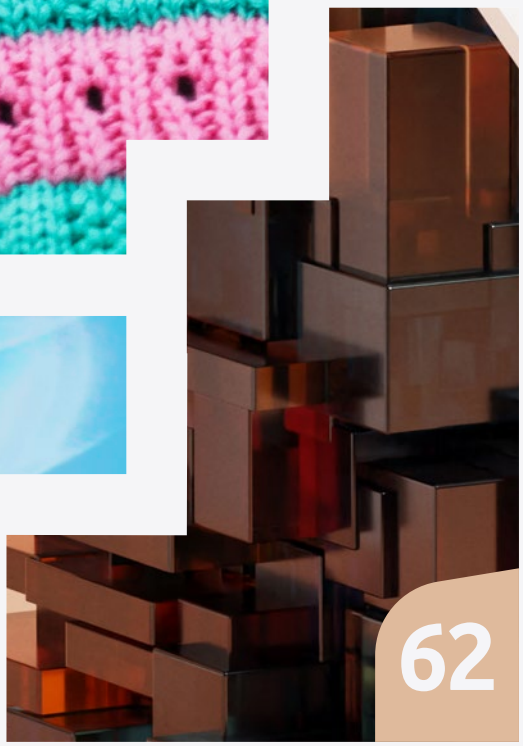




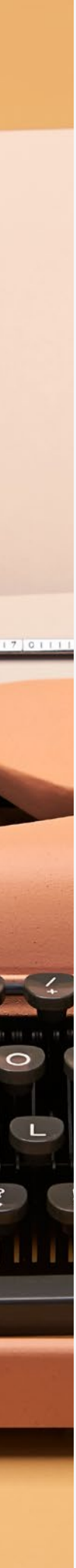








Introduction
Vibe coding



A quiet but significant transformation is taking place in the way work is done. Not only our tools are changing, but thinking, speaking and creating are merging together. The rise of *vibe coding*, an AI-driven practice that turns intentions expressed in natural language into functioning designs or code, shows how technology is reshaping the very foundations of work.

What began as an experimental tool for developers is now finding its way into many other areas, including marketing, human resource management, consultancy and decision-making. In some situations, a single prompt can produce results that once required several people to plan or execute. This changes not only how work is carried out, but also how it is coordinated, how roles are divided and what it means to be productive.

In this report, we examine how this development is unfolding through three scenarios for the future of work. We explore how organisations may be moving towards a form of *vibe architecture*: an infrastructure of AI systems that do more than merely assist. They contribute original ideas, help build software and sometimes even initiate decisions. Alongside concrete examples, we address the organisational and ethical questions emerging from this shift.

For executives and policymakers, it is important not to regard this movement as a temporary fad, but as a sign of deeper structural change. This report does not provide a fixed recipe for success. It suggests concepts, questions and reference points that can help anticipate what lies ahead.

The manner in which this report was created, is an illustration of the shift towards a new world of work. It was not written in a conventional, linear way, but through iterating interactions with language models. In fragments that happened in the car on the way to work, between appointments or at the kitchen table. By asking questions, testing ideas and thinking aloud in an ongoing dialogue with AI, a text emerged, shaped by *vibe writing*. A form of writing where human intention and generated language gradually merge into content. What you read here is therefore not only a description of a possible future, but an expression of it.

Chapter 1
The birth of software engineering

When a group of engineers, scientists and military officials gathered in Garmisch-Partenkirchen in 1968, the term *software engineering* was minted. It did not refer to an established field, but was rather an urgent call to create one. Participants spoke of a *software crisis*: systems were expanding faster than our ability to understand them. Large-scale projects were spiralling out of control – sometimes with life-threatening consequences – because no one could fully grasp how the rapidly expanding software systems functioned. The meeting marked a turning point. From then on, software was to be designed with the same engineering rigour and reliability as other complex systems like bridges or aircraft.

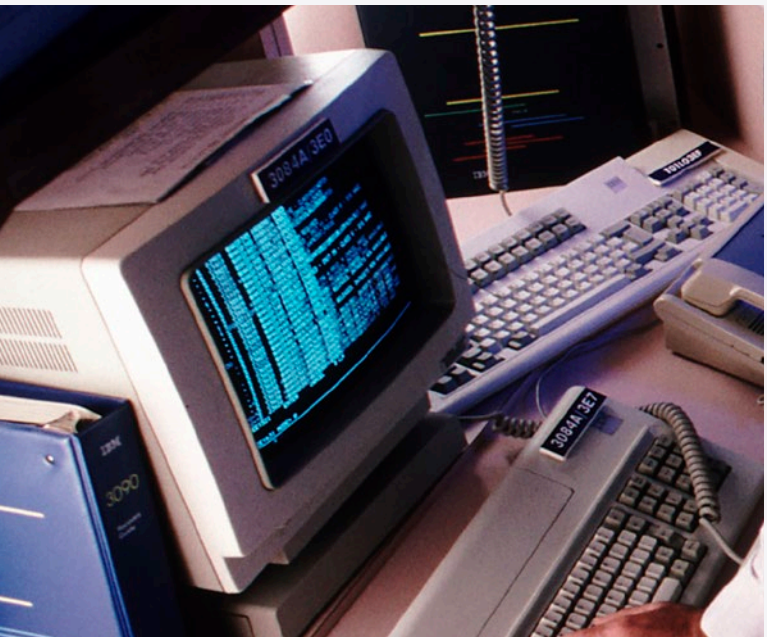
To understand how radical this shift was, we must return to the early decades of computing. Well into the 1950s, the focus was entirely on hardware: tubes, transistors and circuits. Software was a secondary concern, merely a way to operate the machine. Programmers were pioneers who worked with stacks of punch cards, recording logic hole by hole. Documentation was rare, standardisation almost non-existent. The craft resembled improvisation more than engineering – an art form shaped by intuition and trial and error.

As systems grew in scale and complexity, this approach became untenable. In the United States, the Air Force sought to build a national command system, NASA required software to guide the Apollo rockets, and across Europe, large automation projects for railways, telephony and defence were underway. Failures became increasingly frequent: deadlines slipped, budgets ballooned, and – once completed – systems proved almost impossible to maintain. Software had become invisible infrastructure – its fragility revealed only when it failed.

This accumulation of failures prompted a search for structure. In the decades that followed, successive methodological waves emerged: the waterfall model with its linear progression from analysis to implementation, and later agile and scrum practices that emphasised short iterations and continuous feedback. The 1990s introduced maturity models such as CMMI, which sought to measure and standardise the development process. Each of these approaches was an attempt to gain control over something that, by its very nature, resisted it. Software engineering gradually became more than a technical discipline. It evolved into a cultural and organisational endeavour, the art of collectively constructing something invisible until it came to life.

That legacy continues to shape the present. Behind every major system today, from medical devices and financial infrastructures to air traffic control, lies a tradition of norms, standards and control mechanisms created to contain complexity. Software matured through discipline, structure and shared methods.

And yet, we are once again at a turning point. In the past, the concern was that there was too little order; today, the concern is that order is being eclipsed by convenience.



AI models can now generate code from prompts written in natural language. You no longer need to be a programmer to create applications. The developer's role is shifting toward that of a prompter. Code is no longer written line by line but orchestrated – already there in potential, waiting for the right instruction to bring it to life. What once demanded deep technical expertise now relies mainly on direction, tone, and intention. Those who craft prompts need not grasp the underlying logic to produce complex results.

The term *vibe coding* was introduced in early 2025 by Andrej Karpathy and quickly gained traction in the tech world. Its promise was alluring: simply describe your intention in natural language – the *vibe* of what you want – and AI will create it for you. No syntax, no architecture diagrams, no endless refactoring, just a sense of purpose, a direction, and a prompt. The rest is generated.

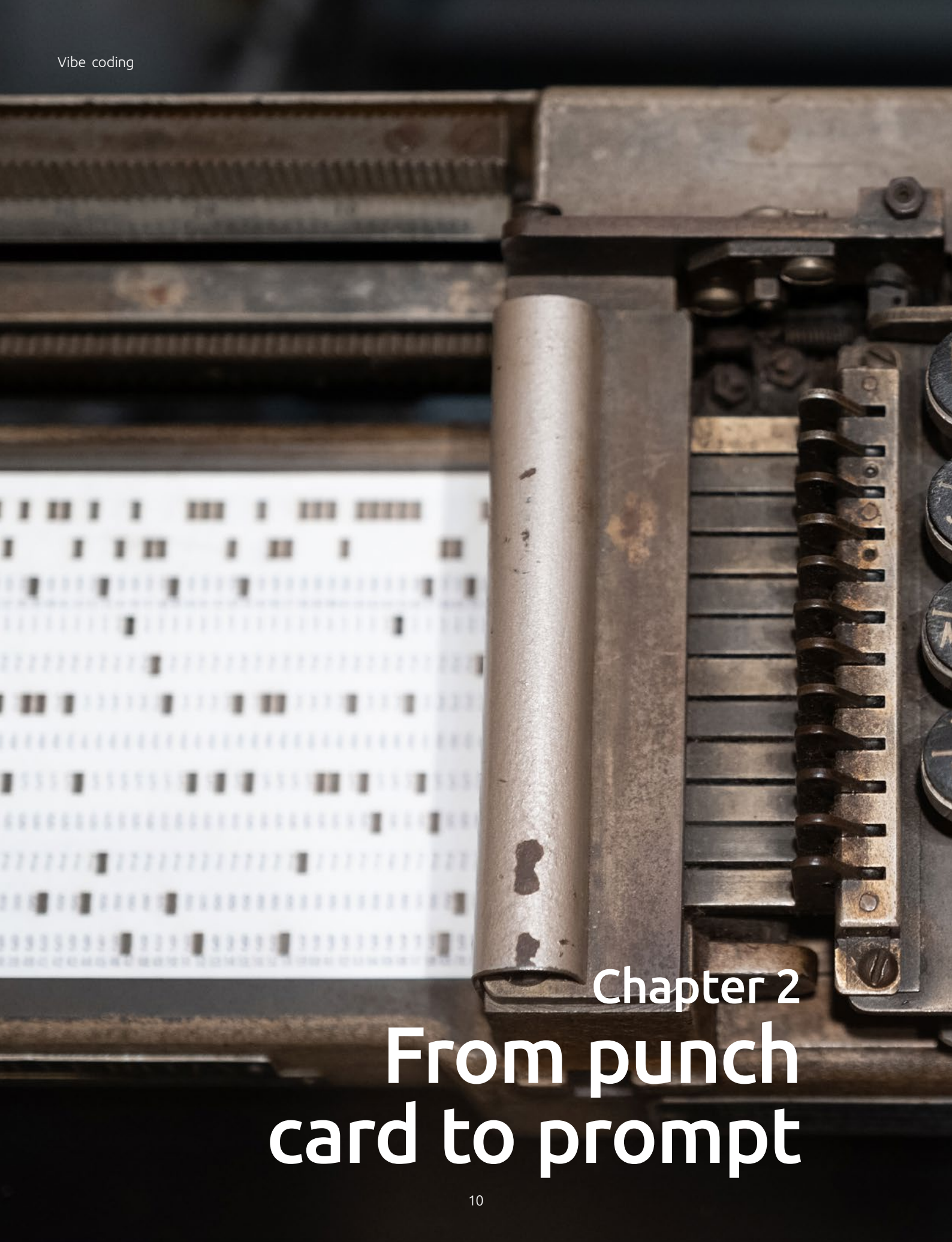
Although vibe coding focuses primarily on generating source code, its influence reaches much further. Software engineering has always been about more than writing code; it also involves testing, documenting, modelling, maintaining, collaborating and evaluating. Yet it is precisely that first creative moment that transforms the rest of the process. Where previously every choice was explicit and open to discussion, it now disappears into the invisible logic of a prompt. The apparent simplicity at the front end conceals a growing complexity at the back end, and it is there that new blind spots begin to appear.

Old concerns therefore lie hidden beneath new promises. Who still understands the code that is being generated? Who guarantees its security, scalability and maintenance? What happens to craftsmanship when the craft itself dissolves into prompts? And, perhaps most importantly, how do we maintain control over an infrastructure that we no longer explicitly design?

These questions are not rhetorical. They strike at the heart of what it means to use technology within complex organisations. Vibe coding exposes that tension once more: between speed and understanding, between innovation and responsibility, between what is possible and what remains desirable. This report seeks ways to interpret and navigate that tension. It examines the practice in which vibe coding takes place, the risks it creates, the governance it requires, and its broader implications for organisations, work and culture.

What was recognised in 1968 as a crisis of improvisation now reappears in a reversed form. Then, the challenge was the need for structure; today, it is the need for understanding. Vibe coding may represent the most natural continuation of the digital revolution, yet precisely for that reason it is essential to ask what we wish to preserve.

In the chapters that follow, we approach this as an exploration of a landscape where software is increasingly written by language itself, and where humans must rediscover what it means to keep thinking.



Chapter 2
From punch card to prompt

The long evolution towards vibe coding

Every technology begins with a promise and a translation. The promise is that machines will lighten, accelerate or enrich our work. The translation is the question of how we convey human intentions to a non-human system.

In the early days of the digital revolution, this translation was literally punched into cardboard. Programmers used punch cards, each hole representing an instruction and each card a line of code. The logic was tangible, rigid and linear. A single mistake meant re-punching the entire card, and anyone who wanted to understand a programme had to read it mechanically, card by card and line by line.

Then came higher programming languages: first assembler, then Fortran, Pascal and C. These were languages the machine could understand, but that already reflected human patterns of thought. Programming remained a demanding, abstract profession, yet the thinking behind it became more intuitive. It was no longer expressed in zeros and ones, but in structures, functions and logical blocks. Humans gained control, even as the distance to the underlying hardware increased.

The next leap was the graphical user interface. Windows, the mouse, drag-and-drop functions and visual metaphors transformed interaction into image, action and movement. Software became more user-friendly, while its inner workings grew more complex. The blue screen of death (reflecting a serious system error) was the final reminder of this hidden complexity. The act of translation had shifted from code to click.

In the 2010s and 2020s, *no-code* and *low-code* platforms emerged, adding abstraction upon abstraction. Platforms such as Bubble, OutSystems and Mendix promised that software could be built without writing code. It was a democratisation of development, yet still based on templates, visual interfaces and pre-programmed components. The translation had been simplified, but it had not disappeared.

What's next? We speak.

With the advent of large language models such as ChatGPT, Claude and Mistral, a new threshold has been crossed. Artificial intelligence is not only taking over the writing of code, but also its *understanding*. Users no longer need to know syntax, study API documentation or install libraries. An intention is enough. A wish. A vibe.

'Build a dashboard that retrieves real-time temperature measurements from this API.'

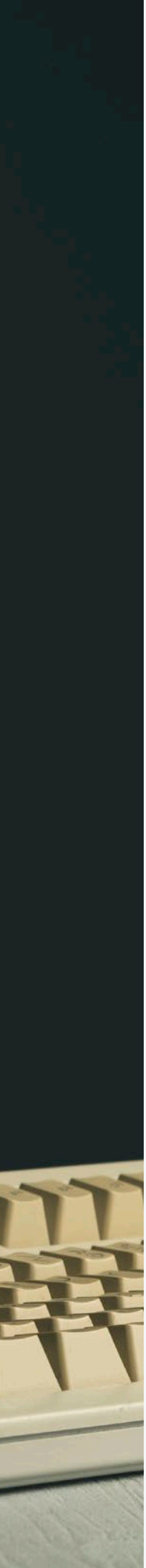
The machine understands the intent and does the required coding, delivering a working application.

