

# The Future of (Public Sector) Product Management in a Vibe-Coded World

---

*by* Ben Welby



## PRINT EDITION

A dedicated print-first edition of the web text, composed as a publication rather than exported as a webpage.

# Edition note

---

This edition is set from the material published at [vc-product.wel.by](http://vc-product.wel.by) and rebuilt as live HTML/CSS so the cover, headings, spacing and typography render cleanly in PDF.

Rather than reproducing the website page for page, it treats the material as a booklet: cover, edition note, contents, chapter openers and continuous reading pages, all composed for A4 print.

### AUTHOR

Ben Welby

---

### SOURCE TEXT

[vc-product.wel.by](http://vc-product.wel.by)

---

### TEXT BASIS

WordPress export

---

### COMPOSITION

HTML/CSS print template

---

### TRIM

A4


---

### OUTPUT

PDF

---

Built around the cover treatment and extended into a restrained editorial system for print.



## CONTENTS

# Table of contents

---

<b>FOREWORD</b>	The kairos moment and the changing material
<b>01</b>	Why vibe-coding matters (and what it isn't)
<b>02</b>	What changes for product management
<b>03</b>	Steady standards in a fast world
<b>04</b>	Governance in the whole system
<b>05</b>	Towards a Highway Code for Digital?
<b>06</b>	Reimagining the team and its roles
<b>07</b>	How we organise when building is cheap
<b>08</b>	Stages as lenses: learning, testing, and scaling responsibly
<b>09</b>	User research in the age of instant prototypes
<b>10</b>	From tasks to outcomes
<b>11</b>	Buying change, not a plan
<b>12</b>	Conclusion: the craft of confidence
<b>ABOUT</b>	About the author

---

FOREWORD

# Foreword: The kairos moment and the changing material

---

Public service teams are at a *kairos* moment: a time when a new technical capability invites us to rethink almost everything about how we build and deliver services. But our centre of gravity isn't swayed by "AI magic". It rests on civic values we already know to be true: start from user needs; follow evidence over hunches; prize outcomes over outputs; prefer pragmatism to hype; fix the basics; and above all, put human worth and inclusion at the core. These are the ground we stand on when anything new arrives.

The "material" of digital development itself is changing. AI-assisted development collapses the time between intent and software: we can describe what we want in natural language and see working systems emerge at remarkable speed. Over the last year, this has moved from novelty to normality. Teams can now go from an idea to software in hours rather than weeks.

You'll sometimes hear this called *vibe coding*. In the rest of this document I'll use that label sparingly, because the bigger shift is not a vibe - it's a capability: AI woven through the disciplines we already rely on, from engineering to policy to operations.

The biggest change is not simply that engineers can go faster — it's that *more people can now shape software as a medium*. And that medium won't always look like a page in a browser.

And it will arrive unevenly. Some people will meet us through chat and apps; others through phones, letters, Jobcentres and third-party front doors. Delivering omni-channel services means being agnostic about channels but highly opinionated about ensuring unevenness doesn't turn into unfairness.

Policy, ops, analysis and content can now express intent as something runnable, not merely describable. That collapses much of the old translation layer where ideas waited weeks to become concrete. The consequence is profound: multidisciplinary teams don't just collaborate around plans; they collaborate around deployed code. This will change how teams work, how roles overlap, and what "multidisciplinary" looks like in practice.

That's why our rigour matters. This isn't about knocking together flimsy prototypes or automating away the craft of coding; it's about accelerating responsible development. Used well, with peer review, testing, accessibility, security and operational thinking baked in, AI assistance can produce production-grade software quickly and safely.

But the point of this shift isn't *faster code*. **The point is quicker value in real users' hands**, consistent with the Service Standard: showing the thing sooner, learning sooner, and iterating based on evidence.

There is a particular hazard in this moment: when software can be conjured quickly, it becomes easier for leaders to confuse a working thing with a ready thing. A prototype can look like a service. It can feel "almost done". And that is exactly when public sector judgement matters most, because the distance between a demo and a duty-of-care service is not measured in pixels. It's measured in evidence, operability, accessibility, lawful data handling, and the humility to say "not yet" even when the screen looks convincing.

I've tried to write this as a manual for the world we're in rather than to catalogue fast-changing tools. Digital services may be becoming more malleable, but our responsibility is to shape them toward human benefit. This document aims to help the public sector do that - harnessing the potential without losing what makes our services safe, fair, and trustworthy.

CHAPTER 01

# Why vibe-coding matters (and what it isn't)

---

## A compression worth caring about

AI assistance matters for public sector product management because it collapses the time between intent and something real. It's not just that delivery teams can move faster; it's that more of the multidisciplinary team can express ideas as working software early enough to make contact with reality while the stakes are still low.

What used to take months can now be achieved in days. A concept that might once have languished in a backlog can get built in a morning and tried with real users by the afternoon.

This acceleration matters because improving services can tangibly improve people's lives – getting a payment to someone sooner, reducing avoidable contact, making a form less frustrating, giving caseworkers tools that load straightaway. If we can do that faster, we should.

But it's worth being clear about what's actually changing, because there are two accelerations happening at once, and one deeper consequence.

## Two accelerations, one consequence

First, those who wouldn't call themselves technical can increasingly build runnable, testable services - not just describe them. Product managers can bring an end-state into view instead of making slideware. Policy colleagues can express intent as rules you can run and levers you can pull. Ops can sketch an operating model as a walkthrough you can actually step through. Content can generate variants and see how they behave in context. Those things used to wait on a dev team to translate them. Now they can exist early, while the questions are still alive.

Second, engineers can use the same tools to compress delivery and feedback loops: less boilerplate, less waiting, more iterations with proper tests and controls. That doesn't diminish engineering; it changes where the effort goes. Some of the typing shrinks, but the craft of making things robust, secure, accessible, observable and maintainable becomes more central.

There was once a chasm between "a sketch on the wall" and "something real in front of users". AI-assisted delivery narrows it from both sides. The outcome is not that engineering disappears; it's that the whole team can collaborate in the medium of software sooner, and therefore argue less about hypotheticals and more about what actually happens.

And that's the deeper consequence: the unit of delivery becomes even more plainly the team. Not policy then delivery. Not design then build. Not a relay race of documentation. A multidisciplinary team working in the same medium, able to get to something testable quickly, and then to learn together from what reality feeds back.

## Where misunderstanding creeps in

This is the moment where some people reach for the phrase *vibe coding*, and it's exactly where misunderstandings creep in. From here on, when I'm talking about the disciplined practice, I'll mostly say **AI-assisted delivery** - and I'll keep *vibe coding* for the cultural shorthand and the low-rigour end of the spectrum.

And vibe coding also isn't synonymous with flimsy "demo-ware". Some people hear the phrase and picture shiny-but-hollow prototypes: impressive screenshots that crumble under real policy, real data, real accessibility needs, and real operational constraints. That is one end of a spectrum, but it's not the whole story. Yes, plenty of vibe-coded artefacts are throwaway sketches, but AI assistance can also generate code that's close enough to production to warrant taking seriously.

Other phrases are starting to emerge - and *agentic engineering* is a good one for what's happening in the more narrow confines of the engineering discipline: orchestrating agents, scrutinising output, keeping the quality intact. But what I'm trying to name is the operating model for public services: the way we work collectively to work in the same medium and add public value more quickly. In government the 'clever' bit is rarely the code. So that's why I'm using **AI-assisted delivery**. Because this is a team pursuit.

## The bar doesn't move

AI-assisted delivery isn't theatre, and it isn't a shortcut; it's a powerful new tool in our toolkit - but only when used responsibly and intelligently. It doesn't eliminate the need for empathy, context, or wise decisions. If anything, it amplifies them: when code can be produced at the speed of thought, the limiting factor becomes whether we are building the right thing and whether we are

upholding our duties to users. It can supercharge delivery, but it only delivers public value when it is guided by sound judgement, inclusion, and evidence.

It isn't a licence to bypass hard product thinking or abandon good practice. The real work was never just writing code: it's making sense of what people need, agreeing the right response, and then proving - through evidence and iteration - that we've met it. That proof is still the priority; because when you can build quickly, you can also build the wrong thing quickly.

The dividing line isn't the tool; it's whether the work meets the bar: reviewed, tested, accessible, secure, supportable, measurable, and safe to put in front of the public.

So the discipline becomes: what is this for? If it's experimentation, treat it like an experiment: learn, document, discard. If it's headed towards live, then build it as if it will have to survive contact with reality - because it will.

Another way to put it: we now have a more powerful engine in the delivery vehicle. It can dramatically increase our speed, but the steering wheel remains in human hands - guided by the Service Standard, by user research and iteration, and by the hard work of navigating policy, risk, and operational reality. AI assistants can generate code; they cannot own consequences. They do not reliably know what is ethical, accessible, or appropriate in context. That accountability remains ours.

So this is not a replacement for human-driven product management and service design; it's an augmentation. It can collapse the time between idea and reality - so long as we keep checking those realities are desirable, fair, and effective.

Public sector teams can now potentially respond to change faster than ever: a new policy intent or emerging user need can be met with something working in days, and tested early with real users - in the open, where assumptions can be challenged. That rapid policy-to-delivery cycle is a dream of digital government - now it's

within reach. But to seize it, we have to dispel any notion that vibe coding is “cheating” or “hype” or just “demo-ware”. It’s neither a silver bullet nor a trick; it is simply the next evolution of agile tooling.

It still requires discipline.

High speed doesn’t change the core of what we do: do our users benefit, and do outcomes improve? AI assistance increases our capacity to try; product leadership must increase our capacity to learn: fast feedback loops, effective measurement, and firmly held standards. It can let us practice our core values faster, which is a compelling proposition for public good, but it isn’t a shortcut around those values. If anything, the faster we go, the more carefully we must steer.

## **When ‘the service’ isn’t a webpage**

There is another shift underneath all of this: what we mean by “the service” is becoming less stable.

In the GDS era we earned trust by getting good at the web: clear journeys, fewer dead ends, pages that matched policy, transactions that worked. That work still matters. The public will still meet government through letters, phones, walk-ins, and third-party front doors.

But increasingly the thing that changes people’s experience won’t always be “a service” in the familiar sense. It might be rules expressed as code. An API used inside someone else’s interface. A shared component that quietly shapes outcomes across multiple journeys. An agent operating through a channel we don’t own. Sometimes it will be invisible unless it fails.

That changes the craft. We're no longer only designing pages and flows. We're designing behaviour in systems: how decisions get made, how exceptions get handled, what happens on a bad day, and how trust is sustained when the user never sees the machinery.

So yes: build the front door well. But treat the deeper layers as “the service” too, because that is where fairness, explainability, and operability either exist...or don't.

CHAPTER 02

# What changes for product management

---

The section above widens what we mean by “the service”. This section brings it back down to the practical question: what changes in the day job when building stops being the main constraint?

## **Pace changes the rhythm**

As AI-assisted delivery becomes normal, the most obvious change is **pace**. The tempo of delivery can increase by an order of magnitude: work that used to fill a two-week sprint may be doable in a morning, and features that once spanned months can land in days. The cadence starts to feel less like a relay race and more like a continuous flow. You might take an idea into a Monday stand-up and have it in front of users within that week.

That speed forces a rethink of standard product routines because assumptions become runnable, testable, and evidenced far sooner, and because the pressure to call something ‘done’ will arrive earlier from those inspired by your work.

## **Backlogs stop being the main artefact**

One immediate implication is how we handle backlogs and planning. When it’s quicker to build and test something than to debate it or park it in the icebox, the backlog changes shape. Long lists of user stories waiting for developer attention can shrink, with

the backlog refocusing on higher-level problems and outcomes. The small stuff - minor bugs, content tweaks, low-risk improvements - can be handled on the fly, by whoever spots the issue, rather than waiting for the next sprint.

This means product teams can live up to their ambitions: curating the right problems to solve, rather than prioritising every trivial fix. Product management becomes more about managing a flow of experiments, and less about tending a long to-do list.

### **When delivery gets cheaper, direction gets pricier**

Another change is the relationship between strategy and roadmaps. When teams can accomplish more in a given period, direction becomes even more crucial.

### **The strategic role of product doesn't go away – if anything, it looms larger.**

In a quarter, an AI-assisted team might deliver what would once have taken a year. Without a clear north star, that speed becomes chaos or wasted effort.

But product clarity is not conjured in a vacuum. A team can run an outcome-led rhythm brilliantly and still be trapped inside an organisation that cannot decide what it is trying to achieve. In a strategy-poor system, AI won't create progress, it will just spin the wheels. If building gets cheaper, then mission becomes more precious: a shared theory of change that connects digital capability to real-world impact, not just a shopping list of features.

So product managers must double down on outcomes: articulating what success looks like and measuring progress toward it.

This isn't a new idea. Outcome-led roadmaps have always been the point. AI-assisted delivery makes good practice easier to sustain. Roadmaps become less a dated list of features and more a set of ambitious outcomes (e.g. “Reduce processing time for X from 2 weeks to 2 days” or “Increase uptake of the service by 20%”), with

the team rapidly experimenting to get there. The roadmap becomes a living hypothesis, reshaped by early evidence, updated often, and held together by clarity of purpose.

### **Blurred boundaries, not vanished roles**

Crucially, the product manager's role in an AI-assisted team leans further into facilitation and integration. Multidisciplinary teamwork remains essential, but boundaries blur: anyone in the team might now be expressing intent as working software. Engineers harden what's worth keeping and make it robust, secure, accessible, observable and maintainable. The hierarchy of who does what becomes less rigid - it's all hands on deck to respond to feedback as it arrives. Product managers still guide the orchestra, while playing some notes themselves.

One might worry: does this mean we no longer need as many people, or certain roles?

Teams may be smaller or differently composed. If a pair can do in a day what once took several people a couple of weeks then the economics of team size will change. But AI-assisted delivery doesn't remove the need for the essentials, it rebalances them. User research, design, technical quality, content, and data still need clear ownership. The difference is that individuals and teams can cover more than one base at times, with less waiting and fewer handoffs. A designer and developer, supported by AI tooling, might produce functional code in hours, with researchers testing it with users the next day. The product manager's job is to keep that faster cycle rooted in user needs and outcomes, and to make sure that speed doesn't turn into burnout or loss of direction.

### **Risk moves faster too**

Another shift for product management is the nature of risk and evidence. Faster building compresses the learning cycle, but it also compresses the window between a mistake and real-world impact. The product manager's mindset must remain vigilant about quality

and outcomes. We must ask not just “Can we build it by Friday?” but “Should we build it at all, and what will tell us if it’s working?” This is nothing new: it’s the discipline of hypotheses and success metrics, now running at a higher cadence. With rapid deployment and good instrumentation, we can see real effects in days and decide with more confidence.

But it remains the product manager’s job to act on that evidence promptly: stop what isn’t working, double down on what is, and explain the why to stakeholders who may be astonished at how quickly things are moving.

### **The job is still the job**

In sum, product management in a vibe-coded world is still about **making sure the right thing gets built** and delivers value. The core questions haven’t changed: “Is this solving a real problem? How will we know? What’s the smallest thing we can do to learn more? How does this align with our mission?”

What’s changed is the context in which we answer them - building isn’t the bottleneck anymore. The product manager becomes the real-time translator between vision, evidence, and execution, aligning these new capabilities with user need and policy intent.

CHAPTER 03

# Steady standards in a fast world

---

Speed may be the headline of a vibe-coded world, but trustworthiness remains the bottom line. When we can deploy changes with unprecedented pace, our standards and guardrails matter more than ever. If you're going to drive faster, your brakes and seatbelts need to be reliable. We must uphold, and in places strengthen, our quality benchmarks, so velocity yields faster **value**, not faster failures.

The UK Government's Service Standard (alongside equivalent guidelines in other countries) is the foundation here. If delivery accelerates, its criteria don't get diluted just because we can move quickly. On the contrary, they become load-bearing. Any adoption of AI-assisted delivery should happen *within* the Service Standard, not around it or in defiance of it.

And if that's the bar, then the next question is practical: where does that bar live, day to day, when more people can ship?

## **Standards have to live in our defaults**

A vibe-coded world changes where standards have to live. When more people are getting their hands dirty in the codebase, then we need to find ways to ensure quality is baked into the defaults of our tools, including the prompts, policies, and guardrails we give to AI agents, as well as the working practices of the team itself.

So even if we can now spin up a working service in a day, we still meet the basics before it is in front of real users. If AI helps to write code, it still goes through review, testing, and continuous integration. Accessibility, security, data protection, and operational readiness don't become optional because the tool is quick. In fact, they become more central, because you can now change more, more often.

The good news is that AI can strengthen the basics too: generating test cases, improving monitoring, and helping teams spot issues earlier. But none of that removes judgement. It just makes it harder to justify not doing what we already know protects our users.

## **Make the safe path the fast path**

If we want standards to hold at higher speed, we have to stop treating safety as an extra step. Modern platform infrastructure can make the safe path the fast path: secure, scalable environments with the right defaults baked in; one-click deploys; monitoring and logging; feature flags; and easy rollback.

We don't want a world where a team can build an MVP in a day, but then spend three months waiting for security clearance or someone to provision an environment. Keeping standards steady is not only a team discipline. It is an organisational one. It means investing in enabling systems - platforms, pipelines, automated testing, and operational tooling - so that the path of least resistance is also the right one.

In many public sector organisations, the bottleneck to rapid iteration isn't writing the code. It's the surrounding bureaucracy and legacy infrastructure. Overcoming that is as much culture and leadership as it is tech: insisting that governance and delivery pipelines adapt, and making it easy to deliver secure, accessible, documented and supportable change.

### **“Done” does not get cheaper just because “build” does**

A fast world tests discipline. There will be temptations to skip steps: “It's just a trial, can we bypass some tests or design reviews to save time?”. The discipline is to resist. If we use AI to generate code, we can also generate tests and run them immediately. If we release to a small percentage of users, we put analytics and feedback channels in place from day one. Our definition of “done” does not soften because “done” arrives sooner. Every change still has to meet the bar: does it work for users, is it accessible, and does it operate as well on a bad day as a good one?

The most dangerous moment is often the one where it looks finished to a busy senior person, while the team can still see all the ways it isn't ready to carry the public.

And “done” has consequences beyond the deployment. It has a human dimension. AI can make delivery faster; it cannot make people inexhaustible. If speed is coming from heroics and burnout, then we haven't become more capable. We've just shifted the cost onto individuals, and it will surface in the health of the team and the reliability of the service.

“Done” also has a footprint as well as a finish. AI-assisted delivery leans on compute, and compute has real costs: financial, environmental, and sometimes geopolitical. Stewardship means we don't treat that as someone else's problem. Be deliberate about where we use these tools, avoid wasteful patterns just because iteration is easy, and make it normal to ask: *what are we spending - in money, energy and dependency - to get this result?*

## Trust is what people take away

Faster does **not** mean sloppy. Faster means more value delivered so *long as* quality is maintained. If speed comes at the cost of excluding users, exposing data, or confusing people, it's not progress at all. So our job is to ensure that doesn't happen.

Because the broader implication of steady standards is public trust. Citizens may not know anything about vibe coding or agile or AI, but they certainly experience the outcomes. If government services start improving visibly week by week, that can build confidence in public institutions' responsiveness. But if things break faster, or change unpredictably without proper support, then trust erodes.

Keeping standards means keeping the implicit social contract: we move quickly **and** fix things. We innovate **and** uphold reliability. That balance reassures the public, and those who oversee our work, that we aren't adopting new tools as a reckless gamble but as a thoughtful evolution.

Faster practices demand a level head and a steady hand. By holding firm to service standards and our ethics, we can ensure increased velocity translates into increased public value, not just activity. We haven't taken our eyes off the destination: services that are accessible, secure, accountable, and centred on the people who need them. We just have an opportunity to reach that destination sooner.