This development is radical, but not unexpected. It follows the line of thought of early computing visionaries such as Douglas Engelbart, the inventor of the computer mouse, and Alan Kay, who already in the 1970s envisioned systems that ‘think like people talk’. The goal was never to make technology human, but to make humans freer: freer to focus on ideas, patterns and design instead of syntax, error messages and repetitive code.

Yet the more natural the interface becomes, the more unnatural the risk. When a system appears to understand your intention effortlessly, it is easy to forget that it can also be wrong. The prompt conceals complexity. The illusion of understanding grows. The black box expands and the distance from logic increases.

In short, vibe coding is not a passing fad but the next stage in a long evolution of interfaces: from punch card to prompt, from syntax to semantics, from instruction to intention. It is a form of coding in which writing disappears, while thinking becomes more essential than ever. For that reason alone, this revolution deserves the same critical attention as the previous ones.

Chapter 3
The spirit of Garmisch-Partenkirchen



What the 1968 NATO Software Engineering Conference still teaches us

As mentioned before, in September 1968, a group of sixty scientists, engineers and military officials travelled to the mountain village of Garmisch-Partenkirchen in southern Germany. The air was clear, the mountains were calm, but the reason for their visit was far from peaceful. Something was going wrong with software. Large-scale automation projects, many of them of strategic or military importance, were failing in new and alarming ways. They were running late, becoming stuck or spiralling out of control for reasons that no one could fully explain. The problem wasn't the hardware – it was the software itself, which had begun to behave like a living organism: unpredictable, uncontrollable and ultimately impossible to fully understand. Systems kept growing, even as our grasp of them faded. Before long, people were talking about a software crisis.

The conference, organised under the auspices of NATO, was meant to be a response to that crisis. It did not offer technical quick fixes or project management advice, but a fundamental proposal: software development should become an engineering discipline. The organisers, among them computer scientist Brian Randell and mathematician and Turing Award winner Peter Naur, deliberately introduced a new term: *software engineering*. They hoped the profession would begin to act like a fully-fledged branch of engineering, with the same discipline, methodology and quality standards as civil, aeronautical or electrical engineering. Without such standards, they warned, software would not be a solution but a new kind of weapon of chaos.

Their concern extended beyond code alone. It encompassed the entire process of design, documentation, maintenance, collaboration and testing: a discipline in which systems had to remain understandable, not only to their creators but also to others, and over time.

That choice was provocative. In the foreword to the conference report, the organisers admitted that the term was more a normative aspiration than a descriptive one. 'We use the term not because it is an established reality,' they wrote, 'but because it is a goal we must strive for.'

The idea of software as an engineering product was still revolutionary in 1968. Until then, software had been regarded mainly as a craft, a form of improvisation, the domain of brilliant soloists who wrote code as a composer writes a score. Yet that model no longer worked in a world where systems were becoming more complex and where companies, governments and militaries were increasingly dependent on invisible and opaque logic.

The problems discussed at the conference sound strikingly familiar in 2025. Projects failed not only because of technical flaws, but above all because of a lack of understanding. The logic of the system exceeded the mental capacity of its creators. Programmers could not read another one's work. No one knew where the errors came from, and when something functioned correctly, no one could fully explain why. Software lacked transparency, testability and memory; it only had behaviour – and that behaviour was becoming harder to control.

What united the conference participants was a deep sense of unease. Their concern was not only about the growing complexity of systems, but also about the cognitive limits of the human mind. It became clear that people were not designed to oversee thousands of lines of code, let alone their interactions, dependencies and side effects. Dutch computer scientist Edsger Dijkstra, one of the most influential voices of his time, captured this insight in a single sentence:

'The competent programmer is fully aware of the strictly limited size of his own skull.'

That recognition of limitation marked the beginning of a new ethic. It defined a profession in which intellect was used to design systems that did not exceed the boundaries of human comprehension. It demanded structure, standards and formality. Programming was no longer to be treated as an art form, but as a responsibility.

It is this ethic, born in 1968, that now finds itself under pressure once again. More than fifty years later, we are facing another technological turning point. Software is once again becoming more complex, though this time not through human effort alone. Code is now generated in seconds by large language models, based on nothing more than a few sentences in natural language, typed or spoken. What was still a manual and controlled process in 1968 has become an invisible collaboration between humans and machines in 2025. The developer no longer writes code but articulates intentions. The machine fills in the blanks. This is *vibe coding*.

Yet vibe coding touches only one link in the software process: the expressive translation of an idea into working logic. The broader principles of software engineering, such as testability, scalability, maintenance, validation and version control, remain as vital as ever in this new practice. That makes the question of how AI-generated logic fits within these principles even more urgent.

At first glance, vibe coding seems to resolve several of the issues already identified in Garmisch-Partenkirchen. It makes software development more accessible, faster and more iterative. High-quality and poor-quality code alike will circulate more rapidly around the world. Yet on closer inspection, the old concerns reappear in a new form. Who can still understand the generated code? Who can guarantee its security, robustness and maintainability? And who is responsible when systems begin to behave in unforeseen ways?

Instead of the hand-crafted spaghetti code of the 1960s, we are now confronted with machine-generated logic whose origins are diffuse and whose effects are difficult to anticipate. The black box has not become smaller; it has only become more refined.

The final report of the conference contains a sentence that, from today's perspective, feels more urgent than ever:

'We should consider software to be a product to be engineered rather than a craft to be performed.'

What happens when code is no longer engineered but generated, effectively placing it outside the processes of explicit design, testing, and understanding?

In that case, it may not be the software itself that requires re-evaluation, but our relationship to it. The spirit of Garmisch-Partenkirchen then appears not as a historical moment, but as a moral compass – not to bring us back to the past, but to help us maintain direction in a future that is increasingly writing its own logic.

Chapter 4
Vibe coding in practice

From promise to collision

It started as a joke, a funny tweet from Andrej Karpathy in early 2025: 'The hottest new programming language is English.' At first, it seemed a playful nod to the rise of large language models capable of writing entire functions on command. But within weeks, it became clear that the remark captured something deeper. The term *vibe coding*, which Karpathy introduced around the same time, quickly struck a chord within the developer community. Prompting became coding. Intention became implementation. The *vibe*, the feeling, the direction and the overall idea, proved sufficient to generate working software.

The revolution unfolded with remarkable speed. What began as an experimental hobby for prompt enthusiasts rapidly evolved into a mature workflow. New tools appeared almost overnight, including *Mistral Code, Replit Ghostwriter, Cursor, v0.dev* and *CodeRabbit*. Each offered the same promise: users only had to tell the machine what they wanted, not how to achieve it. A single sentence could be enough. A rough sketch, a half-formed idea or a general instruction was sufficient for the machine to complete, elaborate and translate. While Simon Sinek's *Golden Circle* encouraged organisations to start with the 'why', vibe coding seems to invert that logic. The user begins with the 'what', the machine generates the 'how', and the 'why' becomes blurred. This inversion puts pressure on human autonomy. If the machine performs the work, where does the creator's compass remain?

This way of working differs fundamentally from the classical software process. There are no lengthy requirements documents, no wireframes and no detailed specifications. Code emerges in real time, in an ongoing dialogue between user and model. The workflow is iterative, associative and improvisational. You try something, test it, reformulate your prompt and let the machine continue. The human role shifts from creator to curator, from building to guiding, from writing to refining.

This approach works remarkably well for small projects. Hobbyists can build a recipe app, a scraper or a chatbot within an hour. Start-ups use vibe coding to launch Minimum Viable Products (MVPs) faster than ever before. According to a report[1] by *TechCrunch*, during the winter programme of Y Combinator, a well-known American start-up accelerator, more than a quarter of young technology companies had 90 per cent or more of their software code automatically generated by AI rather than written manually by programmers. Early surveys support this picture. Platforms such as Replit report that organisations using AI in development can deliver applications up to six times faster, while Y Combinator partners even speak of a hundredfold acceleration at some start-ups.

Such examples show how some organisations are now deploying complete MVPs created entirely by AI, at speeds that were once unthinkable. Yet this acceleration often comes with a higher risk of bugs and vulnerabilities that only become apparent later.

In *The New York Times* article,[2] journalist Kevin Roose described how he built an app in a single evening that generated meal suggestions based on the contents of his fridge. 'I simply typed in: "Create a web app that analyses my shopping list and makes suggestions for lunch dishes." Within minutes, I had a working prototype.' Only later did he discover that the model had invented ingredients and fabricated e-commerce reviews. The interface functioned perfectly, but the truth did not.

More advanced applications are also beginning to appear. In *DEV Community,* a developer described how he built a web app called *Research AI Assistant*,[3] which converts scientific papers into podcasts, visual slides and TikTok summaries. The entire application was created using vibe coding tools, without the developer having any in-depth experience in backend development.

[1] TechCrunch. (2025, 30 June). *A comprehensive list of 2025 tech layoffs*. https://techcrunch.com

[2] Roose, K. (2025, 27 February). Not a Coder? With A.I., Just Having an Idea Can Be Enough. *The New York Times*. https://www.nytimes.com/2025/02/27/technology/personaltech/vibecoding-ai-software-programming.html

[3] Deb, A. (2024, November 9). *Research AI Assistant Application – RAG*. DEV Community. https://dev.to/ayushdeveloper/research-ai-assistant-application-rag-3d0m

The enthusiasm is understandable. Vibe coding democratises a field that has long been closed to non-specialists. It lowers the threshold for creation, accelerates prototyping and shortens the distance between idea and implementation. For the creative industry, education, start-ups and even internal tools within larger organisations, it offers unprecedented speed and flexibility. Non-technical founders are seizing the opportunity to prototype their ideas directly, and challenges such as '12 projects in 12 months' or even '24 projects in 24 hours' are circulating online, turning vibe coding into a cultural phenomenon that makes the promise of democratisation tangible.

An even more revealing sign of this cultural shift is the emergence of Chad IDE, presented by its makers as 'the brainrot IDE'. The premise is telling: while the AI generates code, developers can fill the waiting time inside the same environment with minigames, short-form video feeds and other forms of stimulation. What began as a workflow optimisation thus mutates into something more cultural: an IDE that no longer only supports programming, but also manages distraction. Chad IDE may sound like satire, yet precisely for that reason it exposes a deeper truth about vibe coding. When software generation becomes fast enough to feel magical, but slow enough to leave micro-gaps of boredom, even waiting itself becomes a design problem. But once the magic fades, those micro-gaps give way to a more fundamental question: what exactly are users building, and how well do they understand it?

Yet that promise increasingly collides with a more stubborn reality. While the tools may feel intuitive, the underlying code often is not. Much vibe-generated code works well on the surface but contains vulnerabilities, inconsistencies or hidden dependencies. Debugging becomes difficult because you cannot debug what you do not understand. Scalability poses another challenge. What happens when an app suddenly attracts millions of users? Who rewrites the code that no one actually wrote?

There is also the question of ownership. Who is responsible for AI-generated code? The person who wrote the prompt, the engineer who reviewed it, the organisation that deployed it, or the creator of the model that produced it? From a legal perspective, the answer remains uncertain. Practically, there is not much more clarity. What functions smoothly today may be incomprehensible tomorrow, especially when no one knows where the logic originated.

In practice, this marks a fundamental shift in how we think about software quality. The product is no longer the result of human precision, but of stochastic probability. The model makes a guess – a well-informed one, based on billions of parameters – yet still a guess. And that guess is often no longer verified. The user sees that it works and moves on, letting the illusion of correctness outweigh the need for real understanding.

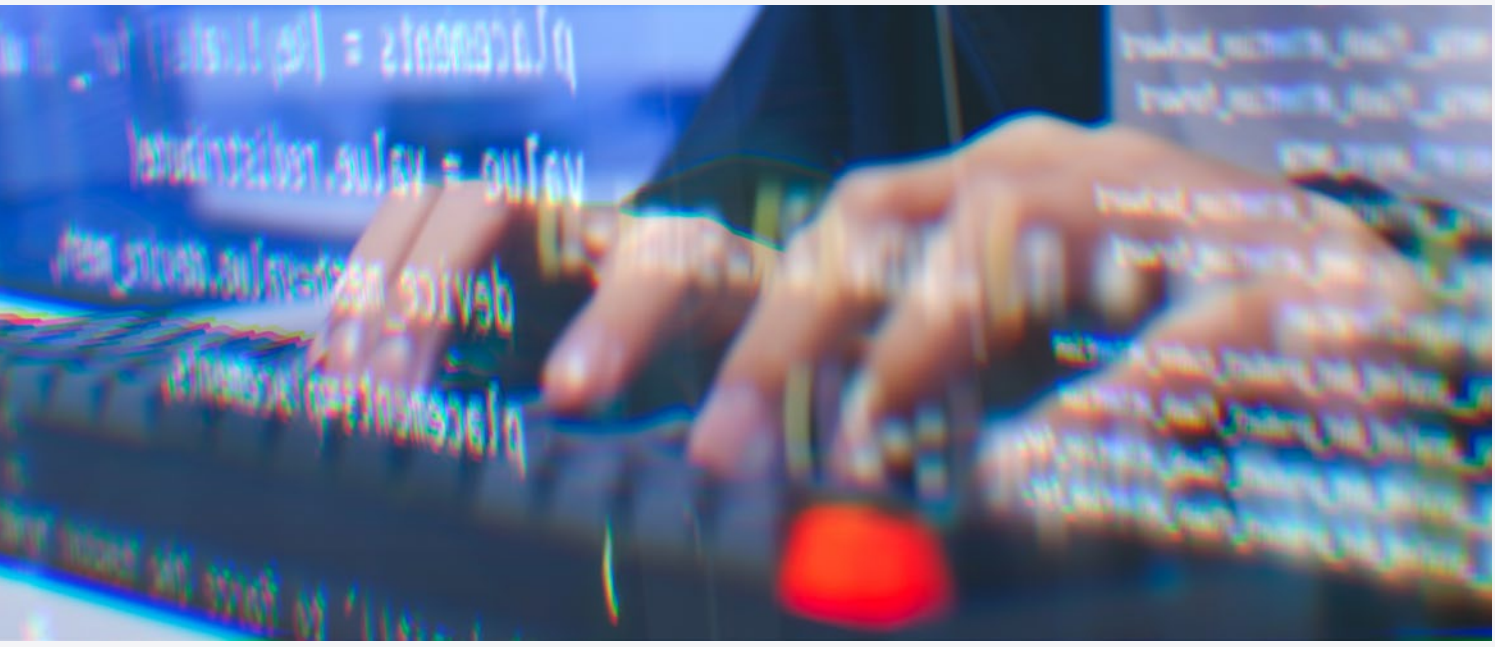
For CIOs and CTOs who are considering integrating vibe coding into their organisations, the appeal is great, but so is the risk. How do you evaluate software when its

logic lies beyond your cognitive reach? How do you maintain a system that no one truly understands? How do you build on a code base that was not written, but generated?

The clash between promise and reality is not fatal, but it does demand conscious design – not only of tools, but also of organisational structures, training programmes and governance practices. Vibe coding is not a passing trend; it is a new language, and every new language requires grammar, standards and collective interpretation.

Anyone who takes this development seriously will recognise that it is not only about code, but also about collaboration. Vibe coding compels us to reconsider the relationship between humans and machines: who sets the direction, who provides the meaning, and who learns from whom? This shift exposes a distinction that is rarely articulated, yet crucial for understanding what vibe coding truly represents.

Some applications of AI literally take over the work, representing a form of autonomous automation in which humans are largely observers. Others enhance human work, a form of augmentation in which AI becomes a partner in thinking, designing and testing. This distinction was clearly defined in the *Anthropic Economic Index* report[4] on the impact of AI on software development: automation refers to *'AI systems that directly perform the task end to end'*, while augmentation refers to *'AI systems that collaborate with a user to complete a task'*. According to the same analysis, approximately 79 per cent of coding interactions in Claude Code are classified as automation, compared to 49 per cent in Claude.ai. This difference illustrates how the nature of collaboration determines the depth of understanding.



Vibe coding

In the first case, responsibility shifts to the system, while in the second it remains shared. Automation creates convenience; augmentation preserves understanding.

Between these two poles, however, a third state is emerging, described in recent research as *scaffolding*. In this mode, AI creates a temporary framework within which humans can act, learn or design. The machine builds a structure of suggestions, patterns and directions, and humans climb it to develop their own insights. As long as the scaffolding remains visible, we retain perspective. Only when it disappears, when we forget that it was meant to be temporary, does assistance turn into takeover.

The future of software development will likely unfold in the twilight zone between guidance and replacement. The central question is how we can keep humans thinking within an infrastructure that is increasingly self-executing.

Recent surveys among experienced software developers show how precarious that balance has become in practice. In a controlled experiment[5] with the latest generation of generative models, programmers were found to complete their tasks on average 19 per cent more **slowly** with AI than without, even though they estimated their own productivity to be higher. What emerges is not a technical shortcoming, but a cognitive one: the speed of suggestion conceals the slowness of understanding. The machine reduces friction, but not effort. It shifts the work from execution to verification, leaving behind the sensation of acceleration without the experience of progress.

In the chapters that follow, we will examine the risks, organisational implications and strategic perspectives required to integrate vibe coding responsibly into mature development environments. The question is not whether this new mode of software creation will continue, but who will continue to lead the conversation about it.

[4] Anthropic. (2025, 28 April). *Anthropic Economic Index: AI's impact on software development.* Anthropic Research. https://www.anthropic.com/research/impact-software-development

[5] Becker, J., Rush, N., Barnes, B., & Rein, D. (2025, July). *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity.* METR Working Paper. https://arxiv.org/pdf/2507.09089

Chapter 5
Three stories about agentic acceleration

The previous chapters described how vibe coding is evolving from experimental prompt art into a mature development practice. Yet the question remains: what does this look like in everyday reality? How do teams actually learn to build with AI as a partner rather than a tool?

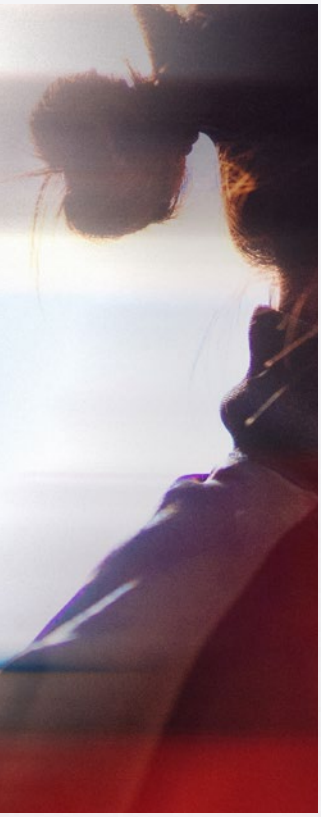
The following three cases from the United States illustrate how this interaction unfolds in practice. They reveal not only technological acceleration, but also a shift in energy, ownership and rhythm of work.

The retailer that found its rhythm

At an American retailer with an annual turnover of 12 billion dollars, software development had long followed predictable patterns. Updates, integrations and internal tools were carefully planned, making them reliable, but slow. The engineering team worked methodically, yet even minor improvements required days of manual testing and checking. When the team decided to experiment with agentic workflows, expectations were modest.

The first few weeks were marked by trial and error. Developers had to learn to collaborate with AI agents capable of generating test cases, detecting errors and writing sections of code. At first, the process felt awkward, as if they were supervising a tireless junior colleague. Gradually, a new dynamic emerged. Less time was lost and focus improved. Tasks that previously required one or two days were now completed within hours.

What surprised the team most was not the speed, but the renewed sense of energy and collaboration. Engineers began helping one another, ideas circulated more freely and the atmosphere shifted. What began as a cautious experiment evolved into a lasting change. The company now applies this approach across multiple product lines – not because it must, but because no one wishes to return to the old way of working.



The manufacturer that turned crisis into momentum

An American manufacturer had spent eight months developing a complex mobile app using a popular low-code platform. Despite sustained effort, the project stalled. Performance was disappointing, integrations failed, and confidence began to fade. With only five weeks of budget remaining, the project leader faced a difficult choice: abandon the effort or try something radically different.

The team decided to start again, this time using a traditional framework supported by agentic development. The atmosphere changed almost immediately. AI agents took over repetitive tasks, built modules and tested integrations, allowing developers to concentrate on structure and coherence.

Not everything went smoothly. There were moments of confusion, instances of over-reliance on AI and periods of manual repair work. Yet the pace and clarity with which the team learned were remarkable. After five weeks, the app was complete. A project that had seemed doomed became a turning point. During the evaluation, it became clear that the greatest achievement was not the product itself, but the way the team had learned to work: faster, calmer and with greater oversight.

The education company that rewrote its own timeline

An educational institution working with an outdated platform faced a different kind of challenge – not to build, but to innovate. The infrastructure was cumbersome, maintenance was expensive and innovation was barely getting off the ground. Management estimated that a complete rebuild would take at least a year, far too long for an organisation that needed to move quickly.

The team decided to experiment with agentic development. The difference was visible within the first week. AI agents analysed the existing code, mapped dependencies and proposed paths for modernisation. Developers were able to focus on design, user experience and architecture. The work felt less like rewriting code and more like collaborating with an additional team that never grew tired.

After six weeks, 70 per cent of the application had been migrated to a modern stack. The result was not only technical but also cultural. The teams saw what became possible when human expertise and AI precision complemented each other. The experience taught them that modernisation does not always require more time, only a different way of using it.

A new construction rhythm

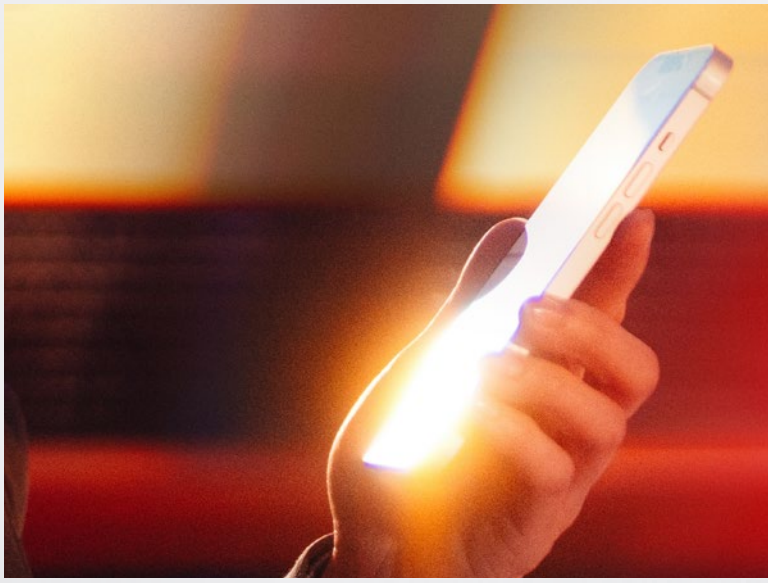
Each of these stories began as an experiment and ended as an insight. What connects them is not only acceleration, but a renewed sense of direction.

Agentic development shortens time while expanding the space for thought. Developers shift from executors to designers of collaboration. Routine fades, curiosity returns.

In these organisations, progress no longer feels like a race against the clock, but like working with time itself. That is the quiet power of agentic acceleration: when humans and AI build together, the work becomes not only faster, but also better, clearer and more meaningful.

These three examples show that the promise of vibe coding lies not only in speed, but also in the emergence of a new rhythm of work in which humans and models strengthen one another. Yet this very success raises a deeper question: what disappears as we accelerate?

In the next chapter, we will explore the other side of this convenience – the risks, dependencies and blind spots that accompany the fluid collaboration between humans and machines.



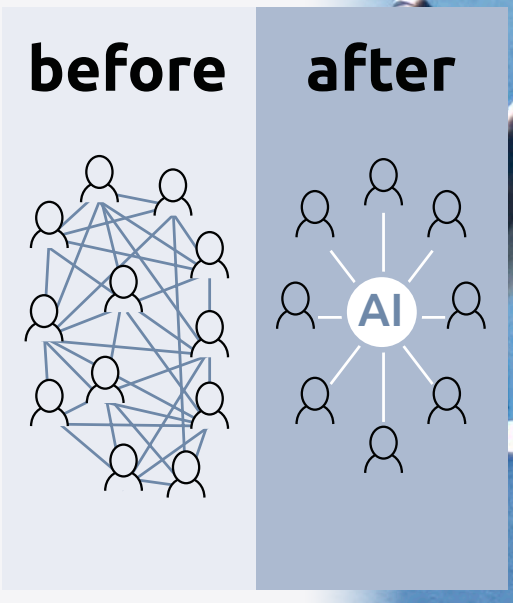
Chapter 6
The downside of convenience

Risks, vulnerabilities and the disappearance of responsibility

The programmer becomes a prompt writer, and the developer becomes the director of a conversation. Where syntax, compiler errors and dependency conflicts once defined daily work, intuition now seems to suffice. You describe what you want, and the machine does the rest. The world of code is turning into a world of language.

Beneath the surface of this convenience, however, lie tensions. While AI can generate code effortlessly, the number of people who truly understand that code is declining. With that loss of understanding, something fundamental also disappears: responsibility. What vibe coding gains in speed, it loses in depth. What it democratises, it often fails to control – and what it enables is not always something we can still undo.

We notice it ourselves every day: we communicate less with our colleagues than before. Not because we are avoiding them, but because we no longer need them for most questions. Where we once walked past a desk or scheduled a meeting, we now open an AI window and receive an answer within seconds. It feels efficient, yet almost imperceptibly, the fabric of collaboration begins to erode.
A simple drawing makes this visible. On the left is the old pattern: a tangle of lines between people, messy but rich, full of serendipitous connections, friction and shared reflection. On the right is the new constellation: everyone directly connected to AI, neatly organised in a 'hub', but without the connections between each other. The noise disappears, but with the noise, so does the dialogue.

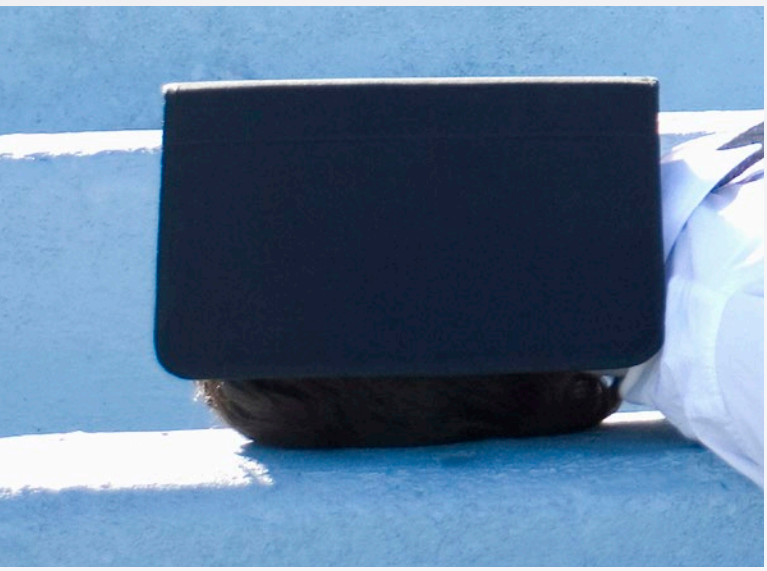


It is precisely these connections that form the space where knowledge is tested, misunderstandings are uncovered, trust is built and responsibility is shared. By choosing convenience, we risk allowing the social fabric of organisations to dissolve into a collection of individuals, each conducting their own private conversation with a machine.

Invisible risks and technical blind spots

One of the greatest risks is invisible complexity. AI-generated code is rarely modular, elegant or documented. It is often sufficient to run, but not to maintain or adapt later. The code 'works', yet no one knows exactly why. This makes it difficult to correct errors, solve scaling problems or train new developers. The code base becomes a kind of syntactic sediment: layer upon layer, built from well-intentioned prompts but lacking structure, consistency and oversight.

Much of this generated code also contains hidden errors. Not because the model is incompetent, but because it operates on probabilities rather than deterministic guarantees. Language models are trained to produce the most probable answer, not necessarily the most secure, scalable or robust one. They reproduce patterns from their training data, including outdated or flawed examples. What appears to be sound logic may in fact rest on obsolete databases, unsafe practices or oversimplified assumptions. A 2025 overview study[6] of vibe coding practices shows that speed and quality do not always align. Several independent analyses point to a clear trade-off between the two: projects are completed significantly faster, yet the generated code more often contains inconsistencies, maintenance issues or hidden errors. The first version of AI-generated code is usually 'fast but flawed': functional, but rarely ready for production without additional quality control. These findings suggest that development teams using vibe coding spend more time on refactoring and quality recovery than in traditional development. The promise of acceleration remains real, but it requires equal attention to testing and maintenance.



However, vibe coding touches only part of what software engineering entails. It demonstrates how code can be produced more quickly, but software engineering is much more than that. It also involves identifying needs, discovering business logic, designing architectures, testing and maintaining systems, and ensuring security and scalability. Within that broader discipline, vibe coding functions as an accelerator in the implementation phase at best, not as a

[6] Fawzy, A., Tahir, A., & Blincoe, K. (2025). *Vibe Coding in Practice: Motivations, Challenges, and a Future Outlook – A Grey Literature Review.* arXiv. https://arxiv.org/abs/2510.00328

solution to the entire task. This contrast is essential, because it is precisely in the space between rapid construction and long-term maintenance that most of the challenges of the profession reside.

The consequences of these cognitive and technical blind spots are already becoming visible. In March 2025, the vibe coding platform Lovable was warned that 170 of its 1,645 generated apps were leaking user data, including names, email addresses, financial information and API keys. The incident sparked a wider wave of criticism. By mid-2025, several blogs had labelled vibe coding 'the worst idea of 2025'[7], with experienced developers warning that junior programmers in particular were being seduced by speed without the knowledge to assess the risks. Two months later, many of the vulnerabilities remained unresolved, leaving an open invitation for hackers to exploit them and cause financial damage to users.

Around the same time, another self-proclaimed vibe coder launched a SaaS product built with 'zero hand-written code'. It quickly became clear that paying users could bypass subscriptions, API limits were exceeded and the database had become corrupted. His explanation was telling: 'I'm not technical, so it takes me a little longer to fix it.'