CHAPTER 04

# Governance in the whole system

---

## Governance can't stay remote

When delivery speeds up and teams become more fluid, governance cannot remain static or remote. In the public sector, 'governance' isn't a technicality – it's how democratic accountability, legal compliance, and ethical oversight show up in practice. So it needs to evolve in step with AI-assisted delivery to cover the *whole system* of service delivery, end-to-end and top-to-bottom: from code quality and day-to-day delivery all the way up to ministerial intent and public legitimacy.

Fast, AI-enabled delivery won't succeed if approvals, audits, funding decisions, and policy alignment still operate on six-month or annual cadences. And the challenge is no longer only "teams can ship faster". It's that **more of the team can now ship safely**. When policy, ops, analysis and content can all express intent as runnable code, governance has to sit inside multidisciplinary work: continuous, not episodic.

## From sign-off to a trail

So we need to move away from governance as periodic gatekeeping towards governance as ongoing collaboration - not as a softer version of assurance, but as a more practical one. Agile already pushed us in this direction, but AI assistance makes it unavoidable. When teams can iterate in hours, a distant sign-off becomes both

bottleneck and risk: you can ship multiple iterations before the next governance set-piece. The only workable pattern is for those functions to be represented within or alongside the team from the start.

This is also the most pragmatic way to protect senior accountability in an AI-assisted world. When the cadence speeds up, the old model doesn't produce certainty; it produces theatre - confidence manufactured through paperwork while the real risk sits in what nobody has looked at. Continuous governance gives leaders something better than reassurance: a live view of exposure, evidence, and risk. It replaces "sign-off as a moment" with "confidence as a trail".

## **Governance moves left**

In practice, that means the *whole system* is present in the room - physical or virtual - where decisions are made. This is multidisciplinary work in its fullest sense: delivery and legitimacy, side by side. Policy, legal, security, finance and operations don't hover outside the team as occasional sign-offs; they work alongside delivery, continuously. When the team is shaping an improvement, the policy lead can clarify the intent ("what Parliament intended here is X") and help turn it into testable criteria. And when an AI-generated change touches personal data, privacy and security concerns can be flagged - and mitigated - before anything goes live. Done well, this doesn't slow teams down; it removes rework and waiting - and if your team already enjoys these ways of working, AI assistance simply lets you go further, faster.

But governance also has to "move left" into the machinery. If more people can produce deployable code, then quality can't depend on individual heroics or after-the-fact review. Guardrails have to live in defaults: the platform, the pipeline, the component library, access controls, feature flags, and patterns for prompting and testing. In other words: governance becomes something you **do** through the system, not something you **attend** in a meeting.

There's a close cousin here in terms of AI safety too: if responsibility isn't built into the work, it doesn't exist, it just turns up later as harm.

## Conditions for telling the truth

The leadership task in a high-speed medium is not to demand launch. It is to create the conditions where teams can tell the truth. That means rewarding the call that something isn't ready. It means asking for evidence, not theatre. It means funding capability, not squeezing headcount. And it means treating "no" and "not yet" as part of responsible pace, not obstruction.

## Policy as something you can run

Whole-system governance also recognises the political context. Public sector products do not exist in a vacuum; they operate in a landscape of ministers, policies, and public opinion. In a rapid development environment, teams can pivot quickly to emerging ministerial priorities or urgent user needs - but doing so responsibly requires tight coupling with the policy-making side, who should also be working with these tools. Governance here is not just "compliance" but *alignment*: ensuring the service reflects legitimate policy goals and that policymakers understand the trade-offs and evidence coming back from the service.

And the opportunity is bigger than faster implementation. If we can prototype services at pace, we can also prototype *policy*: building simulations and decision engines with levers you can pull, assumptions you can inspect, and models you can adjust – instead of trying to compress complexity into twenty slides. That opens up a different kind of conversation between policy and delivery: not "here is the deck", but "here is the engine – here are the choices, here is what we think happens, do you want to explore it and leave your feedback?" This is the time to deliver on the promise of rules as code.

Political leadership needs to be kept informed and that cadence should increase. They may not want daily updates on prototypes, but they will benefit from engaging with Show and Tells, and seeing what the team has tested, what has been learned, and what the models assume. In turn, political decisions - like changing an eligibility rule - can be modelled and then implemented and trialled in near-real-time. But that requires trust from governance bodies that speed won't create chaos or injustice, and confidence that the modelling is transparent, testable, and grounded in evidence. Building trust has always been a critical part of the job. The non-negotiable question now is whether we are setting a higher bar for trustworthiness as we go faster.

### **Shared fluency, shared legitimacy**

Another pillar of whole-system governance is shared digital fluency. When policy, legal, finance, and other partners are intertwined with delivery, it helps enormously if they have a working grasp of digital concepts and AI capabilities. And it works the other way too: delivery teams need fluency in risk, law, operational reality, and public value. This is a cultural exchange: the AI-assisted future works best when everyone has some of each other's literacy. Training and cross-pollination matter - short rotations, shadowing, joint reviews, shared exposure to data and decisions - so that when people come together to govern a fast-moving project, they can speak a common language.

Governance in a vibe-coded world stops being a separate layer that periodically scrutinises the work, and becomes part of how the work is done every day. It's governance as an *activity* rather than an *event*: automated checks in delivery pipelines; multidisciplinary workshops that set guardrails and success criteria; and transparency about what's being built and learned. Whole-system governance means speed doesn't cut corners, because the people responsible for those corners are in the room when decisions are made, not invited in afterwards.

The paradox is that smart governance doesn't slow delivery; it enables it. Clear guardrails, explicit thresholds for scaling, and honest evidence give teams permission to move quickly without becoming reckless. That is how we keep speed honest - and why governance has to live inside the work, not outside it. It's delivery and oversight working hand in hand, so speed serves legitimacy, and legitimacy serves better, safer outcomes for the public.

CHAPTER 05

# Towards a Highway Code for Digital?

---

If AI-assisted delivery changes the physics of how we build, then the bigger question becomes how we *govern* that new pace safely. As we navigate this higher-speed world, perhaps it is helpful to think in terms of a living **Highway Code for Digital** – a shared set of rules, signals and expectations that everyone in public service abides by.

## **Everyone is a road user**

The Highway Code doesn't just apply to drivers; it covers pedestrians, cyclists, and passengers too. Everyone who uses the road system shares a baseline of understanding about how it works and what keeps it safe.

Digital should be the same.

And it has to be living. Not a laminated poster, but a maintained set of patterns, defaults and expectations that evolves as the terrain changes, like a design system or a platform. The point is not to freeze the world, or prescribe a curriculum. It's to give people enough shared grammar to move quickly and enough curiosity to keep updating it.

Whether you're a policy adviser drafting regulations, a director signing off a budget, or an engineer deploying a service, you are a road user in the digital state. You have duties and rights. You need to know the basics of how the system functions and how your actions affect others.

## **Not everyone needs to code, but everyone needs judgement**

This is not about turning everyone into coders. It's about setting a shared civic expectation: that all public servants understand the digital terrain they operate in well enough to make informed, responsible decisions. It's the difference between being a passenger who knows the car moves, and someone who understands the road system well enough to act responsibly within it - whether you're driving, cycling, or crossing at the lights.

That baseline literacy is what keeps the whole system coherent.

In the same way, every official should know what an API is, what "open data" implies, why accessibility and security cannot be bolted on later, how user research should influence policy, and what it means to depend on a model you don't control - in cost, carbon, and sovereignty terms. These are not technical specialisms; they are the grammar of modern government.

## **From "digital as a function" to "digital as literacy"**

At present, we still treat "digital" as a function - something done by those with a DDaT or GDaD job title. Even as we talk up AI and automation, the instinct is still to ringfence capability: set a target for "digital professionals", launch an AI skills campaign, and hope for transformation to follow. But this risks entrenching the wrong mindset: that digital is specialist work, and everyone else is a passenger.

The reality we need is the opposite: not a tenth who are digital, but a hundred percent who are digitally fluent. We will always need deep specialists - engineers, architects, data people - but basic digital judgement can't be delegated. If you're setting policy, commissioning a supplier, approving spend, running operations, or managing risk, you're already making digital choices whether you call them that or not. Digital is that kind of literacy: it shapes how policy is made, how services are delivered, how risk is managed, and how citizens experience the state.

**The bottom line is that this is too important to be left to the experts.**

### **A baseline already exists**

The aim of a Highway Code for Digital is not to elevate digital specialists above others, but to normalise a baseline of competence for all. And lots of people have done the hard work to think about what that baseline needs to look like already. For example, the OECD in its Framework for Digital Talent and Skills in the Public Sector sets out the following five digital government user skills for every public servant:

- recognising the potential of digital for transformation
- understanding users and their needs
- collaborating openly for iterative delivery
- the trustworthy use of data and technology
- and data-driven government

None of this makes digital specialism redundant. Any road system needs people who build it, maintain it, and keep it safe - and digital transformation in the public sector is no different. We will need engineers, researchers, designers and architects to keep things functional, extend its reach, and help others to travel safely. But the Highway Code exists so it isn't only experts who hold the system

together. Its full value comes when everyone else can travel responsibly: respecting the rules, anticipating knock-on effects, and knowing when to call for help.

### **What the Highway Code actually *does***

So a vibe-coded future requires us to democratise digital understanding across the public service, not to mystify it. A Highway Code for Digital would codify a handful of basics: how to recognise a user need, how to think about service performance, how to spot a privacy or security risk, and how to work with multidisciplinary teams rather than throw things over walls. These expectations shouldn't live in a slide deck; they should be built into induction, leadership, appraisal, and everyday decision-making.

And like any Highway Code, it would come with a few simple disciplines for life at pace: signal early, check your mirrors, keep your distance. In digital terms: communicate openly, consider knock-on effects, and don't outrun research, accessibility, review, or operational readiness just because you can.