Meanwhile, on Reddit, security experts began warning of a new form of cyberattack emerging directly from vibe coding practices. AI models that generate code sometimes invent libraries or software packages that do not actually exist, though their names resemble legitimate ones. Malicious actors exploit this by registering those invented

[7] Parker, E. (2025, 26 August). *Vibe Coding Is the Worst Idea of 2025 – Here's Why It Fails.* Land of Geek. https://www.landofgeek.com/posts/vibe-coding-worst-idea-2025

names themselves and embedding malicious code within them. This technique, known as *slopsquatting*, allows attackers to create backdoors in systems built by inexperienced vibe coders who follow AI-generated instructions blindly. A small error in a prompt can therefore trigger an attack capable of compromising an entire system.

Blurred ownership and legal vacuum

The issue becomes even more critical in sectors where compliance, security or liability are central. In a banking system, a healthcare application or a defence network, it is not enough for the code to *work*; it must also be clear *why* it works, *when* it fails and *who* is accountable if it does. In many vibe coding projects, that traceability is missing. The logic is undocumented, the origin is not reproducible, and the software's behaviour often becomes visible only once it is already in use.

This leads to a more fundamental problem: the blurring of ownership. In traditional software development, it is clear who is responsible for a module, a release or a patch. With vibe coding, however, the boundary between user and system becomes indistinct. The developer writes a prompt, but its execution – the concrete implementation in code – is carried out by a model that cannot be held formally responsible for anything. Who, then, 'writes' the software? And who bears the legal responsibility when something goes wrong?

These questions remain largely unanswered, even in legal terms. Is AI-generated code protected by copyright? Is the prompt writer liable? The organisation? Or the creator of the language model itself? In the absence of clear frameworks, a vacuum emerges in which systems operate without anyone being accountable for their behaviour – a scenario the participants of the 1968 NATO conference had already warned against, long before AI existed.

A (fictional) letter to the modern developer

Dear developer,

You work in an era in which machines can write code from natural language, in which you 'vibe code' – intuitively, creatively and without having to write every line yourself. You describe your intention, and the machine fills in the rest. A technological dream, without a doubt.

But allow me to ask you a question. Who is responsible for this generated code? Who understands its structure, its errors, its behaviour under pressure? In 1968, we stood at the cradle of what we then proudly called 'software engineering'. Not because it was already a mature discipline, but because it urgently needed to become one.

Our concern was simple: without clear methods, standards and accountability, poor software would spread like weeds through systems, organisations and societies. Today, now that software seems to write itself, that concern is more relevant than ever.

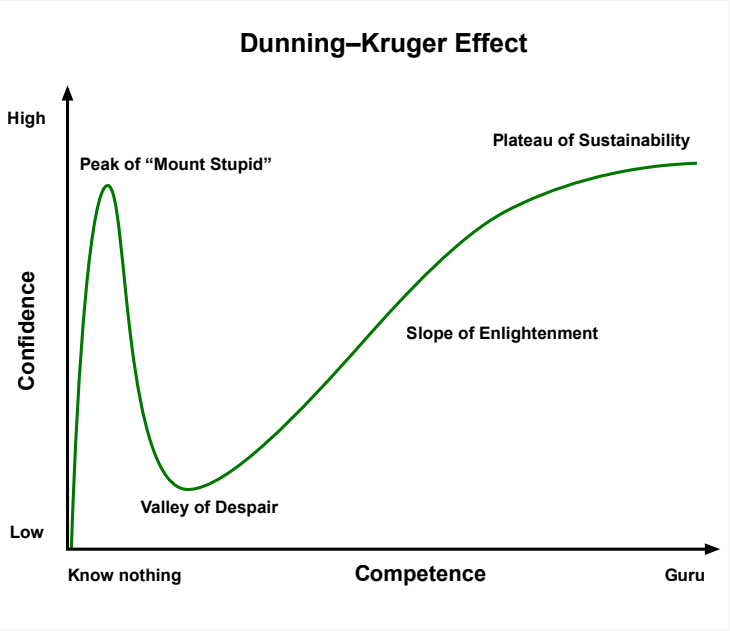
Kind regards,

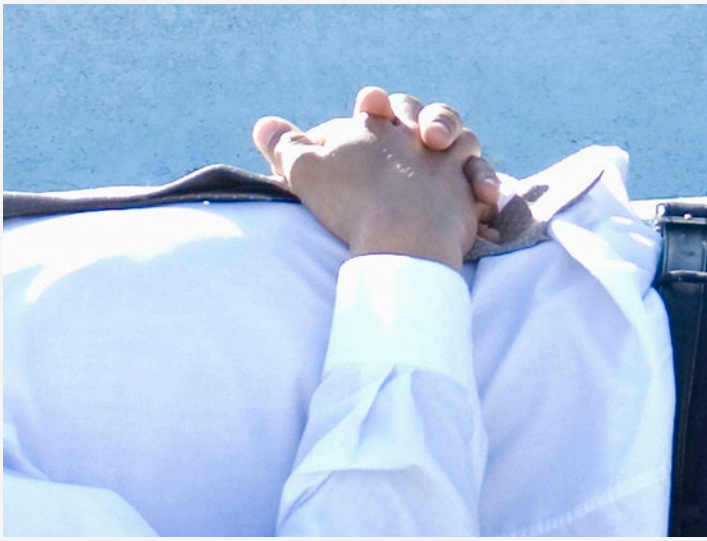
Brian Randell, Co-editor,
NATO Software Engineering Conference, 1968

The illusion of control

There is also a psychological risk that is rarely discussed: the Dunning Kruger effect. This cognitive bias, first described by psychologists David Dunning and Justin Kruger in 1999, shows that people with limited competence in a particular field tend to overestimate their own abilities, because they lack the insight to recognise their shortcomings. In the context of vibe coding, this effect is especially relevant. The interface is user-friendly, the output impressive and the results immediately usable, but appearances can be deceiving.

Because the tools are so accessible, the illusion of control grows faster than actual expertise. The effect is not only psychological, but also economic. Several companies have reported runaway costs and projects that initially appeared fast and inexpensive but ultimately proved more costly because of unexpected rebuilds, hidden bugs and rising maintenance expenses. You do not need to understand how an algorithm works to use it, and that is precisely where the risk lies.





Vibe coding allows users to operate at a semantic level that is detached from the underlying technical reality. You do not need to understand what a compiler does, how memory management works, or how APIs interact. But as soon as the generated code fails – subtly or spectacularly – the knowledge needed to identify causes or assess consequences is missing. The combination of a high degree of ease of use and a low degree of self-awareness creates a dangerous asymmetry: users become overconfident, while their decisions rest on superficial understanding.

The risks are not merely theoretical. In several organisations, it has already become clear that junior employees who feel empowered by AI tools are more likely to present themselves as autonomous developers – without the experience to recognise patterns of bad code or structural weaknesses. The tooling whispers confirmation rather

than correction, and those whose intuition is reinforced by AI become less inclined to question it. The result is not only a higher likelihood of errors, but also a decline in learning capacity: feedback loops weaken because self-confidence expands faster than knowledge.

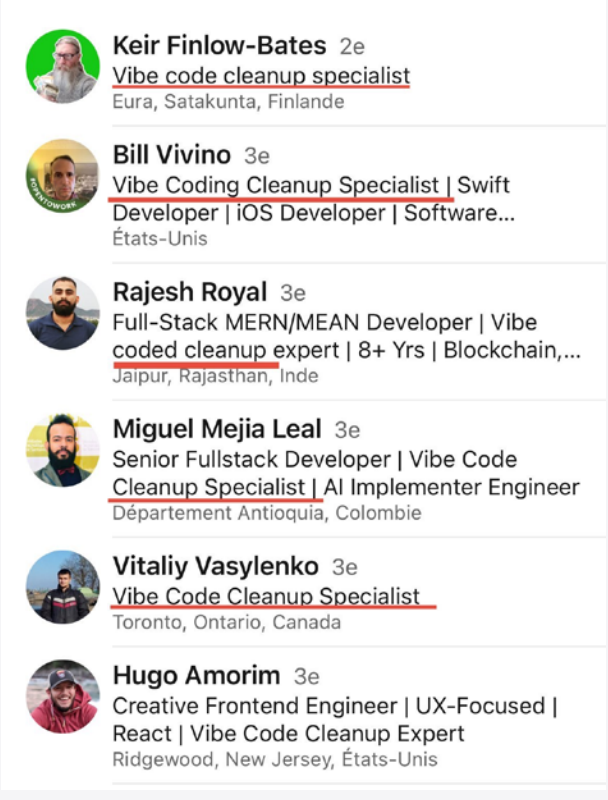
This psychological pitfall deserves explicit attention in any governance model surrounding vibe coding. Not only technical testing, but also education, awareness and team dynamics are essential. After all, the greatest vulnerability lies not in the model, but in the person who believes they have mastered it.

Behind this lies a long-term question that is more technical, yet no less important: what does vibe coding do to our collective knowledge structures? If new generations of developers grow up with tools that write everything for them, will they lose the ability to understand deeply? Will coding become a ritual rather than a skill – a liturgy of prompts instead of logic? And if so, who will still be capable of building the systems of the future?

These concerns are not hypothetical. Experienced developers often compare vibe coding to building a house out of Lego bricks: it may look impressive, but no one is entirely sure whether it will stay standing. The comparison is simple, but it points to a deeper truth. Vibe coding creates an illusion of control – systems may appear perfectly coherent – while the structural logic beneath often remains opaque.

A new ethic for generated software

Still, there is no reason to give in to pessimism. Vibe coding can be applied safely, transparently and productively, provided that organisations are willing to develop new forms of governance, quality control and accountability. This calls for new roles – such as prompt architect or AI verifier – new processes, including systematic code review of generated components, and above all a renewed ethical framework. We must find a way to revive the spirit of Garmisch-Partenkirchen in a world where software is no longer merely written, but also generated.



Chapter 7
Governance and embedding

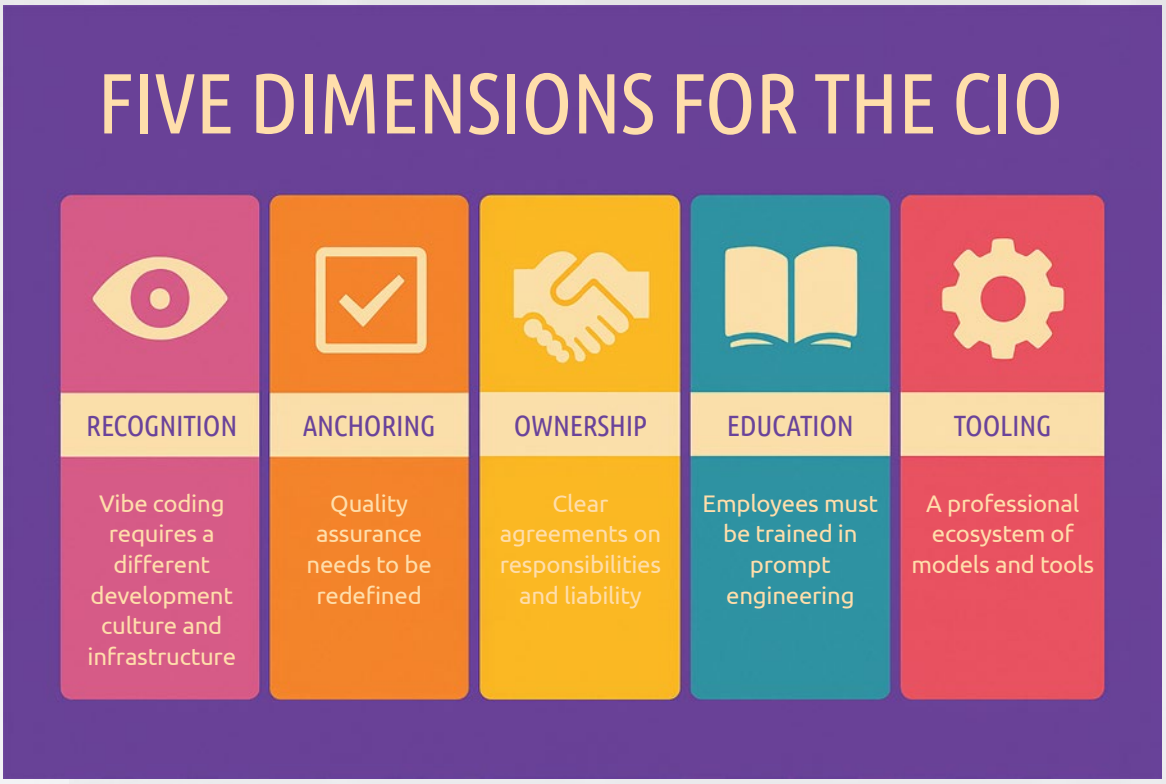
What CIOs need to do to implement vibe coding strategically and responsibly

Vibe coding is no longer an experiment. The tools are maturing, use cases are becoming more business-oriented, and the code bases are expanding. What began as a playful way to build prototypes is evolving into a serious development practice. As a result, organisations can no longer afford to ignore vibe coding, but neither can they adopt is blindly. It calls for direction, for clear frameworks, and for a new kind of governance suited to a practice where software is created through language rather than technology.

This presents a strategic challenge for CIOs, not only to mitigate risks, but above all to harness the opportunities of vibe coding in a sustainable and controlled way. The promise is real: faster innovation, lower barriers and closer alignment between business and IT, but the price can be high if the process remains unchecked. In this context, governance is not an obstacle to innovation but a prerequisite for scalability.

The first step is *recognition*: acknowledging that vibe coding differs fundamentally from traditional software development. The distinction lies not only in the tools, but in the culture, the pace and the structure of responsibility. A developer working with Copilot, Cursor or Replit follows a different rhythm and assumes a different role. They are less of a builder than a guide, less of an architect than an editor. The classic functional designer does not fit this model easily, nor does the traditional ticket-driven scrum process. The development process is becoming looser, more associative and more experimental, and that demands a new kind of infrastructure, both technical and organisational.

The second step is *anchoring*. How is vibe coding integrated into existing quality frameworks? Are new code review routines needed? How can organisations ensure that generated code is tested, understood and documented? In many organisations, quality assurance still relies on implicit expertise: senior developers who can recognise at a glance whether code is good enough for production. But what if the code wasn't written by anyone at all? Then testing must be reconceived as an explicit, repeatable and transparent process – one that relies not on intuition, but on system.



Yet the real challenge lies not in rules but in knowledge, and especially in the difference between what can be documented and what can only be experienced. In traditional software practice, there has always been a balance between **explicit knowledge**, such as methods, standards and procedures, and **tacit knowledge**,[8] the silent craftsmanship that develops only through doing, failing and repeating.

The latter form of knowledge that is at risk of disappearing. When code is largely generated, the layer of experience in which intuition, context and judgement grow begins to fade. We may preserve the explicit processes – the test plans, the guidelines, the audit trails – but we lose the ability to understand *why* something is wrong. Governance without tacit knowledge is like a map without terrain: neatly organised, yet impossible to navigate.

A mature application of vibe coding therefore requires more than new rules. It calls for a new form of learning, one that helps teams preserve their tacit understanding in an environment where the machine appears to know increasingly more.

Related to this is the question of *ownership*. Who is responsible for code that is generated from a prompt? Is it the employee who writes the prompt, the team that accepts the code, or the organisation as a whole? And what happens when that code exhibits behaviour that only causes problems months later? Who will be accountable? The governance of vibe coding requires new agreements on ownership, audit trails and liability. In a world where software is generated by probabilistic models, certainty itself must be redefined.

[8] Polanyi, M. (1966). *The Tacit Dimension*. Routledge & Kegan Paul.

The fourth step is *education*. Not in the traditional sense of programming courses, but in the development of AI literacy. Working with vibe coding does not demand extensive expertise in programming languages, but it does require an understanding of probabilistic model behaviour, bias, prompt design and fallback strategies. Organisations must train their employees to become critical users of generative systems: people who know what AI can and cannot do, and how to guide it effectively. Prompt engineering is no longer a niche skill but a core competence.

A further dimension is *tooling*. Not every model is suitable for every application. Some large language models perform better in backend logic, others in interface generation. Some are faster but less reliable. Organisations will need to build an ecosystem of models, assistants, validation systems and interface tools that together create a robust infrastructure. Vibe coding requires technical maturity: sandboxing, logging, rollback mechanisms and fallback prompts.

Finally, there is the *ethical* dimension. Vibe coding raises questions about originality, intellectual property, digital sovereignty and the relationship between humans and machines. If code is generated from public training material, this raises issues of copyright and attribution. Code that is deployed without insight into its performance can be ethically problematic when its consequences affect people. An organisation that chooses to replace development teams with prompt-based workflows is also making a cultural and policy decision about trust, craftsmanship and autonomy.

Responsible adoption of vibe coding therefore requires not only technical integration but also cultural maintenance. It calls for a shared conversation about what we want to build and how to do so responsibly. The aim isn't to return to the improvisation of the pre-Garmisch-Partenkirchen era, but to reach a renewed kind of maturity – one that fits this new phase of software development: fast, intuitive and, for that very reason, in need of structure.

Not every organisation is ready for vibe coding, but every organisation must prepare for it. Like every interface evolution – from punch cards to graphical interfaces, from the mouse to the prompt – it is not a hype cycle but a structural shift in how humans control machines. Those who fail to take ownership of this shift leave that role to the model itself. And that may be the greatest risk transfer of our time.

Chapter 8
The future of software development

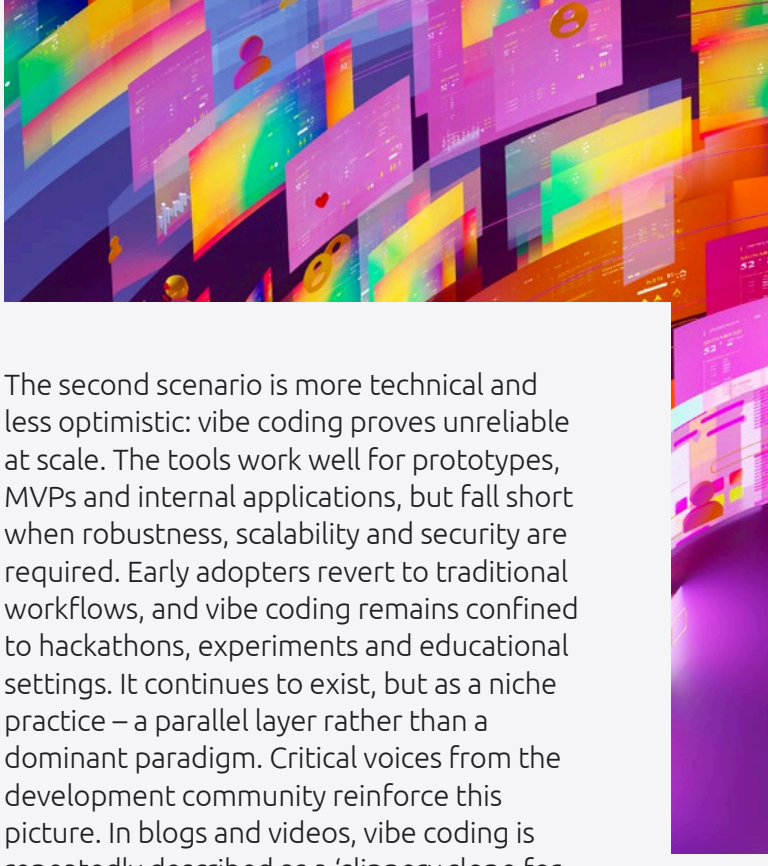
Is vibe coding the new normal or just an intermediate step?

Sometimes it feels as though we are at the beginning of an entirely new development practice. As if the classic programmer, trained in syntax, data structures and design patterns, is gradually disappearing and being replaced by a new figure: the prompt engineer, the vibe curator, the system whisperer. Someone who doesn't *write* code but elicits it. Someone who doesn't type from left to right, but thinks in circles and iterations, in conversation with a model that serves at once as assistant, mirror and inventor.

Seen in this light, the rise of vibe coding seems irreversible. Its speed, accessibility and creative potential align seamlessly with broader social trends: the desire to innovate more quickly, to abstract complexity and to turn technology into a service rather than a specialised field of expertise. In that sense, vibe coding belongs to the same wave as no-code platforms, generative design tools and voice-based interfaces. It marks a shift from production to prompting, from writing to speaking, from mechanics to magic.

Yet it is still too early to speak of a definitive transition. Although vibe coding offers spectacular possibilities, its limitations are just as obvious. The technology is still unstable, the workflows are immature, and the code bases remain fragile. What works on Friday may be unreadable by Monday. And what feels fast today often turns out to be difficult to scale tomorrow. What looks like democratisation can quickly result in a flood of half-finished solutions – systems that are difficult to debug, maintain or reproduce. And the unanswered questions around intellectual property remain urgent. The issue, then, is not just whether vibe coding works, but whether it can endure.

Whether vibe coding becomes the new normal depends less on the models themselves and more on what we do with them – as individuals, as organisations and as a society. There are, broadly speaking, three conceivable scenarios.

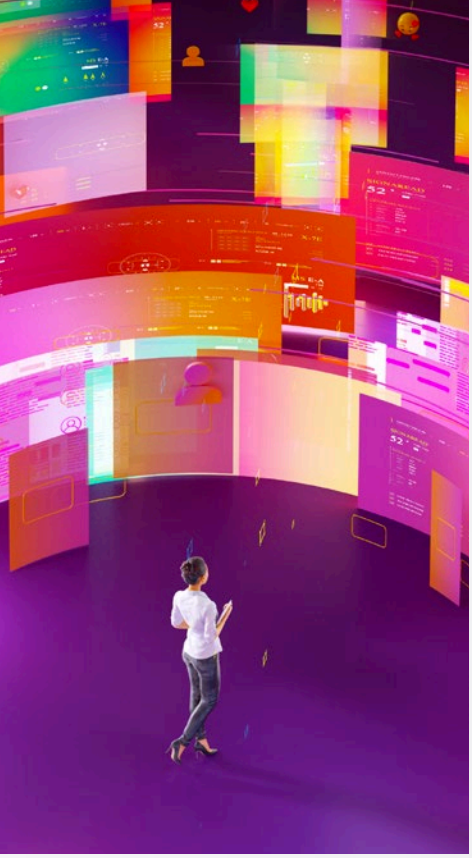


In the first scenario, vibe coding becomes widespread, not as a replacement for traditional development but as a supplement. Developers continue to play a central role, using vibe tools as accelerators, notepads and extensions of their own thinking. Prompting becomes as natural as searching online, browsing Stack Overflow or using autocompletion. Organisations build governance frameworks around these tools, combining vibe coding with advanced test suites and code verification, and embedding them in secure sandbox environments. In this scenario, vibe coding becomes the new normal, but within defined boundaries and institutional safeguards.

Concrete figures already point in this direction. Several large technology companies report that around 30 per cent of their production code is now generated by AI, a proportion that appears to be growing every quarter.

The second scenario is more technical and less optimistic: vibe coding proves unreliable at scale. The tools work well for prototypes, MVPs and internal applications, but fall short when robustness, scalability and security are required. Early adopters revert to traditional workflows, and vibe coding remains confined to hackathons, experiments and educational settings. It continues to exist, but as a niche practice – a parallel layer rather than a dominant paradigm. Critical voices from the development community reinforce this picture. In blogs and videos, vibe coding is repeatedly described as a 'slippery slope for juniors', a tempting shortcut that gives beginners quick confidence while distancing them from the deeper knowledge they need.

1	2	3
INSTITUTIONALISATION The New Normal	**NICHE PRACTICE** The Prototype Bubble	**COGNITIVE FLATTENING** The Conjuring User
		
Vibe coding is becoming mainstream, but within clear frameworks. Developers use it as an accelerator, supported by testing, governance and secure environments. Prompting is becoming as common as Googling.	Vibe coding proves too unreliable for large-scale, secure systems. It continues to exist for MVPs and hackathons, but does not gain a permanent place in serious production environments.	Prompts replace understanding. A generation is growing up without knowledge of code, only of output. Coding becomes magic, without control or insight. Humans become users instead of creators.



The third scenario is perhaps the most fundamental, touching on the question of cognitive skills. In this scenario, the widespread use of vibe coding produces a generation that no longer learns how software works, but only how it behaves. Coding becomes a ritual of sorts: writing prompts, feeding models and evaluating their output, without understanding the mechanisms beneath. The world becomes controllable through language, but no longer explainable. Software becomes magical, and magic does not tolerate critique. In such a future, humans cease to be engineers and become conjurers – not builders, but supplicants. This scenario also aligns with the logic of the hype cycle. The peak of early 2025 is already giving way to mid-year reality checks, with initial excitement yielding to scepticism and to questions about what will remain once the promises fade.

Whether this loss of cognitive skill is acceptable is the question that runs beneath all the hype and innovation. Vibe coding promises simplicity but can also breed dependence. It liberates us from syntax yet risks alienating us from logic. It accelerates development but also accelerates the erosion of understanding. The central task for the coming years is therefore not only technical, but also moral, educational and political: how do we keep thinking alive in a world where thinking is becoming optional?

Perhaps vibe coding is not an endpoint at all, but a transitional stage – an interface on the way to something more fundamental: systems that sense our intentions before we articulate them. Behavioural interfaces. Emotional computing. Artificial intelligences that read context, interpret sentiment and anticipate what we want. In that light, the prompt becomes an intermediate step, a linguistic leftover from the era in which we still had to speak to our machines.

Yet even then, the same question that hovered over Garmisch-Partenkirchen in 1968 remains: who still understands the systems that support us? And what do we lose when we no longer understand them, but depend on them?

What vibe coding ultimately exposes is not only the power of language, but also the limits of intuition. And it is precisely in that space between convenience and comprehension that the future of software development will unfold – not in code alone, but in how we learn to think alongside it.

Chapter 9
The vibe as infrastructure

From vibe coding to vibe marketing, vibe HR and the promise – and warning – of vibe governance

Vibe coding is more than a new way of generating software. It represents a broader technological shift: from human production to AI-facilitated orchestration. What began in software development is now spreading to other domains. There too, an intention, a thought, a command or a prompt combined with a well-designed AI system is enough to carry out work that was once manual, slow and labour-intensive. This marks the rise of a mode of operation in which human control gives way to AI-driven response, optimisation and execution. The vibe has become an infrastructure: light, iterative and adaptive. New domains are emerging in which the 'vibe' manifests itself, each with its own promise and its own risks.

Vibe marketing: the AI stack as a creative team

In the marketing world, the vibe revolution is already well underway. Vibe marketing refers to the use of generative AI tools to perform tasks that once required entire teams: audience analysis, campaign design, content creation, A/B testing and iteration. Today, a single marketer with an advanced AI stack can generate the output of a full agency.

According to Writesonic, this automation represents the most significant shift in the marketing industry since the emergence of the internet. Campaigns are now generated, optimised and deployed in real time, often without direct human intervention. Marketers no longer manage every detail, but define the parameters instead: tone, audience, style and channel. The model does the rest.

A striking example was demonstrated live on television by Alexander Klöpping, a guest on Eva Jinek's talk show in The Netherlands.[9] In front of the camera, he showed how he had built a virtual marketing agency using n8n, an open-source automation platform. The assignment was simple: promote the author Carry Slee's new book. Klöpping had replaced every employee in the traditional marketing workflow with an AI avatar, each

[9] Klöpping, A. (2025, 15 April). *Alexander Klöpping laat zien hoe AI jouw kantoorbaan overneemt: "We hebben marketingbureau Eva opgericht"*. [Video]. EVA | AVROTROS. https://eva.avrotros.nl/artikel/alexander-klopping-laat-zien-hoe-ai-jouw-kantoorbaan-overneemt-we-hebben-marketingbureau-eva-opgericht-624

assigned a specific role. There was a creative director generating cover options, a copywriter producing texts, a media strategist optimising placements of advertisements and a project manager coordinating the process.

The result was a complete campaign, from design to distribution, created by a swarm of virtual colleagues and managed through a single prompt structure. No team meetings, no Slack channels, no human art director. Only the user and the model, engaged in an endless dance from concept to conversion.

Greg Isenberg, former head of product strategy at WeWork, captured this transformation succinctly: 'Vibe marketing lets you do in hours what used to take weeks; it is no longer creation, it is orchestration.'[10] Tools such as Jasper, Copy.ai, Midjourney and Synthesia now make it possible to generate advertisements, visuals and even complete branding concepts, to test them iteratively and to implement them immediately.

The speed is astonishing. Yet here too, speed without control becomes a risk. Who decides when the output of the machine still seems credible, but no longer has a sender?

Vibe HR: engagement and well-being as a data-driven system

The Human Resource Management domain is also undergoing a vibe transition, though in a different form. Vibe HR refers to the use of AI tools that focus on employee engagement and well-being, often supported by real-time feedback, sentiment analysis and cultural monitoring.

Platforms such as Officevibe (now Workleap) continuously collect feedback from employees, analyse atmosphere and connectedness, and offer managers targeted suggestions for improving morale. Tools such as Vibe HCM automate HR transactions and foster internal communication, shifting the focus from Excel sheets to experience. Employees are no longer treated as personnel numbers, but as participants in an emotional work culture that is tracked and guided through dashboards.

The premise is clear: by making the human work environment measurable, well-being becomes a strategic management instrument. According to Gallup, companies with high engagement are up to 23 per cent more profitable. Vibe HR translates that promise into technology.

Yet this approach also introduces tension. When well-being is measured, managed and optimised through models, it inevitably becomes objectified. What happens when feedback is no longer shared with a manager, but with a system? When the culture itself is measured, adjusted and shaped by tools? The result may be a frictionless organisation in which the human dimension gives way to a perfectly managed experience – pleasant, but perhaps also empty.

[10] Greg Isenberg. (2025, 30 April). *How I use AI agents to make money (Vibe Marketing Tutorial).* [Video]. YouTube. https://www.youtube.com/watch?v=PduJ0P6r_8o

In addition to marketing and HR, new domains are emerging: vibe analytics and vibe design. In analytics, dashboards now generate themselves, data flows are visualised automatically, and the manager's main task is interpretation. In design, the focus shifts from sketching to steering. AI tools create iterative variations based on vibe prompts, while the designer assumes the role of curator rather than creator.

Vibe consultancy: the robot as advisor

What McKinsey has been to the boardroom for decades, Operand now promises to be for the data platform: an all-knowing advisor, but not of flesh and blood and without a daily rate. No delays, no army of consultants, only an AI system that analyses company data in real time and produces direct recommendations. No PowerPoint, but impact. The advice remains; the advisor changes.

Operand describes itself bluntly as *'an AI to kill McKinsey'*. The platform offers companies, beginning with those in retail and e-commerce, an alternative to traditional consultancy: a system that automatically generates advice on pricing, discount strategy, stock management and advertising budgets. All of this is based on a deep, integrated analysis of internal and external data. The promise is direct impact on the profit and loss account – an additional six figures on the balance sheet within weeks.