A Highway Code for Digital, then, is about shared literacy and shared accountability. It recognises the value of professional drivers and mechanics, but expects everyone else to know how to travel safely, to respect the rules, and to notice when something isn't working as it should. It's how we keep our footing as the machinery speeds up - ensuring that every public servant, from the most junior official to the most senior minister, can move with confidence, safely, and in the right direction.

CHAPTER 06

# Reimagining the team and its roles

---

AI means public service teams are now building in a different medium. In a vibe-coded world, the tempo changes, and that forces a reckoning: we cannot keep the same division of labour and pretend nothing else will alter.

Some roles will shrink. Some will merge. Some will reappear as a different kind of work entirely.

That is not because the work was pointless, but because a chunk of what used to require a person now looks like a capability. Drafting, scaffolding, first-pass analysis, boilerplate, testing, documentation, data schemas, even the first cut of an interface: AI assistance makes those things cheaper. So the question becomes less “how many hands do we have?” and more “where is judgement still needed?”

This is where we might get carried away. Quality working code can be written quickly now but that does *not* mean the organisation can safely ship everything quickly. The work that remains is less visible: clarity, intent, evidence, operability, and responsibility.

So teams may be smaller, or at least *denser*. We may end up with the same number of staff across more teams. We may build more in-house, and maintain it properly. We may finally pay down the legacy we have been living around for years, including the

unglamorous refactoring of COBOL and FORTRAN estates that still hold up large parts of the state. All of that is in service of improving our services.

And some old boundaries will blur.

### **Boundaries blur; standards don't**

Do we still need “frontend vs backend vs webops” as hard categories if more of the team can work full-stack on small slices? Probably less than we do today.

Do we still need “QA” as a separate gate at the end? Much less.

Do we still need quality, testing, accessibility assurance, performance validation, security review, privacy practice? More than ever.

“QA” doesn't go away but the *shape* of it changes. Manual regression-as-a-phase becomes harder to justify when the team can generate tests, run them continuously, and ship in smaller slices. But someone still has to care that the service behaves under real conditions, that accessibility holds, that failure modes are humane, that changes don't create unfairness, and that metrics are not lying. If we remove the people who hold that line, we will not go faster; we will simply push the costs into incidents, complaints, and frontline burden.

This is why the multidisciplinary team remains non-negotiable. If anything, it becomes more important, because at higher speed you can institutionalise blind spots: you can move faster with fewer hand-offs, but you cannot move faster with fewer perspectives.

### **Two rings that behave as one team**

A useful way to make this concrete is to think in two rings that behave as one team:

### **The core delivery ring**

Product, design, research, content and engineering: accountable for user needs, interaction and content design, code and tests, measurement and iteration.

### **The institutional ring (policy-in-the-loop)**

Policy, legal, commercial, assurance, security, data protection, finance, operations, communications, risk and ethics: not as periodic sign-off, but as partners close enough to shape the work while it is still cheap to change.

This is not an org chart. It is a practical claim about what it takes to move quickly without losing your footing. In a vibe-coded world, each discipline still exists, but the cadence changes. Less waiting. Less theatre. More working evidence. More responsibility.

I've had a go at naming the roles that have to be present in the system. Not because every team needs every job title in the stand-up, but because every team needs access to these perspectives, at the right moments, in a way that is fast enough to be real.

### **Roles and accountabilities in a vibe-coded world**

This is not an org chart. It's a statement of **work that must be covered** if we are to move quickly without becoming reckless. Some of these accountabilities sit in the core delivery ring. Some sit in the institutional ring. Some are best held in enabling teams that create paved roads for everyone else.

Each role description is intentionally short: what this discipline exists to protect; what changes when building is cheap; what "good" looks like when you can ship in days.

Product

- **Where it sits:** Core ring
- **Still accountable for:** user need, outcome focus, sequencing, and stopping work that doesn't earn its place.

- **In a vibe-coded world:** less slideware; more working code; more decisions made in the presence of evidence.
- **Signs it's working:** a small set of outcomes that stay stable; clear hypotheses and stop rules; learning captured and reused; fewer late surprises.

### Engineering

- **Where it sits:** Core ring + Enabling (for paved roads)
- **Still accountable for:** reliability, security, accessibility, operability, and maintainability.
- **In a vibe-coded world:** less typing; more review and verification; more defaults baked into pipelines; more focus on the parts that fail in the real world.
- **Signs it's working:** small reversible changes; tests and checks run by default; preview environments and feature flags are normal; the service can be operated calmly.

### Design

- **Where it sits:** Core ring
- **Still accountable for:** usability, coherence, accessibility as experience, not compliance.
- **In a vibe-coded world:** faster iteration in real components; more “showing” earlier; less debating hypotheticals.
- **Signs it's working:** awkward states tested; accessibility holds under change; drop-offs and confusion reduce; design decisions cite evidence not taste.

### Content

- **Where it sits:** Core ring
- **Still accountable for:** clarity, accuracy, inclusion, and words that match the service's behaviour.
- **In a vibe-coded world:** drafting accelerates; editing and truth-checking become more central; comprehension is tested sooner.

- **Signs it's working:** fewer avoidable contacts caused by wording; clearer error messages; consistent tone; fewer “confident but wrong” moments escaping into production.

## Research

- **Where it sits:** Core ring
- **Still accountable for:** ethical, inclusive research; insight you can trust; keeping the team close to reality.
- **In a vibe-coded world:** continuous discovery; prototypes used as research artefacts without replacing real users with simulated ones.
- **Signs it's working:** regular user contact; synthesis that changes decisions; fewer untested assumptions reaching live; clear confidence levels on findings.

## Delivery

- **Where it sits:** Core ring
- **Still accountable for:** flow, coordination, team health, and making work visible without bureaucracy.
- **In a vibe-coded world:** fewer ceremonies; more cadence; tighter WIP; release choreography; guardrails enforced through defaults.
- **Signs it's working:** pace without heroics; predictable throughput; clear ownership; fewer “urgent” handovers; learning absorbed rather than piled up.

## Policy

- **Where it sits:** Institutional ring
- **Still accountable for:** intent, proportionality, fairness, and explainability.
- **In a vibe-coded world:** policy is expressed earlier as runnable rules and worked examples; constraints are co-authored while change is still cheap.
- **Signs it's working:** fewer late blockers; fewer mismatches between policy and service behaviour; clearer traceability; confidence earned through working evidence.

## Legal

- **Where it sits:** Institutional ring
- **Still accountable for:** lawfulness, transparency, equality duties, accountability.
- **In a vibe-coded world:** joins the cadence; shapes constraints and notices alongside delivery; keeps DPIA/ROPA live as behaviour changes.
- **Signs it's working:** fewer last-minute rewrites; notices match what the service does; decisions survive scrutiny without reconstruction.

## Security

- **Where it sits:** Institutional ring + Enabling
- **Still accountable for:** protecting people, data and services; proportionate controls; abuse resistance.
- **In a vibe-coded world:** controls codified in pipelines; threat models kept living; AI-specific risks named and tested.
- **Signs it's working:** vulnerabilities caught early; secure defaults; rehearsed incident/kill-switch routines; fewer “we’ll harden later” surprises.

## Information governance & data protection

- **Where it sits:** Institutional ring
- **Still accountable for:** lawful basis, minimisation, retention, transparency, auditability.
- **In a vibe-coded world:** obligations expressed as checks; tool/model inventories kept current; “no PII in prompts/logs” treated as normal hygiene.
- **Signs it's working:** notices match behaviour; low PII sprawl; DSAR/FOI manageable because records exist; fewer privacy risks discovered late.

## Governance & assurance

- **Where it sits:** Institutional ring
- **Still accountable for:** proportionate assurance; alignment to the Service Standard; accountable decisions.

- **In a vibe-coded world:** fewer big gates; more frequent, lighter evidence check-ins on the running thing.
- **Signs it's working:** quick decisions under clear guardrails; stop rules used; issues caught in preview not production; assurance that reduces rework.

#### Commercial & procurement

- **Where it sits:** Institutional ring
- **Still accountable for:** value for money, fairness, contestability, sensible exit.
- **In a vibe-coded world:** tranche funding against evidence; evaluating working slices not slideware; treating AI tooling as supply chain.
- **Signs it's working:** smaller bets; clean off-ramps; contracts requiring operability and transparency; portability tested, not promised.

#### Operations & support

- **Where it sits:** Institutional ring (close to core cadence)
- **Still accountable for:** reliability on a bad day; humane failure modes; frontline reality; incident response.
- **In a vibe-coded world:** co-design in preview; rehearsing failure; pacing change using error budgets and real signals.
- **Signs it's working:** fewer escalations; faster recovery; clear runbooks; staff trust; fewer workarounds becoming the service.

#### Data & performance analysis

- **Where it sits:** Institutional ring / Enabling
- **Still accountable for:** measurement integrity; baselines; honest interpretation; guarding against gaming.
- **In a vibe-coded world:** near-real-time signals; experiment design; confidence stated; metrics treated as instruments, not weapons.
- **Signs it's working:** shared definitions; decisions improve rather than thrash; numbers used to learn, not to posture.

## Communications

- **Where it sits:** Institutional ring
- **Still accountable for:** clarity about what changed and why; trust; accessible comms.
- **In a vibe-coded world:** comms aligned with progressive rollout; tight loops with support and ops; quick, humane incident comms.
- **Signs it's working:** fewer “what happened?” contacts; fewer surprises; consistent language across channels; trust maintained through change.

## Risk & ethics

- **Where it sits:** Institutional ring
- **Still accountable for:** proportionality; fairness; accountable choices; seeing harm early.
- **In a vibe-coded world:** risk stays live and tied to flags/ measures; targeted checks where stakes are higher; provenance and decision logs maintained.
- **Signs it's working:** clear stop rules; harms prevented rather than explained afterwards; fewer “the model said it was fine” moments.