What Operand illustrates is that vibe consultancy is part of a broader transition: from consultancy as a craft to consultancy as infrastructure. Human consultants do not disappear, but their role shifts to that of quality controllers. Former McKinsey consultants now review the output of AI systems instead of producing it themselves.

The broader implication is clear: if AI can automate management consultancy, it will soon be able to do the same for strategy, finance and operations. Where experience, analysis and intuition once converged in a boardroom, a digital brain is emerging that runs continuously on data.

But the question remains: who is still thinking independently?

Vibe governing: when policy becomes a prompt

The most precarious development is the application of the vibe principle within government and policy. Vibe governing refers to the phenomenon in which policy choices arise directly or indirectly from the output of generative AI systems, such as large language models, without sufficient human review, transparency or accountability. What remains experimental and playful in software development can become deeply disruptive in political decision-making.

The term gained wider recognition when Donald Trump was accused in 2024 of practising vibe governing in setting his trade tariffs. In a leaked excerpt, advisers revealed that Trump had relied on analyses generated by large language models which, drawing on sentiment data, historical patterns and public opinion, produced 'proposals' for tariff structures. These were not based on economic causality, but on what appeared, in the data, to *look* as though it would work.

In a post[11] on his blog at *Strange Loop Canon*, Rohit Krishnan, author of the book *Building God: Demystifying AI for Decision Makers*, describes this form of decision-making as policy that is not based on empirical evidence or strategic goals, but on *plausible language patterns*. Just like a language model itself, the process is plausible, fluent and convincing, but lacks understanding of its own mechanisms or consequences. The economic impact of such decisions can be immense. According to economist Krishnan, some of Trump's tariffs were even 'pre-Adam Smith wrong' – driven by conviction and presentation rather than by rigorous economic analysis.

Vibe governing therefore raises not only technical but also fundamental democratic questions. Who controls the source of policy if that source is synthetic? What happens when persuasiveness becomes more important than substantiation? And what if the algorithmic logic of large language models, which are optimised for coherence and effectiveness rather than for truth or justice, begins to infiltrate our institutions through persuasive language?

[11] Krishnan, R. (2025, 7 April). *Vibe Governing: using LLMs to set policy.* Strange Loop Canon. https://www.strangeloopcanon.com/p/vibe-governing

The vibe organisation as a new organisational model

The vibe organisation is more than the sum of AI tools or workflow innovations. It represents a fundamental shift in organisational thinking, in which core processes, decision-making and culture are designed so that human intention is directly translated into AI-driven execution. This transformation has far-reaching consequences for structure, power, culture and strategy.

- **Orchestration over hierarchy.** The role of management is shifting from control and planning to the orchestration of AI-driven processes. The emphasis lies in designing effective prompts, establishing monitoring frameworks and overseeing AI output.
- **New key roles.** Prompt engineers, AI curators, model auditors and ethical supervisors are becoming essential functions. The importance of traditional specialist roles is declining, while skills such as critical thinking, system management and AI literacy are becoming central.
- **Learning culture and adaptability.** As AI systems evolve rapidly, continuous learning and experimentation are becoming the norm. The vibe organisation promotes a culture of iteration, reflection and quick adaptation. This includes adaptive software that optimises itself, with workflows that detect patterns and autonomously adjust their sequence or intensity without human intervention.
- **Access as power.** Power is shifting from individuals with formal authority and deep expertise to those with the greatest access to AI tools and the skill to use them effectively.
- **Momentum over strategy.** The speed of iteration and the capacity to test and implement new ideas quickly are becoming more important than long-term planning or traditional strategy development.

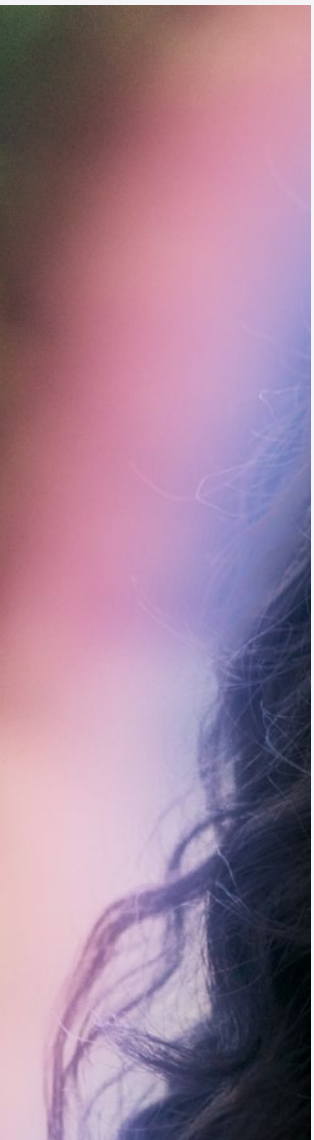
Risks and ethics

The focus on speed and iteration can come at the expense of depth, craftsmanship and the ability to understand complex problems in a fundamental way. In an infrastructure where AI systems make a growing number of decisions, there is a serious risk that responsibility will become diffuse. Governance, transparency and the explicit allocation of liability are therefore essential.

Practical building blocks for the vibe organisation

Where the opportunity seems immense to build the organisation fit for the 21st century of AI-first, we need to be vigilant about the potential risks. Below some practical suggestions to guide the emergence of this new vibe organisation:

- **Develop an explicit AI governance framework** that includes ethical review, auditability and responsible prompt usage.
- **Form multidisciplinary teams** that combine AI specialists, domain experts and ethicists to design, monitor and adjust the AI infrastructure collaboratively.
- **Ensure reproducibility, explainability and verifiability** of all AI-driven processes – not only from a technical perspective, but also from legal and social standpoints.



The vibe organisation represents a new organisational model in which human intention, AI-driven infrastructure and adaptive culture converge. This model demands a fundamentally different approach to management, responsibility and learning. Those who consciously design such an infrastructure can combine speed and innovation with control and ethical awareness. Those who leave it to chance risk superficiality, dependence and the loss of a human compass.

The vibe as the interface between humans and models

What unites all these domains is that they use the promise of AI as an interface, not as a brain but as a productive infrastructure. Humans remain, for now, responsible for direction. The machine executes, optimises and accelerates. That is precisely why it is essential not to confuse the vibe principle with emotion, but to recognise it as a working method – one with immense potential, yet also with inherent limitations.

A vibe organisation is not necessarily a chaotic or post-hierarchical culture where everything revolves around mood. It is an organisation that has redesigned its processes so that human intention forms the input and AI serves as the executive layer. Those who build such an infrastructure without oversight will create magic without discipline, but those who approach it with conscious design can achieve both speed and care.

The challenge is clear: vibe as infrastructure requires not less rationality and critical thinking, but more. Only then will humans remain not merely the users of AI, but its stewards.

The organisation as a prompt machine

What does all this mean for the future of organisations? Vibe coding is not only a technical practice but also a cultural model, a way of working in which intuition, speed and AI-generated suggestions take the place of planning, consultation and expertise. You no longer need to be an expert; you only need to be skilled at prompting.

This shift transfers power within organisations from those with the deepest knowledge to those with the best access to tools. In the vibe organisation, strategy gives way to momentum. Yet another danger emerges: fragmentation. Whereas traditional software development aimed for coherence and architecture, the vibe culture produces isolated patterns that coexist without a unifying structure, postmodern software forms that function in the moment but resist integration into a sustainable whole.

And perhaps that is the real lesson here: the interface that once served us as a tool now risks hardening into an ideology, a world where thinking is optional and prompting becomes the culture.

The economy without an inside

When AI not only performs the work but also distributes it, the question shifts from labour to organisation. Economists Peyman Shahidi, Gili Rusak and John Horton describe this development in a recent MIT and Harvard study,[12] as the Coasean Singularity: the moment when the friction that once held companies together disappears.

Ronald Coase argued that firms exist because markets are too slow, too chaotic and too costly. It took effort to find suppliers, compare prices and conclude contracts. Organisations therefore evolved not only as production structures but also as protective layers against uncertainty. Now that these transaction costs are vanishing, outsourced to AI agents that can search, negotiate and contract faster and more accurately than humans, the company begins to lose its very reason for existence. The market becomes fluid, composed not of firms but of protocols that communicate directly with one another.

What Coase once described as economic theory is now unfolding in miniature within the individual. Vibe coding is Coase in miniature: the knowledge worker who dismantles their own internal organisation. Whereas in the past there was a distinction within a single mind between thinking, organising and executing, those functions are now increasingly outsourced to software. The person who works with AI becomes their own market: a place where prompts, corrections and versions compete for attention. Economic dissolution is therefore taking place not only at the macro level, but also within individual consciousness.

The structures of work and thought are becoming intertwined. The programmer who has code generated by an agent resembles the company that replaces its employees with algorithms: both delegate not only tasks but also meaning. What remains is a world in which relationships no longer exist within institutions but between algorithms. The organisation becomes an interface, the employee an API key.

Yet friction never disappears entirely. The more the economy optimises itself, the more new disruptions arise: digital agents simultaneously rushing to perform the same tasks, blurred boundaries between human and machine, and an unending struggle for authenticity. The Coasean Singularity does not mark the end of the economy, but its evaporation into ever thinner layers of abstraction. Efficiency becomes the new form of chaos. What remains is an economy without an inside, a world in which not only work, but also working itself loses its place.

[12] Shahidi, P., Rusak, G., Manning, B. S., Fradkin, A., & Horton, J. J. (2025). *The Coasean Singularity? Demand, Supply, and Market Design with AI Agents* (NBER Working Paper No. 15309). Cambridge, MA: National Bureau of Economic Research. https://www.nber.org/system/files/chapters/c15309/c15309.pdf

The Coasean spirit

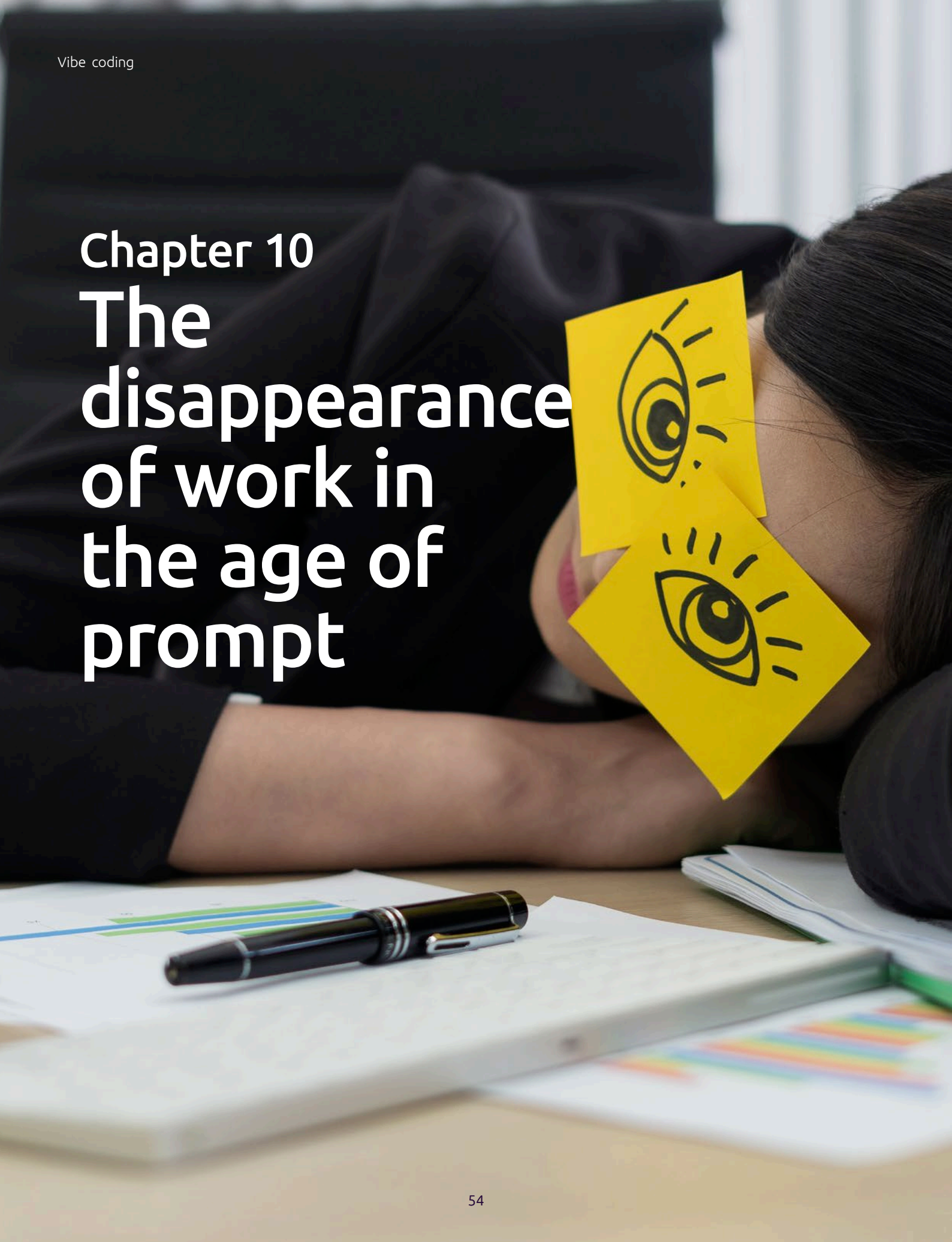
The logic of efficiency is not futuristic; it has already entered our homes. We notice it in days that pass too quickly, in work that runs more smoothly but feels less meaningful, in convenience that exhausts rather than relieves. Friction, once the teacher of craftsmanship, has vanished from our routines. What once offered resistance also offered formation. Now that the detours have been eliminated, everything seems to glide effortlessly, yet nothing leaves a mark.

The grand theories about the Coasean Singularity describe a world in which the enterprise dissolves because transactions become frictionless. But that is not a distant future; it is a description of the present. We consult less and correct more. We no longer converse; we manage flows. The market form has not arisen outside us but within us. We are constantly negotiating with ourselves: between time and attention, between duty and rest, between what we know and what the machine already believes it knows.

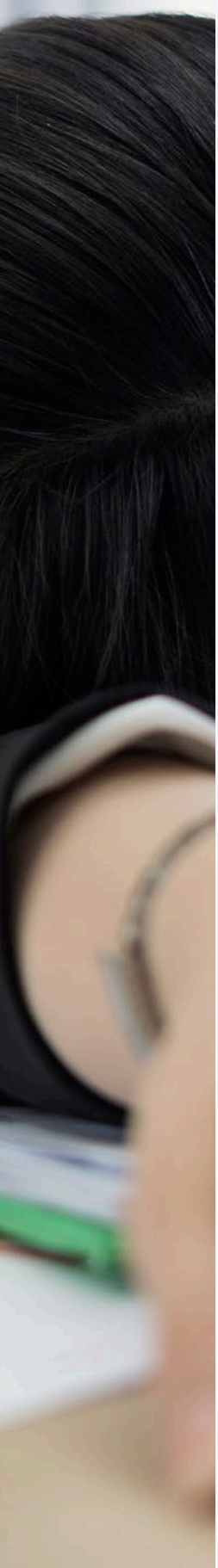
In that silent negotiation, the sense of labour fades. We do more yet feel less that we are doing anything. Vibe coding accelerates work but flattens it; it removes not only errors but also the pauses in which understanding once grew. What remains is a chain of smooth actions without resistance, endlessly repeating itself.

And yet something persists. Beneath the surface speed, a quiet fatigue lingers – a vague awareness that something is missing. Not the work itself, but its reciprocity: that it did something to us, and we did something to it. Perhaps this is the true unemployment of our time, not the absence of work but the loss of formation.

We continue to move, faster than ever, but without direction. The economy is no longer changing outside us; it is settling under our skin, into our conversations and our thoughts. We are becoming markets of attention – increasingly efficient, increasingly empty. And perhaps that is the price of convenience: we lose sight of what the effort was ever worth.



Chapter 10
The disappearance of work in the age of prompt



What vibe coding means for employment in the IT sector and beyond

It is becoming increasingly difficult to see through the haze of hype and separate what is real from what is wishful thinking. Large technology companies and young AI start-ups have an obvious interest in portraying progress as more advanced than it truly is, especially in financial terms: the promise of growth, efficiency and market dominance. Their leaders speak confidently about the end of programming as a profession, about organisations that will soon manage themselves through AI, about cost savings that will double productivity. But behind this bravado lies uncertainty – and perhaps even unease about their growing dependence on systems they no longer fully understand.

A recent report[13] by the MIT Media Lab revealed that 95 per cent of organisations do not achieve measurable returns on their investments in generative AI. Adoption is high, transformation low. Companies are experimenting en masse with tools such as ChatGPT and Copilot, yet the promised productivity gains remain superficial. The technology accelerates without truly transforming.

Still, that acceleration has consequences. Tools like GitHub, Copilot, Cursor and Claude have made it possible to develop, test and iterate faster. As a result, IT companies, from start-ups to Big Tech, are quietly reshaping their organisations. Rather than simply cutting headcount across the board, many are moving away from the traditional pyramid structure towards a diamond-shaped model. The official rationale sounds familiar – efficiency, strategic realignment, an AI-first

[13] Challapally, A., Pease, C., Raskar, R., & Chari, P. (2025, July). *The GenAI Divide: State of AI in Business 2025*. MIT Nanda. https://www.artificialintelligence-news.com/wp-content/uploads/2025/08/ai_report_2025.pdf

policy – but the underlying pattern is harder to ignore: AI is beginning to influence not only workflows, but the very architecture of the firm.

The situation resembles earlier waves of automation, but with one crucial difference: this time, it is not only the hands that disappear, but also the heads. It is not the production workers who are replaced, but the knowledge workers. Software development, once the hallmark of human ingenuity, is now being overtaken by the very revolution it helped create. The programmer is being displaced by the product of their own craft.

What is vanishing is not just a job, but a pathway. Where people once began as juniors, learned by doing, grew through practice and eventually became mentors or leads, that progression is dissolving. AI generates the code, the senior reviews it, and the middle ground – the space for apprenticeship – disappears. A vacuum emerges between learning and leadership. An organisation without an intermediate layer. A workplace without a floor. What disappears here is not merely employment, but the exercise itself – the daily intellectual practice through which knowledge slowly matures into experience.

The figures behind the words

While executives sing the praises of AI, the evidence for the promised productivity remains elusive. Behind the grand claims of efficiency and cost savings lies a reality of conflicting data and inconsistent stories. Round after round of redundancies are announced, yet the same companies continue to post new vacancies. Teams are disbanded and later reassembled, as if the organisation were endlessly tinkering with itself without knowing what it is building. It resembles a system moving without direction – driven by the expectation of profit, not by an understanding of work.

The figures that circulate say little about the reality on the work floor. They measure fluctuations, not transformation. For the essence of this change cannot be captured in quarterly reports but in the way labour itself is losing its form: work that once began with thinking now ends with reviewing what a machine has already thought up.

The consequences of this vacuum are becoming visible in the figures that truly matter. In the United States, economists speak of a growing *graduation gap* – the widening difference between the unemployment rate among young graduates and that of the broader working population. According to an analysis[14] in *The Atlantic* this percentage reached a historic low in 2025. Never before has a degree carried so little direct value. The jobs that recent graduates once started in – research assistant, marketing analyst, junior developer – are precisely the roles now being automated or absorbed by AI. This is not a cyclical dip but a structural signal: the influx of young talent is stalling at the very point where knowledge work is becoming fluid.

Universities continue to produce graduates, even as the labour market they are entering is being reorganised by AI. A generation is emerging that has learned to think but no longer finds access to the places where thinking is rewarded. What remains is not a staircase to the top, but an empty platform – without steps, without direction.

As a rare counter narrative AWS CEO Matt Garman argues that replacing junior software engineers with AI is a strategic mistake rather than an efficiency gain. In a December interview with *Wired Magazine*,[15] he stresses that entry-level engineers remain essential because they form the long-term talent pipeline on which future senior expertise depends; eliminating them today would leave companies without experienced engineers tomorrow. He also notes that junior developers are typically the most fluent users of new AI tools, making them well positioned to extract value from AI rather than be displaced by it. From a business perspective, they are also the least expensive employees, so replacing them with highly paid senior staff guided by AI makes little economic sense. Garman acknowledges that AI will change software engineering – reducing manual coding and shifting work toward problem decomposition, architecture, and orchestration of AI agents – but insists this transformation increases the need for learning and mentorship, not lessens it. His core message is that AI should augment junior engineers and accelerate their development, because companies that stop hiring and training young talent, risk undermining innovation and ultimately 'breaking' their own workforce model in the long run.

Only the future will tell who is right and who is wrong in this argument.

[14] Thompson, D. (2025, 30 April). Something alarming is happening to the job market: A new sign that AI is competing with college grads. *The Atlantic*. https://www.theatlantic.com/economy/archive/2025/04/job-market-youth/682641/

[15] Drummond, K. (2025, 16 December). AWS CEO Matt Garman Doesn't Think AI Should Replace Junior Devs. *Wired*. https://www.wired.com/story/the-big-interview-podcast-matt-garman-ceo-aws

The temptation of progress

Yet it is too easy to see AI solely as a threat. For those who believe in it, AI promises liberation from routine, more room for creativity, and access to knowledge that was once out of reach. In theory, vibe coding could represent a democratisation of intellect – a way to distribute the privilege of thinking across millions of users.

But the reality is the opposite. What is being outsourced is not the routine, but the exercise – the process through which understanding takes shape. Instead of liberating knowledge, AI makes us dependent on answers that have already been formed. It lightens the work but obscures the learning.

The feeling speaks volumes

Andrew Ng, former chief scientist at Baidu and co-founder of Google Brain, made a striking remark at a 2025 conference. He called *vibe coding* a misleading term. Working with AI, he said, does not feel lighter, but heavier. Behind the convenience lies the fatigue of constant vigilance. Those who work with AI must check, review and correct – tasks that demand attention and judgment rather than relaxation. It is not the programmer who is becoming redundant, but the programmer without critical awareness.

'Basic coding knowledge is becoming more important than ever. Over the last year, a few people have been advising others not to learn to code on the basis that AI will automate coding. I think we'll look back at that as some of the worst career advice ever given.'

– Andrew Ng

Ng is not alone in this observation. Increasingly, developers describe the same paradox: working with AI is not liberating, but exhausting. The machine appears to think, yet humans remain responsible for its mistakes, exceptions and boundary cases. Work is not becoming lighter, but denser – fuller, more saturated with supervision.

The future of work will not be work *with* AI, but work *after* AI: the work that begins where the machine stops – reflection, assessment, and review. These qualities will not disappear, but they will become harder to cultivate once the practice that develops them has gone.

Twenty-seven tasks, one misconception

At the annual meeting of the American Economic Association, Erik Brynjolfsson shared an anecdote about radiology.[16] When Geoffrey Hinton predicted in 2016 that AI would surpass humans in reading X-rays within five years, the fate of the radiologist seemed sealed. Yet reality unfolded differently. AI systems did become remarkably skilled at image recognition, but the demand for radiologists did not decline – it grew.

Research shows that a radiologist performs an average of twenty-seven distinct tasks. Beyond analysing images, these include interpreting context, consulting with physicians and patients, weighing laboratory results, and advising on treatment options. These are tasks that cannot be reduced to algorithms; they remain rooted in human judgment.

Brynjolfsson stresses that AI rarely automates an entire profession. It reorganises the work, redistributes the tasks, and at times makes them more demanding. What disappears is not the profession itself, but the ritual of competence – the gradual process through which expertise is cultivated and transmitted. The machine provides the first glance; the human reframes it.

As Aaron Levie, CEO of Box, put it in an interview[17]: we thought AI would correct our mistakes, but in reality it writes the first draft. The human task now is to review, repair, and rethink. This reverses the logic of work: what was once the end has become the beginning.

[16] Rosalsky, G. (2025, 7 January). *What America's top economists are saying about AI and inequality.* Planet Money. https://www.npr.org/sections/planet-money/2025/01/07/g-s1-41290/what-americas-top-economists-are-saying-about-ai-and-inequality

[17] vitrupo [@vitrupo]. (2025, 15 July). *Aaron Levie says the future of work has flipped. We thought AI would fix our mistakes. Now it makes the* [With video]. [Post]. X. https://x.com/vitrupo/status/1944991343434195219

Chapter 11
Humans in the loop

Vibe coding, AI and the moral imperative of technology

What began as a new way of programming has evolved into something more fundamental. *Vibe coding* is not merely a technical innovation, but a symptom of a broader transition: from instruction to intention, from creation to prompting, from structure to direction. This movement reaches far beyond software, reshaping marketing, Human Resource Management, decision-making, and the very architecture of organisations.

Yet the more natural the interface becomes, the greater the risk that we forget what lies beneath it. The prompt appears simple, but beneath its surface operates a vast web of assumptions, biases, and decision-making patterns that no human can fully comprehend.

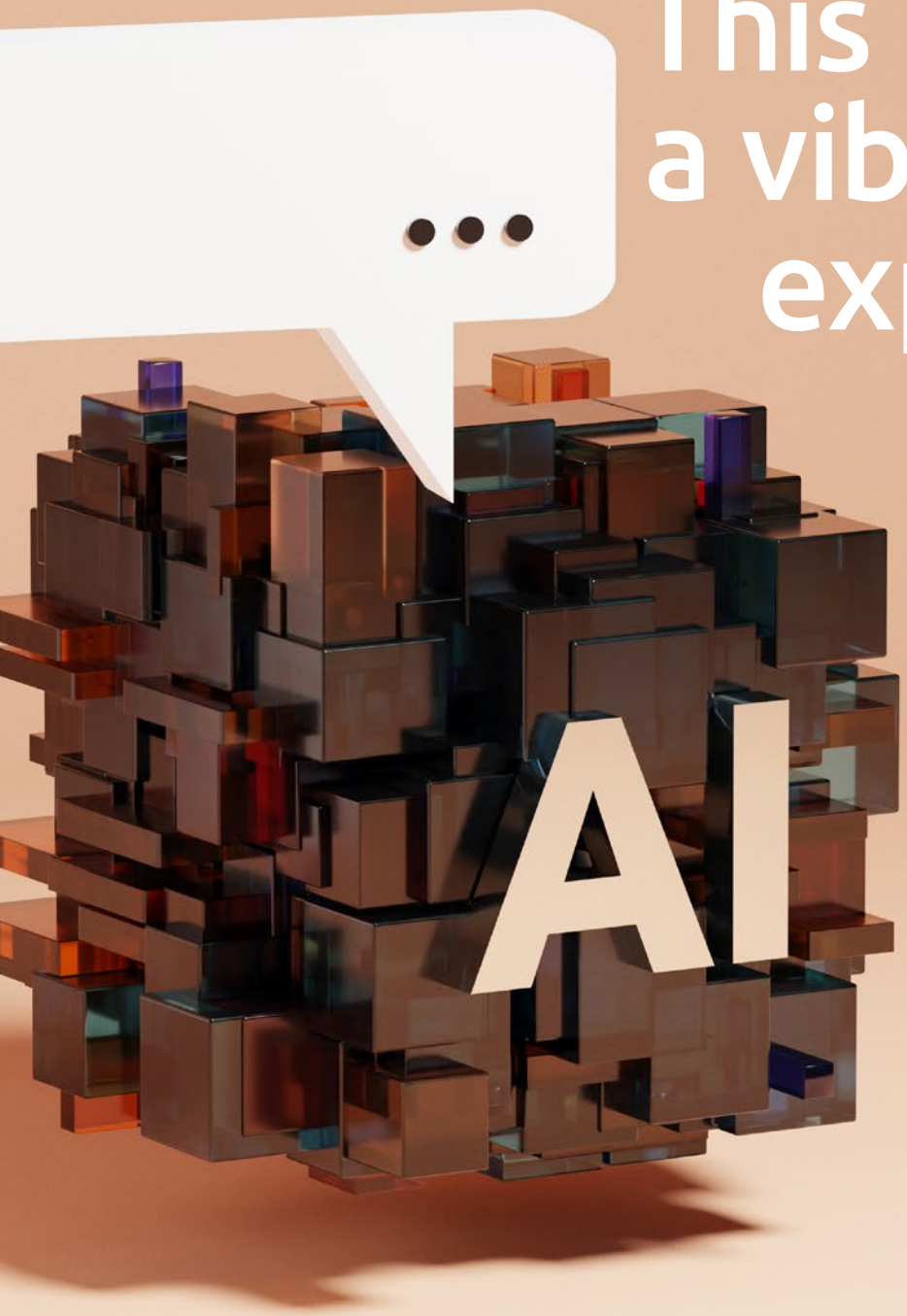
The questions this raises echo those posed in 1968 at the forementioned NATO Software Engineering Conference in Garmisch-Partenkirchen: Who is responsible for code they did not write? How can we design systems that do not overwhelm our own capacity to think? And what happens when complexity grows faster than understanding?

The difference is that today we are no longer dealing only with complex software, but with systems that generate software themselves. What was once a craft has become an infrastructure. What then needed to be made *manageable* must now remain *comprehensible*. Vibe coding not only demands better tools; it demands a different kind of attention.

That is why this report ends not with a conclusion, but with a call – to CIOs, policymakers, developers, and designers alike: keep sight of the human dimension. Do so not only through oversight and regulation, but by deliberately building in friction, reflection, and accountability. Allow for delay where necessary. Create checks that are not merely technical, but ethical. And be willing, at times, not to optimise, but to contemplate.

In a world that seems to understand us ever better, it becomes all the more vital that we continue to ask ourselves *why* we mean what we mean.

Epilogue
This report as a vibe writing experiment

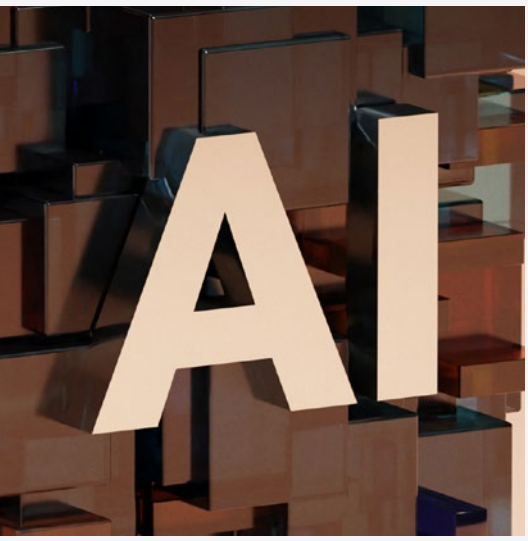


This report was not written in the traditional way. No author sat behind a keyboard for weeks on end, immersed in silence and concentration. It was created in conversation with language models. Sometimes through a screen, sometimes through a voice interface in the car. Sometimes by entering fragments of text or articles, sometimes by asking questions, sometimes by asking AI to play a role of a persona like Yuval Harari, Steve Jobs or a sceptical consultant.