CHAPTER 07

# How we organise when building is cheap

---

If building becomes cheaper and faster, the question isn't how to spend less on people - it's how to spend differently on capability. Badly-built software is a drain on government: brittle services, repeated rebuilds, solving the same problems multiple times. The answer remains: fund teams, not projects - durable, multidisciplinary teams who own a problem, learn continuously, and keep improving the service they steward over time.

What changes in a vibe-coded world is the *shape* and *rhythm* of that organisation. Teams no longer need to be huge to have impact. With the right mix of skills, and good platforms beneath them, a smaller cross-functional team can deliver at a pace that used to demand a programme. The goal isn't to reduce the size for the sake of it but to make every pound and every hour count towards a better outcome.

## **Two models that work together (*standing + mission*)**

When we can have impact in short bursts, how do we avoid defaulting to short-lived project teams? I suggest we need two models that work together:

1. Standing service teams that own and evolve a product over time,

## 2. Mission teams that surge around specific outcomes or policy moments.

The standing team holds the continuity - reliability, feedback, iteration - while the mission team brings some extra focus to seize an opportunity or unblock something, then folds its learning back into the core. Neither replaces the other. Together they form an ecosystem of continuous service stewardship and adaptive delivery.

This isn't the return of "keep-the-lights-on" skeleton crews. Quite the opposite: the standing team has to be set up to iterate, not merely maintain. It's where institutional memory lives, where user feedback lands, and where change accumulates safely. Its health determines whether a service matures gracefully or fossilises. That means giving those teams stable funding and permission to keep improving, not just to keep running.

At higher speeds, this could be how we get speed without chaos: surge capacity without organisational amnesia. Missions give you a way to concentrate effort on what matters, without constantly rebuilding the scaffolding of a programme. And they stop specialist disciplines being treated as intermittent obstacles. Accessibility, data, legal, security - these are not people you contact at the end. They're perspectives you need alongside you while you're still deciding what the thing is. Again: not a new idea. Just multidisciplinary work, practised properly.

### Funding that follows evidence

And when building becomes cheap, do we also get the freedom to match our funding models to the same logic? Instead of having to secure huge upfront budget for a multi-year plan (with all the expectations and rigidity that comes with it), can we move to greater tranche-based funding: funding released in slices, tied to evidence and outcomes. Fund a discovery and alpha mission; if it

yields a working prototype and promising data - improved takeup, reduced processing times, fewer avoidable contacts - then release the next chunk of funding to scale and harden it.

Funding becomes iterative, just like development.

This reduces waste - we stop pouring money into ideas that don't survive contact with reality - but it also changes the discipline of decision-making. Spending can follow proof of value: a working thing, credible measures, clear (and openly published) stories about what we are learning. That is easier to govern well. Because funding is released against evidence and a model of assurance that happens continuously, in proportion, and without waiting for a single all-or-nothing business case or a spending review cycle.

### **Platform as tarmac (not just tooling)**

All of this depends on the health of our shared digital infrastructure - but we should be careful what we mean by 'infrastructure'. In a vibe-coded world, the limiting factor is not compute or code. It's whether teams can move quickly on a smoothly tarmaced road: clear standards, shared ways of working, and common building blocks.

Yes, some of that is technical: environments that are quick to spin up; routine deployment; security, monitoring and accessibility baked in; common components like identity, payments and notifications; approaches to data that unlock its value across three tenses. But the platform play is bigger than tooling. It also includes the shared guidance and best practice that stops every team relearning the same lessons; spending, governance and assurance that work at pace; and a serious commitment to digital inclusion so faster doesn't just mean faster-for-the-already-confident.

That's the job of long-lived enabling teams and functions: not to build the service for everyone else, but to keep that common machinery healthy - technical and institutional - in good order, and to remove friction for delivery teams. When those foundations are

weak, teams compensate by rebuilding them locally, in different ways, at different quality levels. At higher pace, that failure mode accelerates. If we want mission teams to surge without chaos, we need to strengthen this Government as a Platform enabling layer - and give it both the responsibility and the authority to make other teams faster, safer, and more focused on the needs of their users.

## **Talent is part of the platform**

Finally, there's the cultural side - though in truth it's not really finally at all. Talent and capability are part of the platform. If AI assistance makes routine tasks easier, then that changes how we think about early-career roles. The challenge is not to hollow out the junior ranks, although there may be fewer of them, it's to reimagine what it means to learn the craft.

If all we think juniors do is low-level, repetitive work, we misunderstand the point of apprenticeship. Learning to design public services has never just been about roadmaps or code; it's about judgement, ethics, teamwork and care. The new generation should still have places to practise those - through reviewing AI output, working with users, understanding policy context, and shaping the human side of systems.

Our organisations may become flatter in terms of who contributes ideas - because if anyone can test an idea cheaply, more voices can lead to more experiments. But flat doesn't mean leaderless. Leadership's job in this environment is to set direction, create the environment for success, and curate learning: to channel energy towards outcomes, not let a thousand prototypes bloom into noise.

When building becomes cheaper in time and money, organising well means being light on our feet without losing coherence. Small teams that form, deliver, and reform elsewhere. Funding that follows the evidence of value. An enabling platform - technical and institutional - that liberates teams rather than constrains them. And an ethos that good ideas can come from anywhere, because the cost of trying them is low. But the discipline of learning still

matters. This is a vision of a more *organic* approach to public sector organisations: one that can continuously reconfigure itself in response to new challenges, while staying accountable, inclusive, and trustworthy.

CHAPTER 08

# Stages as lenses: learning, testing, and scaling responsibly

---

For more than a decade, the rhythm of Discovery, Alpha, Beta and Live has structured how digital public services come to life in the UK. That rhythm remains vital: civic scaffolding that helps us learn safely, stage by stage, on the path from curiosity to confidence.

In a vibe-coded world, where ideas can be prototyped as deployed code before a meeting ends, the meaning of those stages deepens. They are not hurdles to clear, but lenses for understanding: of users, of systems, and of what it takes to deliver a safe, sustainable service.

## Discovery is where we see clearly

The core discipline - *begin with users and their needs* - does not change. If anything, speed makes the early stages of discovery *more* important, not less. Discovery isn't about what can be built; it's about what *should* be built, and why.

It's where we pause long enough to see the world through our users' eyes, map the journeys and constraints that shape their experience, and surface the organisational, policy and data realities that will make or break the service later. When building

becomes fast, *understanding* becomes precious. Discovery endures because it forces us to look before we leap. **Its essence is not documentation but discernment.**

AI tools can help: quick visualisations, simulated flows, prompts co-created with users. But quick sketches are not the same as insight. Discovery demands the discipline to slow down enough to keep asking, *is this still the right problem?* Prototypes will start to appear, but they do so to help us illuminate the problem, not jump to the solution.

The discipline is to stay curious: keep asking “*why does this exist?*” before racing to answer “*what could we build?*”

### **Alpha is where we learn cheaply**

Here, understanding is tested. Ideas take form - lightly and provisionally - so we can explore directions without committing too early. AI-assisted delivery teams might generate multiple prototypes in a morning, test with users the same day, and carry forward the most promising. The artefacts may multiply but the output is still insight.

Product and design shift from planning builds to curating and comparing them: which best meets the need we uncovered? Which breaks on policy, legal, operational, or data constraints? What hidden assumptions have surfaced?

Alpha becomes a rhythm of rapid, responsible experiments: the smallest thing that tests the riskiest assumption, with the right mix of people involved in the learning.

Alpha is allowed to be untidy but abundantly so - it should produce multiple versions and expect to throw most of them away. But too often “Alpha” runs as polite waterfall to deliver One True Service. When building gets cheaper, we lose some excuses: it becomes practical, and healthier, to explore properly and let evidence choose what survives.

And yet, AI assistance introduces a new tension. The old adage - *discard Alpha code* - exists for good reason: prototypes are built for learning, not for longevity. But what if your AI pair can generate something secure, accessible, tested, observable and operable by default? Isn't it wasteful to throw it away?

If an Alpha slice meets the standards you'd demand in Beta, and the team has capacity to maintain it, then build on top of it. Otherwise, keep the learning and let the code go. We must never smuggle Alpha code into production simply because it looks tidy. The point is not to build something beautiful. It's to learn something true.

You don't end Alpha because your prototype is polished but because you have greater clarity: *we've understood the problem, tested the main risks, and identified a direction worth pursuing.*

This isn't linear. Research informs design; design provokes better research. A Discovery question today might become an Alpha experiment tomorrow. You can hold both mindsets at once - learning about users while showing them working ideas to test your own assumptions. The teams who thrive will be those who keep evidence flowing in both directions - research shaping what gets built, and working ideas surfacing new questions to explore.

## **Beta is where learning meets reality**

It's where teams build something that works end-to-end for real users, integrates with real systems, and operates under real-world conditions. Here, performance, reliability, accessibility and operational readiness get tested properly.

Beta is also the bridge to long-term ownership.

Teams must start thinking about lifecycle: who will care for the service when the spotlight moves on? What does 'good' look like for frontline staff? Going fast here doesn't mean cutting corners; it

means building in the open, showing your working, and inviting feedback from users, colleagues and peers so that decisions are tested socially as well as technically.

With AI in the loop, Beta can also expand its scope. The question becomes not just “*does it work?*” but “*is it working fairly, securely and as we intended?*” This is where new assurance shows up: checking model outputs for bias, confirming prompts don’t leak sensitive data, validating that AI-generated content still meets accessibility and clarity standards.

AI assistance changes *what* must be tested, not *whether* we test.

In practice, Beta may fragment into progressive rollouts, feeding live data straight back into iteration. A service might scale from 5% to 20% in days, but only if each step earns its place through evidence. This is evidence, not vibes. You don’t skip evidence just because you’re getting more confident in how you’re orchestrating AI assistance.

## **Live is not the end**

It’s where stewardship begins: running safely, at scale, with a clear owner, published metrics, and a rhythm of continuous improvement. In a vibe-coded world, the expected daily rhythm should no longer be exceptional. The line between late Beta and Live can blur but Live is where accountability crystallises: a commitment to own consequences, listen to users, and fix issues promptly.

And beyond Live is the unglamorous but necessary discipline of retiring a service well - handling data properly, redirecting users cleanly, and ensuring that institutional learning isn’t lost. Ending well is as important as starting well.

**Across all these stages, the theme is evidence, not assumption.**

We move on because we've learned enough to proceed responsibly - not because we get the paperwork right; not because a prototype looks good; not because the code runs.

So while loops may shorten and transitions become more fluid, their purpose remains. The stages are not gates to rush through but disciplines of reflection:

- Discovery asks *why this?*
- Alpha asks *how might we?*
- Beta asks *can this work for real?*
- Live asks *how do we sustain and improve it?*

AI-assisted delivery can change the medium of evidence - more tangible, more immediate, more participatory - but it doesn't change the need for evidence itself.

The challenge for teams in a vibe-coded world is not to skip stages but to tighten feedback loops while preserving their integrity. Building might have got easier; learning is still hard. The Service Manual remains our guide not because it slows us down, but because it translates speed into safety, quality and value.

The future of delivery won't abandon these stages. It will fold them tighter. Discovery will happen within Beta; Alpha may re-open after Live. Teams will move through them in days, not months, circling back as new questions emerge. The boundaries between the stages might soften but the learning will deepen.

What holds it together is a commitment to earn confidence with proof, not promise. To stay honest at speed. In that sense, AI-assisted delivery gets us closer to our aspirations: **delivery as a practice of learning in public** - disciplined enough to keep pace

with the tools it uses, humble enough to keep asking what users actually need, and rigorous enough to keep the whole endeavour safe.

CHAPTER 09

# User research in the age of instant prototypes

---

What does user research look like when a team can spin up an instant prototype at the drop of a hat? It doesn't change *why* we do research - we still need to understand users in the round. But it may change *how*, *when*, and *what counts as evidence*.

The fundamentals remain unchanged: go to the source (real users), observe context and behaviour, combine qualitative and quantitative insight, and iterate accordingly. But because building and releasing prototypes is now quicker and cheaper, we can narrow the gap between “research phase” and “delivery phase”. The cadence can increase, and our path from question to evidence shortens.

In the old days (which was perhaps earlier this year...), code was the expensive part. Research de-risked that: interviews, surveys, journey mapping, paper prototypes - all ways to learn before committing to build.

That logic hasn't vanished, but the economics has inverted.

## **Code as a research artefact**

It's still unwise to build a huge system without any evidence of need, but the main constraint is not in the cost and time of producing code. The expensive part is *not having evidence* - not

talking to users, not understanding the system you're altering, not knowing whether the thing you've made makes life better or worse. AI-assisted delivery turns code into a readily available research artefact: high-fidelity prototypes we can create in hours to probe a question, and discard when we've learned what we needed to learn.

This flips some assumptions. We can try a small thing simply to see how users react, because we haven't sunk a lot of time into it and we can throw it away. Code becomes another tool in the researcher's toolkit, like sketches or diagrams, except it behaves like the real experience, which can reveal frictions that a neat mock-up will hide.

## **Synthetic scaffolding**

This is also where synthetic data becomes genuinely useful - not as a replacement for the voice of a user, but as scaffolding for better research into the contours of reality. Synthetic profiles, journeys, personas and datasets can help teams make the picture more concrete: which variations matter, where operational complexity might bite, what 'normal' and 'edge' cases look like, and what the service would need to handle. It can help you *prepare* research, generate hypotheses, design safer tests, and spot obvious blind spots early. But it's no substitute for lived experience. The discipline is to treat synthetic work as provisional: a way to sharpen questions - and then validate, correct, and deepen it with real people.

The point is not that a synthetic persona is "true". It's that it lets you hold more possibilities in view at once. A team can generate a spread of plausible life situations, constraints, and failure modes, and use them to pressure-test a journey before you ever put it in front of someone real. Done well, that can widen empathy rather than flatten it, because it is another opportunity to invite questions about who benefits from our assumptions and our ambitions, and who does not.

A direct consequence of instant prototypes is that research can become continuous rather than phase-based or milestone oriented. Teams can adopt a steady cadence of small learning loops. One week you test a concept with a prototype built in a day; the next week you test the same idea behind a feature flag in the real service; the week after you observe how frontline staff use an admin interface updated based on what you learned. The boundary between “researching” and “building” becomes a tight loop.

This shifts the constraint from production to learning. When the team can build in hours, the limiting factor becomes evidence: finding the right users, running research responsibly, and making sense of the results fast enough to steer. We spend less time waiting for environments to be configured and more time staying close to people and reality.

It should also change how we decide to try things. Previously, before committing valuable engineering time, teams would reasonably want high confidence up front. Now, if a hunch emerges from partial evidence, we can use our services to help us learn more - through controlled, reversible experimentation. Release a change to 5% of users, watch completion and drop-off rates, read feedback, and revert quickly if confusion appears. Instead of lengthy debates about whether a form flow might help, tightly designed tests can let the evidence decide.

### **Evidence still has to touch people**

In the public sector, duty of care remains non-negotiable. We do not run random experiments on people when stakes are high, and we don't expose vulnerable users to unresearched journeys simply because iteration is easy. But many services include lower-risk elements where experimentation is appropriate: wording, navigation, onboarding, reminders, optional guidance, internal tools, non-critical flows. With the right safeguards, we should make

better use of A/B testing and multivariate experiments than we might have done in slower delivery cycles. The opportunity is to be more empirical, not more cavalier.

Faster iteration also produces richer data streams. Every small release generates evidence: analytics, support tickets, contact-centre patterns, live chat transcripts, operational signals. Researchers increasingly weave those quantitative traces together with qualitative insight - working closely with data and performance colleagues to set up measures that help the team learn, not merely report.

This is also where AI can help in a way that isn't about prototyping at all. It can take on some of the heavy lifting: marshalling long documents and messy datasets, pulling themes across dozens of transcripts, summarising patterns in support logs, and helping teams move back and forth between the qualitative and the quantitative without losing the thread. The risk is obvious: you can get a fluent answer that isn't grounded. So the discipline is the same as everywhere else in this paper: treat it as a partner for sensemaking, not an oracle, and keep the link back to the underlying evidence clear enough that a human can challenge it.

But speed has a cost if learning doesn't keep up. Five experiments a month is easy; making sense of them, choosing measures that matter, and turning results into decisions the team can stand behind is not. Synthesis becomes a craft in its own right: filtering signal from noise, connecting what users said with what they did, and telling a coherent story about what changed and why. Researchers help ensure the team actually learns from all this data and doesn't just drown in it.

### **Beware 'vibes without people'**

There's a particular danger in the era of instant prototypes: beware 'vibes without people'. With all this power to create and test quickly, a team might be tempted to rely too much on their own instincts, simulated feedback, or the model's reassurance ("the AI

says this is good”, “the flow feels nice to us”). A small, multi-skilled team using AI at the expense of their users’ input can become an echo chamber at high speed. The antidote is old-fashioned and still essential: regular contact with real users, in real contexts, as a non-negotiable rhythm. Instant prototypes don’t replace talking to humans; they give humans something real to react to.

AI assistance and vibe coding shouldn’t be confined to engineers and product folk. In a faster world, policy, ops, frontline leaders, analysts, and everyone else can author working models of their assumptions - quickly, cheaply, and early - as part of the same multidisciplinary loop. Policy can express intent as a rules engine and stress-test it with edge cases. Ops can prototype an operating model and walk it through with staff. Analysts can build the monitoring view that defines what success looks like in practice.

The point isn’t to generate authority by producing software; it’s to make the thinking visible so it can be challenged, improved, and validated - with real users externally, and real users internally too. Caseworkers, policy owners, assurance, lawyers, finance and commercial partners aren’t “stakeholders at the end”; they’re part of what makes the service operable, lawful and sustainable, so they belong inside the learning cadence.

So user research in a vibe-coded world becomes more integrated and iterative than ever: we research while we build, and build while we research. The barriers to trying something are lower, which can broaden exploration but only if we keep the work anchored in real people, real contexts, and honest evidence. Faster artefacts are not the same thing as deeper understanding. The task is to use speed to get closer to reality, not further from it.

CHAPTER 10

# From tasks to outcomes

---

‘Outcomes over outputs’ is what good product has always been trying to do. The problem is that our day-to-day mechanics - long backlogs, granular user stories, estimation rituals, ticket queues - have a way of pulling teams back into task management, even when everyone agrees that impact is the goal.

AI-assisted delivery doesn’t create a new philosophy. It changes the economics of execution. When small fixes and even more substantial features can be delivered almost as soon as they’re identified, the effort of tracking every task starts to look disproportionate. What remains hard, and therefore valuable, is clarity: what problem are we solving, what would success look like, and what evidence will tell us whether we’re helping or harming?

## **The backlog as a register of outcomes**

A traditional backlog tries to be a list of everything: features, bugs, chores, refinements. In practice it becomes either endless or obsolete, and often both. When a typo, misaligned button, or small content defect can be fixed immediately, the ‘log a ticket and wait’ threshold rises. A lot of work that used to sit in Jira for weeks can simply get done.

What belongs in the backlog is what is enhanced by shared focus and deeper thought: problems to solve, hypotheses to test, outcomes to pursue. So the backlog can become a register of opportunity statements rather than a laundry list. Items like:

- Improve accessibility of the appointment booking flow
- Reduce drop-off at step three of the claim journey
- Reduce average handling time for caseworkers processing complex cases

These aren't bigger tickets. They're better prompts. They invite the team to decide how to pursue them and keep the conversation anchored in purpose: if a change doesn't plausibly move an outcome, why are we doing it?

This also clarifies the craft of product and delivery leadership. The job isn't to manage a queue; it's to cultivate and prune: keep the most important problems visible, retire what no longer matters, and stop the backlog becoming a museum of stale intentions.