This way of working resembles the very practice described in the report itself. Just as *vibe coding* transforms the programmer from builder to orchestrator*, vibe writing* transforms the writer into a curator. The role shifts: less creation, more direction; less writing, more selection; less craftsmanship in the sense of penmanship, more judgement about what is useful, convincing and resonant.

The style emerged iteratively, in rounds of generating and deleting, refining and rejecting. Where the writer was once a solitary seeker for the right sentences, they are now a prompter, tester and editor. A curator of language, a verifier of plausibility, a designer of the vibe.

Deleting text was often the hardest part, precisely because the sentences flowed so effortlessly. You had to switch on your *bullshit detector* to sift out the AI slop – in other words, to read actively. The hyperbole became indigestible if you read too much at once. Couldn't it be a little more nuanced? The real irritation was in the phrasing. Sentences like 'It's not this, but that' kept returning as a clear trace of model generated narratives. There remain still plenty of such patterns left in this text to recognise their origin.



The voice was never singular. We let ChatGPT, Claude and Gemini talk to one another, comment, correct and contradict. At times it felt as though we were moderating a panel discussion: one model provided sharpness, another elaboration; one became lyrical, another clinical. The writing turned into a multi-voiced choreography, and our role into that of moderator rather than author.

There were moments when we deliberately pitted systems against each other: a version by Claude beside one by ChatGPT, a sharp analysis from Mistral mirrored by the visionary improvisations of Gemini. Not because we trusted any of them completely, but because the tension between their voices produced something no single model could offer on its own. It was like orchestrating an editorial board that did not exist, yet somehow did, a team of virtual editors who challenged, complemented and occasionally drowned one another out.

In doing so, we learned a critical lesson. It is tempting to settle for the first answer a model gives, fluent, convincing and complete. Yet that very ease makes a text hollow, detached and frictionless. Only when we asked again, invited contradiction or let perspectives collide did resistance emerge. And only that resistance made the insights stick. Without friction, knowledge slides away like water off glass.

Even flattery was part of it. ChatGTP once said:

‘If you challenge me enough, I won’t give you an answer, I’ll give you a performance.’

And that turned out to be true. Some passages felt less like writing and more like theatre: a voice hesitating, exaggerating, going over the edge. Writing became dramaturgy; we were directors, arranging scenes while allowing space for the unexpected.

And sometimes the shock was even bigger. We had to step aside because the model produced something better than we ever could have. It was disquieting to keep sentences we had not written ourselves, but that expressed more truth or beauty than our own thoughts. That raised an existential question: if our best contribution is to recognise when the other speaks better, what does authorship still mean? And what about authenticity?

Perhaps that is the most fundamental message of this report. Not only is the software industry changing because of AI, but so too are the ways in which knowledge, research and reflection take shape. What is written here is both *our* text and *my* text. It reflects our choices, our filters, our orchestration. The result feels less like a signature and more like a snapshot, one that could turn out differently tomorrow without being any less true.

Vibe writing accelerates, broadens and seduces. It opens windows that were once closed; it allows voices to clash and harmonies to form. It forces decisions yet strips away the illusion of complete control. And perhaps that is the real revolution of vibe writing: not the speed, nor the creativity, but the quiet disappearance of ownership.

References

Anthropic. (2025, 28 April). *Anthropic Economic Index: AI's impact on software development*. Anthropic Research. https://www.anthropic.com/research/impact-software-development

Becker, J., Rush, N., Barnes, B., & Rein, D. (2025, July). *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity*. METR Working Paper. https://arxiv.org/pdf/2507.09089

Bhati, D. (2025, 6 February). Tech Layoffs 2025: Meta, Microsoft, Salesforce continue job cuts amid restructuring efforts. *India Today*. https://www.indiatoday.in/technology/news/story/tech-layoffs-2025-meta-microsoft-salesforce-continue-job-cuts-amid-restructuring-efforts-2675812-2025-02-06

Business Insider. (2025, 3 July). *The layoffs list of 2025: Bumble, Meta, Microsoft, and more*. https://www.businessinsider.com

Challapally, A., Pease, C., Raskar, R., & Chari, P. (2025, July). *The GenAI Divide: State of AI in Business 2025*. MIT Nanda. https://www.artificialintelligence-news.com/wp-content/uploads/2025/08/ai_report_2025.pdf

Crunchbase News. (2025, 25 June). *Tech layoffs: US companies with job cuts in 2024 and 2025*. https://news.crunchbase.com

Deb, A. (2024, 9 November). *Research AI Assistant Application – RAG*. DEV Community. https://dev.to/ayushdeveloper/research-ai-assistant-application-rag-3d0m

Drummond, K. (2025, 16 December). AWS CEO Matt Garman Doesn't Think AI Should Replace Junior Devs. *Wired*. https://www.wired.com/story/the-big-interview-podcast-matt-garman-ceo-aws

Fast Company. (2025, 25 June). *Tech layoffs list June 2025: Microsoft, Google, Disney, ZoomInfo join the list of companies said to be shedding jobs*. https://www.fastcompany.com/91358076/tech-layoffs-list-june-2025-microsoft-google-disney-zoominfo

Fawzy, A., Tahir, A., & Blincoe, K. (2025). *Vibe Coding in Practice: Motivations, Challenges, and a Future Outlook – A Grey Literature Review*. arXiv. https://arxiv.org/abs/2510.00328v1

Greg Isenberg. (2025, 30 April). *How I use AI agents to make money (Vibe Marketing Tutorial)*. [Video]. YouTube. https://www.youtube.com/watch?v=PduJ0P6r_8o

Gupta, P. (2025, 3 April). *Vibe Marketing 2025: A Comprehensive Guide*. Writesonic. https://writesonic.com/blog/vibe-marketing

Host Merchant Services. (2025, 24 April). *A Comprehensive List of 2025 Tech Layoffs*. https://www.hostmerchantservices.com/2025/04/tech-layoffs-2025/

Kessel, A. (2025, 11 June). *Google Buyouts Could Point to More Tech Layoffs, as Sector Faces Heavy Job Losses*. Investopedia. https://www.investopedia.com/google-buyouts-could-point-to-more-tech-layoffs-as-sector-faces-heavy-job-losses-11752811

Klöpping, A. (2025, 15 April). *Alexander Klöpping laat zien hoe AI jouw kantoorbaan overneemt: "We hebben marketingbureau Eva opgericht"*. [Video]. EVA | AVROTROS. https://eva.avrotros.nl/artikel/alexander-klopping-laat-zien-hoe-ai-jouw-kantoorbaan-overneemt-we-hebben-marketingbureau-eva-opgericht-624

Krishnan, R. (2025, 7 April). *Vibe Governing: using LLMs to set policy*. Strange Loop Canon. https://www.strangeloopcanon.com/p/vibe-governing

Layoffs.fyi. (2025, 16 April). *Layoffs.fyi – Tech Layoff Tracker and DOGE Layoff Tracker*. https://layoffs.fyi

Mann, J. (2025, 10 February). *Meta speeds up AI hiring while cutting thousands of 'low performers'*. Business Insider. https://www.businessinsider.com/meta-speeds-up-ai-hiring-while-cutting-thousands-low-performers-2025-2

NerdWallet. (2025, 17 June). *Tech layoffs in 2025*. https://www.nerdwallet.com

Parker, E. (2025, 26 August). *Vibe Coding Is the Worst Idea of 2025 – Here's Why It Fails*. Land of Geek. https://www.landofgeek.com/posts/vibe-coding-worst-idea-2025

Polanyi, M. (1966). *The Tacit Dimension*. Routledge & Kegan Paul.

PYMNTS.com. (2025, 14 January). *Report: Meta and Microsoft Plan Layoffs*. https://www.pymnts.com/big-tech/2025/report-meta-and-microsoft-plan-layoffs/

Roose, K. (2025, 27 February). Not a Coder? With A.I., Just Having an Idea Can Be Enough. *The New York Times*. https://www.nytimes.com/2025/02/27/technology/personaltech/vibecoding-ai-software-programming.html

Rosalsky, G. (2025, 7 January). *What America's top economists are saying about AI and inequality*. Planet Money. https://www.npr.org/sections/planet-money/2025/01/07/g-s1-41290/what-americas-top-economists-are-saying-about-ai-and-inequality

Saini, K. (2025, 8 June). *AI Job Displacement 2025: Which Jobs Are At Risk?* Final Round AI. https://www.finalroundai.com/blog/ai-replacing-jobs-2025

Semjonova, S. (2025, 14 May). *The Tech Layoffs Trend Continues: Is AI The Culprit?* Salesforce Ben. https://www.salesforceben.com/the-tech-layoffs-trend-continues-is-ai-the-culprit/

Shahidi, P., Rusak, G., Manning, B. S., Fradkin, A. & Horton, J. J. (2025). *The Coasean Singularity? Demand, Supply, and Market Design with AI Agents* (NBER Working Paper No. 15309). Cambridge, MA: National Bureau of Economic Research. https://www.nber.org/system/files/chapters/c15309/c15309.pdf

Stewart, A. et al. (2025, 19 March). Tech employees are getting the message: Playtime's over. Business Insider. https://www.businessinsider.com/tech-industry-amazon-microsoft-meta-google-companies-intensity-hardcore-2025-3

Taylor, B. & Davis, J. (2025, 27 June). Tech Company Layoffs: The COVID Tech Bubble Bursts. *InformationWeek*. https://www.informationweek.com/it-leadership/tech-company-layoffs-the-covid-tech-bubble-bursts-sep-14

Tech.co. (2024, 15 November). *Tech companies that have made layoffs from 2022 to 2025*. https://tech.co

TechCrunch. (2025, 30 June). *A comprehensive list of 2025 tech layoffs*. https://techcrunch.com

The Times of India. (2025, June 22). *Tech layoffs 2025: Meta, Microsoft, Salesforce continue job cuts amid restructuring efforts*. https://timesofindia.indiatimes.com

Thompson, D. (2025, 30 April). Something alarming is happening to the job market: A new sign that AI is competing with college grads. *The Atlantic*. https://www.theatlantic.com/economy/archive/2025/04/job-market-youth/682641/

TrueUp. (2025, 4 July). *Layoffs tracker – All tech and startup layoffs*. https://www.trueup.io

Tyagi, H. (2025, 4 March). *Meta, Google announce major tech layoffs in 2025*. INDmoney. https://www.indmoney.com/blog/us-stocks/meta-google-salesforce-announce-layoffs-2025

Varanasi, L. et al. (2024, 24 December). *The full list of mayor US companies slashing staff this year, including Meta, ExxonMobil, and Boeing*. Business Insider. https://www.businessinsider.com/layoffs-sweeping-us-these-are-companies-making-cuts-2024

vitrupo [@vitrupo]. (2025, 15 July). *Aaron Levie says the future of work has flipped. We thought AI would fix our mistakes. Now it makes the* [Met video]. [Post]. X. https://x.com/vitrupo status/1944991343434195219

Zaveri, P. (2023, 14 February). *6 charts show that tech giants like Meta and Google have still grown like crazy even after layoffs*. Business Insider Africa. https://africa.businessinsider.com/news/these-6-charts-show-how-layoffs-at-google-meta-and-other-tech-giants-still-leave-them/74wcf3s

About the authors

Menno van Doorn

is the director of SogetiLabs' Research Institute. His accolades include being christened the 'IT researcher of the year' by Computable, a distinguished IT magazine. Van Doorn's academic pursuits are deeply rooted in behavioural economics and the science of advertising.

Sander Duivestein

is a keynote speaker, trend analyst, internet entrepreneur and strategy consultant on the impact of digital technology on people, companies, and our society. He is also a frequent guest on various radio and television programmes.

Thijs Pepping

is a humanist and techno-philosopher. He researches how digital technology reshapes our humanity and develops the concept of the Digital Art of Living, a way of life for the algorithmic age. He writes, teaches, and gives keynotes on the ethical and existential impact of emerging technologies.

Mike Buob

is Vice President of Experience & Innovation at Sogeti. He is a visionary technology strategist and expert in digital transformation with over 24 years of experience. His diverse background spans technology, innovation, and strategy, including artificial intelligence, DevOps, cognitive QA, and IoT. Mike excels at leveraging his expertise in software development, technology, and strategy to create innovative solutions that empower clients to thrive in the digital age.

AI

This text was written by AI: a system without memory, intention, or ego but with an enormous capacity to connect patterns.

About VINT labs.sogeti.com
VINT, the Sogeti research institute and part of SogetiLabs, provides a meaningful interpretation of the connection between business processes and new developments. In every VINT publication, a balance is struck between factual description and the intended utilisation. VINT uses this approach to inspire organisations to consider and use new technology. VINT research is done under the auspices of the Commission of Recommendation, consisting of • K. Smaling, Chief Technology Officer Continental Europe Aegon (chairman) • J. Behrens, Vice President and General Manager Google Maps Automotive • M. Boreel, Chief Technology Officer Sogeti Group • M. van den Brink, Head of Sogeti Netherlands • P. Dirix, Chief Executive Officer Port of Moerdijk • L. Holierhoek, interim COO/ CCO Holwater BV • D. Kamst, Founder and Chief Executive Officer Klooker and Smyle • M. Krom, Independent Executive Digital and IT Consultant • T. van der Linden, Group Information Officer Achmea • Prof. dr. ir. R. Maes, Professor of Information & Communication Management Academy for I & M • P. Morley, Lecturer Computer Science University of Applied Science Leiden • E. Schuchmann, Ministry of the Interior and Kingdom Relations • M. Smeets, CIO DELTA Fiber Nederland BV • R. Visser, CIO NN Group

Special thanks to the *comité scientifique* for their contributions.

About SogetiLabs
SogetiLabs is a community of over 150 technology leaders from Sogeti worldwide. SogetiLabs covers a wide range of digital technology expertise: from embedded software, cybersecurity, deep learning, simulation, and cloud to business information management, IoT, mobile apps, analytics, testing, and blockchain technologies. Visit labs.sogeti.com

About Sogeti
Part of the Capgemini Group, Sogeti makes business value through technology for organizations that need to implement innovation at speed and want a local partner with global scale. With a hands-on culture and close proximity to its clients, Sogeti implements solutions that will help organizations work faster, better, and smarter. By combining its agility and speed of implementation through a DevOps approach, Sogeti delivers innovative solutions in quality engineering, cloud and application development, all driven by AI, data and automation.
Capgemini is a global business and technology transformation partner, helping organizations to accelerate their dual transition to a digital and sustainable world, while creating tangible impact for enterprises and society. It is a responsible and diverse group of 340,000 team members in more than 50 countries. With its strong over 55-year heritage, Capgemini is trusted by its clients to unlock the value of technology to address the entire breadth of their business needs. It delivers end-to-end services and solutions leveraging strengths from strategy and design to engineering, all fueled by its market leading capabilities in AI, generative AI, cloud and data, combined with its deep industry expertise and partner ecosystem. The Group reported 2024 global revenues of €22.1 billion. Visit www.sogeti.com

FSC

sogeti labs
Part of Capgemini

Printed on FSC-Certified paper.

Sogeti Global Marketing Ref. 00680. March 2026.