### **A learning log matters as much as a to-do list**

Speed increases the risk of organisational amnesia: repeating experiments, re-litigating decisions, rebuilding solutions that failed last year. Documenting, preferably in the open like this excellent work from NHS England, what was tried and what happened prevents teams from unknowingly revisiting dead ends and helps new joiners understand why the service is the way it is.

Something as simple as: *'We tried X to address Y; it didn't move the needle; here's what we learned; here's what we'd do differently next time'* can save weeks of repeated work and turn speed into cumulative progress rather than churn.

## **Planning becomes mission-shaped**

Whether you keep two-week sprints or move towards continuous flow, planning works best when it is anchored in outcomes, not task lists. If the team can accomplish in two days what used to take two weeks, then a sprint becomes less a commitment to a static list of tasks and more a time-bounded mission within which the exact work will evolve as the team learns.

Stand-ups and boards still matter, but they shift purpose: not defending estimates, but synchronising attention - what changed, what we learned, what we're doing next.

This makes it easier to behave the way agile always said we should: respond to change over following a plan. AI-assisted delivery doesn't invent that principle, but it does make deviation cheaper and therefore more likely, so the discipline has to sit at the level of outcomes and evidence.

## **Roadmaps become hypotheses with targets**

Outcome-oriented roadmaps are also not new. Instead of committing to a sequence of features, the roadmap becomes a sequence of value targets and questions. For example:

- Increase completion rate of online claims from 70% to 85%
- Reduce average processing time from five days to two
- Extend reach to a new user group, with explicit equity and access measures

Under each outcome, you might hold provisional approaches: simplify the form, add guided support, automate a verification step, redesign a back-office workflow, but the roadmap is not a promise to ship a specific solution on a specific date. It is a promise to tackle the problem, to measure honestly, and to keep trying until the evidence says you've made meaningful progress.

This aligns well with an environment where if one idea doesn't work, you can try another without paying a huge switching cost. But it does require leadership that is comfortable with adaptive delivery: trust that the team will deliver value by the end of the quarter even if the exact outputs evolve. In exchange, there will be more value, not less: teams that remove small impediments continuously and test multiple approaches empirically without clinging to pet features just because they were presented in slides weeks ago.

### **Reporting becomes a narrative of impact**

In faster environments it's easy to confuse activity with impact. This is where communication rituals like weeknotes become more valuable, not less, but their content shifts. The purpose isn't to list what was built; it's to capture what has changed.

A weeknote might say: *'We set out to reduce confusion in error messaging. We released two iterations, and calls to the help line dropped by 10%. Next week we'll test whether that holds, and then pivot to the next source of avoidable contact.'* That kind of narrative keeps everyone oriented around outcomes and forces the team to articulate what it's learning.

### **Metrics-driven development becomes more practical**

To cement 'outcomes over tasks', teams can organise work directly around the measures that matter to them: uptake, completion, avoidable contact, processing time, cost per transaction - with ideas for experiments under each of them. For public services we can think more expansively about how to include things like equity of access, differential outcomes for different groups, and operational sustainability.

The point isn't to reduce everything to numbers. It's to ensure that every meaningful piece of work can link to public value, and that you can quickly identify when that is absent.

Measures are a sharp tool. They can illuminate reality, and they can be weaponised. In public services, numbers are often harder than they look: baselines shift, user behaviour changes seasonally, policy changes contaminate results, and proxy metrics invite gaming. It is risky to hand a brittle metric to a difficult stakeholder who will treat it as a performance target rather than a learning signal. The discipline is not to avoid numbers; it is to treat them with the same care we treat code: define them, version them, attach confidence, and pair them with narrative and context. A good metric is a learning instrument first, and a performance instrument second.

### **The deeper prize: removing work from the system**

If you care about deeper efficiency, and we should, the goal isn't "ship more". It's to remove work from the system: fewer steps, fewer handoffs, fewer avoidable contacts, fewer appeals, fewer workarounds, fewer duplicate tools doing the same job. AI makes it easier to change services. The real prize is needing fewer changes because the service is simpler, and because the underlying estate is being paid down rather than patched around.

### **Managing the anxiety of "what exactly will you deliver?"**

Some stakeholders will feel a loss of predictability. *'What exactly will you deliver, and when?'* is not an unreasonable question, especially in environments shaped by business cases, programme controls, and supplier contracts. The answer is not to pretend certainty; it's to be explicit about your confidence in the outcome you're pursuing, the evidence you'll use and the transparency in how you tell the story.

There is more integrity in saying, *'By June we aim to halve the backlog of pending cases; we will try a number of approaches and publish what we learn,'* than in promising a specific set of features - a promise history tells us is often late, brittle, or not successfully solving the real problem.

## **Bringing it together**

This is not a new doctrine. The above are tried and tested practices that lead to good outcomes for users - and in a vibe-coded world we have fewer excuses not to do them. Shifting from tasks to outcomes means reimagining the artefacts and conversations that organise the work: the backlog fulfilling its role as a strategic tool; sprints as missions; roadmaps articulating value targets and learning questions; reporting that tells stories about impact, evidence and the value being added.

This is not a more casual approach. AI-assisted delivery makes it easier to live up to the ambitions of agile more fully: focus on the problem, iterate rapidly, measure honestly, and judge success by what improved for users and the wider system, not by how many tickets got closed.

CHAPTER 11

# Buying change, not a plan

---

A vibe-coded world doesn't only change how we build. It changes what it means to fund and procure delivery responsibly. Much of our commercial machinery still assumes that working software is expensive and slow, so it asks for confidence up front, manufactured through documents.

AI-assisted delivery collapses that premise. When real progress can be shown in weeks, and when the right answer may only emerge through iteration, the old model starts demanding false certainty. If funding and procurement stay locked to rhythms measured in months while delivery gets measured in days, the result is not control. It is either paralysis, or teams sprinting ahead and leaving legitimacy to be rebuilt later.

So the shift is simple to state and hard to do well: move from **front-loaded certainty to ongoing validation**. This is not less rigour, it is different rigour. A rigour based on proof, not fortune-telling.

## **Fund in tranches, release against evidence**

The most practical move is evidence-based tranche funding. Not a single £5m bet placed on a plan, but a sequence of smaller commitments released against what the work has actually shown.

An initial tranche funds an Alpha mission with a clear expectation: *show something working, show what it changed, show what you learned, show what risks remain*. That “something” might be a prototype tested with users. It might be a policy engine exercised against edge cases. It might be an operational walkthrough run with frontline staff. It might be code in production behind a feature flag with early metrics. The point is that the funding decision is anchored in reality, not rhetoric.

The next tranche is released when the evidence justifies it: not because the paperwork is immaculate, but because the team can credibly say, *this is working, this is safe to scale, and here is what we will prove next*. And if it isn't working, tranche funding gives you a dignified off-ramp: you stop early, capture the lessons, and avoid “completing” a programme that no longer deserves to exist.

This can look like stage-gating. The difference is that the gate is not a document. It is evidence: user outcomes, operational impact, risk reduction, and observable service performance.

But there is room for caution: tranche funding can easily become micro-gatekeeping - death by a thousand finance reviews. Adopting this model requires clear delegations, predictable and lighter-touch cadences, and a bias towards reusing assurance once it has been earned. Otherwise you simply swap one slow system for a slower one.

### **Value for money becomes easier to show (and harder to fake)**

In a vibe-coded world, “value for money” should become easier to demonstrate. And harder to bluff.

When change is cheap, the question is no longer “can we afford to build it?” but “are we moving the numbers that matter?” Completion rates, processing time, avoidable contact, cost-to-serve, staff handling time, errors, appeals, complaints - and, crucially,

equity of access and differential outcomes. These are not afterthoughts. They are the public value of the service, made visible.

This is where finance and product language meet. A good tranche review is a short story, told with evidence:

- What outcome did we target?
- What changed for users and staff?
- What did it cost to achieve that change?
- What risks did we remove, and what risks did we discover?
- What do we believe now that we didn't believe six weeks ago?

That last question matters. In government, “learning” is often treated as an indulgence. In this model, it is the mechanism by which you protect the public purse.

### **Buy capability and partnership, not monolithic deliverables**

Procurement has to change shape as well. Large contracts that specify a fixed scope years out don't fit a world where the right answer is discovered in contact with reality. The stance becomes: buy small missions, buy specialist help in bursts, and keep ownership of the evolving service.

That can mean more use of flexible call-offs, outcome-shaped statements of work, and multi-supplier arrangements that allow you to pull in expertise when you need it - accessibility, security, content, data engineering, service design, performance analysis.

AI assistance also changes who can credibly deliver. It can level the playing field for smaller suppliers with deep domain expertise and strong engineering discipline, because the productivity gap between a “big team” and a “small expert team” narrows. Procurement should be open to that: favour agility and demonstrable competence, not just scale and familiar logos.

And if you want a cleaner test of capability, you can increasingly ask for working evidence early: not “a beautiful proposal”, but a short, time-boxed mission (that you fund) which produces a prototype, an analysis, or a working integration in a sandbox environment, so claims meet reality quickly.

### **Make “the safe path” a contractual requirement**

If more of the multidisciplinary system can contribute to runnable artefacts, then standards cannot live only in people’s heads. They have to live in defaults, and you can insist on those defaults commercially.

Contracts should require modern delivery practices and shared visibility, so government can see what is being built as it is built. This is the shift from black-box delivery to partnership-based work that can be safe at pace.

At minimum, suppliers should be required to:

- work in the open with shared code and shared environments where appropriate
- provide the artefacts that make services operable (runbooks, monitoring, alerts, incident patterns)
- treat accessibility, security and data protection as non-negotiable “definition of done”, not bolt-ons
- support feature-flagged release and safe rollback
- provide evidence of performance and user outcomes, not just “scope delivered”

In other words: if delivery can go faster, the bar for operability needs to go up.

## **Treat AI as part of the supply chain**

If AI tools are woven into delivery, they become part of the supply chain, and therefore part of the commercial and assurance landscape. It would be easy to ban them, but that would be counterproductive. We need to be explicit about:

- what tools are used, and in what contexts
- what data can and cannot be shared with them
- how provenance is managed (what generated what, when, and under which constraints)
- how AI-produced code is held to the same standards as human-written code
- how IP, reuse and exit are handled so government is not locked out of its own service

In a vibe-coded world, prompts, policies and agent guardrails are important artefacts. As accountability is non-negotiable, then traceability is part of the price of admission. Not to punish teams, but so that when something goes wrong we can understand what happened, and fix it properly.

There's also a sovereignty question hiding in the plumbing. If our ability to deliver depends on tools we can't explain, can't exit, or can't run without a single external choke-point, we have moved risk, not removed it. In some cases that may still be acceptable but it has to be named, governed, and designed around, not discovered during an incident.

We will not certify the whole upstream story through procurement alone, but we can insist on a floor: clarity about what tools are in use, what data they touch, what gets retained, and what the exit looks like if we need to change course. If we rely on it, we should be able to describe it plainly to an auditor, a minister, and ourselves.

## **Rethink the on-ramps: procuring the road, not just the vehicle**

All of this collapses if the on-ramps are broken.

There is no point having AI-assisted delivery if the route to a safe environment takes three months; if security review is a queue rather than a collaboration; if teams can generate change quickly but can't deploy it safely. In a vibe-coded world, the bottleneck shifts from "writing code" to "getting onto the road".

That means investing in enabling infrastructure: secure-by-default environments, one-click preview deployments, identity and access patterns, logging and monitoring, test automation, component libraries, feature flagging, and the guardrails that let more people contribute without lowering the bar.

Cross-government procurement and strategy needs to reflect that reality: less bespoke infrastructure per programme, more shared platform capability that makes the safe path the easiest path. This is not only a technical procurement question. It is organisational: who owns the platform, how it is funded, and how it is governed so it serves teams rather than slowing them down.

## **Bringing it together**

The thread running through all of this is that evidence becomes the mechanism by which pace is earned. In a vibe-coded world, we should walk into funding and assurance conversations with something real: "try it", "see what changed", "here's what users did", "here's what staff said", "here's what the metrics show", "here's the risk we removed". The story stops being "trust us". It becomes "come and see". The strategy is delivery.

Funding, commercial and evidence practices have to evolve to keep pace with a vibe-coded world. The goal is not more speed for its own sake. It is faster learning, earlier proof, cleaner off-ramps, and a stronger ability to scale what works.

The shape of it is familiar by now: fund in tranches, procure for partnership and outcomes, insist that standards live in defaults, treat AI as part of the supply chain, and invest in the on-ramps.

CONCLUSION

# Conclusion: the craft of confidence

---

In the end, this paper isn't really about AI. It's about confidence.

Not hype-fuelled bravado. Not the brittle confidence that comes from pretending we already know the answer. The kind of confidence public service actually needs: earned through evidence, inclusion, and diligent practice - the confidence to move faster because we trust the methods we've honed over many years, methods that keep us honest and our users at the centre.

In a vibe-coded world, building gets cheaper. Confidence doesn't. We can make things real faster than before, but the cost of being wrong hasn't fallen.

A vibe-coded world compresses the distance between intent and reality. That creates an opportunity, and a temptation. The opportunity is that we can deliver value earlier, learn sooner, and stop wasting months defending guesses. The temptation is that speed becomes a substitute for thought, or that we treat "working software" as if it were the same thing as a legitimate, supportable service. The craft is not letting speed blur that line: using it to get closer to reality, not further from it.

## **A week that's now plausible**

Picture a week that would have sounded implausible in government not long ago, but now sits within reach.

A team notices a spike in confusion and avoidable contact. By the afternoon they've designed and shipped a small change to a safe environment. The next day it's in front of users and frontline colleagues and they observe what happens. Then it's out in the open, behind a feature flag for a small percentage of live traffic, actively monitored, with rollback ready if needed. Twenty-four hours later the evidence is clear about whether the change helps or doesn't; and if it doesn't, it's removed without drama, with the learning captured. By the end of the week the team can reflect: this is what we tried, this is what changed, this is what we learned, and this is what we will do next.

Nothing in that rhythm is reckless. It is cautious where it should be, and bold where it can be. The speed comes from smaller steps, clearer evidence, and less waiting - not from gambling, not from heroics, and not from "trust us".

That is what I mean by the craft of confidence: designing a way of working where each step is small enough to be safe, visible enough to be accountable, and evidence-rich enough to justify the next move. It's how you "seek forgiveness not permission" without drifting into arrogance, because you are not bypassing governance; you are meeting it continuously, in public, through what you ship and what you measure.

## **Confidence for all**

This is also why this isn't only a product or engineering story. In a vibe-coded world, more of the multidisciplinary team can express intent as runnable artefacts. Confidence has to be built in teams, yes, but also in the people who fund work, assure it, operate it, support it, and ultimately answer for it.

That has implications.

For leaders, the question shifts. Not “did you deliver to the plan?” but “did you earn the right to go faster?” Did you build a system that can learn safely at pace? Did you make the safe path the easiest path? Did you invest in the on-ramps (platforms, environments, governance built into delivery) so teams aren’t forced to choose between speed and legitimacy?

For practitioners, it’s similar. The best response to scepticism or anxiety is not a manifesto; it is practice. Try the tools. Use them on low-risk work. Notice where they help and where they mislead. Then double-down on what they cannot do: empathise, exercise judgement, weigh trade-offs, notice who is excluded, and defend what “good” means. If AI compresses the time to build, then human responsibility becomes the scarce resource we protect.

And for the public, ideally, most of this is invisible. People do not need to know what tools we used. They need to experience services that work: clear, accessible, reliable, humane; improving steadily; honest about mistakes; quick to fix what breaks. If we do this well, trust can grow not because we promise less change, but because we demonstrate competence in change.

## **Humility as a discipline**

There is one more dimension to confidence that matters here: humility.

Real confidence is not certainty. It is the ability to act without pretending to be infallible. An AI-assisted world will make our wrongness show up faster, which is a gift if we are willing to see it. The iterative discipline has always implied “we might be wrong, so we test”. AI-assisted delivery makes that easier than ever, because testing is cheaper and learning loops are tighter. But that only helps if we keep our curiosity alive: curiosity about what users actually

experience, curiosity about what the data is really saying, curiosity about what we don't yet understand, and the willingness to slow down when the evidence demands it.

If you want a summary of what this paper is arguing, it is this: **a vibe-coded world doesn't invent new values; it extends our capacity to live up to our disciplines.** It reduces the distance between policy intent and experienced reality.

If you want something to take away, I think this paper boils down to this:

- **Have a go, but keep it bounded and honest.** Use AI on low-risk problems first. Time-box experiments, name the hypothesis, capture what happened, and discard without shame.
- **Do not mistake “it runs” for “it's ready”.** Working code is cheaper now. Readiness is still expensive: accessibility, security, lawful data handling, operability, monitoring, support paths, and rollback. Protect teams when they say “not yet”.
- **Make proof the currency of progress.** Replace confidence-by-document with evidence-by-default: show the thing, measure the effect, record what you learned, and let that trail earn autonomy, funding, and the right to scale.
- **Put standards and governance in the machinery, not in ceremonies.** If more people can ship, guardrails must live in defaults: pipelines, platforms, component libraries, feature flags, and transparent decision logs, with policy, legal and ops close to the team, not waiting at the end.
- **Let outcomes set the pace, and be willing to undo your own work.** Organise around the numbers that matter, be explicit about uncertainty, and scale only when the evidence holds. When it doesn't, stop early, and publish what you learned.

That is how you go faster without becoming reckless, and make it easier for the next team: less faith in plans, more confidence in proof.

The future of public sector product management in a vibe-coded world is not AI doing everything. It is human values, human judgement, and human collaboration mattering even more, at a higher cadence, with less waiting to hide sloppy thinking, and fewer excuses not to learn in public.

That is a demanding future. But it is also a hopeful one: a future where we can improve services faster than we once thought possible without losing our integrity, because we have learned how to build confidence as a craft.

ABOUT THE AUTHOR

# About the author

---

I work at the intersection of digital delivery, public governance, and the practical work of earning trust.

Over the last 15+ years I've helped design and deliver public services in the UK and internationally. That has included work at the UK Government Digital Service (GDS) on GOV.UK and early thinking on Government as a Platform, as well as longer-running work on service design, digital identity, and the enabling conditions that make delivery safe at pace.

I've also contributed to OECD work on digital government and digital identity, including the [OECD Recommendation on the Governance of Digital Identity](#) and the [Good Practice Principles for Service Design and Delivery](#). After leaving the OECD I've advised and supported work with the World Bank, UN University, the governments of Azerbaijan, Kosovo, and Thailand, NHS England and DWP on the realities of digital transformation in complex, high-accountability environments.

In recent years I've been using AI-assisted development tools in two places: personal side projects, and in the context of public service transformation to bring ideas to life. What has struck me is not simply the speed, but the way these tools collapse the distance between an idea and something you can put in front of users. That

is powerful. It also raises the bar on judgement, standards, and governance, because the pressure to treat “it runs” as “it’s ready” arrives earlier.

Ultimately my work is driven by a belief in the power of digital tools to improve lives, foster equity, and strengthen trust in public institutions. I’m passionate about building a state that is inclusive by design and accountable in practice.

This paper is my attempt to describe that moment plainly, and to argue for a version of faster delivery that remains public-service shaped: evidence-led, inclusive, and accountable. I hope you find this exercise valuable.

I’ve kept the phrase “a vibe-coded world” because it names a real cultural pattern, and because it provokes the right question: what happens when software becomes easier to produce than to govern?

I used AI tools (including ChatGPT and Gemini) in drafting and editing. I did so in the same spirit the paper argues for: faster iteration, but with responsibility staying human, and with my standards and values at the heart.

[Connect with me on LinkedIn](#)

[Things I've written](#)